Multiprocessor Priority Scheduling Algorithm in Real-time Systems

Kun-Shan Wu
Department of Computer Science and Engineering
Tatung University
Email: e9506012@ms.ttu.edu.tw

Liang-Teh Lee
Department of Computer Science and Engineering
Tatung University
Email: ltlee@ttu.edu.tw

***Abstract*— In recent years, multiprocessor systems are used widespread. However, in real-time systems many scheduling algorithms are developed based on single processor systems, such as rate-monotonic (RM) scheduling algorithm and earliest deadline first (EDF) scheduling algorithm. There are some different properties between single processor systems and multiprocessor systems. Those scheduling algorithms designed for single processor systems are not suitable for applying to multiprocessor systems. For the hard real-time system, it is most important to keep all tasks to meet their deadlines. If the system's utilization is high, it is hard to approach this goal. In this paper, we propose a scheduling algorithm, Multiprocessor Priority (MPP) scheduling algorithm, for multiprocessor real-time systems. The scheduler assigns all tasks and processors different priorities to prevent the processor time's fragmentation. Moreover, it reserves longer available processor time for the task that requires longer execution time. This kind of scheduling strategy could increase system's schedulable rate when the processor's load is heavy. The simulation results show that the proposed algorithm is efficient even when the system load is heavy.**

***Index Terms*— scheduling algorithm, priority, multiprocessor system.**

I. Introduction

Multiprocessor systems have been used widely and rapidly in recent years. The scheduling strategies have become a problem in multiprocessor real-time systems [2]. The duty of the scheduler is to determine when to execute the tasks and execute it in which processor without missing its deadline [9]. An important difference between the single processor system and the multiprocessor system is that the multiprocessor system owns several processors for scheduling. The scheduler needs to determine when to execute the task, and execute the task in which processor. For scheduling the task, the scheduler needs to consider the states of all processors. It becomes a complicated and interactive problem. Furthermore, the scheduling algorithm needs to consider that the number of processors might increase in the future.

Many scheduling algorithms are developed based on single processor real-time systems. The rate-monotonic (RM) scheduling algorithm and earliest deadline first (EDF) scheduling algorithm are used widespread in single processor real-time systems. However, these algorithms may not be suitable for multiprocessor real-time systems. Some problems might come up when these algorithms are used in multiprocessor real-time systems directly [10].

Motivation

Multiprocessor real-time systems have many characteristics different from single processor real-time systems. It is not efficient to use single processor scheduling algorithms directly in multiprocessor systems. The problems of applying these algorithms in multiprocessor systems will be discussed later. Briefly, the scheduler for multiprocessor system should guarantee all tasks be executed without missing their deadlines. It should have good scheduling capability even the system has heavy loading. Moreover, the scheduler of the multiprocessor real-time systems should consider the number of the processors might increase in a system. To provide a scheduling algorithm that could schedule tasks efficiently, multiprocessor priority scheduling algorithm is proposed and implemented in this paper.

In the rest of this paper, some essential background issues related to this paper will be discussed

in chapter 2. Chapter 3 introduces the algorithm and the implementation of the proposed scheduling algorithm. Chapter 4 shows the experiments and results. Conclusions of this research will be presented in chapter 5.

II. Background

2.1 Real-time scheduler

The purpose of the original single-processor scheduling algorithm is to guarantee the tasks without missing their deadline and increase the utilization of the system. It is done by establishing the promotion to a higher priority [3] for the periodic tasks, otherwise the task will miss its deadline. The RM and EDF scheduling algorithms are used in the single processor real-time system widely. In the recent years, the multiprocessor systems are evolved rapidly. In the single processor system the scheduler decides when the task should be executed, but in the multiprocessor system the scheduler should decide not only when to execute the task but also which processor to execute the task. Besides, the multiprocessor scheduler should consider the increasing of the number of processors dynamically. We will discuss the problems of the RM and EDF scheduling algorithms in the multiprocessor real-time systems in the following sections.

2.2 RM scheduler

The rate-monotonic (RM) was a fixed priority algorithm where the task's priorities are determined by the periods of tasks. The task with tight period will get higher priority. The RM scheduler scheduled the tasks from the highest priority to the lowest priority one by one. It works well in the single processor real-time systems. However, in the multiprocessor real-time systems, the RM scheduler is not efficient. Even the loading of the system is not heavy, the system will result in un-schedulable by applying RM scheduling algorithm. Figure 2.1 is an example for the RM scheduling algorithm in the multiprocessor real-time system. The system consists of two processors and there are three tasks need to be executed. The total system loading is 80%. However, at tick 100th, the task T3 will miss its deadline.

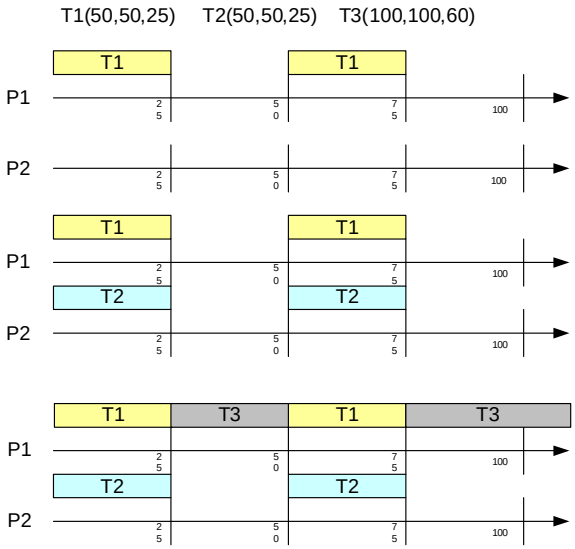


Figure 2.1: RM scheduler example

2.3 EDF scheduler

The EDF scheduling algorithm [5] scheduled the tasks based on the task's earliest deadline. EDF scheduler scheduled the tasks dynamically and it was a preemptive scheduling algorithm. Unfortunately, it is not straightforward in multiprocessor real-time systems. Figure 2.2 shows an illustrative example. In the figure, the system consists of two processors and three tasks. The loading of the system is about 75%. Firstly, the T1 and T2 are scheduled to be executed in P1 and P2 respectively. When T1 finished at tick 25th, T3 is dispatched to P1. At 50th tick, T3 is preempted by T1, and at 75th,

T3 is dispatched to P1 again. Finally, T3 misses its deadline at tick 110th.

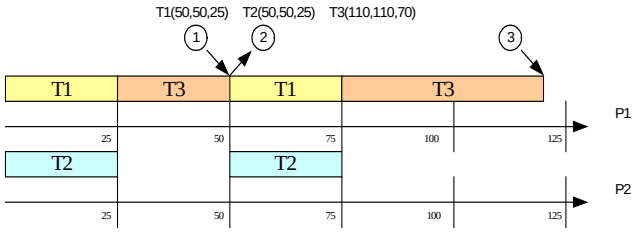


Figure 2.2: EDF scheduling algorithm in the multiprocessor real-time systems

2.4 Dual Priority Algorithm

The dual priority [DP] algorithm [10, 15] was designed for multiprocessor systems. It used a global scheduler to schedule tasks. The scheduler separates the processor's available time slice into two phases, dynamic phase and static phase. In dynamic phase the periodic tasks get lower priorities, and in static phase the periodic tasks own higher priorities. To guarantee the tasks would meet their deadlines, the scheduler sets the task to static phase at its promotion time. The promotion time was defined as below.

Promotion = deadline – worst case execution time

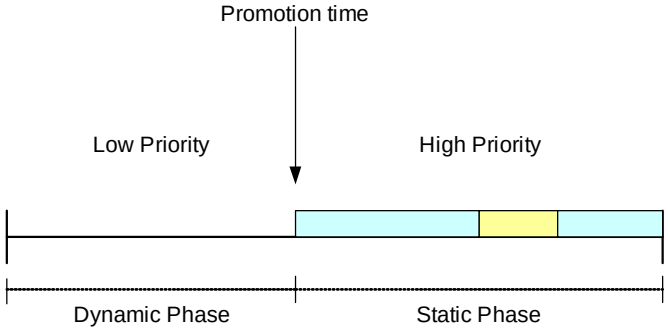


Figure 2.3 Periodic tasks allocation phases in Dual Priority Algorithm

The DP implemented a global scheduler (GS) to select the first N tasks from the queue and executed them on the N processors. During runtime, an arriving task will be queued in the Global Queue (GQ). In this queue, aperiodic tasks have higher priority than periodic tasks and they are queued in FIFO order. The High Priority Queues (HQ) is used to queue promoted periodic tasks. With this scheme, all periodic tasks' deadlines are guaranteed.

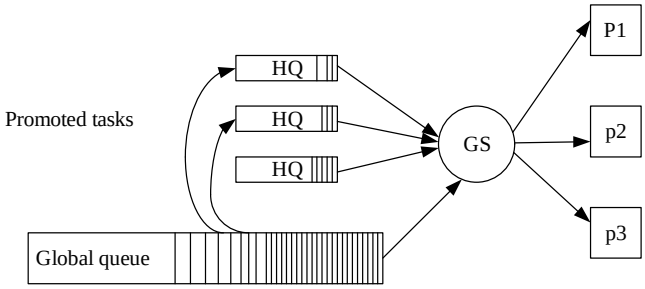


Figure 2.4 Global scheduler of Dual Priority Algorithm

The aperiodic tasks have good response time in the DP scheduler. When the system load is heavy, the DP scheduler can achieve better performance than an optimal local scheduler as the Slack Stealing scheduler [14]. The DP scheduler can obtain the higher performance of the system with a small number of processors. Nevertheless, the DP scheduler still has the fragmentation problem between the processors and it only works well in the system with few processors [10].

III. Design and Implementation of the Proposed Scheduling Algorithm

3.1 Multiprocessor Priority Algorithm

In this chapter we will describe the proposed Multiprocessor Priority (MPP) scheduling algorithm that is suitable for the multiprocessor systems. The Multiprocessor Priority scheduling algorithm is a static pre-runtime scheduling algorithm. The MPP has following characteristics: (1) it guarantees all tasks without missing their deadlines. (2) It

could schedule tasks on multiprocessor real-time systems when the system has heavy loading. (3) It reserves the possibility and flexibility to assign different priorities for the processors that have different capabilities and properties.

The MPP scheduler for the multiprocessor real-time systems is based on the offline computation of periodic task's worst case execution time. By calculating the worst case execution time of the tasks, the scheduler could guarantee the tasks to be executed without missing their deadlines. The MPP scheduler defines each processor with its own priority. The processor's priorities are defined in the order of the processor scheduling sequence. The MPP scheduler selects the task that has the highest priority in the ready queue, and then searches the processor that could execute the task without missing its deadline from the highest priority processor to the lowest priority processor.

The MPP scheduling algorithm schedules the tasks according to the processor's priority. It will schedule the task to the highest priority processor if the task does not miss its deadline. In Figure 3.1, there are three tasks scheduled in two processors. Based on the MPP scheduling algorithm, the first processor (P1) is assigned a high priority and the second processor (P2) is assigned a low priority. Firstly, the MPP scheduler selects the highest priority task (T1) to schedule. Because P1 has higher priority, the T1 will be dispatched to P1. Then, both of P1 and P2 can execute T2 without missing its deadline, but P1 has higher priority, the MPP scheduler will dispatch T2 to P1. After T1 and T2 are scheduled, the time slices of P2 are still available. It means that P2 still has capability to execute the task, such as T3, that requires long execution time to complete. In this case, both of the rate-monotonic (RM) scheduling algorithm and earliest deadline first (EDF) scheduling algorithm are non-schedulable. Task T3 will miss its deadline at tick 100th, since these two scheduling algorithms make processor's available time slices become fragmentation.

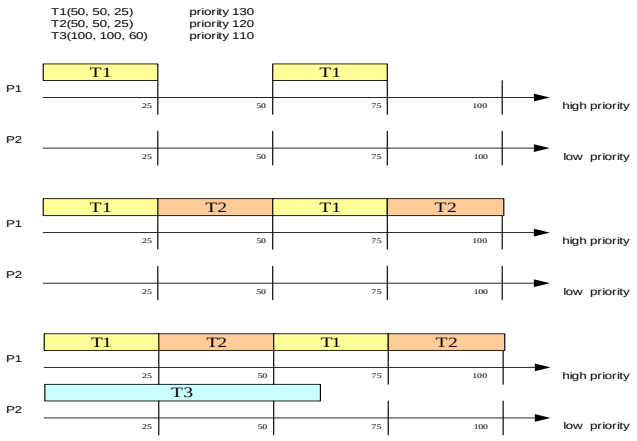


Figure 3.1: Using MPP scheduler to schedule 3 tasks with worst case execution time in 2 processors system

3.2 Framework and Assumptions

Considering a multiprocessor real-time system with N symmetrical processors and shared memory. Every processor P has a pre-assigned priority Pi. All tasks work independently and they can be preempted at any time. Each task Ti has a period of expire time Tei, deadline Tdi, and worst case execution time Tci. Assumed to satisfy Tci <= Tei. The priorities of the tasks are assigned as Tpi. The overheads for context switching, task scheduling, task preemption, and migration are assumed to be zero.

3.3 Implementation

For evaluating the proposed MPP scheduler, a simulator with applications has been constructed. The simulator is written in C language and can be

compiled by gcc or other compiler. Figures 3.2 is the flow diagram of the algorithm, respectively.

3.3.1 Flow chart of MPP

Figure 3.2 shows that the scheduler selects the highest priority task in the task queue firstly. Then the scheduler searches the available processor from the highest priority processor, but not the first available processor. If the processor can guarantee the task without missing its deadline, the task will be dispatched to this processor. If the highest processor can't guarantee the task without missing its deadline, the scheduler will test it to next processor one by one until all processors are tested. If there is no processor that could guarantee the selected task be executed without missing its deadline, it means that the system is un-schedulable.

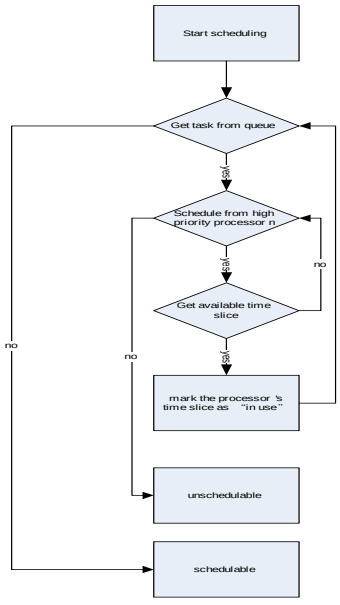


Figure 3.2: MPP flow diagram

The traditional scheduling algorithms select the processor depending on the processor's available time. These scheduling algorithms select the processor that could execute the task earliest and guarantee the task without missing its deadline. These strategies could have good response time. But its drawback is that the system's available time slices will become fragmentation in each processor in the multiprocessor real-time systems. The MPP scheduler does not select the processor that could execute the task earliest, but select the processor depends on the processor's priority. Hence, the processors that have higher priority usually execute more tasks; the processors that have lower priority usually execute fewer tasks. This strategy makes the MPP scheduler to avoid the fragmentation of the system time.

IV. Experimental Results

In this chapter, we evaluate the performance of the proposed scheduling algorithm and compare it with RM and EDF scheduling algorithms by simulation. The simulation data is generated randomly based on the worst case execution time percentage. The simulations include 4 tasks, 6 tasks, 8 tasks and 12 tasks to be scheduled in 2 processors and 4 processors in this chapter.

4.1 The notation of the simulation

The notations listed below are used to generate the task's worst case execution time for the simulation. All tasks will generate a random number and calculate the worst case execution time of the task based on the task's period. The required parameters in the formula are the task number, the processor number, the utilization of the system and the period of the task.

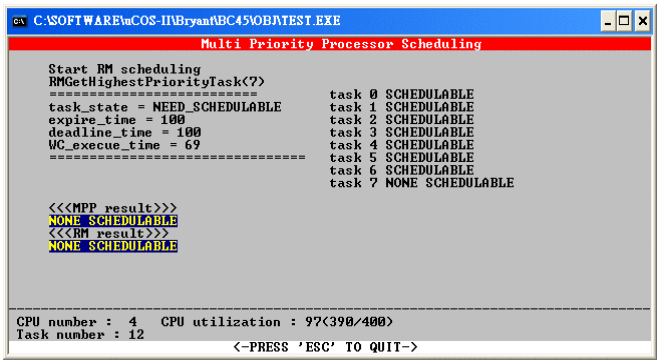
n : Total number of tasks
Pn : Total number of processors
Su : The system's utilization
i : The task i
TEP : The expiration period of the task.
R : The random number (0-1)
WCE : The worst-case execution time of the task

$$WCEi = TEPi * R / \Sigma_{i \in n} (TEPi * R) * TEPi * Pn * Su$$

4.2 Simulation with RM, EDF, and MPP algorithm

The RM scheduler schedules the task by the priority based on the task's period. The EDF scheduler schedules the task based on the earliest deadline. The MPP scheduler schedules the task based on both the task's priority and the processor's priority. In the simulation for MPP scheduler, we give the task's priorities the same as the RM scheduler. The assumption is that all processors have the same capabilities, so the processors' priorities are just defined sequentially.

In the simulation, the MPP, RM, and EDF scheduling algorithms schedule the tasks with different system utilizations. When the system utilization is lower than 50%, almost all test cases are schedulable by using these three algorithms. When the system's utilization is more than 90%, almost all test cases are un-schedulable by using these three algorithms. So in this simulation, the system's utilizations are defined from 50% to 90%. Finally, the comparisons of the efficiencies of these scheduling algorithms are made. Figures 4.1 to 4.3 are the test cases for RM, EDF and MPP scheduler for 4 processors system. The test cases are simulated in a computer with Intel® Pentium® M 1.86 GHz processor. The operation system of the simulation is µC/OS-II.



4.3 Schedulable rate in various utilizations

To compare these scheduling algorithms, the scheduler should keep all the tasks be executed without missing their deadlines. Once any task misses its deadline, the simulation application will judge it as un-schedulable case. The experiment includes four simulation cases as following sections.

4.3.1 Simulation of 4 tasks in 2 processors

Test case A is the simulation to schedule 4 tasks in the system with 2 processors by using RM, EDF, and MPP scheduling algorithms. The system's utilizations are 50%, 60%, 70%, 80% and 90%.

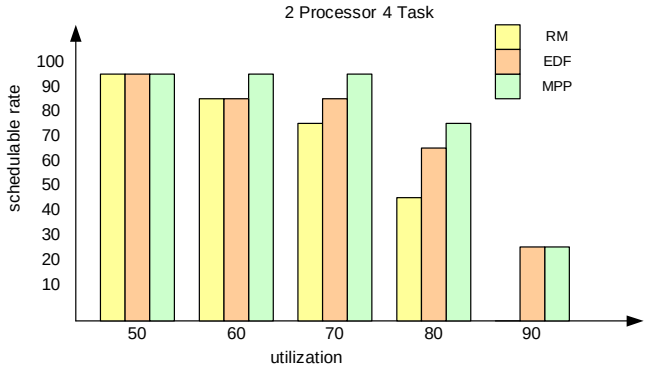


Figure 4.4: Result of 4 tasks scheduled in 2 processors with 50% to 90% system utilization

4.3.2 Simulation of 6 tasks in 2 processors

Test case B is the simulation to schedule 6 tasks in the system with 2 processors by using RM, EDF, and MPP scheduling algorithms. The system's utilizations are 50%, 60%, 70%, 80% and 90%.

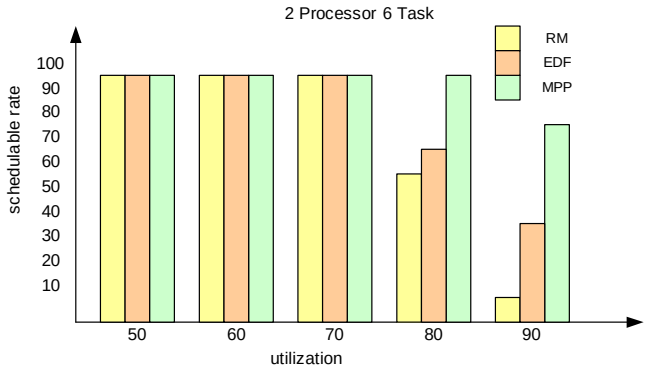


Figure 4.5: Result of 6 tasks scheduled in 2 processors with 50% to 90% system utilization

4.3.3 Simulation of 8 tasks in 4 processors

Test case C is the simulation to schedule 8 tasks in the system with 4 processors by using RM, EDF, and MPP scheduling algorithms. The system's utilizations are 50%, 60%, 70%, 80% and 90%.

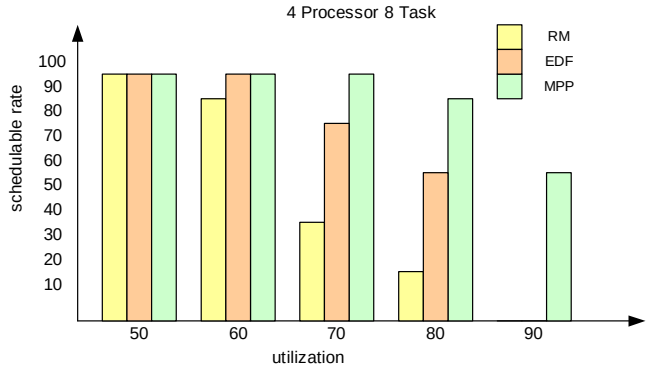


Figure 4.6: Result of 8 tasks scheduled in 4 processors with 50% to 90% system utilization

4.3.4 Simulation of 12 tasks in 4 processors

Test case D is the simulation to schedule 12 tasks in the system with 4 processors by using RM, EDF, and MPP scheduling algorithms. The system's utilizations are 50%, 60%, 70%, 80% and 90%.

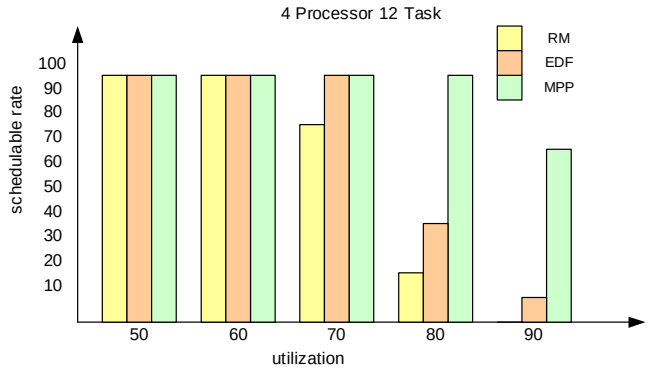


Figure 4.7: Result of 12 tasks scheduled in 4 processors with 50% to 90% system utilization

The values of the worst case execution time are generated randomly for the experimental. It simulates various distributions of the worst case execution time in these algorithms. Moreover, the simu-

lation includes various numbers of processors and tasks to compare these algorithms in different environments.

In Figures 4.4 to 4.7, the experiment shows that these scheduling algorithms could schedule the tasks well when the system's utilization is 50% and 60% in the cases A, B, C and D. When the system's utilization increases to 70%, 80% and 90%, there are some test cases become un-schedulable. The RM scheduler's schedulable rate decreased quickly especially in 4 processors test cases C and D. When the system's utilization increases to 90%, the RM scheduler is almost un-schedulable in all test cases. Even using the EDF scheduler in cases A, B, C and D, the schedulable rate is low when the system's utilization increases to 80% and 90%. For the MPP scheduler, all the test cases are schedulable when the system's utilizations are 50% to 70%. Even the system's utilization increases to 80% and 90%, the MPP scheduler still has efficient scheduling capability.

In the experiment, it shows the schedulable rates of three schedulers in multiprocessor real-time system and the schedulable rate's variation in different system's utilizations. Generally, the MPP scheduler's efficiency is better than EDF scheduler and RM scheduler; the EDF scheduler's efficiency is better than RM scheduler in the multiprocessor real-time system.

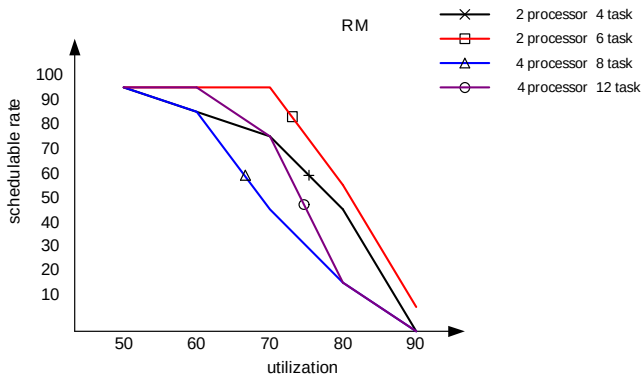


Figure 4.8: Result of the efficiency of the RM scheduling algorithm

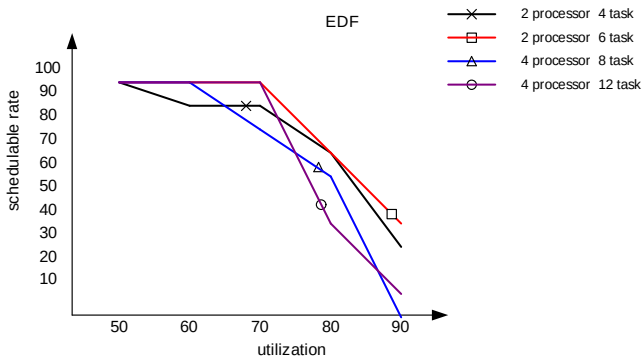


Figure 4.9: Result of the efficiency of the EDF scheduling algorithm

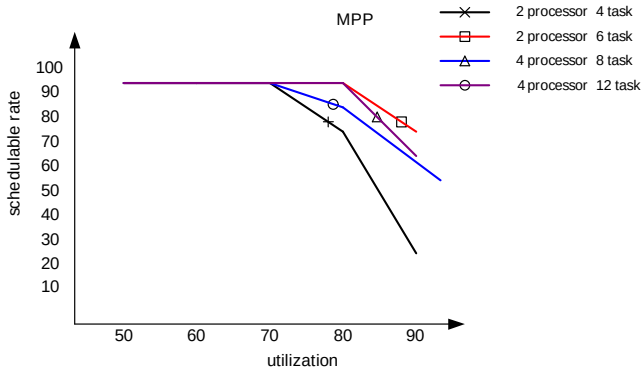


Figure 4.10: Result of the efficiency of the MPP scheduling algorithm

In Figures 4.8 to 4.10, when the system's utilization is 60%, there are some un-schedulable test cases by using RM scheduler. There are two

un-schedulable test cases by using EDF scheduler. But the first un-schedulable test case appears by using MPP scheduler when the system's utilization is 80%.

When the system's utilization is increased to 90%, the RM scheduler almost becomes un-schedulable in all test cases. The RM scheduler's schedulable rate is only 10 % in the 2 processors for 6 tasks test case; in all of the other test cases, the RM scheduler is un-schedulable at all. The EDF scheduler's schedulable rates are from 0% to 40% when the system's utilization is 90%. It means that the EDF scheduler could not work well when system has heavy loading in the multiprocessor real-time system. The MPP scheduler's schedulable rates are from 30% to 80% when the system's utilization is 90%. The experimental result shows that when the system's utilization increases to 90%, the MPP scheduler is un-schedulable in some test cases. But the MPP scheduler is still more efficient than RM scheduler and EDF scheduler in the multiprocessor real-time system when the system has heavy loading.

After the un-schedulable case appears, the slope of the un-schedulable rate in the Figures 4.8, 4.9 and 4.10 is RM > EDF > MPP. It means that when the un-schedulable test case appears, the efficiency became worse quickly by using RM and EDF scheduling algorithm.

4.3.5 Power saving

In the experiments, the MPP scheduler always dispatches the tasks to the higher priority processor firstly. This strategy causes that the higher processors execute the tasks firstly. Hence, the processors with lower priorities might not need to execute any tasks when the system's loading is not heavy. It means that these processors are in the idle state at all times. Figure 4.11 shows an example of the system with an idle processor. The system includes four processors to schedule seven tasks. After the system scheduled by MPP scheduler, the 4th processor (P4) does not execute any task at all times.

If the processors of the system have Dynamic Voltage Scaling (DVS) capability, the system could turn off those processors that are in the idle states. Moreover, the system designer could use fewer processors in the system to reduce the system cost. By turning off the processors that are in the idle state or reducing the number of the processors, the power consumption of the system is reduced.

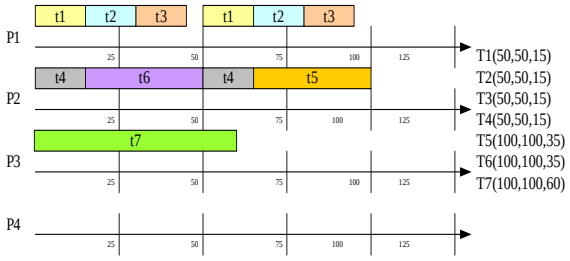


Figure 4.11: Example for power saving

4.4 Summary

In this chapter, the experiments show the results in multiprocessor real-time systems by using MPP, RM, and EDF scheduling algorithms. The experiment was simulated with various cases with different system loading and system with different number of processors. To meet the general cases, the worst case execution times of the tasks are generated randomly. Generally, the MPP scheduler gets a better scheduling efficiency in the multiprocessor real-time system in all simulation cases. When the system's loading is heavy, the schedulable rate of MPP scheduler is still acceptable. Moreover, the

MPP scheduler gets a higher possibility to save the power consumption of the system in general cases.

V. Conclusions

In this paper, we propose an efficient scheduling algorithm for multiprocessor real-time systems. The experimental results show that the schedulable rates of MPP scheduler in all simulation cases are better than RM and EDF schedulers.

Furthermore, when the system is schedulable, the MPP scheduler prevents the system available time slice from fragmentation. It means that once a task requires a long worst-case execution time, the scheduler gets a better probability to schedule this task.

In addition, the MPP scheduler keeps the flexibility for the multiprocessor real-time system. If the processors of a system have different capabilities and characteristics, the MPP scheduler will assign a suitable priority to each processor. Hence, tasks can be scheduled with some strategies as the user's design by defining the processors' priorities of the system.

In this paper, the experiment assumes that the capabilities of the processors are all the same in the system. Hence, the priorities of the processor are predefined statically. In the simulation, some test cases of the MPP scheduling algorithm are un-schedulable with the predefined processor priority. If the MPP scheduler could change the processor's priority dynamically then the system might become schedulable. The scheduler changes the processor's priority based on the processor's loading, the heavier loading the higher priority. Through this method, tasks will be more concentrated in high priority processors and the system's fragmentation can be decreased further.

In the future, the processors of a system may have different capabilities. The strategy to define the priorities of the processors for the system and the strategy to improve the system performance by applying the proposed algorithm are subjects need to be further studied.

REFERENCE

[1] A. Burns and A. Wellings, Real Time Systems, and Programming Languages, 3rd Ed, http://www.cs.york.ac.uk/rts/books/RTSBookThirdEdition.html

[2] A. Burns, "Scheduling Hard Real-Time Systems," Software Engineering, Volume 6, Issue 3, pp 116-128, Journal, 1991

[3] B. Anderson, S. Baruah, and J. Jonsson, "Static-Priority Scheduling on Multiprocessors," 22nd Real-Time Systems Symposium, pp. 41-43, 2003

[4] B. Sprunt, L . Sha, and J. P. Lehoczky, "Aperiodic Task Scheduling for Hard Real-Time Systems," Real-Time Systems Journal, Volume 1, pp. 27-60, June 1989

[5] J. Gossens, S. Funk, and S. Baruah, "EDF scheduling on Multiprocessor platforms: some (perhaps) counterintuitive observations," Real-Time Computing Systems and Applications, pp. 321-330, 2002

[6] J. M. Banús, A. Arenas, and J. Labarta, "An Efficient Scheme to Allocate Soft-Aperiodic Tasks in Multiprocessor Hard Real-Time Systems," Parallel and Distributed Processing Techniques and Applications, Volume 2, pp. 809-815, 2002

[7] J. P. Lehoczky, L. Sha, and Y. Ding, "The Rate-Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior," Proceedings of Real-Time Systems Symposium, pp. 166-171, 1989

[8] J. P. Lehoczky and S. Ramos-Thuel, "An Optimal Algorithm for Scheduling Soft-Aperiodic Tasks in Fixed Priority Preemptive Systems," RealTime Systems Symposium, Volume 2, Issue 4, pp. 110-123, December 1992

[9] K. Ramamritham, J. A. Stankovic, and W. Zhao, "Distributed Scheduling of Tasks with Deadlines and Resource Requirements," IEEE Transactions on Computers, Volume 38, pp. 1110-1123, August 1989

[10] M. Banús Josep, Alex Arenas, and Labarta Jesús, “Dual Priority Algorithm to Schedule Real-Time Tasks in a Shared Memory Multiprocessor,” IEEE Computer Society Washington, DC, pp. 2-12, USA, 2003

[11] M. Joseph and P. Pandya, "Finding Response Times in a Real-Time System," British Computer Society Computer Journal, Cambridge University, Volume 29, Issue 5, pp. 390-395, 1986

[12] M. L. Dertouzos and A. K. Mok, "Multiprocessor On-Line Scheduling of Hard-Real-Time Tasks", IEEE Transactions on Software Engineering, Volume 15, pp. 1497-1506, 1989

[13] M. R. Garey and D. S. Johnson, "Complexity Results for Multiprocessor Scheduling under Resource Constraints," SIAM Journal on Computing, Volume 4, Issue 4, pp. 397-411, 1975

[14] P. Lehoczky John and R. Thuel Sandra, "Scheduling Periodic and Aperiodic Tasks using the Slack Stealing Algorithm," Prentice-Hall, Chapter 8, pp. 175-197, 1994

[15] R. Davis and A. Wellings, "Dual Priority Scheduling", 16th Real-Time Systems Symposium, pp.100, 1995

[16] µC/OS-II, The Real-Time Kernel
http://www.micrium.com/products/rtos/kernel/rtos.html