The `rtsched` package for LaTeX (version 1.0)

Giuseppe Lipari

September 29, 2011

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, multido and pstricks, all widely available on any TeXdistribution.

The drawing capabilities are completely based on the PSTricks: for this reason, it may not be safe to use `pdfLaTeX` to compile a document that uses the `rtsched` package, due to the fact that at the time of this writing, PSTricks is not well supported by pdfLatex. If you want to produce pdf files (for example for making slides with Beamer), consider using the following compilation chain:

```
latex doc.tex && dvips doc.dvi && ps2pdf doc.ps
```

As said, the style works also with Beamer, and it is also possible to use animations. A separate file (rtsched-beamer-doc.tex) is provided to show the animation capabilities of this tool.

You can find more example 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` if 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 arival time, and `reld` is the relative deadline; an absolute deadline will be drawn at `t + reld`.

In this example there are a lot of repetitions. These can be avoided if you use the `multido` macro, as shown in the example of Figure 2.

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

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

The first one is use 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`.

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 11 that uses command `TaskRespTime`.

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 visualize 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.

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.

2.3 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 arbiteary label inside a shadow box with the following command:

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

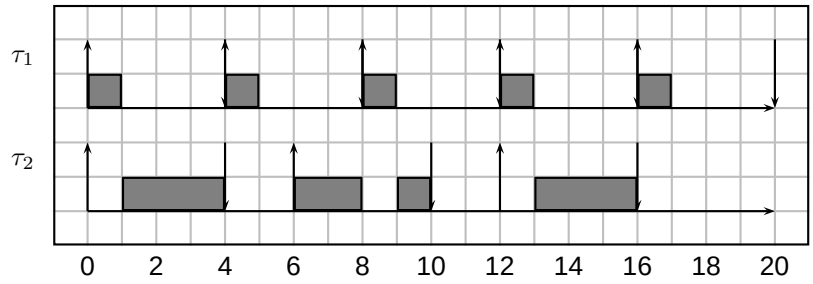
which draws a boxed label at position `x,y` in the grid.

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

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

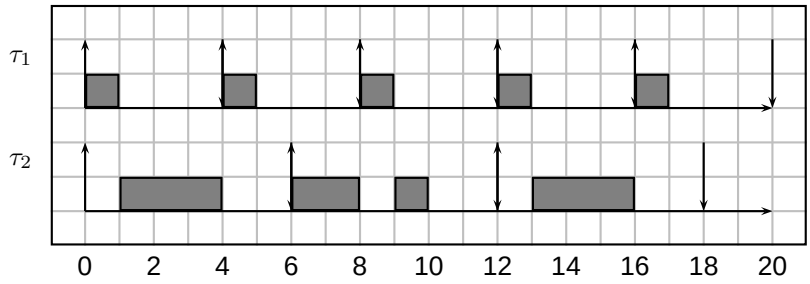
Notice that the order with which the objects are drawn is exactly the same as the order in which they are specified in the code. For example, in Figure 9, the executions of all the task are drawn on top of the box. You can try to move the `RTBox` command at the end to see what happens.

2.4 Priority Inheritance



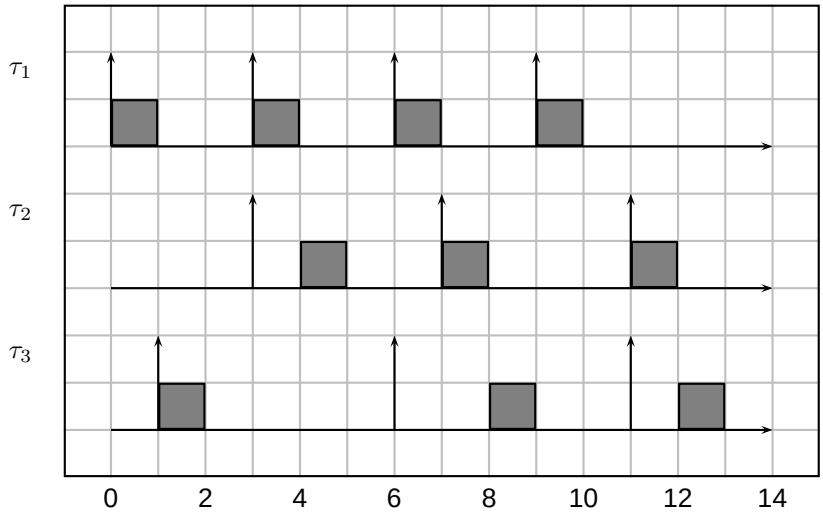

```
% 2 tasks, for 20 units of time
% we specify the width (10cm is the default
% value, so we will stop specifying it from now on)
\begin{RTGrid}[width=10cm]{2}{20}
  %% the first job of task 1 arrives at time 0,
  %% with a relative deadline of 4
  \TaskArrDead{1}{0}{4}
  %% the second job arrives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



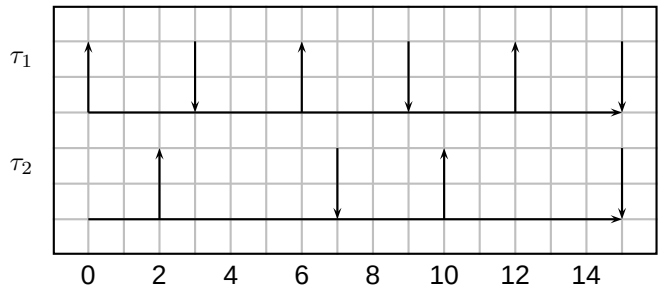

```
\begin{RTGrid}{2}{20}
  \multido{\n=0+4}{5}{        % 4 instances of period 4
    \TaskArrDead{1}{\n}{4}    % draw the arrival and deadline
    \TaskExecDelta{1}{\n}{1}  % draw execution (highest priority),
                              % from \n to \n+1
  }

  \multido{\n=0+6}{3}{        % 3 instances of period 6
    \TaskArrDead{2}{\n}{6}    % draw the arrival and deadline
  }
  % no simple formula for lowest priority, sorry!
  \TaskExecution{2}{1}{4}
  \TaskExecution{2}{6}{8}
  \TaskExecution{2}{9}{10}
  \TaskExecution{2}{13}{16}
\end{RTGrid}
```

Figure 2: Using multido to avoid repetitions



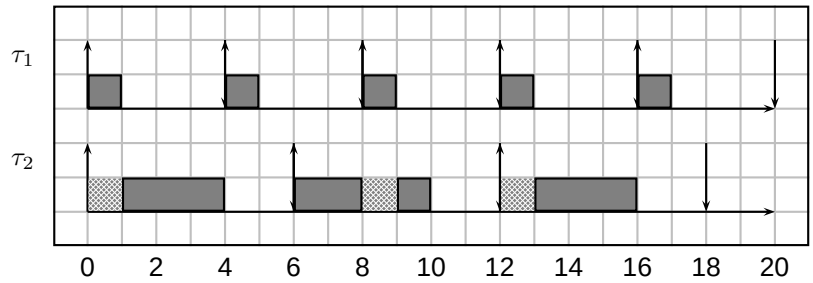

```
\begin{RTGrid}{3}{14}
  \multido{\n=0+3}{4}{        % 4 instances of period 3
    \TaskArrival{1}{\n}       % draw only the arrival
    \TaskExecDelta{1}{\n}{1}} % draw execution (highest priority),
                              % from \n to \n+1

  \multido{\n=3+4}{3}{        % 3 instances of period 4, starting from 3
    \TaskArrival{2}{\n}}      % draw only the arrival

  \TaskExecDelta{2}{4}{1}
  \TaskExecDelta{2}{7}{1}
  \TaskExecDelta{2}{11}{1}

  \multido{\n=1+5}{3}{        % 3 instances of period 5, starting from 1
    \TaskArrival{3}{\n}}      % draw only the arrival

  \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



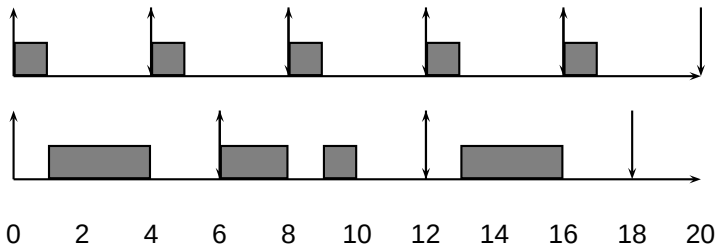

```
\multido{\n=0+6}{3}{
\TaskArrDead{1}{\n}{3}}
\multido{\n=2+8}{2}{
\TaskArrDead{2}{\n}{5}}
```

Figure 4: Deadlines less than periods



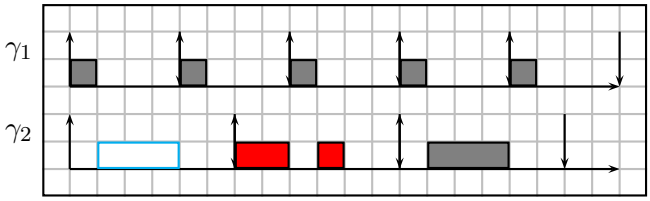

```
\begin{RTGrid}{2}{20}
  \multido{\n=0+4}{5}{
    \TaskArrDead{1}{\n}{4}
    \TaskExecDelta{1}{\n}{1}}
  \multido{\n=0+6}{3}{
    \TaskArrDead{2}{\n}{6}}

  \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



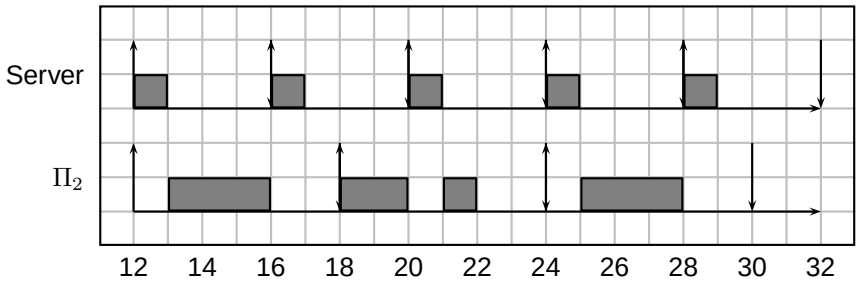

```
%% no grid and no symbols
\begin{RTGrid}[nogrid=1,nosymbols=1]{2}{20}
  \multido{\n=0+4}{5}{
    \TaskArrDead{1}{\n}{4}
    \TaskExecDelta{1}{\n}{1}}
  \multido{\n=0+6}{3}{
      \TaskArrDead{2}{\n}{6}}
  \TaskExecution{2}{1}{4}
  \TaskExecution{2}{6}{8}
  \TaskExecution{2}{9}{10}
  \TaskExecution{2}{13}{16}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

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



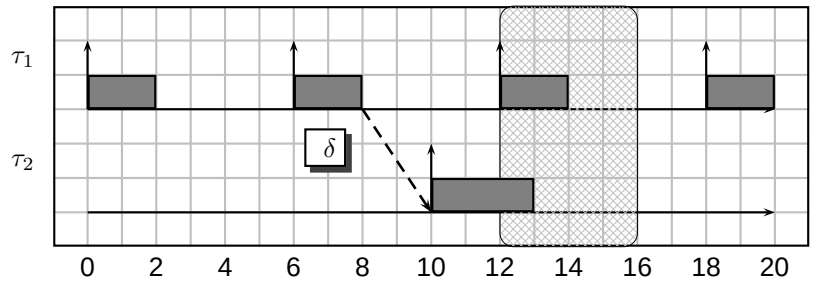

```
%% specify 1) no numbers on the time line, 2) a different symbol, 3)
%% a different size of the symbol (default is 8pt).
%% Notice that you should not use the math mode in the
%% specification of the symbol, as the symbol is already used in a
%% math environment inside the macro
\begin{RTGrid}[symbol=\gamma,nonumbers=1,labelsize=11pt]{2}{20}
  \multido{\n=0+4}{5}{
    \TaskArrDead{1}{\n}{4}
    \TaskExecDelta{1}{\n}{1}}
  \multido{\n=0+6}{3}{
    \TaskArrDead{2}{\n}{6}
  }
  %% 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 size 11pt), no numbers, a different task color



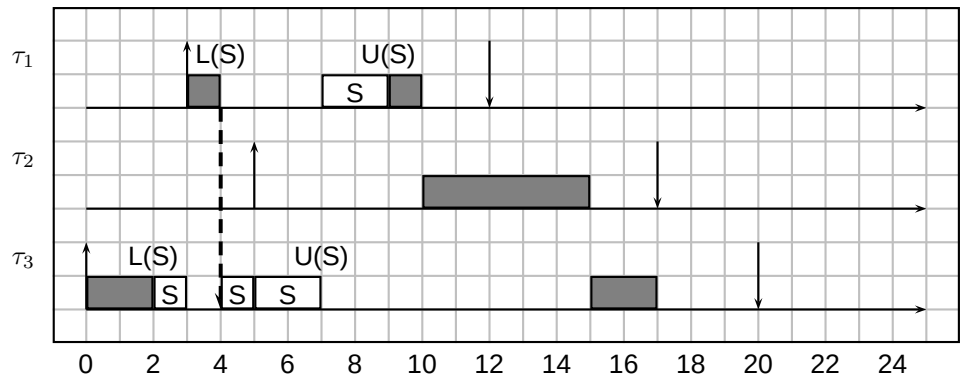

```
%% specify 1) no numbers on the time line, 2) number starting from 12
\begin{RTGrid}[nosymbols=1,numoffset=12]{2}{20}
  %% the symbol for the first row
  \RowLabel{1}{Server}
  %% the symbol for the second row
  \RowLabel{2}{$\Pi_2$}
  \multido{\n=0+4}{5}{
    \TaskArrDead{1}{\n}{4}
    \TaskExecDelta{1}{\n}{1}}
  \multido{\n=0+6}{3}{
    \TaskArrDead{2}{\n}{6}
  }
  \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}
  \multido{\n=0+6}{4}{
    \TaskArrival{1}{\n}
    \TaskExecDelta{1}{\n}{2}}
  \TaskArrival{2}{10}
  \TaskExecDelta{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




```
\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 10: Priority Inheritance example