Implementing CBS support for NanoRK

UBC | CICS 525

In this assignment you will continue to extend NanoRK with support for the Constant Bandwidth Server (CBS) for guaranteeing utilization to tasks and to isolate tasks from overruns of other tasks. You will use your modified version of NanoRK that supports EDF for this assignment. (You do not, however, need to combine this with SRP.)

1 Tasks

T1: Implement constant bandwidth servers. The default Nano-RK system supports periodic tasks and provides a CPU reserve for each task. When a task exceeds its reserve, an error is registered. An alternative model that is useful in multiple cases (e.g., soft realtime systems where each task merely needs a guaranteed fraction of processing time and when dealing with aperiodic tasks) is the constant bandwidth server that works with earliest deadline first scheduling. Read the chapter on *Dynamic Priority Servers* in the book *Hard Real-Time Computing Systems* by Buttazzo. In essence, CBS allows tasks that exceed their nominal CPU reserve to continue execution - or stay in the ready queue - by postponing the deadline of the task. Nano-RK provides methods for creating and managing periodic tasks. A sample fragment of code for adding a task looks like this:

```
nrk_task_set_entry_function( &TaskOne, Task1);
nrk_task_set_stk( &TaskOne, Stack1, NRK_APP_STACKSIZE );
TaskOne.prio = 1;
TaskOne.FirstActivation = TRUE;
TaskOne.Type = BASIC_TASK;
TaskOne.SchType = PREEMPTIVE;
TaskOne.period.secs = 0;
TaskOne.period.nano_secs = 250*NANOS_PER_MS;
TaskOne.cpu_reserve.secs = 0;
TaskOne.cpu_reserve.nano_secs = 50*NANOS_PER_MS;
TaskOne.offset.secs = 0;
TaskOne.offset.nano_secs= 0;
nrk_activate_task (&TaskOne);
```

For this assignment, add a new type of task that would be managed by a constant bandwidth server. Creating a task that is managed by a CBS would be similar to creating a regular periodic task and would like this:

```
nrk_cbs_set_entry_function( &CBS1, Task1 );
nrk_cbs_set_stk( &CBS1, Stack1, NRK_APP_STACKSIZE );
CBS1.prio = 1;
CBS1.FirstActivation = TRUE;
CBS1.Type = CBS_TASK;
CBS1.SchType = PREEMPTIVE;
CBS1.period.secs = 0;
CBS1.CBS1.period.nano_secs = 250*NANOS_PER_MS;
CBS1.cpu_reserve.secs = 0;
CBS1.cpu_reserve.nano_secs = 50*NANOS_PER_MS;
CBS1.offset.secs = 0;
CBS1.offset.nano_secs= 0;
nrk_activate_cbs ( &CBS1 );
```

Add the primitives needed to manage constant bandwidth servers at the programming interface and within the kernel/scheduler. A software developer should be able to build an application that uses a combination of regular periodic tasks and constant bandwidth servers. Tasks are scheduled using EDF therefore the priority field is used only to break ties when two jobs have the same absolute deadline. You may assume that a job of a task is not released until the previous job of the same task is complete.

The goal, in this assignment, is to use CBS only to isolate tasks and guarantee CPU reservations in terms of utilization. CBS can be used to schedule aperiodic tasks but you need not focus on aperiodic tasks for this assignment.

2 Documentation and submission

Write a clear report for Task 1. Describe your implementation for the task. The report should be submitted as a PDF file and should include the names of the group members. Notice that we do not provide any tests for correctness. In engineering practice, you should consider testing mechanisms in conjunction with the design. You should *describe clearly* the methods you used to verify the correctness of your implementation.

Use `handin` to submit your work. For your implementation effort, you will (and should) only submit the files from the NanoRK source code that were changed. Invoke `handin` from `remote.mss.icics.ubc.ca` as follows after making sure that all the necessary files are in a directory named `nrk-cbs`: `handin cics525 nrk-cbs`