# Lecture

Topics:

- Further scheduling theory: Dependent processes, multi-processing
- Discussion of the TIMES tool

# Readings

Sections 11.8-11.14 in [BW], but with special focus on sections 11.8, 11.9, 11.10.1, 11.11.1 (covered in Week 6), 11.13, and 11.14.

Examples of multiprocessor anomalies may be found in [Buzzato], Chapter 2.

For documentation of the TIMES tool, see the tool homepage: `www.timestool.com`. For the current lecture, see especially the start of the paper:

> *TIMES: a Tool for Schedulability Analysis and Code Generation of Real-Time Systems*
> `www.it.uu.se/research/group/darts/times/papers/afmpw-formats03.pdf`

This article may also be found on DTU Learn.

# Supplementary Readings

If you are further interested in multiprocesser scheduling, you may consult the survey paper *A Survey of Hard Real-Time Scheduling for Multiprocessor Systems* by Davids and Burns.

For an elaborate description of attempting to use UPPAAL with stopwatches to model scheduling, see:

> *Model-Based Framework for Schedulability Analysis Using UPPAAL 4.1*
> `www.cs.aau.dk/~adavid/publications/40-bookchap.pdf`

[Chapter 4 of Nicolescu and Mosterman (eds.), *Model-Based Design for Embedded Systems*, CRC Press 2010.]

You may also find these articles on DTU Learn.

# Exercises

*You are expected to do these exercises as part of the preparations for the exam, but you may prefer to work on the Mandatory Assignment at the labs and do the exercises later.*

Exercise 2 and 3 use the TIMES tool which may be downloaded from the tool homepage mentioned above.

**Be sure to turn off the "Use 2 clock scheduler ... " optimization in the verifier.** This option does not seem to work properly.

## Exercise 1: Polling

Consider the problem of Exercise 11.13 (and also 11.14) in [BW]: An external event does not release a task by itself, but has to be "detected" by a periodic monitoring task through *polling*. It is understood that the monitoring task tests for the event at the very start of it *execution*. If the event appears later than this, it will not be detected until the next execution of the task.

Assume the deadline for the external event $e$ is $D_e$ and that is takes $C_e$ time to compute the response, determine the scheduling parameters of the periodic process which should monitor $e$.

You may optionally try to apply the result to the event set of Exercise 11.14 and verify that the set of monitoring tasks are schedulable.

## Exercise 2: Get acquianted with the Times tool

[In these exercises, we do not use the facilities provided by the "Project", "Task" or "Precedence" tabs in the TIMES tool editor.]

Start up the tool, declare a few periodic tasks and run them using the simulator. Try to vary timing parameters and scheduling policy (including preemption mode) to obtain both schedulable and non-schedulable situations.

You may like to start from the example presented at the lecture to be found at CampusNet as `Example.xml`.

## Exercise 3: Controlled Task

A task $t$ is to be released aperiodically in (small) bursts according to the following constraints:

- Two releases of $t$ are separated by at least 5 time units.
- Within any period less than 25 time units there are at most two releases.

**Question 3.1:** Model the releases of task $t$ by a timed automaton (template) in the TIMES tool. You may use the deterministic `Control1.xml` as a starting point, but the releases should be as non-deterministic as possible.

Hint: Use two clocks and (after a startup phase) keep track of which one was last used.

**Question 3.2:** Assuming the burst task to have $(C, D) = (1, 5)$, try to see if it can be accommodated within the following task set (using fixed priorities):

| Task | T | C |
|---|---|---|
| $a$ | 25 | 3 |
| $b$ | 9 | 2 |
| $c$ | 40 | 4 |
| $d$ | 13 | 5 |

It may be the case that you cannot get the verification through without removing one or more low-priority tasks.

To see if the burst task can run together with the full task set, you may then instead use a deterministic worst case burst task obtained by setting the parameters of `Control1.xml` properly.

## From Exam F13:

## PROBLEM 4 (approx. 40 %)

*The questions in this problem can be solved independently of each other.*

A real-time system to be run on a single-processor computer has three periodic tasks `a`, `b` and `c`, with the following parameters:

| | $T$ | $C$ |
|---|---|---|
| a | 5 | 2 |
| b | 10 | 3 |
| c | 25 | 4 |

The deadline of each task is equal to its period. Initially, the tasks are considered to be independent.

**Question 4.1:**

(a) Calculate the load of each task and the total load of the system.

(b) Based on the total load, which scheduling principles can be concluded to be feasible for scheduling the given set of tasks?

In the given system, the set of task is to be scheduled by a preemptive fixed-priority scheduler (FPS) using rate monotonic priority assignment.

**Question 4.2:**

Calculate the response time of each task.

Now, a shared protected resource $M$ is to be used by all three tasks. Each task is always going to use the resource $M$ for exactly 1 of its computation time units $C$. Furthermore, the usage of $M$ will take place at the end of the computation for each of the three tasks.

Recall that the *blocking time* $B_t$ of a task $t$ is the maximum time the task $t$ may experience being blocked waiting for *lower priority tasks* to release resources (not including any preemption by tasks with higher priority than task $t$).

**Question 4.3:**

(a) Explain why the blocking times of tasks b and c are 1 and 0 respectively.

(b) Determine the (worst-case) blocking time $B_{\mathsf{a}}$ of task a and illustrate using a scheduling scenario how this amount of blocking may occur.

[Notice that blocking of task a may be subject to *priority inversion*.]

(c) Taking the blocking times into account, use response time analysis to determine whether the task set is schedulable when $M$ is used.

**Question 4.4:**

(a) Assume that the principle of the *immediate ceiling priority protocol* (ICPP) is applied to the use of $M$.

State the blocking time of each task under this principle and determine the response time of each task.

(b) Discuss briefly how the response time of the lowest priority task in general is affected by application of ICPP.