Report Number: WUCSE-2006-55

2006-01-01

Flexible Maximum Urgency First Scheduling for Distributed Real-Time Systems

Yingming Chen and Chenyang Lu

2006-55

Flexible Maximum Urgency First Scheduling for Distributed Real-Time Systems

Authors: Yingming Chen, Chenyang Lu

Type of Report: Other

Department of Computer Science & Engineering - Washington University in St. Louis
Campus Box 1045 - St. Louis, MO - 63130 - ph: (314) 935-6160

In the rest of this report, we first describe the design of the FMUF scheduling strategy. Then, we present simulation results to demonstrate that FMUF can provide performance isolation for mission-critical tasks.

2 FMUF Scheduling

The following are some definitions for *FMUF* Scheduling strategy:

-Two types of tasks: *mission-critical* tasks and *best effort* tasks,categorized by its importance specified by users.

-All real-time priorities are divided into two classes: a high priority class and a low priority class. A task whose priority is in the high priority class is in the *high* group. The other tasks are in the *low* group. That is, a task in the *high* group always has a higher priority than any tasks in the *low* group. *Inside* each group, all tasks are ordered by their end-to-end deadline, and end-to-end Deadline Monotonic scheduling algorithm is applied to assign subpriorities.

The basic idea of FMUF is to utilize an adaptive mechanism to dynamically adjust the priority of *best effort* tasks (i.e., move some best effort tasks into or out of the *high* group) according to the current conditions at runtime. In the rest of this section, we present two approaches to implement this strategy.

2.1 End-to-end Task Model

A system has m end-to-end periodic tasks $\{T_i|1 \le i \le m\}$ executing on n processors $\{P_i|1 \le i \le n\}$. Task T_i is composed of a chain of subtasks $\{T_{ij}|1 \le j \le n_i\}$ that may be allocated to multiple processors.A subtask $T_{ij}(1 \le j \le n_i)$ cannot be released for execution until its predecessor $T_{i,j-1}$ is completed. a non-greedy synchronization protocol (e.g., release guard [1]) is used to enforce the precedence constraints between subsequent subtasks.Hence, each subtask T_{ij} of a periodic task T_i is also periodic and shares the same rate as T_i. Each task T_i is subject to an end-to-end relative deadline related to its period.

2.2 FMUF based on deadline miss ratio

In this approach the priorities of some *best effort* tasks are adjusted based on the history of deadline miss. We first introduce several notations:

- T_0: the *best effort* task whose priority has just been adjusted in the latest adaptation operation
- $U_{i,k}$: the total utilization of processor $)P_i$ during the k^{th} sampling period
- S_i: the set of processors that host subtasks of T_i

The Logic of FMUF

1. At the start time of the system, place all *mission-critical* tasks into the *high* group(by assigning high priority to them); and add all *best effort* tasks into the *low* group(by assigning low priority to them); set $T_0 = null$

2. In the end of every sampling period, monitor the total utilizations $\{U_{i,k}|1 \leq i \leq n\}$ of all processors and check the deadline miss of all tasks
   (a) **If** there is deadline miss in the *high* group, move a best effort task T_s from the *high* group to the *low* group(by decreasing its priority), and set $T_0 = T_s$
   (b) **If** (i) there is no deadline miss in the *high* group for consecutive N ($N > 1$) sampling periods and (ii) there are deadline misses in the *low* group, choose one best effort task T_s from the *low* group according to some selection policies(e.g., shortest end-to-end deadline first)
      i. **If** $T_s \neq T_0$, **then** move T_s to the high group (by increasing its priority), and set $T_0 = T_s$
      ii. **Else** calculate $\Delta U_i = U_{i,k-1} - U_{i,k}$ for all processors in S_s; if $\min_{P_i \in S_s} \Delta U_i \geq$ *threshold*, then move T_s to the high group

In this appraoch, an adaptation operation can be triggered by several reasons, such as the arrival and departure of mission-critical tasks, and significant variation of execution times. The *threshold* is defined to avoid oscillation of the system, i.e., one task has been moved into and out of the high group repeatedly. The value of *threshold* need to be tuned.

Here is one simple example to demonstrate how FMUF works. In this example, we set $N = 5$ and *threshold* $= 5\%$. The workload includes 7 tasks: three *mission-critical* tasks, T_1 with end-to-end deadline $D_1 = 400$, T_2 with $D_2 = 500$, and T_3 with $D_3 = 600$; four *best effort* tasks, T_4 with $D_4 = 350$, T_5 with $D_5 = 300$, T_6 with $D_6 = 400$, and T_7 with $D_7 = 550$. There are two processors. Each task has one subtask on each processor. In the beginning of the $(k-1)^{th}$ sampling period, there is no deadline miss for all tasks. All three *mission-critical* tasks are in the *high* group, whereas there are two *best effort* tasks, T_4 and T_5, in the *high* group. Other two *best effort* tasks are in the *low* group. During the $(k-1)^{th}$ sampling period, execution time of T_1 increases significantly, which incurs T_3 miss deadline. An adaptation operation is triggered at the end of $(k-1)^{th}$ sampling period. The best effort task with the longest end-to-end deadline in the *high* group,T_4 here, is moved to the *low* group. T_0 is set to T_4. For the next 10 sampling periods, there is no deadline miss in the *high* group while deadline miss occurs in the *low* group. Adaption operation is triggered in the end of each sampling period. However, as T_s is always equal to T_0 and the utilization deviations of two processors are both less than the *threshold* 5%, no task will be moved to the *high* group according to the algorithm. At the end of $(k+10)^{th}$ sampling period, T_2 departs from the system. The utilization of either T_{21} or T_{22} is greater than 5%. Then after the adaption operation, T_4 will be moved back to the *high* group. In this example, we select the *best effort* task to adjust priority according to the end-to-end deadline in order to decrease the priority inversion. We can easily replace this with other selection policies.

2.3 FMUF based on *AUB condition*

In this algorithm, we try to adjust the priority of some *best effort* tasks based on Aperiodic Utilization Bound(AUB) for end-to-end tasks [2]. We use AUB to guarantee all tasks in *high* group meet their deadlines. The schedulability condition of Aperiodic Utilization Bound is computed by:

$$\sum_{j=1}^{N} \frac{U_j(1 - U_j/2)}{1 - U_j} \leq 1 \qquad (1)$$

N is the number of processors;U_j is the estimated utilization of subtasks in *high* group on processor j, $U_j = \sum_{T_i \in G_j} C_{ij}/D_i$. For simplicity, in the rest of this section, we call inequation 1 *AUB Condition*.

Before we describe the logic of the algorithm ,we first introduce two notations:

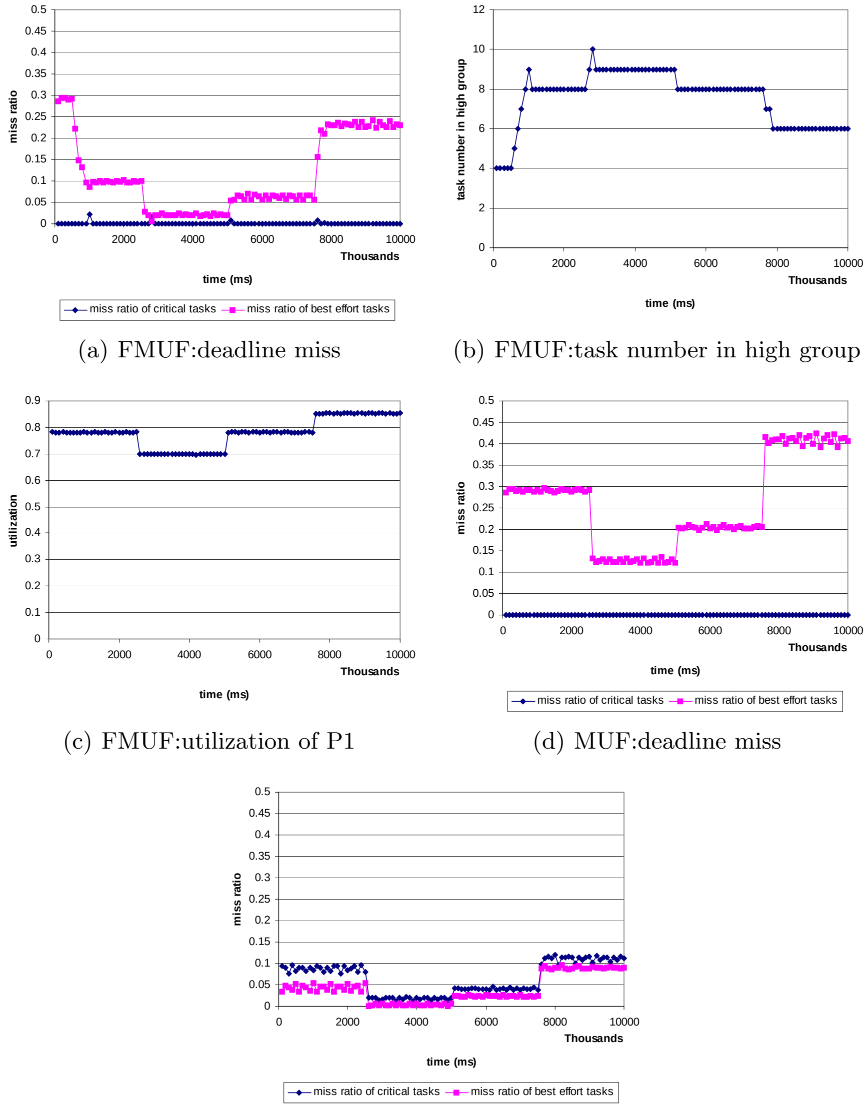
- B_h: set of *best effort* tasks in the *high* group
- B_l: set of *best effort* tasks in the *low* group

The Logic of FMUF

1. At the start time of the system, place all *mission-critical* tasks into the *high* group(by assigning high priority to them), and add all *best effort* tasks into the *low* group(by assigning low priority to them);
2. in the end of every sampling period, check the AUB condition
   (a) **If** AUB condition is satisfied: select one task T from B_l and check the new AUB condition for the *high* group after including T; if new AUB condition is not violated, move T into the *high* group; select another task from B_l and repeat this test until all tasks in B_l have been tested
   (b) **Else** AUB condition is violated: move one task T' from B_h to the *low* group and check the new AUB condition for the *high* group after excluding T'; if new AUB condition is not satisfied yet, select another task from B_h and repeat this process until the new AUB condition is satisfied or B_h is empty

3 Simulation Results

In this section, we evaluate the performance of the first approach presented int Section 2.2. The simulation runs on the EUCON simulator with a medium-sized workload that comprises of 11 end-to-end tasks. The task set includes 4 mission-critical tasks and 7 best-effort ones. Each task has 6 subtasks, which are deployed on 6 different processors(for simplicity, subtask i on processor i, $1 \leq i \leq 6$). The workload parameters are shown in Table 1. ET1-ET6 are corresponding to the execution times of subtask1-subtask6. There are only 6 different execution times: 35,40,55,65,70, and 90. Criticality = 1 means miss-critical task; criticality = 0 is corresponding to best-effort task. The time unit of execution time and period is millisecond. We assume each task's end-to-end deadline is equal to its period. In the beginning, only first 10 tasks run. Task 11 will arrive later. The control period $T_s = 100s$. Other two parameters for FMUF are chosen like this: N = 5, threshold = 5%.



(a) FMUF:deadline miss

(b) FMUF:task number in high group

(c) FMUF:utilization of P1

(d) MUF:deadline miss

(e) DMS:deadline miss

Fig. 1. Experimental results: FMUF, MUF, and DMS

In the experiment, the system runs for 10000 seconds. We create three scenarios to evaluation the performance of FMUF: 1) at 2500 seconds, execution times of all subtasks are decreased by 10%; 2) at 5000 seconds, task 11 arrives; 3) at 7500 seconds, execution times of all subtasks are increased by 10%. In order to evaluate the performance of FMUF, we compare it with two baselines: MUF and DMS. We plot the results in Figure 1. The FMUF control operation is trig-

Table 1. workload parameters(ms)

Tasks	ET1	ET2	ET3	ET4	ET5	ET6	period	criticality
1	55	40	65	55	40	65	700	1
2	90	65	70	90	65	70	1000	1
3	65	70	65	70	55	65	900	1
4	35	40	40	35	35	40	500	0
5	70	65	65	55	65	70	800	0
6	70	65	90	70	90	70	1200	0
7	40	55	35	40	40	35	600	0
8	65	55	55	70	40	55	700	0
9	70	65	65	90	70	65	900	0
10	35	40	35	35	40	35	400	0
11	65	55	65	70	65	70	700	1

gered four times: in the initial stage, three best-effort tasks are moved into high group; after scenario 1, two more best-effort tasks are moved into high group; after scenario 2, two best-effort tasks are move back to low group; after scenario 3, another two best-effort tasks are degraded into low group.The results shows that FMUF can provide performance isolation for mission-critical tasks.

References

1. Jun Sun, Jane W.-S. Liu, I.: Synchronization Protocols in Distributed Real-Time Systems. ICDCS(1996)
2. Tarek F. Abdelzaher, Gautam H. Thaker, Patrick J. Lardieri, I.: A Feasible Region for Meeting Aperiodic End-to-End Deadlines in Resource Pipelines. ICDCS (2004) 436-445