COSC 4331/6384 Real-Time Systems – Spring 2023
Assignment 2 (Programming) - Due: Wednesday, March 29, 2023

This assignment is motivated by the fact that portable, battery-operated devices such as mobile phones and monitoring instruments not only need to meet timing constraints (for example, in video conferencing) but also ensure that these devices operate for as long as possible between battery charging. In a real-time system, we can reduce the processor's voltage and thus the speed of execution of a task while still meeting its specified completion deadline if there is laxity. This results in an overall quadratic reduction in the energy required to complete the task. This assignment consists of two parts: (1) read and understand the referenced paper [1], and (2) write a program to implement the power-aware scheduling algorithm for sporadic tasks in this referenced paper (Algorithm STS).

A sporadic task is denoted by T = {a, c, d, s}, where a is the arrival time, c is the computation time, d is the deadline, and is the context switching time. In this assignment, it is assumed that the speed/voltage of the system/processor running these tasks can be scaled in the interval (0, 1.0].

A scheduling decision is made whenever any of the following two events occurs: (i) Event-1: a new sporadic task arrives and is accepted into the system by the acceptance test, and (ii) Event-2: the current task completes its execution. When an Event-1 occurs, the scheduler updates the optimal voltage schedule of the processor for all the tasks including this new one in the task queue. The variable voltage scheduling algorithm is shown as Algorithm STS in [1]. When an Event-2 occurs, the completed task is removed from the task queue and we execute the task at the head of the task queue following the current voltage schedule.

What you need to do when implementing the STS algorithm [1]: Given an input - a set of sporadic tasks arriving dynamically, schedule the tasks in an online manner to (1) meet all deadlines or discard the newly arriving task if it would cause any accepted/uncompleted task to its deadline, including the new task itself; and (2) to schedule the tasks in optimal speeds in order to minimize the energy consumption. Output: The complete task schedule and the optimal speed schedule.

You will need to implement two versions of the STS algorithm. In the 1st version you will calculate the clock speed as a continuous value, much like in Figure 1 of [1]. For the 2nd version we will provide for you a set of discrete speeds that you may select when changing the speed of the CPU. For instance, we may limit the speeds to a range of number between (0, 1.0] such as (0.1, 0.3, 0.5, 0.7, 0.9, 1). When changing the speed of the algorithm you will need to only use one of these values provided you.

What you will be given for the input is a file input.txt. You program needs to read the input from this file. In the file, the format of the input is as follows. The parameters a, c, d, and s for each task are respectively its arrival time, computation time at the reference (highest) voltage, deadline, and context-switching time (from it to another task at the completion of the first task or its interruption) at the reference (highest) voltage. Note that the referenced paper [1] only considers a, c, and d. Therefore, s in the paper is always 0. This assignment makes the power-aware scheduler more realistic by considering context-switching times.

Context switching should be considered whenever you switch from one task to another. So if you change from T_1 -> T_2 your total context switching time should be $S_1 + S_2$. Note, if T1 is the last task running, you will not need to consider the context switching time when it finishes.

Number of tasks: N
Task 1: [a c d s]
Task 2: [a c d s]
….
….
Task N: [a c d s]

Here is an example of the file:

Number of tasks: 3
Task 1: [0 4 11 1]
Task 2: [3 5 10 1]
Task 3: [6 5 16 2]

Please note that the tasks are assumed to arrive dynamically, although the input file makes it looks like static. That is, your system does not know the task until it arrives. You can always assume that all tasks arrive and finish their works in [0, 200]. There is no floating point number in the program and result, rounding the floating point numbers to integers appropriately if necessary.

Bonus (20 points):

For this assignment, we have only required you to implement the STS algorithm for sporadic tasks as outlined in [1]. For the bonus, implement the HPASTS algorithm detailed in section 5.2.2 of [1]. In other words, implement the algorithm for both sporadic and periodic tasks. You may need to look at other papers in order to make sure that this is correct.

For input to the bonus, you can consider the same format as that of the regular assignment only with a 2^{nd} set of tasks representing periodic tasks. Where a period task is denoted T = {a, c, d, s} where a is arrival, c is computation time, d is deadline which you can assume equals period, and s is context switching time.

Number of tasks: N

Possible speeds: [0.1, 0.3, 0.5, 0.7, 0.9, 1).
Task 1: [a c d s]
Task 2: [a c d s]
….
….
Task N: [a c d s]

Here is an example of the file:

Number of sporadic tasks: 3
Task 1: [0 4 11 1]

Task 2: [3 5 10 1]
Task 3: [6 5 16 2]

Number of periodic tasks: 2

Task 1: [0 2 18 1]
Task 2: [0 2 18 1]

Your report should contain:

1. Listing of your source code.
2. Explicit description of how to run your program, and how to add my testing input file into your project.

Turn-in requirements:

Turn in your report, codes and your own test files compressed in a zip file on BlackBoard.

Reference:

[1] I. Hong, M. Potkonjak, and M. B. Srivastava, "On-line scheduling of hard real-time tasks on variable voltage processor," Proc. Computer-Aided Design (ICCAD), pages 653–656, November 1998.