The `rtsched` package for LaTeX (version 2.0)

Giuseppe Lipari

June 21, 2016

List of Figures

1 Introduction

In this document, I give an overview of the `rtsched` LaTeXpackage, which can be used to easily draw chronograms (GANTT charts). These diagrams are quite common in real-time scheduling research.

The package depends on keyval and TikZ/PGF (pgf version 2.10 or greater), both widely available on any TeXdistribution.

The drawing capabilities are completely based on TikZ. Thus, you can compile a document that uses `rtsched` package with modern tools producing pdf document as `pdfLaTeX`, `XeLaTeX` or `LuaLaTeX`.

As said, the style works also with Beamer, and it is also possible to use animations.

You can find more examples of usage of this style in my lectures, which can be downloaded at the following address: `http://retis.sssup.it/~lipari/courses/`.

I prefer to demonstrate the capabilities of the package by a set of examples. You can just cut and paste the examples and play with them as you wish.

2 Basic commands

2.1 Simple example with two tasks

In Figure 1 I show a simple example of the Rate Monotonic schedule of two simple tasks, followed by the code that generated it. To draw the grid, with the numbers, you have to use the `RTGrid` environment:

```
\begin{RTGrid}[options]{n}{t}
...
\end{RTGrid}
```

where `n` is the number of horizontal axis (one per task, in this case), and `t` is the length of the axis in time units. This also draws the task labels on the left, and the numbering on the bottom.

Every job arrival is depicted with an upward arrow; a deadline is depicted by a downward arrow (not very visible here, since it concides with the next arrival, see Figure 4 for deadlines different from periods). The task execution is depicted by a gray box.

The arrival of a job and the corresponding deadline can be obtained by using the following commands:

```
\TaskArrival{i}{t}
\TaskArrDeadl{i}{t}{reld}
```

where `i` is the task index (from 1 to `n` included), `t` is the arrival time, and `reld` is the relative deadline; an absolute deadline will be drawn at `t + reld`. If you only want to draw absolute deadlines, you can simply use the following command:

```
\TaskDeadline{i}{t}
```

that works in the same way as `\TaskArrival{i}{t}`.

In this example there are a lot of repetitions. These can be avoided if you use the periodic versions of some commands, as shown in the example of Figure 2. Available periodic versions of the commands can be found in Table 1. The periodic versions take two additional arguments corresponding to the period and to the number of instances desired.

To draw the execution rectangle, you can use the following command:

```
\TaskExecution{i}{t1}{t2}
\TaskExecDelta{i}{t}{delta}
```

The first one is used to draw an execution rectangle of height 1-unit for the `i`-th task from `t1` to `t2`. The second command draws a rectangle from `t` to `t+delta`.

Command	Periodic version
`\TaskArrival{i}{t}`	`\TaskNArrival{i}{t}{p}{n}`
`\TaskDeadline{i}{t}`	`\TaskNDeadline{i}{t}{p}{n}`
`\TaskArrDeadl{i}{t}{reld}`	`\TaskNArrDeadl{i}{t}{reld}{p}{n}`
`\TaskExecDelta{i}{t}{delta}`	`\TaskNExecDelta{i}{t}{delta}{p}{n}`
`\TaskEnd{i}{t}`	`\TaskNEnd{i}{t}{p}{n}`

Table 1: Table of periodic commands where p stands for the period and n for the number of instances

In Figure 3, you can see how to only draw arrival upward arrows, and how to specify offsets. Finally, in Figure 4 you can see an example with 2 tasks with relative deadlines different from periods (the so-called *constrained deadline tasks*).

It is also possible to visualise preempted tasks with a hatched fill style. An example is in Figure 5 that uses command `TaskRespTime`.

It is sometimes useful to represent the end of a job execution, especially to distinguish it from preemption. In that case, you can use the following command :

```
\TaskEnd{i}{t}
```

and its periodic version detailed in Table 1 that draw little circle(s) at the specified date(s). It works in the same way as `\TaskArrival` command as shown in Figure 2. Alternatively, you can also add the `end=1` key to the `TaskExecution` command as follows:

```
\TaskExecution[end=1]{1}{10}{12}
\TaskExecDelta[end=1]{1}{10}{2}.
```

2.2 Controlling visualization

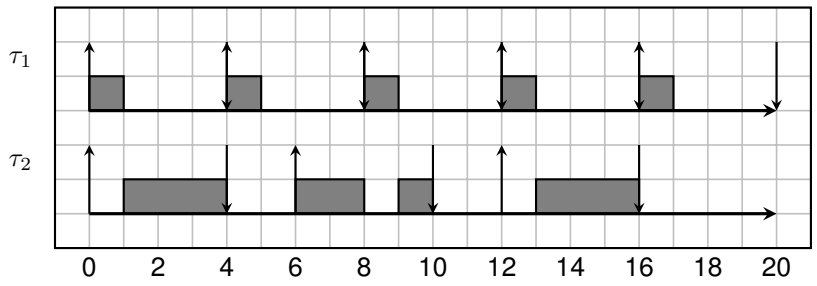
It is possible to specify many options in the `RTGrid` environment. Maybe you don't like the grid: then, you can decide to not visualise it as in Figure 6, where we also removed the task symbols.

The next figure 7 uses different task symbols, does not show the numbers on the time line, and the color of the boxes that denote the execution of the second instance of the second task are changed to red. Also, I am changing the width, so the figure looks smaller. Notice that you can directly specify colors using the TikZ way (color!percentage for example).

Do you want to specify an arbitrary symbol at a certain row? No problem! See the example in Figure 8. Here we use the command:

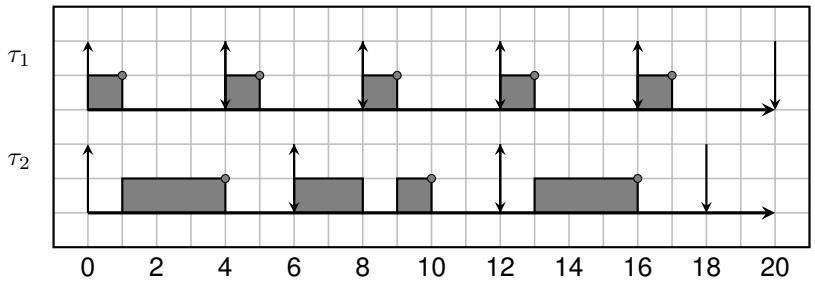
```
\RowLabel{i}{label}
```

which writes the `label` at the specified row (index 1 stays at the top). Here we show also how to specify an arbitrary starting number in the time line, using the `numoffset=12` option.



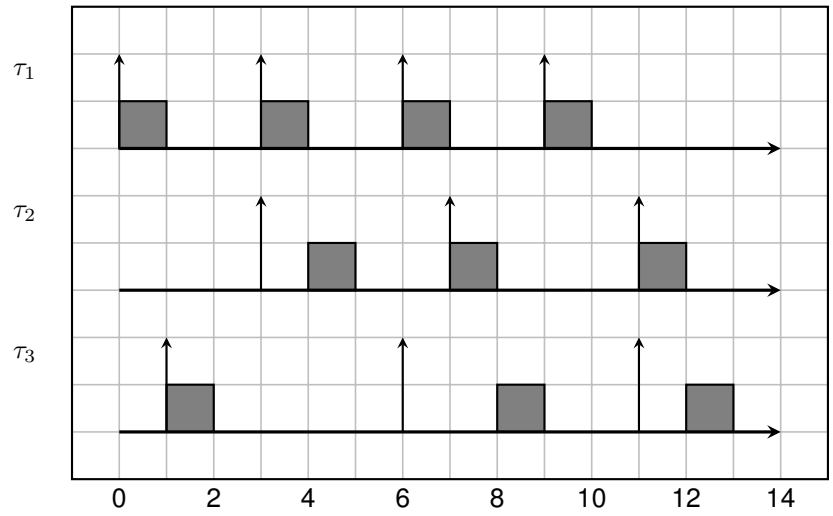

```
% 2 tasks, for 20 units of time
% we specify the width (10cm is the default
% value, so we will stop specifying it from now on)
\begin{RTGrid}[width=10cm]{2}{20}
  %% the first job of task 1 arrives at time 0,
  %% with a relative deadline of 4
  \TaskArrDead{1}{0}{4}
  %% the second job arrives at time 4
  \TaskArrDead{1}{4}{4}
  %% etc
  \TaskArrDead{1}{8}{4}
  \TaskArrDead{1}{12}{4}
  \TaskArrDead{1}{16}{4}

  %% the task executes in intervals [0,1], [4,5], etc.
  \TaskExecution{1}{0}{1}
  \TaskExecution{1}{4}{5}
  \TaskExecution{1}{8}{9}
  \TaskExecution{1}{12}{13}
  \TaskExecution{1}{16}{17}

  %% the second task
  \TaskArrDead{2}{0}{4}
  \TaskArrDead{2}{6}{4}
  \TaskArrDead{2}{12}{4}
  \TaskExecution{2}{1}{4}
  \TaskExecution{2}{6}{8}
  \TaskExecution{2}{9}{10}
  \TaskExecution{2}{13}{16}
\end{RTGrid}
```

Figure 1: Two tasks, with deadline equal to period, RM scheduling



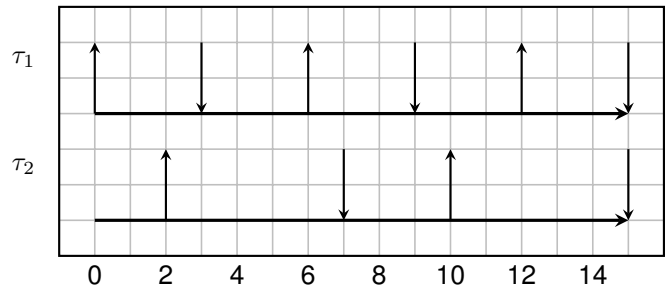

```
\begin{RTGrid}{2}{20}

  \TaskNArrDead{1}{0}{4}{4}{5}    % draws the arrivals and deadlines
  \TaskNExecDelta{1}{0}{1}{4}{5}  % draws executions (highest priority)
                                  % for 5 instances of period 4

  \TaskNEnd{1}{1}{4}{5}           % draws 5 ends of job execution of period 4

  \TaskNArrDead{2}{0}{6}{6}{3}    % draws the arrival and deadline
                                  % for 3 instances of period 6


  % no simple formula for lowest priority, sorry!
  \TaskExecution{2}{1}{4}
  \TaskExecution{2}{6}{8}
  \TaskExecution{2}{9}{10}
  \TaskExecution{2}{13}{16}
  \TaskEnd{2}{4}
  \TaskEnd{2}{10}
  \TaskEnd{2}{16}
\end{RTGrid}
```

Figure 2: Using periodic commands to avoid repetitions




```
\begin{RTGrid}{3}{14}

  \TaskNArrival{1}{0}{3}{4}        % draws only the arrivals
  \TaskNExecDelta{1}{0}{1}{3}{4}   % draws executions (highest priority)
                                   % for 4 instances of period 3



  \TaskNArrival{2}{3}{4}{3}        % draws only the arrivals
                                   % 3 instances of period 4, starting from 3
  \TaskExecDelta{2}{4}{1}
  \TaskExecDelta{2}{7}{1}
  \TaskExecDelta{2}{11}{1}

                                   % 3 instances of period 5, starting from 1
  \TaskNArrival{3}{1}{5}{3}        % draws only the arrivals

  \TaskExecDelta{3}{1}{1}
  \TaskExecDelta{3}{8}{1}
  \TaskExecDelta{3}{12}{1}
\end{RTGrid}
```

Figure 3: Three tasks with offsets, and only arrivals with no deadlines




```
\begin{RTGrid}[width=8cm]{2}{15}
 \TaskNArrDead{1}{0}{3}{6}{3}
 \TaskNArrDead{2}{2}{5}{8}{2}
\end{RTGrid}
```

Figure 4: Deadlines less than periods

2.3 Highlighting and labeling objects

Sometimes it may be important to say that one task has caused the activation of another task. You can use the following command, as shown in Figure 9:

```
\Activation{i}{t1}{j}{t2}
```

which draws an arrow from the baseline of task `i` at time `t1` to the baseline of task `j` at time `t2`. Also, you can put an arbitrary label inside a shadow box with the following command:

```
\Label{y}{x}{label}
```

which draws a boxed label at position `x,y` in the grid.

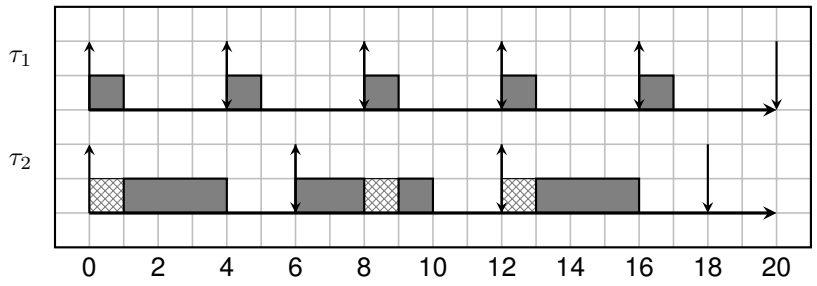
Finally, it is possible to draw a rectangular box with rounded corners to highlight a portion of the schedule with `RTBox`:

```
\RTBox{t1}{t2}
```

Notice that the order with which the objects are drawn is exactly the same as the order in which they are specified in the code, excepted for horizontal axes, arrivals, deadlines and end of job execution that are always drawn on the foreground. For example, in Figure 9, the executions of all the tasks are drawn on top of the box. You can try to move the `RTBox` command at the end to see what happens.

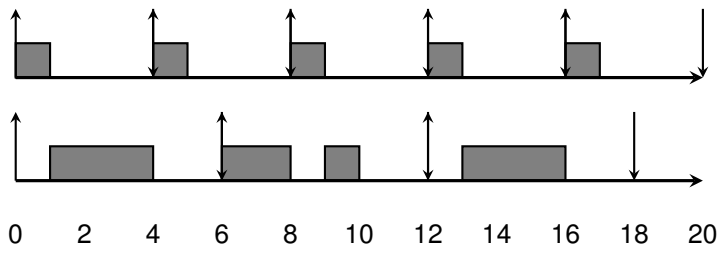
2.4 Priority Inheritance

An example of task locking/unlocking and the use of the Priority Inheritance Protocol is shown in Figure 10. Here, task τ_3 locks resource S at time $t = 2$. This is obtained by using command:




```
\begin{RTGrid}{2}{20}

  \TaskNArrDead{1}{0}{4}{4}{5}    % draws the arrivals and deadlines
  \TaskNExecDelta{1}{0}{1}{4}{5}  % draws executions (highest priority)
                                  % for 5 instances of period 4


  \TaskNArrDead{2}{0}{6}{6}{3}   % draws the arrivals and deadlines
                                 % for 3 instances of period 6

  \TaskRespTime{2}{0}{4}         % draws the hatched rectangle in [0,4]
  \TaskExecution{2}{1}{4}        % draws execution (over the previous rectangle)
  \TaskRespTime{2}{6}{4}         % draws the hatched rectangle in [6,10]
  \TaskExecution{2}{6}{8}        % draws execution
  \TaskExecution{2}{9}{10}       % draws execution
  \TaskRespTime{2}{12}{4}        % draws the hatched rectangle in [12,16]
  \TaskExecution{2}{13}{16}      % draws execution
\end{RTGrid}
```

Figure 5: Example with TaskRespTime




```
%% no grid and no symbols
\begin{RTGrid}[nogrid=1,nosymbols=1]{2}{20}
  \TaskNArrDead{1}{0}{4}{4}{5}
  \TaskNExecDelta{1}{0}{1}{4}{5}
  \TaskNArrDead{2}{0}{6}{6}{3}
  \TaskExecution{2}{1}{4}
  \TaskExecution{2}{6}{8}
  \TaskExecution{2}{9}{10}
  \TaskExecution{2}{13}{16}
\end{RTGrid}
```

Figure 6: Removing visualization of the grid and of the task names

```
\TaskLock{3}{2}{S}
```

Unlock is similarly obtained by using command:

```
\TaskUnlock{3}{7}{S}
```

Task τ_1 tries to lock the same resource at time $t = 5$. The priority of τ_1 is then inherited by τ_3: the inheritance rule is depicted by using a dashed tick arrow from the baseline of τ_1 to τ_3, using command:

```
\Inherit{1}{3}{4}
```

The fact that τ_3 is executing inside a critical section is denoted by putting a label inside the execution block, using the following command:

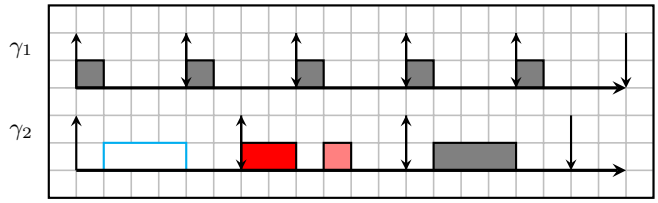
```
\TaskExecution[color=white,execlabel=S]{3}{4}{5}
```

2.5 Jitter

Jitter is often represented as an interval drawn by an horizontal double-headed arrow. As shown in the Figure 12, you can define jitter or any other interval with the following command :

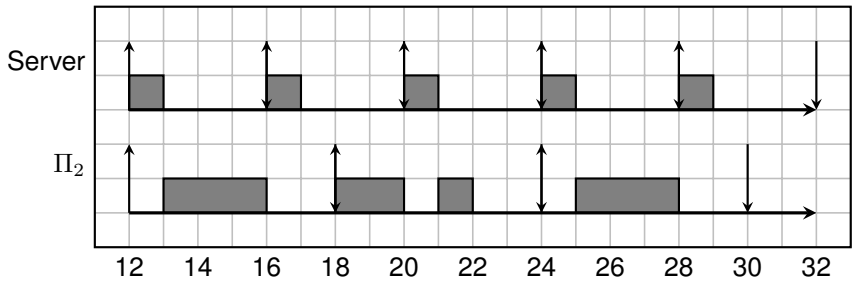
```
\TaskInterval{i}{t1}{t2}{label}
```

This command draws an horizontal double-headed arrow for the `i`-th task from `t1` to `t2` with the specified `label` in the middle.



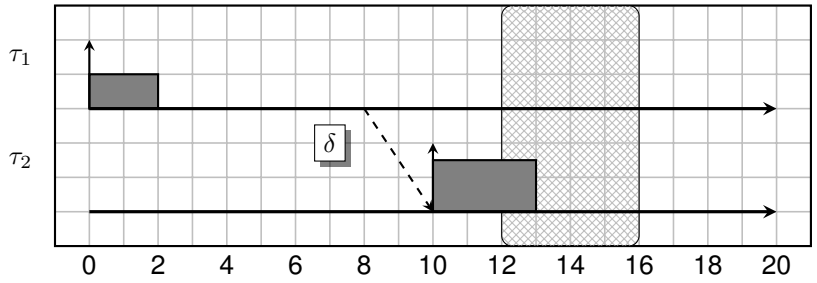

```
%% specify 1) no numbers on the time line, 2) a different symbol, 3)
%% a different size of the symbol (default is \normalsize).
%% Notice that you should not use the math mode in the
%% specification of the symbol, as the symbol is already used in a
%% math environment inside the macro
\begin{RTGrid}[symbol=\gamma,nonumbers=1,labelsize=\Large]{2}{20}
  \TaskNArrDead{1}{0}{4}{4}{5}
  \TaskNExecDelta{1}{0}{1}{4}{5}
  \TaskNArrDead{2}{0}{6}{6}{3}
  %% here, the border changes to cyan, and the fill to white
  \TaskExecution[linecolor=cyan,color=white]{2}{1}{4}
  %% the next two boxes are filled with red instead of gray
  \TaskExecution[color=red]{2}{6}{8}
  \TaskExecution[color=red]{2}{9}{10}
  \TaskExecution{2}{13}{16}
\end{RTGrid}
```

Figure 7: Different symbols (with \Large size), no numbers, a different task color




```
%% specify 1) no numbers on the time line, 2) number starting from 12
\begin{RTGrid}[nosymbols=1,numoffset=12]{2}{20}
  %% the symbol for the first row
  \RowLabel{1}{Server}
  %% the symbol for the second row
  \RowLabel{2}{$\Pi_2$}
  \TaskNArrDead{1}{0}{4}{4}{5}
  \TaskNExecDelta{1}{0}{1}{4}{5}
  \TaskNArrDead{2}{0}{6}{6}{3}

  \TaskExecution{2}{1}{4}
  \TaskExecution{2}{6}{8}
  \TaskExecution{2}{9}{10}
  \TaskExecution{2}{13}{16}
\end{RTGrid}
```

Figure 8: Arbitrary symbols with an appropriate offset, no grid, numbering starting from 12




```
\begin{RTGrid}{2}{20}
  \RTBox{12}{16}
  \TaskArrival{1}{0}{6}{4}
  \TaskExecDelta{1}{0}{2}{6}{4}
  \TaskArrival{2}{10}
  \TaskExecDelta[exeheight=1.5]{2}{10}{3}
  \Activation{1}{8}{2}{10}
  \Label{6}{7}{$\delta$}
\end{RTGrid}
```

Figure 9: Activation (from one task to another one), and an arbitrary label

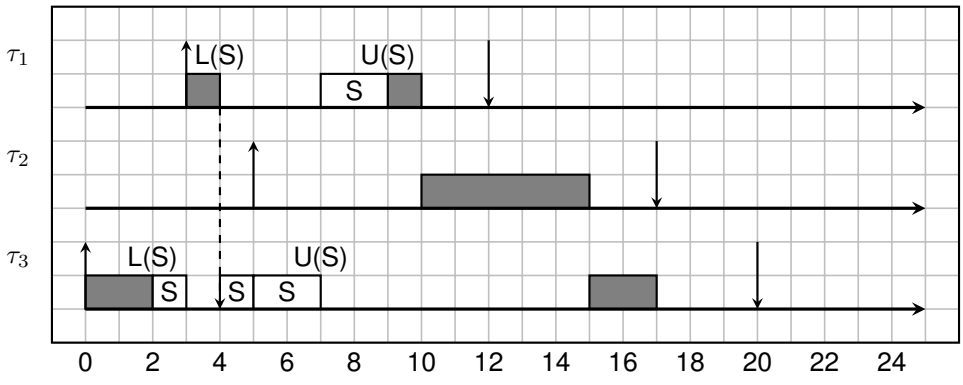
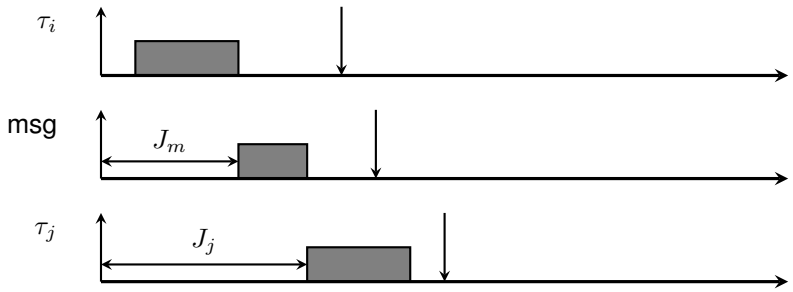


Figure 10: Task blocking on resources: the Priority Inheritance Protocol

```
\begin{RTGrid}[width=12cm]{3}{25}
  \TaskArrDead{3}{0}{20}
  \TaskExecution{3}{0}{2}
  \TaskLock{3}{2}{S}
  \TaskExecution[color=white,execlabel=S]{3}{2}{3}
  \TaskArrDead{1}{3}{9}
  \TaskExecution{1}{3}{4}
  \TaskLock{1}{4}{S}
  \Inherit{1}{3}{4}
  \TaskExecution[color=white,execlabel=S]{3}{4}{5}
  \TaskArrDead{2}{5}{12}
  \TaskExecution[color=white,execlabel=S]{3}{5}{7}
  \TaskUnlock{3}{7}{S}
  \TaskExecution[color=white,execlabel=S]{1}{7}{9}
  \TaskUnlock{1}{9}{S}
  \TaskExecution{1}{9}{10}
  \TaskExecution{2}{10}{15}
  \TaskExecution{3}{15}{17}
  \end{RTGrid}
```

Figure 11: Priority Inheritance example




```
\begin{RTGrid}[nogrid=1, nosymbols=1 ,nonumbers=1]{3}{20}

  \RowLabel{1}{$\tau_i$}
  \TaskArrival{1}{0}
  \TaskExecution{1}{1}{4}
  \TaskDeadline{1}{7}

  \RowLabel{2}{msg}
  \TaskArrival{2}{0}
  \TaskExecution{2}{4}{6}
  \TaskDeadline{2}{8}
  \TaskInterval{2}{0}{4}{$J_m$}  % draws an interval between date 0 and 4 for task 2 with

  \RowLabel{3}{$\tau_j$}
  \TaskArrival{3}{0}
  \TaskExecution{3}{6}{9}
  \TaskDeadline{3}{10}
  \TaskInterval{3}{0}{6}{$J_j$} % draws an interval between date 0 and 6 for task 3 with a
\end{RTGrid}
```

Figure 12: Example with TaskInterval used to represent jitters