Development of Real-Time Systems

July 10, 2016

Assignment 4

In this assignment we are going to use our previously learned skills in FreeRTOS and SimSo to schedule non-periodic jobs. First we will start off by setting up a set of periodic tasks in SimSo and then extend the schedule with a non-periodic job. We will compare difference schedulers here and argue for which one is better for different types of tasks. Then we will use FreeRTOS to implement non-periodic jobs in practice. With the previously learned skills in measuring time, we will measure the response time of non-periodic jobs and argue for or against a given schedule.

1 Simulation assignment

Consider the tasks $T_1(3,\ 0.5)$, $T_2(4,\ 1.5,\ 3)$, $T_3(7,\ 1.0,\ 5)$ and the EDF scheduler. A sporadic job arrives at $t = 50$ having the execution time of 10 and a relative deadline of 30. Create the sporadic task in SimSo by selecting: generate task set and then list of act. Dates to the release time.

Use SimSo to schedule the task set and provide a report answering the following questions:

- **What is the minimum/maximum/average response time of all tasks?**

  This data is shown on the task tab, which is presented on figure 1.

Task	min	avg	max	std dev
Task 1	0.500	0.676	1.500	0.294
Task 2	1.500	1.700	2.000	0.245
Task 3	1.000	1.967	3.500	0.921
Task 4	29.000	29.000	29.000	0.000

Figure 1: Task tab of results window in the first simulation.

- **Is any task missing the deadline? Which task? Where?**

  No, all tasks meets the deadline as can be seen in the simulation log, which is presented on figure 2.

48000000	48.0	Task 1_17 Executing on CPU 1
48500000	48.5	Task 1_17 Terminated.
48500000	48.5	Task 2_13 Executing on CPU 1
49000000	49.0	Task 3_8 Activated.
49000000	49.0	Task 2_13 Preempted! ret: 1000000
49000000	49.0	Task 2_13 Executing on CPU 1
50000000	50.0	Task 4_1 Activated.
50000000	50.0	Task 2_13 Terminated.
50000000	50.0	Task 3_8 Executing on CPU 1
51000000	51.0	Task 1_18 Activated.
51000000	51.0	Task 3_8 Terminated.
51000000	51.0	Task 1_18 Executing on CPU 1
51500000	51.5	Task 1_18 Terminated.
51500000	51.5	Task 4_1 Executing on CPU 1
52000000	52.0	Task 2_14 Activated.
52000000	52.0	Task 4_1 Preempted! ret: 9500000
52000000	52.0	Task 2_14 Executing on CPU 1
53500000	53.5	Task 2_14 Terminated.
53500000	53.5	Task 4_1 Executing on CPU 1
54000000	54.0	Task 1_19 Activated.
54000000	54.0	Task 4_1 Preempted! ret: 9000000
54000000	54.0	Task 1_19 Executing on CPU 1
54500000	54.5	Task 1_19 Terminated.
54500000	54.5	Task 4_1 Executing on CPU 1
56000000	56.0	Task 3_9 Activated.
56000000	56.0	Task 2_15 Activated.

Figure 2: Log tab of results window in the first simulation.

- **Is the sporadic job meeting its deadline?**

  Yes, it meets its deadline. In figure 3 it is shown how the sporadic job finish its execution at $t = 79$ before the deadline which is at $t = 80$.

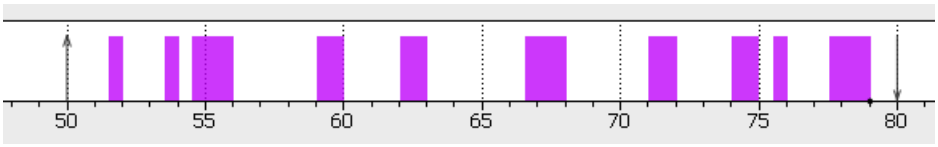

Figure 3: Sporadic task Gantt diagram.

- **What is the response time for the sporadic job?**

  Like it was shown on figure 1, it has a response time of 29.

Consider the tasks $T_1(3,\ 0.5)$, $T_2(4,\ 1.5,\ 3)$, $T_3(7,\ 1.0,\ 5)$ and the RM scheduler. A sporadic job arrives at $t = 50$ having the execution time of 10 and a relative deadline of 30. Create the sporadic task in SimSo by selecting: generate task set and then list of act. Dates to the release time.

Use SimSo to schedule the task set and provide a report answering the following questions:

- **What is the minimum/maximum/average response time of all tasks?**

  This data is shown on the task tab, which is presented on figure 4.

Task	min	avg	max	std dev
Task 1	0.500	0.500	0.500	0.000
Task 2	1.500	1.840	2.000	0.233
Task 3	1.000	1.900	3.000	0.860
Task 4				

Figure 4: Task tab of results window in the second simulation.

- **Is any task missing the deadline? Which task? Where?**

  Yes, Task 4, the sporadic task, miss its deadline. In figure 5, it can be seen that Task 4 aborts at time 80 without finishing its execution.

75000000	75.0	Task 1_26 Executing on CPU 1
75500000	75.5	Task 1_26 Terminated.
75500000	75.5	Task 4_1 Executing on CPU 1
76000000	76.0	Task 2_20 Activated.
76000000	76.0	Task 4_1 Preempted! ret: 1500000
76000000	76.0	Task 2_20 Executing on CPU 1
77000000	77.0	Task 3_12 Activated.
77000000	77.0	Task 2_20 Preempted! ret: 500000
77000000	77.0	Task 2_20 Executing on CPU 1
77500000	77.5	Task 2_20 Terminated.
77500000	77.5	Task 3_12 Executing on CPU 1
78000000	78.0	Task 1_27 Activated.
78000000	78.0	Task 3_12 Preempted! ret: 500000
78000000	78.0	Task 1_27 Executing on CPU 1
78500000	78.5	Task 1_27 Terminated.
78500000	78.5	Task 3_12 Executing on CPU 1
79000000	79.0	Task 3_12 Terminated.
79000000	79.0	Task 4_1 Executing on CPU 1
80000000	80.0	Job Task 4_1 aborted! ret:0.5
80000000	80.0	Task 2_21 Activated.
80000000	80.0	Task 2_21 Executing on CPU 1
81000000	81.0	Task 1_28 Activated.
81000000	81.0	Task 2_21 Preempted! ret: 500000
81000000	81.0	Task 1_28 Executing on CPU 1
81500000	81.5	Task 1_28 Terminated.
81500000	81.5	Task 2_21 Executing on CPU 1
82000000	82.0	Task 2_21 Terminated.

Figure 5: Log tab of results window in the second simulation.

- **Is the sporadic job meeting its deadline?**

  No, it fails and aborts.

- **What is the response time for the sporadic job?**

  This can't be calculated since the sporadic job aborts its execution.

- **Which scheduler is better is better in this example; EDF or RM?**

  The best scheduler is EDF because it can provide feasibility to the system.

2 Programming assignment

The following questions should be solved with programming and the questions should be answered in a report.

- **Is the system fast enough to handle all aperiodic tasks? Why?**

  No, aperiodic tasks tend to be blocked due to their low priority.

- **If not, solve this problem without alter the functionality of any task**

  The solution proposed is to elevate the priority from 2 to 4.

- **What is the response time of the aperiodic task?**

  The response time of the aperiodic task is about 2.08 s, like is shown on the program output on figure 6.

- **Provide a screenshot of the running system**

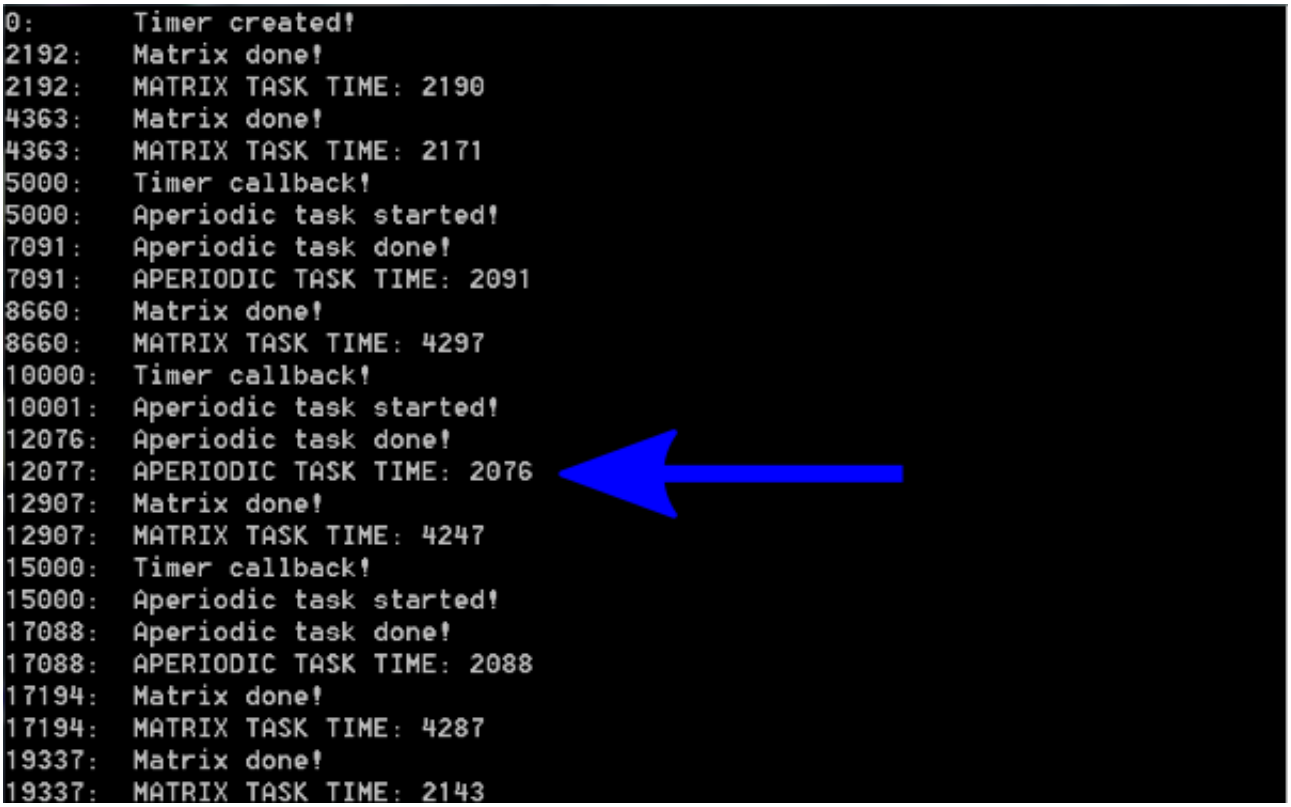

Figure 6: Screenshot of the running system.