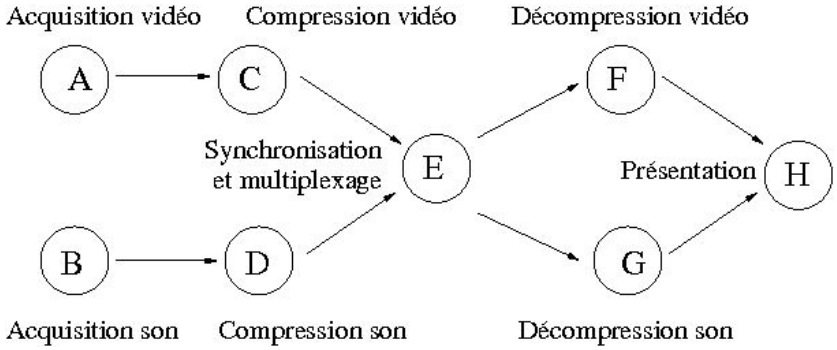
Année 2010-2011

INF342 - COU
Contrôle de Connaissances
28 juin 2011
3h00 - Sans document – Barème indicatif

Veuillez répondre à cette partie sur une feuille séparée

I. Ordonnancement (5 points)



On s'intéresse à une application multimédia dont les traitements sont décrits par la figure ci-dessus. L'application est constituée de 8 tâches soumises à des contraintes de précédence. Les paramètres temporels de cette application sont donnés ci-dessous (exprimés en milli-secondes) :

Tâches	Si	Ci	Di
A	0	3	12
B	0	1	10
C	0	3	20
D	0	1	15
E	0	5	25
F	0	4	37
G	0	1	20
H	0	1	40

Nous utilisons un algorithme DM préemptif. L'ensemble des tâches est exécuté sur un seul processeur. Le graphe de tâches est activé périodiquement toutes les 40 milli-secondes.

Question I-1

Ordonnancer le jeu de tâche sur la période d'étude sans tenir compte des contraintes de précédence. En analysant l'ordonnancement calculé, vérifier si le jeu de tâche est ordonnançable. Les contraintes de précédence sont elles respectées ?

Question I-2

Pour prendre en compte les contraintes de précédence, on va appliquer la méthode de Chetto & Blazewicz en modifiant les Di de chaque tâche. Cette méthode est généralement associée à EDF mais peut tout aussi bien être utilisée avec DM. Donnez les nouveaux Di et le chronogramme d'ordonnancement calculé avec ces nouveaux Di. Vérifiez si les contraintes temporelles et les contraintes de précédence sont respectées.

Question I-3

Tâches	Si	Ci	Di	Pi
A	0	5	12	20
B	0	3	10	40
C	0	3	20	20
D	0	1	17	40
E	0	7	25	20
F	0	3	37	20
G	0	2	20	40
H	0	2	40	40

Pour améliorer les performances de l'application, on souhaite expérimenter l'exécution de notre application sur une architecture multi-processeurs constituée de deux processeurs identiques. On espère, ainsi, doubler la fréquence d'affichage des images. Dans la suite de l'exercice, on ne tient plus compte des contraintes de précédence et seuls les paramètres temporels ci-dessus sont considérésNotez que la période des tâches associées au flot d'images a été divisée par deux.

Donner l'ordonnancement de ces tâches sur la période d'étude selon l'algorithme « global-DM » en mode préemptif. « global-DM » fonctionne de la façon suivante :

- A tout instant, la tâche la plus prioritaire est celle dont l'échéance est la plus petite.
- L'ordonnanceur attribue à chaque instant les m processeurs aux m tâches/travaux avec les échéances les plus courtes donc.
- Il préempte une tâche/travail lorsque tous les processeurs sont occupés.
- Il autorise les migrations à tout instant.

Question I-4

Pour vérifier le respect des échéances, nous avons simulé l'ordonnancement pendant la période d'étude. Dans le cas d'architectures mono-processeur, il est possible de vérifier le respect des échéances par le calcul du pire temps de réponse. Pour un ordonnanceur préemptif à priorité fixe, l'équation classique est la suivante :

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{P_j} \right\rceil \cdot C_j$$

Expliquez pourquoi cette équation n'est plus applicable avec notre architecture multi-processeurs.

Veuillez répondre à cette partie sur une feuille séparée

II Ordonnancement de tâches indépendantes périodiques et apériodiques (3 points)

$3*(2^{1/3} - 1) = 0,779$ $4*(2^{1/4} - 1) = 0,756$

On considère un système de tâches indépendantes composé de 2 tâches périodiques (TP1 et TP2) et de 2 tâches apériodiques (TA1 et TA2). Les tâches apériodiques vont être traitées par un serveur de tâches apériodique. On applique un ordonnancement de type RM. Le tableau ci dessous résume les caractéristiques des tâches. Notamment, il indique le temps de calcul et la période pour les tâches périodiques et le temps de calcul et la date d'activation pour les tâches apériodiques.

	Calcul	Période	Activation
Tâche Périodique 1 (TP1)	2	5	
Tâche Périodique 2 (TP2)	3	10	
Serveur Tâches Apériodiques (STA)	1	4	
Tâche Apériodique 1 (TA1)	2		4
Tâche Apériodique 2 (TA2)	2		5

- Ce groupe de tâches est-il ordonnançable par RMS ? On détaillera le raisonnement.
- Donner le diagramme d'ordonnancement lorsque le serveur de tâches apériodique est :
  - Un serveur en arrière plan
  - Un serveur à scrutation
  - Un serveur différé
  - Un serveur sporadique

Veuillez répondre à cette partie sur une feuille séparée

III. Bus et Réseaux Temps Réel (2 points)

Soit un jeu de quatre tâches s'exécutant sur deux sites S1 et S2:

- Les tâches A et C sur le site S1,
- Les tâches B et D sur le site S2,

A et B sont prêtes au temps t=0. C et D sont déclenchées sur reception de message :

- C attend un message émis par B,
- D attend un messages émis par A ,

A et B envoient les messages en fin d'activation; si le medium de communication n'est pas déjà occupé, elles en obtiennent immédiatement l'accès. Le délai de transmission d'un message d'un site à un autre est de 4 unités de temps. On donne les caractéristiques des tâches :

Tâche	WCET	Echéance
A	2	
B	4	
C	3	10
D	3	12

Question III-1

- **a-** Donner un diagramme montrant l'exécution de ces tâches et l'occupation du medium de communication
- **b-** Que peut-il se passer si le temps d'exécution de la tâche B est inférieur à son WCET ? Comment résoudre ce problème si le medium est un bus de type CAN ?

IV. Java Temps Réel (1 point)

Question IV-1

Citer deux moments de l'exécution d'un programme Java au cours desquels la gestion de la mémoire par le support d'exécution (la JVM) peut introduire des facteurs d'indéterminisme. Justifier votre réponse et donner les solutions proposées par RTSJ.

V. Noyaux Temps Réel (1 point)

Question V-1

- Comment l'architecture des systèmes embarque prend-elle en compte la grande variété des plateformes matérielles ?
- D'après ce que vous avez vu en cours et TP, si le modèle applicatif requiert un ordonnancement du type RMS et que le support d'exécution propose un ordonnancement par priorité, comme se fait l'implémentation de tâches RMS ?

Veuillez répondre à cette partie sur une feuille séparée

VI. Tolérance aux Fautes (2.5 points)

1) Définir les termes: zone de confinement d'erreur, disponibilité et « sécurité-innocuité »

On suppose que l'on dispose de plusieurs classes, C1, C2, C3 implémentant une même interface en Java (rappel : une interface définit une API qui contraint la signature de certaine méthodes. Ainsi chaque méthode décrite dans l'interface est implémentée dans chaque classe implémentant cette dernière.) Cette interface définit une fonction de calcul f :
public static int f(int x, int y, int z).

2) Rappeler le principe du N version-programming. En supposant que la méthode f n'a aucun effet de bord, expliquer comment l'appliquer ici pour obtenir une implémentation plus fiable de f. On supposera que les exceptions Java sont utilisées pour signaler l'occurrence d'une défaillance d'une méthode.

3) Nous considérons une architecture de type réplication (donc distribuée).
   a. Décrire ce qu'est une faute byzantine sur une réplique.
   b. Peut on tolérer une de ces fautes par la mise en place d'une réplication passive ?

VII. Langage synchrone (1.5 points)

Donner les huit premières valeurs en sortie du bloc lustre **system (cf après le nœud integrator).**

period	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
input	1.0	2.0	0.0	1.0	3.0	1.0	3.0	2.0
output1x								
output2x								

```
node integrator (input :real ; period :real)
returns (output :real)
let
output =0.0 -> (pre output + (pre input)*period) ;
tel ;

node system (input :real ; period : real)
returns (output1x :real ; output2x :real)
var alternate :bool ;
let
alternate=true -> not pre alternate ;
output2x=current(integrator((input when alternate),
                            ((period*2.0) when alternate))
                           ) ;
output1x=integrator(input,period) ;
tel ;
```

Veuillez répondre à cette partie sur une feuille séparée

VIII. Langage AADL (4 points) :

On considère un système de calcul de vitesse. Ce système (Speed_Calculator) est (entre autre) composé d'un capteur de position (Position_Sensor) qui envoie la position courante du système toutes les 10 millisecondes. Le thread Position_Thread reçoit cette information via son port input et enregistre cette donnée dans un tableau de position (Position_array). Cette écriture est réalisée par un appel au sous-programme Write. Périodiquement (toutes les 100 millisecondes), le thread Speed_Thread lie les informations contenues dans le tableau de positions puis calcul la vitesse du système en exécutant la fonction Read_and_Compute. Le comportement des sous-programmes Write et Read_and_Compute sont donnés grâce à l'annexe comportementale AADL.

```
data Integer end Integer;
data Boolean end Boolean;
data Speed end Speed;
data Position end Position;
data Position_Array end Position_Array;

system Speed_Calculator end Speed_Calculator;

system implementation Speed_Calculator.impl
subcomponents
  The_CPU : processor CPU;
  The_Sensor : device Position_Sensor;
  The_Process : process Speed_Process.impl;
connections
  cnx : port The_Sensor.Output -> The_Process.Input;
properties
  Actual_Processor_Binding => reference (The_CPU) applies to The_Process;
end Speed_Calculator.impl;

processor CPU end CPU;

process Speed_Process
features
  Input: in event data port Position;
  Output: out event data port Speed;
end Speed_Process;

device Position_Sensor
```

```
features
  Output: out event data port Position;
end Position_Sensor;

process implementation Speed_Process.impl
subcomponents
  The_Pos_Th: thread Position_Thread.impl;
  The_Speed_Th: thread Speed_Thread.impl;
  The_Pos1: data Position_array;
  The_Pos2: data Position_array;
connections
  cnx1 : port Input -> The_Pos_Th.Input;
  cnx2 : port Output -> The_Speed_Th.Output;
  cnx3 : data access The_Pos_Th.Pos1 -> The_Pos1;
  cnx4 : data access The_Pos_Th.Pos1 -> The_Pos1;
  cnx5 : data access The_Pos_Th.Pos2 -> The_Pos2;
  cnx6 : data access The_Pos_Th.Pos2 -> The_Pos2;
end Speed_Process.impl;

thread Speed_Thread
features
  Pos1 : requires data access Position_Array;
  Pos2 : requires data access Position_Array;
  Output : out event data port Speed;
properties
  Dispatch_Protocol   => Periodic;
  Period              => 100 Ms;
  Compute_Execution_Time => 0 Ms .. 50 Ms;
end Speed_Thread;

thread Position_Thread
features
  Pos1 : requires data access Position_Array;
  Pos2 : requires data access Position_Array;
  Input : in event data port Position;
properties
  Dispatch_Protocol   => Sporadic;
  Period              => 10 Ms;
  Compute_Execution_Time => 0 Ms .. 3 Ms;
end Position_Thread;

thread implementation Position_Thread.impl
subcomponents
  provide_position: subprogram Write;
calls call_sequence: {
  call: subprogram provide_position;};
connections
  cnx1 : port Pos1 -> provide_position.Pos1;
  cnx2 : port Pos2 -> provide_position.Pos2;
  cnx3 : Input -> provide_position.Input;
end Position_Thread.impl;

thread implementation Speed_Thread.impl
subcomponents
  produce_speed: subprogram Read_and_Compute;
calls call_sequence: {
  call : subprogram produce_speed;};
connections
  cnx1 : port Pos1 -> produce_speed.Pos1;
  cnx2 : port Pos2 -> produce_speed.Pos2;
  cnx3 : port produce_speed.Output -> Output;
end Speed_Thread.impl;

subprogram Write
features
  Input: in parameter Position;
  Pos1: requires data access Position_Array;
  Pos2: requires data access Position_Array;
annex behavior_specification
{**
  variable
    counter: data Integer {Initial_value => 1};
```

```
  states
    fill_buffer1: initial complete state;
    fill_buffer2: complete state;
  transitions
        fill_buffer1 -[on dispatch Input and counter < 10]->fill_buffer1
                {
                        Input?(Pos1[counter]);
                        counter++;
                };
        fill_buffer1 -[on dispatch Input and counter = 10]->fill_buffer2
                {
                        Input?(Pos1[counter]);
                        counter:=1;
                };
        fill_buffer2 -[on dispatch Input and counter < 10]->fill_buffer2
                {
                        Input?(Pos1[counter]);
                        counter++;
                };
        fill_buffer2 -[on dispatch Input and counter = 10]->fill_buffer1
                {
                        Input?(Pos1[counter]);
                        counter:=1;
                };
**};
end Write;

subprogram Read_and_Compute
features
  Pos1: requires data access Position_Array;
  Pos2: requires data access Position_Array;
  Output: out event data port Speed;
annex behavior_specification
{**
  variable
    b1: data Boolean {Initial_Value => false};
    result: data
  states
    read_buffer1: initial complete state;
    read_buffer2: complete state;
  transitions
        read_buffer1 -[on dispatch Input and b1=true]->read_buffer2
                {
                        Compute_Speed!(Pos1, result);
                        Output!(result);
                        b1:=false;
                };
        read_buffer2 -[on dispatch Input and b1=false]->read_buffer1
                {
                        Compute_Speed!(Pos2, result);
                        Output!(result);
                        b1:=true;
                };
**};
end Read_and_Compute;

subprogram Compute_Speed
features
  Input: in parameter Position_Array;
  Output: out parameter Speed;
end Compute_Speed;
```

1 – Quelle propriété faut-il ajouter au modèle AADL ci-dessus pour pouvoir procéder à une analyse d'ordonnançabilité ? A quelle catégorie de composants AADL cette propriété doit-elle être attachée, et pourquoi ?

2 – En considérant le modèle AADL ci-dessus, expliquer pourquoi il n'est pas nécessaire de protéger les données partagées The_Pos1 et The_Pos2, sous-composants du processus Speed_Process.impl.