Practical session on scheduling: Cheddar and Storm

Master EMS

September 2010

1 Case study

We consider the simplified Flight Control System of Fig. 1. This system controls the attitude, the trajectory and the speed of an airplane. It consists of 7 tasks which execute repeatedly at a periodic rate. The fastest sub-system executes at 10ms, it acquires the state of the system (angles, position, acceleration) and computes the feedback law of the system. The order is then sent to the flight control surfaces. The intermediate sub-system is the piloting loop, it executes at 40ms and determines the acceleration to apply.. The slowest sub-system is the navigation loop, it executes at 120ms and determines the position to reach. The required position of the airplane is acquired at the slow rate.

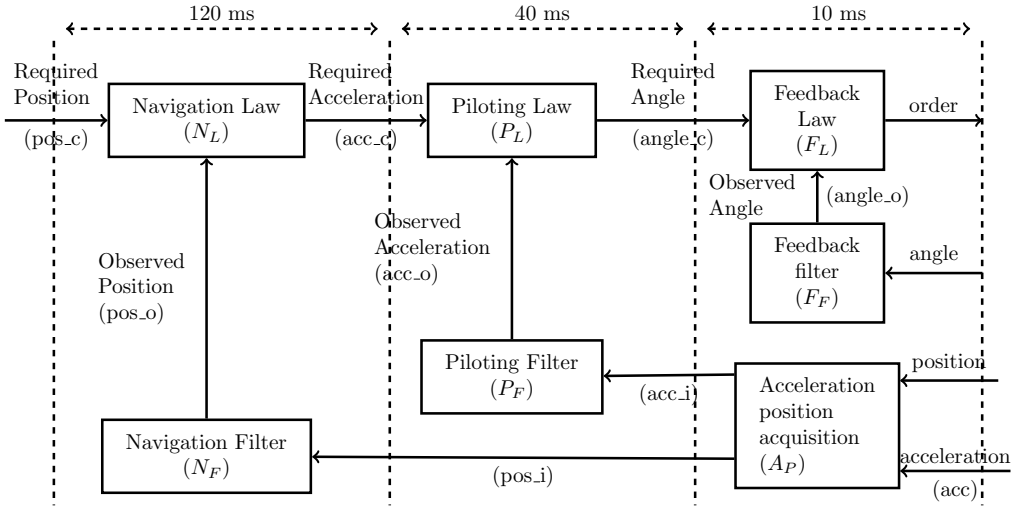


Figure 1: Flight control system

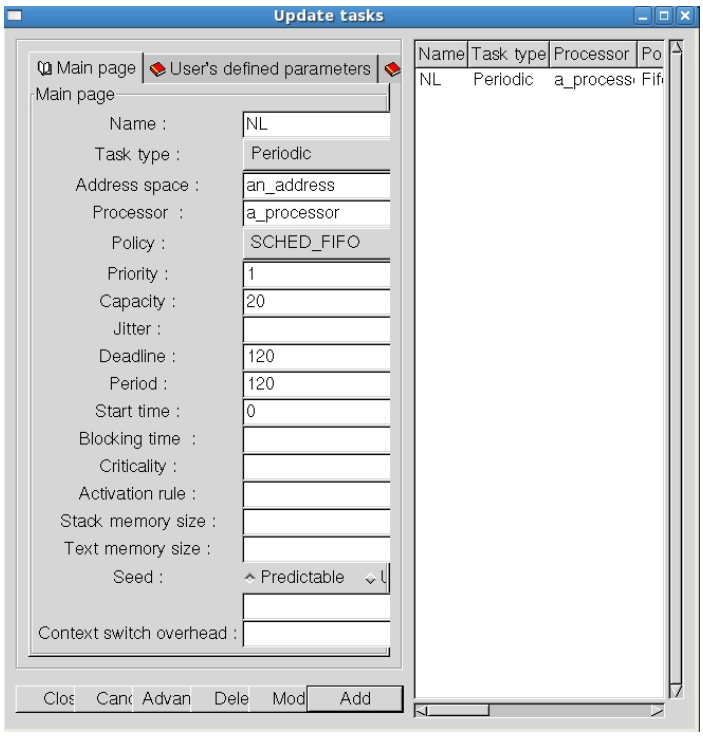
2 Cheddar

CHEDDAR is a free real-time scheduling tool which checks temporal properties on real-time systems. The tool provides a simulation engine and feasibility tests. All documents and binaries can be found in the url:
`http://beru.univ-brest.fr/~singhoff/cheddar/`
ant the tutorial is:
`http://beru.univ-brest.fr/~singhoff/cheddar/ug/cheddar-r2.html`

2.1 Monoprocessor scheduling analysis

Launch CHEDDAR: write cheddar in your terminal. The graphical interface appears. Follow the tutorial recommendation for programming the case study.

1. *add* a processor in the *Edit/Update processors*. For this, give a name, a scheduler policy and precise if it is preemptive or not.
2. *add* a memory address in the *Edit/Update address spaces*. Simply give a name.
3. *add* all the tasks in the *Edit/Update tasks*.



The real-time features of the set of tasks are given below:

Task	Period	WCET	release date	deadline
NL	120	20	0	120
NF	120	10	0	120
PL	40	5	0	40
PF	40	5	0	40
FL	10	2	0	10
FF	10	1	0	10
AP	10	1	0	10

Make a first scheduling analysis with RM and a second using EDF. For each one, try the simulator and the feasibility tests. Click on *scheduling simulation*. The result appears: a Gantt diagram illustrates an worst case execution. Click on *scheduling feasibility*. The tool check if the scheduling never misses any deadline. It computes the processor utilization factor and the response time for each task.

3 STORM

STORM is a Simulation TOol for Real time Multiprocessor scheduling. From a specification of a task set, a scheduling policy and a platform description (these data are specified in an .xml file), the tool simulates a possible execution of the tasks on the multiprocessor platform. All documents and sources are available on the STORM web page: `http://storm.rts-software.org/doku.php`
In the documentation tab, you can find several supports. In particular, the user manual: `http://storm.rts-software.org/download/User-guide-V3.2.pdf`
the designer manual, devoted to the scheduler specification:
`http://storm.rts-software.org/download/Designer-guideV3.3_version1.pdf`
and some examples: `http://storm.rts-software.org/download/Examples.zip`

3.1 Simulation of a pre-defined multiprocessor scheduling

As a first step, we manipulate the tool using pre defined scheduling policies in STORM.

For this, download the executable `storm-3-2.jar` in the download tab. Download also, in the same folder as the executable, the examples provided in the documentation tab. Launch the tool by double clicking on the .jar. The graphical interface opens and we will simulate a downloaded example . Open the example `Test-EDFP1.xml` in your favorite editor and tape in the STORM terminal: `exec Examples/Test-EDFP1.xml` and then `pa` The tool will launch the simulation of this example.

The detailed description of the .xml is given in the user manual.

1. You must give the interval for the simulation: `<SIMULATION duration="50">`. Modify the file `Test-EDFP1.xml` to try a simulation of length 120 et relancez la simulation;

2. you must specify the number of CPUs. Add a second CPU
`<CPU className="storm.Processors.CT11MPCore" name ="CPU B" id="2"></CPU>`
and relaunch the simulation;

3. you must describe the information on the tasks. Add a new task T_4 such that $T(T_4) = 10$, $C(T_4) = 2$, $r(T_4) = 0$ and $D(T_4) = 10$. Then relaunch the simulation;

4. you must give the scheduling policy
`<SCHED className="storm.Schedulers.EDF_P_Scheduler"> </SCHED>` Change the policy by using an other predefined one. Look on the user guide list, in the appendix to find the list of them;

5. other task models are also available such as `<TASK className="storm.Tasks.PTask_NAM"`.

Code the flight control case study, with the following real-time attributes, in STORM, for 2 processors and apply several simulations.

Task	Period	WCET	release date	deadline
NL	120	40	0	120
NF	120	20	0	120
PL	40	10	0	40
PF	40	10	0	40
FL	10	4	0	10
FF	10	2	0	10
AP	10	2	0	10

3.2 Programming a new scheduler

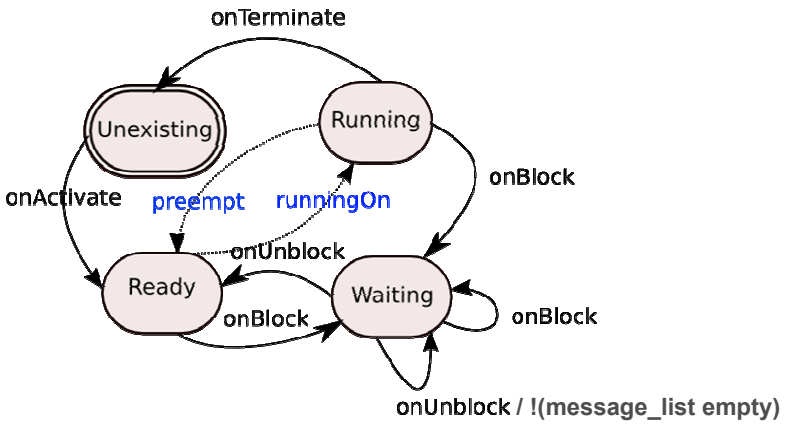


Figure 2: Task model of STORM

Open the designer guide p 12 and copy the code in a file `Mon_EDF_P_Scheduler.java`. *_P* stands for preemptive while *_NP* stands for non preemptive. The task model and

the interactions with the scheduler is described in the Fig. 2. In black, you can see the task actions and in blue the scheduler actions.

Compile the new scheduler `javac -cp ./storm-3-2.jar NewScheduler.java` Save the .class in the folder *storm/Schedulers*. And add the scheduler in the executable: `jar uf storm-3-2.jar storm/Schedulers/*` Storm is launched by using the command: `java -cp ./storm-3-2.jar programme.programme` To test the new scheduler, follow the instruction of the last page of the designer guide.

Exercise 1 *Program the following scheduler:*

1. *modify* `EDF_P` *in order to provide* `GRM_P`*;*
2. *write a partitioned scheduler* `PRM_P`*. In the task description, add the field* `<TASK className="storm.Tasks.PTask_NAM" name ="T1" alloc="1"` *which specifies the number of the assigned processor. To retrieve the value use* `getOwnFieldIntValue("alloc")`*.*
3. *implements the heuristic FFDU instead of using the field alloc.*