Preliminary Performance Evaluation of HEF Scheduling Algorithm

Carlos A. Rincón [†][*] and Albert M. K. Cheng[*]
[†]Networking and Telematics Academic Unit, Universidad del Zulia, Maracaibo, Venezuela. Email: crincon@fec.luz.edu.ve
[*]Real-time Systems Laboratory, University of Houston, Houston, USA. Email: cheng@cs.uh.edu

***Abstract*—The purpose of this paper is to analyze the performance of the Highest Entropy First (HEF) scheduling algorithm for real-time tasks. We generate multiple task sets using the Seoul National University (SNU) real-time benchmark. The tasks were implemented on WindRiver Workbench 3.3 to estimate the WCET. A linear programming solution was implemented to set the period of the tasks aiming to maximize the utilization of the system based on a predefined hyper-period. We measure the performance of HEF scheduling algorithm using as parameters the number of context switches and the deadline-miss ratio. As preliminary result we show that the number of context switches is directly proportional to the number of tasks in a task set. The deadline-miss ratio for all the studied cases was 0%, because the utilization for all the task sets was at most 1 ($U \leq 1$).**

***Keywords*—Highest Entropy First; real-time systems; scheduling; performance.**

I. Introduction

In recent years the use of entropy as a parameter has been proposed as a new approach to schedule real-time tasks [1]. In 2015, Rincon and Cheng [2] presented the mathematical background to measure the entropy of a task set in a real-time system as well as the design, feasibility analysis and implementation of the highest entropy first (HEF) algorithm to schedule real-time tasks in uni-processors.

The HEF algorithm is a new dynamic priority technique to schedule real-time tasks that tries to minimize the uncertainty (based on the probability of the execution of a task during the hyper-period) of the scheduling problem by executing the task with the highest entropy first without missing any deadline.

The purpose of the research is to measure the performance of the highest entropy first scheduling algorithm in order to have a guideline about the behavior of the studied algorithm under certain conditions.

The contributions of this paper are:

- Generate multiple task sets by implementing the programs from the SNU real-time benchmark [3] in Wind River Workbench 3.3 [4] to calculate the WCET and generating the periods by using a linear programming solution aiming to maximize the utilization of the system based on a predefined hyper-period.
- Measure the performance of HEF algorithm to schedule real-time tasks using as metrics the number of context switches and deadline-miss ratio.

The rest of the paper is organized as follows. In the next section, we describe the related work about using entropy as a parameter to schedule real-time tasks. In section 3, we present the design of the HEF scheduling algorithm. Section 4 presents the methodology used to generate the task set for the performance evaluation. Section 5 presents the performance evaluation of the HEF algorithm running the generated task set. We give our conclusions and future work in section 6.

[*]Supported in part by the National Science Foundation under Awards No. 0720856 and No. 1219082.

II. Related Work

A. Entropy as a Parameter for Real-time Scheduling

Entropy is defined as the product of the information generated by an event x and the probability of occurrence of that event ($p_x * I_x$) [5]. Considering a periodic task system with implicit deadlines where the worst case execution time = C_i, period = T_i, hyper-period ($hperiod$) = least common multiple of the periods and applying the information-theoretic concepts, we define the following parameters:

Entropy of a Single Time Unit from a Scheduling Diagram H_{SU}: we define the information generated by a single time unit of the scheduling diagram (I_s) as $log_2(1/P_s)$, where P_s is the probability of a single time unit = $1/hperiod$. The entropy of a single time unit = $P_s * I_s = log_2(hperiod)/hperiod$ bits.

Total Entropy of a Task H_{Task}: is defined as the product of the number of single time units on the scheduling diagram used by a task (number of task instances times C_i) and the entropy of a single time unit (H_{SU}).

$$H_{Task} = \frac{hperiod}{T_i} * C_i * H_{SU} = log_2(hperiod) * \frac{C_i}{T_i} bits \tag{1}$$

Total Normalized Entropy of a Task NH_{Task}: is defined as the total entropy of a task (H_{Task}) divided by its computation time (C_i). This parameter is critical for using entropy in real-time systems because it prioritizes the scheduling based on the task deadlines.

Total Entropy of the System H_{Sys}: is the summation of the entropies of all m tasks (with m=number of tasks).

$$H_{Sys} = \sum_{i=1}^{m} H_{Task_i} = log_2(hperiod) * \sum_{i=1}^{m} \frac{C_i}{T_i} bits \tag{2}$$

Relationship between H_{Sys} and Utilization: Based on Shannon's information theory [5], we know that the maximum value of the entropy (H_x) = log_2(number of possible cases=$hperiod$). Then $H_{Sys} \leq log_2(hperiod)$. Based on this inequality, we have:

$$log_2(hperiod) * \sum_{i=1}^{m} \frac{C_i}{T_i} \leq log_2(hperiod) \tag{3}$$

This inequality is true only if $U = \sum_{i=1}^{m} C_i/T_i \leq 1$.

III. The Highest Entropy First scheduling algorithm

The studied algorithm is a dynamic priority scheduler that uses the normalized entropy of the task (NH_{Task}) and the total entropy of the task (H_{Task}) to decide which task to run first. Basically, every time the scheduler needs to decide which task to run, it will choose the task with the highest entropy (normalized and total), in order to minimize the complexity of the scheduling problem.

A. Algorithm's Design

The proposed scheduling algorithm based on entropy has the following steps:

1) Determine the schedulability of the given task set using the relationship between entropy and utilization proposed in equation (3).
2) Calculate the normalized and total entropy for each task.
3) Select the task to be executed using the following criteria:
   - Select the task with the highest normalized entropy.
   - If two or more tasks have the highest normalized entropy, then select the task with the highest total remaining entropy.
   - If two or more tasks have the highest total remaining entropy and one of these tasks is the one running, then select the task that is running (to minimize preemption), else select the task based on its process identifier (PID).
4) Update the values of T_i and C_i for all tasks.
5) Go to step 2 until time = $hperiod$.

IV. Generating the Task sets

In order to generate the tasks sets to measure the performance of the HEF scheduling algorithm we selected 5 programs from the SNU Real-time benchmark (sqrt.c, fibcall.c, crc.c, minver.c and select.c). Table I shows a description of each selected program.

TABLE I: Selected programs from the SNU real-time benchmark

Task Number	SNU Program	Description
1	sqrt.c	Square root,function implemented by Taylor series
2	fibcall.c	Summing the Fibonacci series
3	crc.c	A demonstration for CRC (Cyclic Redundancy Check) operation
4	minver.c	Matrix inversion for 3x3 floating point matrix
5	select.c	A function to select the Nth largest number in the floating point array size 20

After selecting the programs we implemented them on a server with an Intel i7-3770 processor running at 3.4 GHz, with 16 GB of RAM and 2 TB hard drive using Wind River Workbench 3.3 to calculate the worst case execution time (WCET). We run each program 100 times to average the results. Table II shows the average WCET for the selected programs.

A. Task sets

To measure the performance of the HEF scheduling algorithm we decided to use as independent variable the number of tasks in the task set. We created 4 task sets with 2, 3, 4, and 5 tasks respectively. For each task set we use 100 ms as the hyper-period and applied a linear programming solution to calculate the periods for each task (aiming to maximize the utilization of the system). For the deadlines, we implemented a system with implicit deadlines. Tables III and IV show the generated task sets.

TABLE II: WCET for the selected tasks

Task Number	WCET	ROUND WCET
1	13.34 ms	14 ms
2	7.32 ms	8 ms
3	13.54 ms	14 ms
4	16.41 ms	17 ms
5	25.73 ms	26 ms

TABLE III: Task sets 1 and 2

Task Number	C_i	T_i
1	14 ms	50 ms
2	8 ms	12 ms

Task Number	C_i	T_i
1	14 ms	100 ms
2	8 ms	50 ms
3	14 ms	20 ms

TABLE IV: Task sets 3 and 4

Task Number	C_i	T_i
1	14 ms	100 ms
2	8 ms	34 ms
3	14 ms	50 ms
4	17 ms	50 ms

Task Number	C_i	T_i
1	14 ms	100 ms
2	8 ms	100 ms
3	14 ms	100 ms
4	17 ms	50 ms
5	26 ms	100 ms

The utilization of the generated task sets are: Task set 1 = 0.946666667, Task set 2 = 1, Task set 3 = 0.995294118 and Task set 4 =0.96.

V. Preliminary Performance Evaluation of HEF

With the generated task sets, we run the HEF scheduling algorithm to measure its performance. We calculated the number of context switches and the deadline-miss ratio for each task set. The results are shown in figure 1.

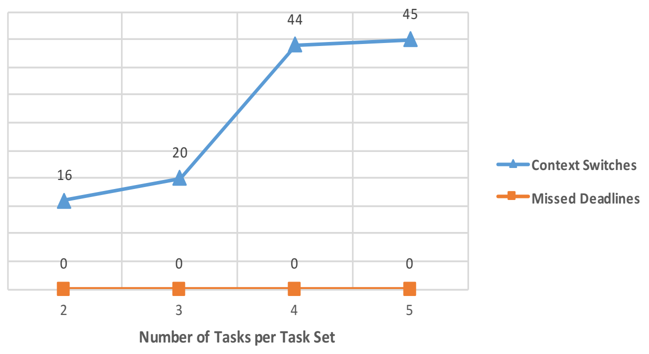

Fig. 1: Number of context switches and deadline-miss ratio for HEF

For task set 1 (2 tasks) the number of context switches is 16 and the deadline-miss ratio is 0%, for task set 2 (3 tasks) the number of context switches is 20 and the deadline-miss ratio is 0%, for task set 3 (4 tasks), the number of context switches is 44 and the deadline-miss ratio is 0% and for task set 4 (5 tasks), the number of context switches is 45 and the deadline-miss ratio is 0%.

These results show that when the number of tasks in the task set increases, the number of context switches increases. The obtained deadline-miss ratio (0 missed deadlines) for all the tasks sets is a consequence of the utilization values (less or equal than 1 for all task sets).

VI. Conclusions and Future Work

In this paper, we have proposed a methodology to generate task sets using the programs from the SNU real-time benchmark (implemented on Wind River Workbench). We calculated the periods for the tasks in the task set using a linear programming solution to maximize the utilization of the system.

The results from the preliminary performance evaluation of the HEF scheduling algorithm show that the number of context switches is directly proportional to the number of tasks in the task set. For the deadline-miss ratio, further analysis must be made to confirm that it depends on the utilization of the system ($U \leq 1$ = no deadline misses).

The HEF algorithm has some similarities with the earliest deadline first algorithm [6] (EDF) because selecting the task with the lowest absolute deadline is the same as selecting the task with highest normalized entropy ($NH_{Task} = log_2(hperiod) * \frac{1}{T_i}$). However when two or more tasks have the same absolute deadline, HEF will select the task that adds more complexity to the scheduling problem using as a parameter the total remaining entropy of the task. Therefore we propose as future work to compare the performance of HEF against EDF using the task sets generated by the methodology proposed in this paper.

References

[1] R. Sharma and Nitin, “Entropy, a new dynamics governing parameter in real time distributed system: a simulation study,” *IJPEDS*, vol. 29, no. 6, pp. 562–586, 2014.

[2] C. A. Rincon and A. M. Cheng, “Using entropy as a parameter to schedule real-time tasks,” in *Real-Time Systems Symposium. WiP Session, 2015 IEEE*, Dec 2015, pp. 375–375.

[3] “Snu real-time benchmark suite,” http://archi.snu.ac.kr/realtime/benchmark.

[4] WindRiver, “Wind river workbench,” http://www.windriver.com.

[5] C. E. Shannon, “A Mathematical Theory of Communication,” *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[6] C. Liu and J. Layland, “Scheduling algorithms for multiprogramming in a hard-real-time environment,” *J. ACM*, vol. 20, no. 1, pp. 46–61, 1973.