Extension d'un outil de trace pour système embarqué temps réel

Encadrants : Laurent Pautet, Jérôme Hugues

Table des matières

TABLE DES MATIÈRES 2

INTRODUCTION 3

I/ RTEMS MONITORING APPLICATION 3

I.1 Présentation de l'application initiale 3
I.2 Bugs initiaux 4
1. Début du monitoring 4
2. Problèmes dans les exemples - le protocole 'immediate PCP' 5
I.3 Ajouts 5
1. Ajout de l'évènement TASK DELETE 5
2. Ajout de l'évènement SLEEP 5
3. Ajout d'informations dans les évènements 6
I.4 Exemples de traces obtenues 6
1. Test PCP 6
2. Test PIP 8

II/ COMPARAISON AVEC CHEDDAR 9

II.1 Présentation de l'outil 9
II.2 Simulation de l'ordonnancement RTEMS 10
1. Définition du processeur 10
2. Définition des taches 10
3. Définition des ressources 11
II.3 Ajout du protocole "immediate PCP" 11

III/ GÉNÉRATION DE SCÉNARIOS 12

III.1 Génération automatique de scénarios pour RTMA et CHEDDAR 12
III.2 Comparaison des ordonnancements obtenus 13
III.3 Exemples 13
1. Test PIP 13
2. Test PCP 15
3. Test RMS 16

IV/ MODE D'EMPLOI 18

IV.1 Génération d'un scénario 18
IV.2 Lancement de RTMA 18
IV.3 Lancement de CHEDDAR 19
IV.4 Comparaison 19

RÉFÉRENCES 20

Introduction

Le but de ce projet est d'observer et de comparer des ordonnancements obtenus lors de l'exécution d'applications temps réel. Cette exécution se fera de deux manières différentes :

D'une part, on exécutera réellement des applications temps réel sous RTEMS, et on observera l'ordonnancement obtenu à l'aide de RTMA, une application de monitoring conçue pour RTEMS.

D'autre part, on simulera l'exécution de l'application grâce à Cheddar, un simulateur d'ordonnancement temps réel, dont on pourra observer la sortie.

Le projet consiste donc à améliorer les traces obtenues par RTMA, automatiser la génération de scénarios temps réel pour RTEMS et Cheddar, faire en sorte que l'ordonnancement puisse être analogue dans les deux cas, et permettre la comparaison des résultats obtenus en sortie.

I/ RTEMS Monitoring Application

I.1 Présentation de l'application initiale

RTEMS Monitoring Application, ou RTMA, a été initialement conçue par Thomas d'Erceville, Alexis Anastassiades, Saïd Lankri, Albert Dalode et Lyes Abdelfettah. Il s'agit d'un outil de trace pour applications RTEMS.

L'utilisateur doit disposer d'une application RTEMS dont il souhaite suivre l'ordonnancement. Il doit aussi indiquer l'intervalle durant lequel il veut que son application soit monitorée. Il suffit alors d'insérer cette application dans l'application de monitoring, et de lancer celle-ci. RTMA se chargera alors de lancer l'application de l'utilisateur, et fournira en sortie différentes informations sur l'ordonnancement obtenu.

Ces informations incluent les évènements suivants :
- TASK START : le démarrage d'une tache par une autre tache
- TASK SWITCH : le changement de contexte entre deux taches
- SEM OBTAIN : la prise d'un sémaphore
- SEM RELEASE : la libération d'un sémaphore
- TASK EXITTED : une tache atteint la fin de ses instructions

Pour chacun de ces évènements sont indiqués les identifiants des taches ou sémaphores qui interviennent, ainsi que l'instant où a eu lieu l'évènement.

Ainsi, la trace la plus simple que l'on puisse obtenir est celle de la figure 1, obtenue lors du monitoring d'une application qui ne lance aucune autre tache, mais se contente de dire 'Hello world'.

Celle-ci permet de comprendre le fonctionnement de base de RTMA. On voit que les temps sont indiqués en ticks, 1 tick = 100 microsecondes. RTMA, l'application de monitoring, correspond à la tache 167837697 (elle a une priorité de 1, c'est-à-dire la priorité maximale). A l'instant 1647, elle laisse la main à la tache 167837698 qui correspond en fait à l'application de l'utilisateur qu'elle vient de lancer. Celle-ci dit 'Hello world' avant de se terminer, à l'instant 1660. C'est alors la tache 151060481, qui correspond en fait au noyau, qui reprend la main, jusqu'à ce que RTMA reprenne la main à l'instant 51647. On constate que c'est en fait 50000

ticks, soit 5 secondes, après le début du monitoring, ce qui correspond au temps de monitoring choisi par l'utilisateur (interval end = 50000). Dans le cas présent, RTMA a donc repris la main alors que l'application de l'utilisateur était terminée, mais si ce n'était pas le cas elle supprimerait celle-ci avant de reprendre la main. Enfin, RTMA affiche les traces à l'écran.

```
*****************************************
* Remote Monitoring Application         *
* (C) Under the GPL version 2+          *
* NO WARRANTY                           *
*****************************************

[RTEMS_MONITOR] Monitor Task Id : 167837697

[RTEMS_MONITOR] RTEMS MONITOR COMPILED WITH NO NETWORK SUPPORT
Buffer size : 32768, interval start : 0, interval end : 50000
[RTEMS_MONITOR] Starting Monitored Task : 167837698

[RTEMS_MONITOR] Time since boot 1493
[RTEMS_MONITOR] << PREEMPT >>
[RTEMS_MONITOR] Monitoring extensions installed
HELLO WORLD !


>> RTEMS MONITOR CONSOLE DUMP START <<

Buffer : 52 bytes used (of 32768 bytes total)
Buffer : [FIXME] first 0 bytes not used
Times are in ticks. 1 tick = 100 microseconds

[TASK SWITCH ] Time       1647 | SwappedOut    167837697 | SwappedIn       167837698
[TASK EXITTED] Time       1660 | Id            167837698
[TASK SWITCH ] Time       1661 | SwappedOut    167837698 | SwappedIn       151060481
[TASK SWITCH ] Time      51647 | SwappedOut    151060481 | SwappedIn       167837697
3 task switches

>> RTEMS MONITOR CONSOLE DUMP END <<

[RTEMS_MONITOR] RTEMS MONITOR END
```

Figure 1 - Trace obtenue par RTMA pour l'application 'Hello world'

I.2 Bugs initiaux

1. Début du monitoring

Comme on l'a vu précédemment, l'utilisateur doit indiquer durant quel intervalle il veut que son application soit monitorée. Ainsi, il indique un 'interval start' et un 'interval end'. RTMA va alors lancer l'application de l'utilisateur, dormir pendant 'interval start', se réveiller pour installer les extensions de monitoring, se rendormir pendant 'interval end', avant de finalement se réveiller et afficher les traces.

Or cette méthode posait un problème dans le cas (courant) où l'on souhaite que l'application soit monitorée dès le début. En effet, on indique alors 'interval start=0', ce qui fait que RTMA fait sleep(0) avant d'installer les extensions de monitoring. Or, faire sleep(0) permet tout de même à l'application de l'utilisateur de prendre la main, même si RTMA va la reprendre presque tout de suite après. Ainsi, les premières instructions de l'application de l'utilisateur avaient le temps de s'effectuer, avant l'installation du monitoring. Ce problème

apparaît clairement pour l'exemple 'hello world', où l'application de l'utilisateur a le temps de se terminer dans l'intervalle du sleep(0). Ainsi les traces obtenues sont:

```
[TASK SWITCH ] Time       1660 | SwappedOut    167837697 | SwappedIn       151060481
[TASK SWITCH ] Time      51660 | SwappedOut    151060481 | SwappedIn       167837697
2 task switches
```

Le 'hello world' s'étant déjà terminé avant que RTMA ne lance le monitoring et rende la main, on voit que c'est simplement le noyau qui reprend la main, jusqu'à la fin de l'intervalle de monitoring.

Ce problème s'est résolu très simplement en demandant à RTMA de ne s'endormir jusqu'au début du monitoring, que dans le cas où 'interval start' est non nulle.

2. Problèmes dans les exemples - le protocole 'immediate PCP'

Des exemples étaient fournis avec RTMA, notamment pour illustrer l'ordonnancement avec des sémaphores en PIP et en PCP. Or les ordonnancements obtenus ne correspondaient pas aux exemples présentés. Pour le protocole PIP, le problème était simplement dû à des problèmes de timings, des temps de travail ou de sommeil trop courts ou trop longs.

Pour le protocole PCP, par contre, le problème venait du fait que ce n'est pas le protocole PCP 'standard' qui est utilisé dans RTEMS, contrairement à ce qui était supposé dans les exemples donnés. RTEMS utilise le protocole 'immediate PCP'.

Ces deux protocoles se basent sur l'attribution d'une priorité plafonnée aux sémaphores, qui correspond à la priorité maximale des taches qui peuvent accéder à ce sémaphore. Dans le protocole PCP, une tache ne peut accéder à un sémaphore que si sa priorité est strictement supérieure à toutes les priorités plafonnées des sémaphores en cours d'utilisation. De plus, une tache bloquante hérite de la priorité de la tache bloquée la plus prioritaire.

Dans immediate PCP, que je nommerai IPCP, dès qu'une tache accède à un sémaphore, elle hérite de sa priorité plafonnée. Ainsi aucune tache qui pourrait essayer d'obtenir ce sémaphore ne peut prendre la main avant qu'il ne soit libéré. C'est cette version de PCP qui est utilisée par RTEMS.

I.3 Ajouts

1. Ajout de l'évènement TASK DELETE

J'ai rajouté l'évènement TASK DELETE aux traces obtenues par RTMA, ce qui permet de voir quand une tache en supprime une autre. Les évènements task_start, task_switch et task_exitted, déjà inclus dans les traces, avaient été implémentés grâce au gestionnaire d'extensions fourni par RTEMS (RTEMS User Extensions Manager). Or ce gestionnaire d'extensions permet de récolter d'autres informations, notamment task_create, task_begin et task_delete. Je n'ai rajouté que cette dernière, les autres ne me paraissant pas nécessaires à la compréhension de l'ordonnancement.

2. Ajout de l'évènement SLEEP

J'ai aussi rajouté l'évènement SLEEP, indiquant l'instant auquel une tache s'endort, et pour combien de temps. C'était nécessaire, en particulier, pour comprendre ce qui se passe lorsqu'une tache de forte priorité perd la main parce qu'elle s'endort.

Pour capturer cet évènement, j'ai créé un wrapper de la même manière que cela avait été fait pour la prise et la libération de sémaphore. C'est-à-dire que j'ai créé la fonction rtems_monitor_task_wake_after, qui enregistre les informations sur la tache qui s'endort avant d'appeler la fonction rtems_task_wake_after.

L'utilisateur n'a pas besoin de changer les fonctions utilisées dans son application, ceci est fait automatiquement par le macro:

```
#define rtems_task_wake_after(X) rtems_monitor_task_wake_after(X)
```

3. Ajout d'informations dans les évènements

Finalement, j'ai rajouté des informations supplémentaires dans les évènements existants, permettant de mieux comprendre l'ordonnancement. Pour chaque tache apparaissant dans les évènements, j'ai indiqué la priorité de la tache au moment de l'évènement, en plus de l'identifiant. Ceci permet d'observer les éventuels changements de priorités au cours de l'application.

J'ai aussi rajouté, pour les prises sémaphores, l'identifiant et la priorité de la tache qui prend ou libère le sémaphore. Ceci pouvait déjà être directement déduit à partir des traces en regardant quelle était la dernière tache courante avant l'évènement, mais de cette façon on peut le voir directement.

Avec ces modifications, les traces obtenues pour l'application 'Hello world' deviennent:

```
[SLEEP       ] Time        1603 | Sleep time        50000     | Task Id          167837697    1
[TASK SWITCH ] Time        1603 | SwappedOut    167837697    1| SwappedIn        167837698    2
[TASK EXITTED] Time        1616 | Id            167837698    2
[TASK DELETE ] Time        1616 | CurrentTask   167837698    2| DeletedTask      167837698    2
[TASK SWITCH ] Time        1617 | SwappedOut    167837698    2| SwappedIn        151060481  255
[TASK SWITCH ] Time       51603 | SwappedOut    151060481  255| SwappedIn        167837697    1
3 task switches
```

I.4 Exemples de traces obtenues

1. Test PCP

Exemple de deux taches partageant 2 sémaphores, le protocole PCP évitant ici l'interblocage.

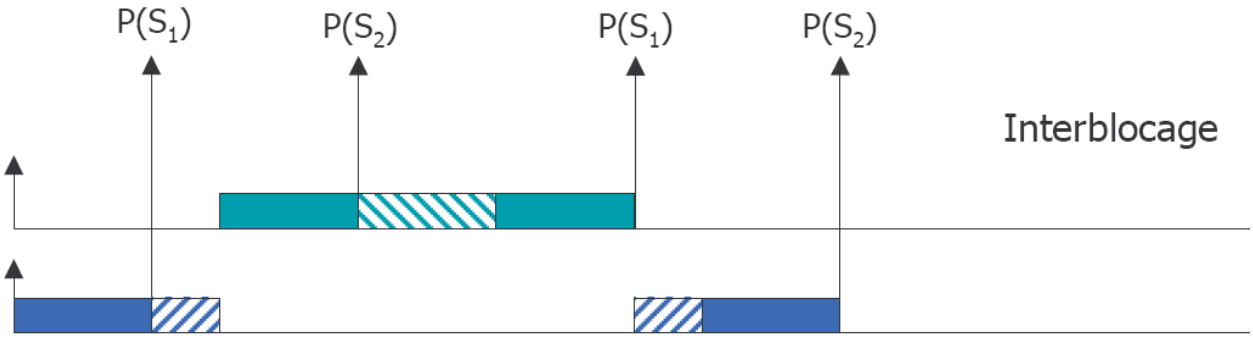


Figure 2 : Interblocage - situation évitée ici

Traces obtenues directement par des printf dans l'application de l'utilisateur :

```
actual current priority:10
Semaphore 1 priority ceiling: 21
Semaphore 2 priority ceiling: 21
task_id :(0x)A010003
Task  1, status = 0, priority = 21
task_id :(0x)A010004
Task  2, status = 0, priority = 22
main -- current priority : 10, is set to (40) :
Task_1 sleeps for 3ms
Task_2 - start work
Task_2 - P on Semaphore 0
Task  2, status = 0, priority = 21  [1]
Task_2 - enters critical section
Task_2 -  working
Task_2 - P on Semaphore 1
Task_2 - enters critical section
Task_2 - exits critical section
Task_2 - V on Semaphore 1
Task_2 - exits critical section
Task  2, status = 0, priority = 21
Task_2 - V on Semaphore 0
Task_1 -  begins working            [2]
Task_1 - P on Semaphore 1
Task_1 - enters critical section
Task_1 - P on Semaphore 0
Task_1 - enters critical section
Task_1 - exits critical section
Task_1 - V on Semaphore 0
Task_1 - exits critical section
Task_1 - V on Semaphore 1
Task_1 -  working
Task  2, status = 0, priority = 22
Task_2 -  stops working
```

[1] On voit ici que la priorité de la tache 2 est élevée à la priorité plafonnée du sémaphore qu'elle vient de prendre. Ainsi la tache 1 ne peut pas reprendre la main tant que le sémaphore n'est pas rendu, ce qui évite l'interblocage.
[2] La tache 2 ayant rendu les 2 sémaphores, sa priorité originale est rétablie, et la tache 1 peut maintenant reprendre la main.

Les traces obtenues par RTMA sont données figure 3 :

```
[SLEEP       ] Time       1603 | Sleep time       50000     | Task Id          167837697    1
[TASK SWITCH ] Time       1603 | SwappedOut   167837697    1| SwappedIn        167837698    2
[SLEEP       ] Time       1603 | Sleep time        1000     | Task Id          167837698    2
[TASK SWITCH ] Time       1603 | SwappedOut   167837698    2| SwappedIn        151060481  255
[TASK SWITCH ] Time       2603 | SwappedOut   151060481  255| SwappedIn        167837698    2
[TASK START  ] Time       2968 | StartingTask 167837698   10| ToBeStartedTask  167837699   21
[TASK START  ] Time       2968 | StartingTask 167837698   10| ToBeStartedTask  167837700   22
[TASK SWITCH ] Time       3012 | SwappedOut   167837698   40| SwappedIn        167837699   21
[SLEEP       ] Time       3033 | Sleep time         30     | Task Id          167837699   21
[TASK SWITCH ] Time       3033 | SwappedOut   167837699   21| SwappedIn        167837700   22
[SEM OBTAIN  ] Time       3052 | Sem Id       436273157     | Task Id          167837700   22
[SEM OBTAIN  ] Time       3180 | Sem Id       436273158     | Task Id          167837700   21
[SEM RELEASE ] Time       3265 | Sem Id       436273158     | Task Id          167837700   21
[SEM RELEASE ] Time       3351 | Sem Id       436273157     | Task Id          167837700   21
[TASK SWITCH ] Time       3352 | SwappedOut   167837700   22| SwappedIn        167837699   21
[SEM OBTAIN  ] Time       3400 | Sem Id       436273158     | Task Id          167837699   21
[SEM OBTAIN  ] Time       3455 | Sem Id       436273157     | Task Id          167837699   21
[SEM RELEASE ] Time       3539 | Sem Id       436273157     | Task Id          167837699   21
[SEM RELEASE ] Time       3593 | Sem Id       436273158     | Task Id          167837699   21
[TASK EXITTED] Time       3612 | Id           167837699   21
[TASK DELETE ] Time       3612 | CurrentTask  167837699   21| DeletedTask      167837699   21
[TASK SWITCH ] Time       3612 | SwappedOut   167837699   21| SwappedIn        167837700   22
[TASK EXITTED] Time       3667 | Id           167837700   22
[TASK DELETE ] Time       3667 | CurrentTask  167837700   22| DeletedTask      167837700   22
[TASK SWITCH ] Time       3668 | SwappedOut   167837700   22| SwappedIn        167837698   40
[TASK EXITTED] Time       3668 | Id           167837698   40
[TASK DELETE ] Time       3668 | CurrentTask  167837698   40| DeletedTask      167837698   40
[TASK SWITCH ] Time       3668 | SwappedOut   167837698   40| SwappedIn        151060481  255
[TASK SWITCH ] Time      51603 | SwappedOut   151060481  255| SwappedIn        167837697    1
10 task switches
```

Figure 3 - Trace obtenue par RTMA pour l'exemple avec PCP

2. Test PIP

Exemple de 3 taches partageant un sémaphore en PIP.

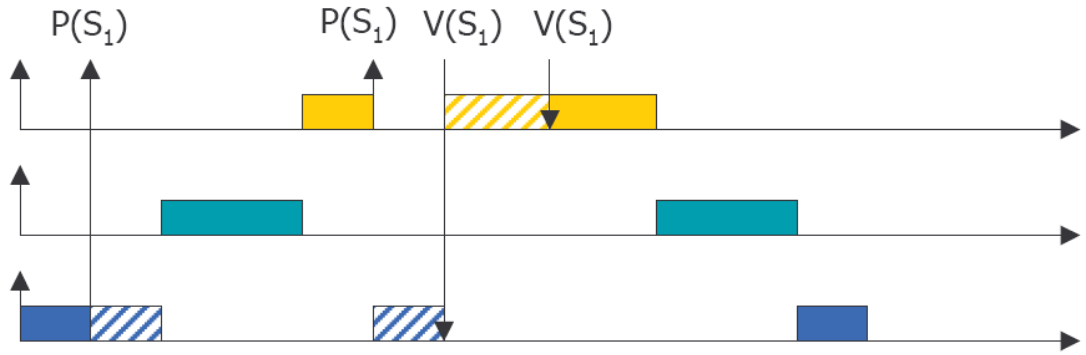


Figure 4 : Exemple du protocole PIP

Traces obtenues directement par des printf dans l'application de l'utilisateur :

```
actual current priority:10
task_id :(0x)A010003
Task  1, status = 0, priority = 21
task_id :(0x)A010004
Task  2, status = 0, priority = 22
task_id :(0x)A010005
Task  3, status = 0, priority = 23
main -- current priority : 10, is set to (40) :
Task_1 - sleeps for 20 ms
Task_2 - sleeps for 10 ms
Task_3 - P on Semaphore 1
Task_3 - enters critical section
Task_3 current priority : 23
Task_2 -  begins working
Task_1 - P on Semaphore 1
Task_3 current priority : 21          [1]
Task_3 - exits critical section
Task_3 - V on Semaphore 1
Task_1 - enters critical section      [2]
Task_1 - exits critical section
Task_1 - V on Semaphore 1
Task_2 -  stops working
Task_3 current priority : 23
main -- current priority : 40
main (init) : exit
```

[1] On voit ici que comme la tache 1 est bloquée par la tache 3 qui a pris le sémaphore avant elle, la priorité de la tache 3 est remontée au niveau de celle de la tache 1 jusqu'à ce qu'elle rende le sémaphore. Sinon, c'est la tache 2 qui aurait repris la main, introduisant un délai supplémentaire pour le déblocage de la tache 1.
[2] La tache 3 ayant rendu le sémaphore, sa priorité originale est rétablie, et la tache 1 reprend la main.

Les traces obtenues par RTMA sont données figure 5.

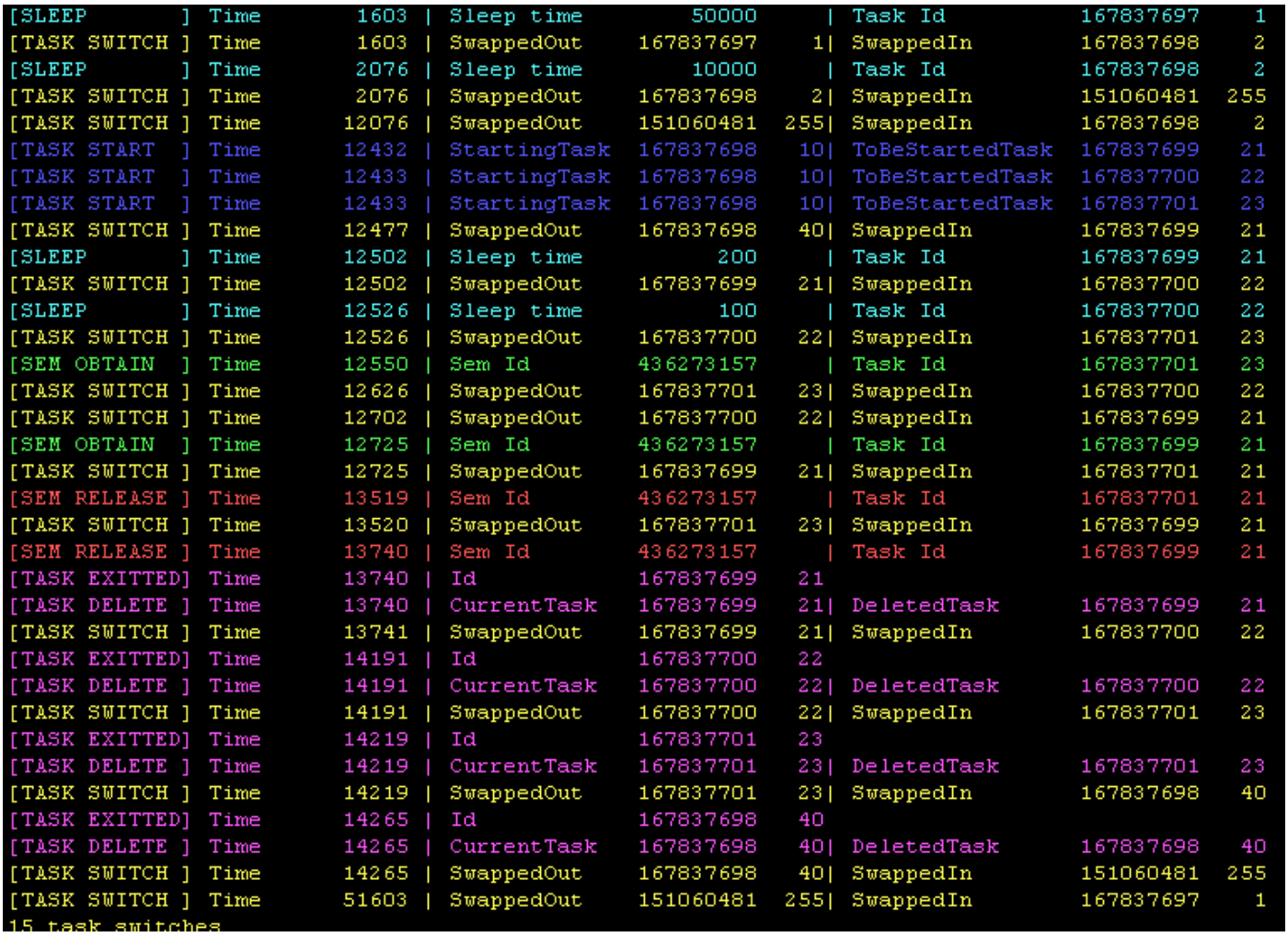


Figure 5 - Trace obtenue par RTMA pour l'exemple avec PIP

II/ Comparaison avec CHEDDAR

II.1 Présentation de l'outil

Cheddar est un outil de simulation d'ordonnancement temps réel, qui a été développé en Ada par le LISYC (Laboratoire d'Informatique des Systèmes Complexes), et principalement par Frank Singhoff. Les possibilités de cet outil sont étendues, nous parlerons principalement ici de celles qui ont été utilisées dans le cadre de ce projet.

L'utilisateur peut définir en entrée un ou plusieurs processeurs et leurs caractéristiques, une ou plusieurs taches et leurs caractéristiques, éventuellement des ressources partagées par les taches, et d'autres options encore qui ont été ou vont être développées.

La propriété la plus importante à définir pour un processeur, par exemple, est le type d'ordonnanceur (par exemple RMS, EDF...). De nombreux types d'ordonnanceurs sont disponibles, et dans le cas où aucun ne conviendrait, l'utilisateur a la possibilité d'en définir un lui-même.

Une fois toutes les entités définies et leurs propriétés précisées, on peut lancer la simulation. On obtient en sortie l'ordonnancement correspondant, mais aussi différentes informations obtenues par l'analyse des évènements (des analyseurs d'évènements pouvant aussi être rajoutés par l'utilisateur).

Pour définir les caractéristiques d'entrées, l'utilisateur dispose de plusieurs solutions. Il peut utiliser directement l'interface graphique (figure 6). Il peut aussi charger un fichier xml de description de scénario, contenant les mêmes informations que celles pouvant être entrées à la main.

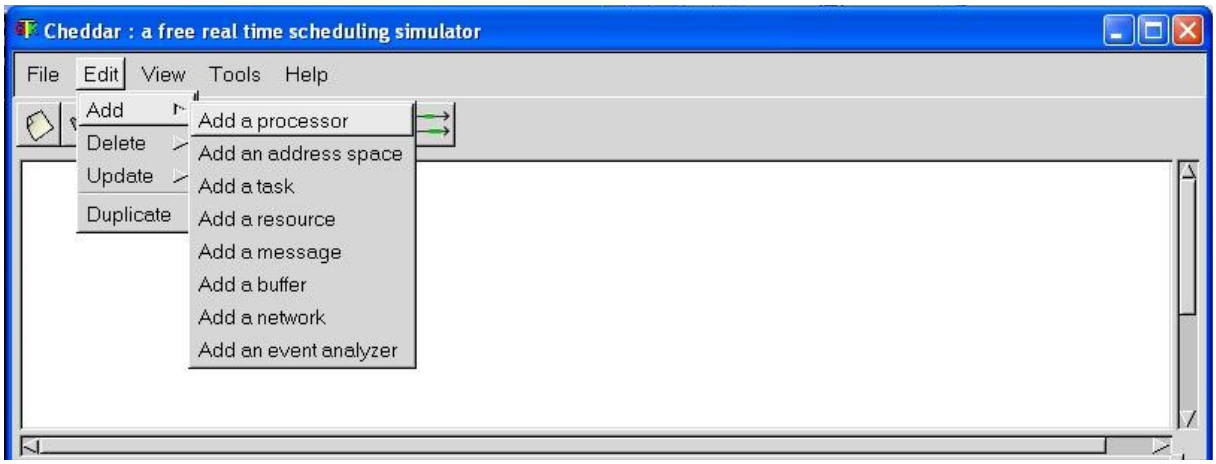


Figure 6 - Ajouts d'informations dans Cheddar par l'interface graphique

En sortie, une autre fonctionnalité utile est la possibilité de sauvegarder les évènements correspondant à l'ordonnancement dans un fichier xml d'évènements. Ces différentes utilisations de fichiers xml, pour la description de scénarios et la sauvegarde du résultat, ont été, comme nous le verrons, largement exploitées dans le cadre de ce projet.

II.2 Simulation de l'ordonnancement RTEMS

Voyons maintenant plus précisément comment simuler des scénarios afin que l'ordonnancement se passe en théorie comme sous RTEMS.

1. Définition du processeur

Les scénarios qui nous intéressent ne font intervenir qu'un seul processeur. Il faut donc indiquer en entrée de Cheddar un processeur, lui donner un nom, et choisir comme ordonnanceur "POSIX 1003.1b/Highest Priority First", qui est l'ordonnancement effectué par RTEMS. Il faut aussi préciser si le processeur est préemptif ou non. Sous RTEMS, ce choix se fait pour chaque tache individuellement. On choisi donc un ordonnancement préemptif, l'option par défaut pour les taches RTEMS étant RTEMS_PREEMPT, option qu'on ne changera donc pas.

Il faut aussi ajouter un espace d'adressage associé au processeur. Pour l'instant, il suffit de lui donner un nom et d'indiquer à quel processeur il est associé.

2. Définition des taches

Il faut ensuite définir les taches et leurs propriétés. Il y a 4 types prédéfinis de taches, periodic, aperiodic, poisson ou parametric. Les seuls types qui nous intéresseront ici sont periodic et aperiodic. Il faut ensuite choisir entre les politiques SCHED_FIFO ou SCHED_RR, qui sont les politiques utilisées pour l'ordonnancement de 2 taches de même

priorité. Ici aussi, nous gardons l'option SCHED_FIFO, qui correspond à l'option par défaut des taches sous RTEMS, c'est-à-dire RTEMS_NO_TIMESLICE.

Il faut bien sur aussi définir la priorité de la tache. Les priorités vont de 1 à 255, la plus élevée étant 255 alors que sous RTEMS la plus élevée est 1. Il faut donc faire attention de bien convertir les priorités lors de la simulation d'un scénario RTEMS sous Cheddar.

Enfin, il faut préciser les temps de départ, de travail, de deadline, et dans le cas des taches périodiques la période. Ces temps sont obligatoirement entiers.

3. Définition des ressources

Enfin, si le scénario fait intervenir des sémaphores, il faut rajouter des ressources. On indique donc le nom de la ressource, sa valeur initiale, et les intervalles durant lesquelles une tache veut accéder à la ressource. Ces intervalles ne peuvent pas être nuls, c'est-à-dire qu'un sémaphore ne peut pas être pris puis relâché immédiatement.

Il faut aussi préciser le protocole de partage de la ressource. Sous RTEMS, les protocoles possibles sont, comme nous l'avons vu, PIP, IPCP, ou aucun protocole. Or sous Cheddar les protocoles disponibles étaient 'No Protocol', PIP ou PCP. Ceci posait donc un problème pour simuler des scénarios utilisant le protocole IPCP sous RTEMS.

II.3 Ajout du protocole "immediate PCP"

Afin de pouvoir simuler des scénarios RTEMS faisant intervenir le protocole PCP, il a donc fallu rajouter cette version du protocole dans les options de Cheddar, en modifiant directement les sources.

J'ai donc implémenté "immediate PCP", ou IPCP, en lui donnant exactement les mêmes caractéristiques que sous RTEMS : une tache qui accède à un sémaphore hérite directement de sa priorité plafonnée. Elle ne retrouvera sa priorité initiale qu'après avoir libéré tous les sémaphores IPCP qu'elle détient.

Ce protocole est donc maintenant disponible, sous le nom IPCP dans l'interface graphique (figure 7) ou IMMEDIATE_PRIORITY_CEILING_PROTOCOL dans les fichiers xml de scénario.

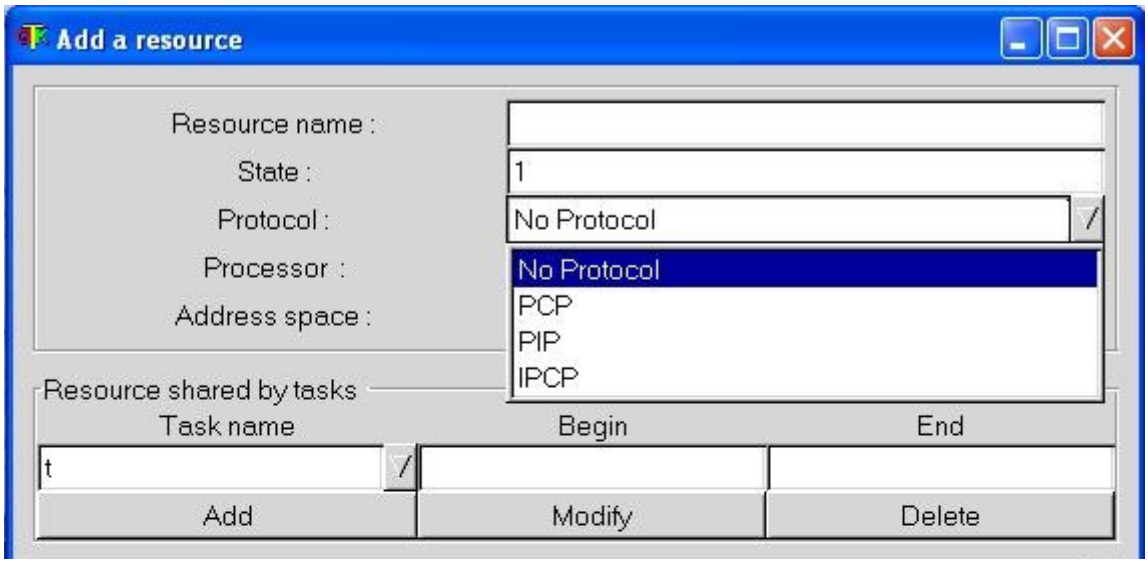


Figure 7 - Les différents protocoles maintenant disponibles pour les ressources

La figure 8 présente l'ordonnancement obtenu pour un des scénarios de tests de IPCP. Dans le cas du protocole PCP classique, la tache T1 aurait pu prendre la main à l'instant 1.

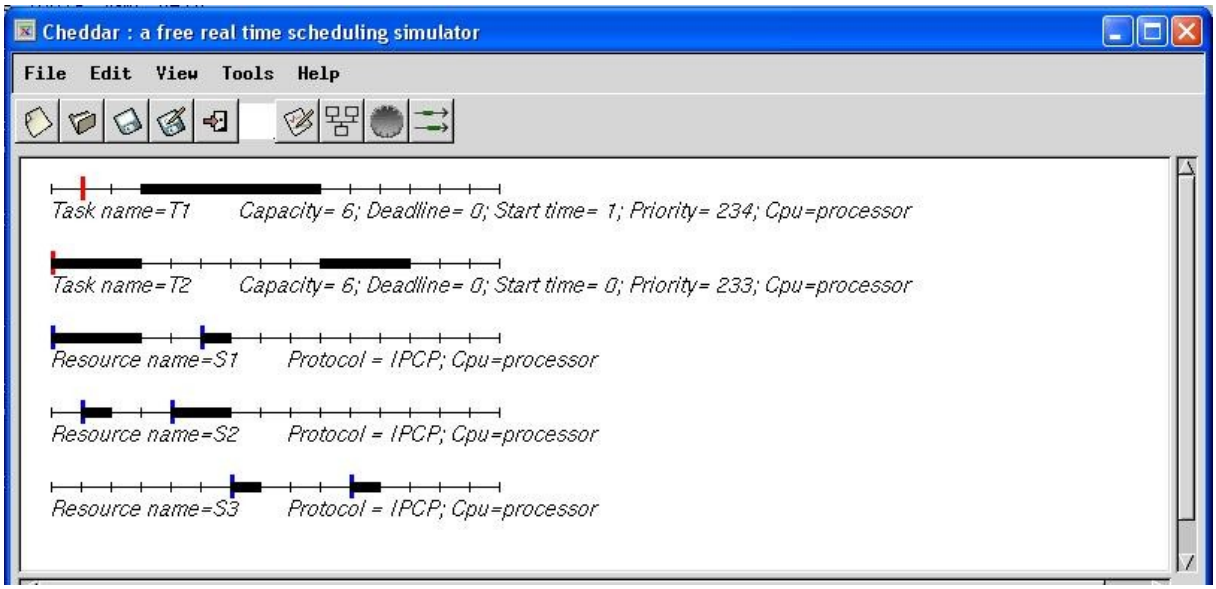


Figure 8 - résultat d'un scénario de test pour le protocole IPCP sous Cheddar

III/ Génération de scénarios

III.1 Génération automatique de scénarios pour RTMA et CHEDDAR

Afin de pouvoir comparer l'ordonnancement obtenu lors de l'exécution d'un scénario sous RTEMS et sous Cheddar, il faut d'abord écrire l'application correspondante sous RTEMS, et entrer toutes les informations nécessaires en entrée de Cheddar, ce qui est long et fastidieux dans les deux cas. C'est pourquoi on a décidé de permettre la génération automatique de ces scénarios.

Deux applications ont donc été crées, l'une permettant de générer du code RTEMS directement utilisable par RTMA, et l'autre permettant de générer le fichier xml de scénario à donner en entrée à Cheddar. Ces deux applications nécessitent bien sur qu'on leur fournisse en entrée un fichier de description du scénario. Ce fichier de description est un fichier texte respectant une structure et une syntaxe simple, le nombre d'options étant relativement limité. Le modèle de description du scénario est le suivant :

```
RUN_TIME : indiquer le temps durant lequel le scénario doit être monitoré
SEMAPHORES : indiquer le nombre de sémaphores utilisés
- Pour chaque sémaphore, indiquer le nom, la valeur initiale, et la
politique
TASKS : indiquer le nombre de taches
- Pour chaque tache, indiquer le nom, la politique (périodique ou non),
dans le cas d'une tache périodique la période et dans le cas d'une tache
apériodique la deadline, la priorité, et l'instant de démarrage
- Pour chaque tache, donner son nom suivi de la suite des instructions à
réaliser
END

Les instructions possibles sont :
S(2): dormir pendant 2 secondes          P(S1) : prendre le sémaphore S1
W(3): tourner pendant 3 secondes         V(S2) : relacher le sémaphore S2
```

A noter qu'il y a quelques différences possibles entre des scénarios pour Cheddar et des scénarios pour RTEMS. Ainsi si on veut générer un scénario pour Cheddar, les temps doivent obligatoirement être des entiers, et l'instruction S (sleep) n'est pas disponible. Les priorités doivent être indiqués en considérant que la plus élevée est 1, la conversion sera faite automatiquement dans le cas d'un scénario pour Cheddar.

Des exemples plus concrets de description de scénarios sont donnés plus bas.

III.2 Comparaison des ordonnancements obtenus

Une fois les scénarios générés et lancés, on obtient pour RTEMS des traces telles que décrites en I/, et pour Cheddar un fichier xml d'évènements. Or c'est assez difficile de comparer les deux, car les descriptions sont faites de manière très différente dans les deux cas.

Afin de faciliter cette comparaison, une dernière application a été créée prenant en entrée un fichier xml d'évènements Cheddar, et affichant en sortie des traces analogues aux traces de RTMA. Les sorties sont ainsi facilement comparables, à quelques détails près.

Voici les différences principales entre les traces obtenues :

- Pour RTEMS les identifiants des taches et des sémaphores sont leurs numéros d'identification, pour Cheddar ce sont leurs noms.
- Les temps sous Cheddar sont des entiers, qui correspondent sous RTEMS à des secondes, soit 10000 ticks. Ces temps sont naturellement moins exacts pour RTEMS, la simulation du temps de travail n'étant pas exacte au tick près.
- Les priorités ne sont pas indiquées pour Cheddar
- Quelques évènements apparaissent dans RTEMS, notamment lors de la création des taches par le main, qui n'apparaissent pas pour Cheddar puisque le main n'y existe pas. Il a cependant été rajouté artificiellement lors de la conversion du fichier xml d'évènements en traces RTMA, afin de minimiser ces différences.
- Dans le cas d'un dépassement de délai, une tache RTEMS va s'interrompre ou être détruite, alors qu'une tache Cheddar poursuivra son exécution.

III.3 Exemples

Pour mieux illustrer le processus de comparaison, voici quelques résultats obtenus.

1. Test PIP

Voici un fichier de scénario correspondant au test PIP vu en I.4.2 :

```
RUN_TIME 11
SEMAPHORES 1
S1 1 PIP
TASKS 3
T1 NONPERIODIC NONE 21 2
T2 NONPERIODIC NONE 22 1
T3 NONPERIODIC NONE 23 0
T1 W(1) P(S1) W(1) V(S1) W(1)
T2 W(4)
T3 P(S1) W(3) V(S1) W(2)
END
```

A partir de ce fichier, on génère automatiquement le code RTEMS correspondant qu'on lance ensuite à partir de RTMA.

On génère ensuite le fichier de scénario xml de Cheddar, on lance la simulation d'ordonnancement et on obtient en sortie le fichier xml d'évènements, que l'on converti en traces RTMA.

Les traces obtenues correspondent aux figures 9 et 10.

```
[SLEEP       ] Time       1604 | Sleep time       130000     | Task Id          167837697    1
[TASK SWITCH ] Time       1604 | SwappedOut    167837697    1| SwappedIn        167837698    2
[TASK START  ] Time       2069 | StartingTask  167837698    2| ToBeStartedTask  167837701   23
[SLEEP       ] Time       2070 | Sleep time        10000     | Task Id          167837698    2
[TASK SWITCH ] Time       2070 | SwappedOut    167837698    2| SwappedIn        167837701   23
[SEM OBTAIN  ] Time       2070 | Sem Id        436273157     | Task Id          167837701   23
[TASK SWITCH ] Time      12070 | SwappedOut    167837701   23| SwappedIn        167837698    2
[TASK START  ] Time      12070 | StartingTask  167837698    2| ToBeStartedTask  167837700   22
[SLEEP       ] Time      12071 | Sleep time        10000     | Task Id          167837698    2
[TASK SWITCH ] Time      12071 | SwappedOut    167837698    2| SwappedIn        167837700   22
[TASK SWITCH ] Time      22071 | SwappedOut    167837700   22| SwappedIn        167837698    2
[TASK START  ] Time      22071 | StartingTask  167837698    2| ToBeStartedTask  167837699   21
[TASK SWITCH ] Time      22071 | SwappedOut    167837698   24| SwappedIn        167837699   21
[SEM OBTAIN  ] Time      32084 | Sem Id        436273157     | Task Id          167837699   21
[TASK SWITCH ] Time      32084 | SwappedOut    167837699   21| SwappedIn        167837701   21
[SEM RELEASE ] Time      51742 | Sem Id        436273157     | Task Id          167837701   21
[TASK SWITCH ] Time      51742 | SwappedOut    167837701   23| SwappedIn        167837699   21
[SEM RELEASE ] Time      61628 | Sem Id        436273157     | Task Id          167837699   21
[TASK EXITTED] Time      71641 | Id            167837699   21
[TASK DELETE ] Time      71641 | CurrentTask   167837699   21| DeletedTask      167837699   21
[TASK SWITCH ] Time      71642 | SwappedOut    167837699   21| SwappedIn        167837700   22
[TASK EXITTED] Time     101689 | Id            167837700   22
[TASK DELETE ] Time     101689 | CurrentTask   167837700   22| DeletedTask      167837700   22
[TASK SWITCH ] Time     101690 | SwappedOut    167837700   22| SwappedIn        167837701   23
[TASK EXITTED] Time     121461 | Id            167837701   23
[TASK DELETE ] Time     121461 | CurrentTask   167837701   23| DeletedTask      167837701   23
[TASK SWITCH ] Time     121462 | SwappedOut    167837701   23| SwappedIn        167837698   24
[TASK EXITTED] Time     121462 | Id            167837698   24
[TASK DELETE ] Time     121462 | CurrentTask   167837698   24| DeletedTask      167837698   24
[TASK SWITCH ] Time     121462 | SwappedOut    167837698   24| SwappedIn        151060481  255
[TASK SWITCH ] Time     131604 | SwappedOut    151060481  255| SwappedIn        167837697    1
13 task switches
```

Figure 9 - traces obtenues pour RTEMS à partir de la description d'un scénario PIP

```
[TASK START  ] Time          0 | StartingTask          MAIN| ToBeStartedTask          T3
[TASK SWITCH ] Time          0 | SwappedOut            MAIN| SwappedIn                T3
[SEM OBTAIN  ] Time          0 | Sem Id                  S1| Task Id                  T3
[TASK SWITCH ] Time          1 | SwappedOut              T3| SwappedIn              MAIN
[TASK START  ] Time          1 | StartingTask          MAIN| ToBeStartedTask          T2
[TASK SWITCH ] Time          1 | SwappedOut            MAIN| SwappedIn                T2
[TASK SWITCH ] Time          2 | SwappedOut              T2| SwappedIn              MAIN
[TASK START  ] Time          2 | StartingTask          MAIN| ToBeStartedTask          T1
[TASK SWITCH ] Time          2 | SwappedOut            MAIN| SwappedIn                T1
[SEM OBTAIN  ] Time          3 | Sem Id                  S1| Task Id                  T1
[TASK SWITCH ] Time          3 | SwappedOut              T1| SwappedIn                T3
[SEM RELEASE ] Time          5 | Sem Id                  S1| Task Id                  T3
[TASK SWITCH ] Time          5 | SwappedOut              T3| SwappedIn                T1
[SEM RELEASE ] Time          6 | Sem Id                  S1| Task Id                  T1
[TASK EXITTED] Time          7 | Id                      T1
[TASK SWITCH ] Time          7 | SwappedOut              T1| SwappedIn                T2
[TASK EXITTED] Time         10 | Id                      T2
[TASK SWITCH ] Time         10 | SwappedOut              T2| SwappedIn                T3
[TASK EXITTED] Time         12 | Id                      T3
```

Figure 10 - traces obtenues pour Cheddar à partir de la description d'un scénario PIP

L'ordonnancement obtenu est donc bien le même, bien qu'il paraisse plus compliqué sous RTEMS à cause de l'intervention de la tache correspondant à RTMA, et des endormissements du main permettant de démarrer les autres taches à différents instants. Les temps sont aussi différents car le temps sous RTEMS est considéré à partir du démarrage de RTMA, mais si on considère le temps en relatif il est bien analogue dans les deux cas.

2. Test PCP

Le test PCP permet notamment de vérifier l'implémentation de IPCP dans Cheddar. Le fichier scénario correspondant au test vu en I.4.1 est donc :

```
RUN_TIME 9
SEMAPHORES 2
S1 1 PCP
S2 1 PCP
TASKS 2
T1 NONPERIODIC NONE 21 1
T2 NONPERIODIC NONE 22 0
T1 W(1) P(S2) W(1) P(S1) W(1) V(S1) V(S2) W(1)
T2 P(S1) W(2) P(S2) W(1) V(S2) V(S1) W(1)
END
```

Les traces obtenues correspondent aux figures 11 et 12.

```
[SLEEP       ] Time        1603 | Sleep time        90000     | Task Id          167837697    1
[TASK SWITCH ] Time        1603 | SwappedOut    167837697    1| SwappedIn        167837698    2
[TASK START  ] Time        2062 | StartingTask  167837698    2| ToBeStartedTask  167837700   22
[SLEEP       ] Time        2063 | Sleep time        10000     | Task Id          167837698    2
[TASK SWITCH ] Time        2063 | SwappedOut    167837698    2| SwappedIn        167837700   22
[SEM OBTAIN  ] Time        2063 | Sem Id        436273157     | Task Id          167837700   22
[TASK SWITCH ] Time       12063 | SwappedOut    167837700   21| SwappedIn        167837698    2
[TASK START  ] Time       12063 | StartingTask  167837698    2| ToBeStartedTask  167837699   21
[TASK SWITCH ] Time       12063 | SwappedOut    167837698   23| SwappedIn        167837700   21
[SEM OBTAIN  ] Time       22051 | Sem Id        436273158     | Task Id          167837700   21
[SEM RELEASE ] Time       32173 | Sem Id        436273158     | Task Id          167837700   21
[SEM RELEASE ] Time       32173 | Sem Id        436273157     | Task Id          167837700   21
[TASK SWITCH ] Time       32174 | SwappedOut    167837700   22| SwappedIn        167837699   21
[SEM OBTAIN  ] Time       42296 | Sem Id        436273158     | Task Id          167837699   21
[SEM OBTAIN  ] Time       52290 | Sem Id        436273157     | Task Id          167837699   21
[SEM RELEASE ] Time       62284 | Sem Id        436273157     | Task Id          167837699   21
[SEM RELEASE ] Time       62284 | Sem Id        436273158     | Task Id          167837699   21
[TASK EXITTED] Time       72278 | Id            167837699   21
[TASK DELETE ] Time       72278 | CurrentTask   167837699   21| DeletedTask      167837699   21
[TASK SWITCH ] Time       72279 | SwappedOut    167837699   21| SwappedIn        167837700   22
[TASK EXITTED] Time       82400 | Id            167837700   22
[TASK DELETE ] Time       82400 | CurrentTask   167837700   22| DeletedTask      167837700   22
[TASK SWITCH ] Time       82401 | SwappedOut    167837700   22| SwappedIn        167837698   23
[TASK EXITTED] Time       82401 | Id            167837698   23
[TASK DELETE ] Time       82401 | CurrentTask   167837698   23| DeletedTask      167837698   23
[TASK SWITCH ] Time       82401 | SwappedOut    167837698   23| SwappedIn        151060481  255
[TASK SWITCH ] Time       91603 | SwappedOut    151060481  255| SwappedIn        167837697    1
9 task switches
```

Figure 11 - traces obtenues pour RTEMS à partir de la description d'un scénario PCP

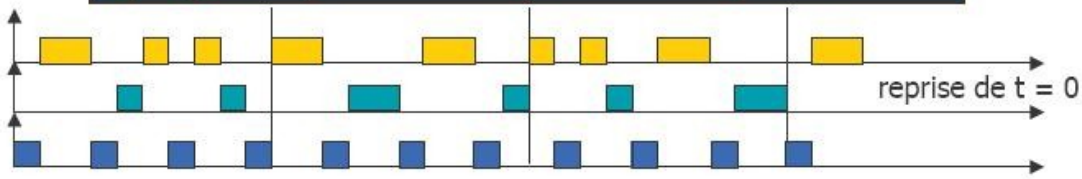
```
[TASK START  ] Time          0 | StartingTask          MAIN| ToBeStartedTask          T2
[TASK SWITCH ] Time          0 | SwappedOut            MAIN| SwappedIn               T2
[SEM OBTAIN  ] Time          0 | Sem Id                  S1| Task Id                 T2
[TASK SWITCH ] Time          1 | SwappedOut              T2| SwappedIn             MAIN
[TASK START  ] Time          1 | StartingTask          MAIN| ToBeStartedTask          T1
[TASK SWITCH ] Time          2 | SwappedOut            MAIN| SwappedIn               T2
[SEM OBTAIN  ] Time          2 | Sem Id                  S2| Task Id                 T2
[SEM RELEASE ] Time          3 | Sem Id                  S1| Task Id                 T2
[SEM RELEASE ] Time          3 | Sem Id                  S2| Task Id                 T2
[TASK SWITCH ] Time          3 | SwappedOut              T2| SwappedIn               T1
[SEM OBTAIN  ] Time          4 | Sem Id                  S2| Task Id                 T1
[SEM OBTAIN  ] Time          5 | Sem Id                  S1| Task Id                 T1
[SEM RELEASE ] Time          6 | Sem Id                  S1| Task Id                 T1
[SEM RELEASE ] Time          6 | Sem Id                  S2| Task Id                 T1
[TASK EXITTED] Time          7 | Id                      T1
[TASK SWITCH ] Time          7 | SwappedOut              T1| SwappedIn               T2
[TASK EXITTED] Time          8 | Id                      T2
```

Figure 12 - traces obtenues pour Cheddar à partir de la description d'un scénario PCP

3. Test RMS

Voici un exemple d'ordonnancement RMS à 100% d'utilisation.

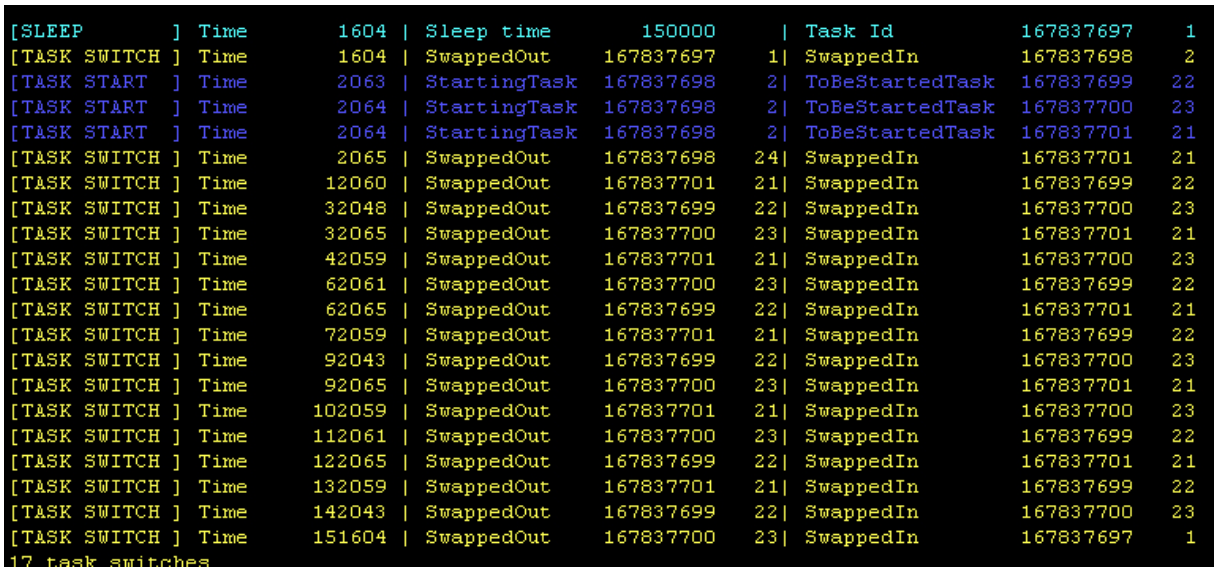
$3x(2^{1/3}-1)=0.78$	période	calcul	utilisation
$\tau'_1 = \tau_1 / 2$	5	2	0.400
τ'_2	15	4	0.267
$\tau'_3 = \tau_3 / 12$	3	1	0.333



Le fichier scénario correspondant :

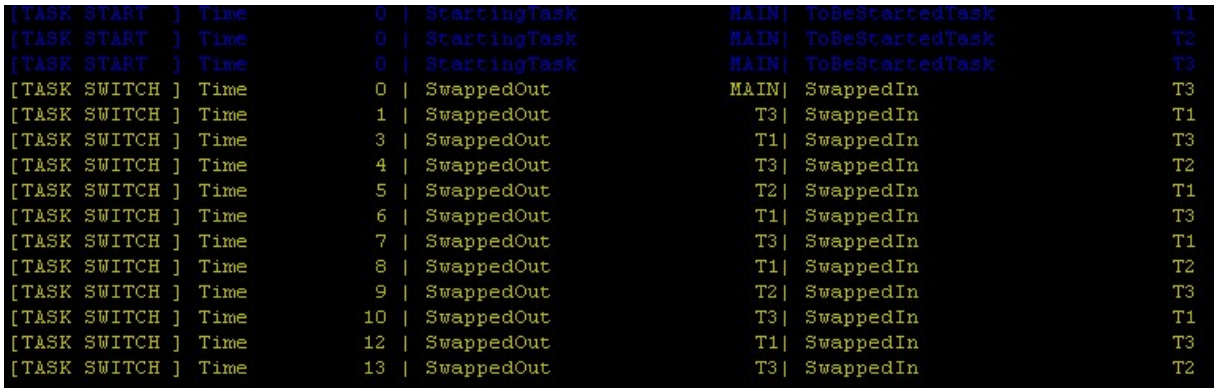
```
RUN_TIME 15
SEMAPHORES 0
TASKS 3
T1 PERIODIC 5 22 0
T2 PERIODIC 15 23 0
T3 PERIODIC 3 21 0
T1 W(2)
T2 W(4)
T3 W(1)
END
```

Les traces obtenues correspondent aux figures 13 et 14.



```
[SLEEP       ] Time       1604 | Sleep time       150000     | Task Id          167837697    1
[TASK SWITCH ] Time       1604 | SwappedOut    167837697    1| SwappedIn        167837698    2
[TASK START  ] Time       2063 | StartingTask  167837698    2| ToBeStartedTask  167837699   22
[TASK START  ] Time       2064 | StartingTask  167837698    2| ToBeStartedTask  167837700   23
[TASK START  ] Time       2064 | StartingTask  167837698    2| ToBeStartedTask  167837701   21
[TASK SWITCH ] Time       2065 | SwappedOut    167837698   24| SwappedIn        167837701   21
[TASK SWITCH ] Time      12060 | SwappedOut    167837701   21| SwappedIn        167837699   22
[TASK SWITCH ] Time      32048 | SwappedOut    167837699   22| SwappedIn        167837700   23
[TASK SWITCH ] Time      32065 | SwappedOut    167837700   23| SwappedIn        167837701   21
[TASK SWITCH ] Time      42059 | SwappedOut    167837701   21| SwappedIn        167837700   23
[TASK SWITCH ] Time      62061 | SwappedOut    167837700   23| SwappedIn        167837699   22
[TASK SWITCH ] Time      62065 | SwappedOut    167837699   22| SwappedIn        167837701   21
[TASK SWITCH ] Time      72059 | SwappedOut    167837701   21| SwappedIn        167837699   22
[TASK SWITCH ] Time      92043 | SwappedOut    167837699   22| SwappedIn        167837700   23
[TASK SWITCH ] Time      92065 | SwappedOut    167837700   23| SwappedIn        167837701   21
[TASK SWITCH ] Time     102059 | SwappedOut    167837701   21| SwappedIn        167837700   23
[TASK SWITCH ] Time     112061 | SwappedOut    167837700   23| SwappedIn        167837699   22
[TASK SWITCH ] Time     122065 | SwappedOut    167837699   22| SwappedIn        167837701   21
[TASK SWITCH ] Time     132059 | SwappedOut    167837701   21| SwappedIn        167837699   22
[TASK SWITCH ] Time     142043 | SwappedOut    167837699   22| SwappedIn        167837700   23
[TASK SWITCH ] Time     151604 | SwappedOut    167837700   23| SwappedIn        167837697    1
17 task switches
```

Figure 13 - traces obtenues pour RTEMS à partir de la description d'un scénario RMS



```
[TASK START  ] Time          0 | StartingTask          MAIN| ToBeStartedTask          T1
[TASK START  ] Time          0 | StartingTask          MAIN| ToBeStartedTask          T2
[TASK START  ] Time          0 | StartingTask          MAIN| ToBeStartedTask          T3
[TASK SWITCH ] Time          0 | SwappedOut            MAIN| SwappedIn                T3
[TASK SWITCH ] Time          1 | SwappedOut              T3| SwappedIn                T1
[TASK SWITCH ] Time          3 | SwappedOut              T1| SwappedIn                T3
[TASK SWITCH ] Time          4 | SwappedOut              T3| SwappedIn                T2
[TASK SWITCH ] Time          5 | SwappedOut              T2| SwappedIn                T1
[TASK SWITCH ] Time          6 | SwappedOut              T1| SwappedIn                T3
[TASK SWITCH ] Time          7 | SwappedOut              T3| SwappedIn                T1
[TASK SWITCH ] Time          8 | SwappedOut              T1| SwappedIn                T2
[TASK SWITCH ] Time          9 | SwappedOut              T2| SwappedIn                T3
[TASK SWITCH ] Time         10 | SwappedOut              T3| SwappedIn                T1
[TASK SWITCH ] Time         12 | SwappedOut              T1| SwappedIn                T3
[TASK SWITCH ] Time         13 | SwappedOut              T3| SwappedIn                T2
```

Figure 14 - traces obtenues pour Cheddar à partir de la description d'un scénario RMS

Les différences observées s'expliquent d'une part par le fait que le temps de calcul simulé sous RTEMS n'est pas exact au tick près. Prenons l'exemple de l'instant 32048. Normalement, comme on le voit dans les traces de Cheddar, la tache T3 devrait prendre la main après T1. Or sous RTEMS on voit que c'est T2 qui prend la main, pendant 17 ticks (soit 1.7 ms). C'est parce que T3 n'a pas travaillé pendant une seconde mais 0.9995 secondes, et T1 n'a pas travaillé pendant 2 secondes mais pendant 1.9988 secondes. Il reste donc 17 ticks de marge avant que la 2ème période de T3 ne commence, et c'est T2 qui en profite.

Une autre différence vient du fait que sous RTEMS, le décompte des périodes pour une tache démarre à partir du moment où une tache prend la main pour la première fois, et non pas à partir du moment où elle est activée. Ainsi la tache T1, qui prend la main à l'instant 12060 et qui a une période de 5, a sa 2ème période qui démarre à 62060, sa 3ème à 112060, etc. Alors que sous Cheddar le décompte démarre lors de son activation, à l'instant 0, et ses périodes démarrent donc à 0, 5, 10 etc.

IV/ Mode d'emploi

IV.1 Génération d'un scénario

Tous les éléments nécessaires à la génération d'un scénario se trouvent dans le répertoire *generate_scenario*.

Il faut d'abord décrire le scénario que l'on veut générer, cette description doit être faite dans un fichier texte. Le modèle pour cette description est le fichier *scenario_model*, où se trouvent les règles de description et de syntaxe, et les options disponibles. Quelques exemples de description de scénarios sont aussi disponibles.

Une fois ce fichier créé, on peut directement générer le code RTEMS correspondant, ou le fichier xml de scénario pour Cheddar. Les applications nécessaires se trouvent dans les répertoires *rtems_generate* et *cheddar_generate.*

Pour RTEMS, le générateur de code, *rtems_generate.c*, se trouve dans le répertoire *rtems_generate*. Le Makefile disponible créera l'exécutable *generate*, qu'il suffit de lancer en donnant en entrée le fichier de description du scénario.

Pour Cheddar, le générateur de code, *cheddar_xml_generate.c*, se trouve dans le répertoire *cheddar_generate*. Le Makefile disponible créera l'exécutable *generate* (mais aussi l'exécutable *display* à partir de *event_table_display.c* dont on verra l'utilisation ci-dessous). Pour générer le fichier xml de scénario il suffit de lancer *generate* avec le fichier de scénario défini précédemment.

IV.2 Lancement de RTMA

Le code RTEMS généré est constitué de deux fichiers, *init.c* et *tasks.h*, le premier contient le main de l'application, et le deuxième contient les taches qui seront lancées par le main. Cette application n'est pas destinée à être lancée de façon autonome, mais uniquement depuis RTMA.

La compilation pour RTMA se fait de manière très simple sur la machine xuri, à partir du répertoire *monitor_src.* Il faut tout d'abord indiquer, dans le fichier *Makefile.inc*, là où se trouve l'application à monitorer (qui doit nécessairement s'appeler *init.c*). Ceci se fait en modifiant la variable USER_ROOT pour donner le chemin d'accès à *init.c* (si l'application a été générée automatiquement elle devrait donc se trouver dans /generate_scenario/rtems_generate).

Une fois cette variable correctement mise à jour, il suffit de lancer le script de compilation *compile.sh.* L'exécutable final sera automatiquement placé dans */tftpboot/spif0.bin*.

Si on ne change pas l'emplacement de cet exécutable, on va donc le télécharger et le lancer sur la carte spif 0.

Pour se connecter sur la carte spif depuis xuri, faire "kermit kermitconf0". Le fichier de configuration *kermitconf0* se trouve dans le répertoire *monitor_build/monitor_exec*. Ensuite, taper c pour se connecter. Si la carte est bloquée, on peut faire un reset depuis xuri en tapant "echo R0>/dev/ttya6".

Enfin, on charge l'exécutable à l'aide de la commande tftpboot, et on le lance en faisant go 100000. A la fin de l'exécution, les traces devraient s'afficher à l'écran.

IV.3 Lancement de CHEDDAR

Le fichier xml généré pour Cheddar à partir du répertoire *cheddar_generate* se nomme *scenario.xml*. Ce fichier doit être utilisé avec la version 1.3p6 de Cheddar.

Cette version n'est, pour l'instant, disponible qu'à partir du dépôt CVS. Les sources sont donc disponibles sur http://beru.univ-brest.fr/~singhoff/cheddar/cvs/. (Tous les packages nécessaires à la compilation se trouvent sur http://beru.univ-brest.fr/~singhoff/cheddar/packages/). Sinon, des binaires pour windows sont temporairement disponibles sur http://beru.univ-brest.fr/~singhoff/cheddar/work/. (La version stable dont les binaires sont directement téléchargeables sur http://beru.univ-brest.fr/~singhoff/cheddar/ est actuellement la version 1.3p5, non suffisante car, entre autres, le protocole IPCP n'y est pas implémenté).

Une fois Cheddar installé et lancé, il suffit de cliquer sur "open XML project" et de charger le fichier *scenario.xml* généré précédemment. Il faut ensuite aller dans Tools/Scheduling/Customized scheduling simulation. Dans la fenêtre qui apparaît, cocher la case "Automatically export of event table", en indiquant éventuellement le nom du fichier xml que l'on souhaite obtenir en sortie (*default_event_table.xml*, par défaut). Après avoir cliqué sur Ok, il faut encore préciser la durée souhaitée de la simulation dans la case "Schedule from 0 to :" (à priori, si on veut des résultats analogues pour RTEMS et Cheddar, cette valeur devrait être choisie identique au RUN_TIME indiqué dans le fichier initial de description de scénario). On peut mettre la même valeur dans "Draw upto :" si on souhaite observer l'odonnancement graphiquement.

Une fois la simulation effectuée, on dispose normalement du fichier *default_event_table.xml* décrivant les évènements survenus lors de la simulation.

IV.4 Comparaison

Finalement pour pouvoir comparer les traces obtenues sous RTEMS aux résultats de Cheddar, il suffit de convertir le fichier xml d'évènements à l'aide de l'exécutable *display* présent dans le répertoire *cheddar_generate*.

Lorsqu'on tape ./display default_event_table.xml, des traces s'affichent à l'écran, semblables aux traces obtenues précédemment sous RTEMS. Les quelques différences ne devraient pas être gênantes pour la comparaison, une fois le lien fait entre les numéros d'identification et les noms des taches et des sémaphores.

Références

- http://www.rtems.com/onlinedocs/releases/rtemsdocs-4.6.5/share/rtems/html/c_user/ "*RTEMS 4.6.5 On-Line Library*"

- T. d'Erceville, A. Anastassiades, S. Lankri, A. Dalode, L. Abdelfettah, "*Spécification technique et réalisation de RTMA*"

- T. d'Erceville, A. Anastassiades, S. Lankri, A. Dalode , L. Abdelfettah, "*The Step by Step Guide to Monitoring RTEMS Applications Using RTMA*"

- http://beru.univ-brest.fr/~singhoff/cheddar/ "*The Cheddar Project*"

- http://beru.univ-brest.fr/~singhoff/cheddar/ug/ug.html "*Cheddar User's Guide*"