TD d’analyse rate monotonic RMA

1.0- Tracer un diagramme temporel d'exécution

1.1-Introduction

Cette technique consiste à vérifier que toutes les tâches respectent leurs échéances en dessinant sur un diagramme temporel la trace (simulation) de l'exécution des tâches en partant du cas pis où elles demandent toutes à s'exécuter simultanément. Il a été démontré que si on trouve une solution a ce cas pis alors le système est ordonnançable.

Ce travail est fastidieux à faire à la main et vite impraticable dès que le nombre de tâches augmente. Le but de ce TD est d'exposer la technique et de l'illustrer sur des exemples simples que l'on vérifie à la main. Il est bien évident que pour un usage professionnel, il y a lieu, soit d'écrire son programme qui construit automatiquement ce graphe en utilisant les règles exposées ci-après, soit d'acheter un produit professionnel qui le fait, voir références a la fin.

1.2- Conditions d'application

Cette technique s'applique à des tâches quelconques dont l'échéance peut dépasser la période et qui peuvent faire intervenir des termes de blocage (synchronisation entre tâches).
L'avantage de cette technique par rapport aux autres plus formelles que nous verrons plus tard est qu'elle donne une vision précise de l'exécution et donc un moyen d'analyse plus claire pour comprendre ce qui se passe. Elle fournit le meilleur guide d'analyse, en cas de détection de non ordonnançabilité, pour évaluer comment essayer d'améliorer les choses.

1.3- Règles de construction du graphe

On construit un tableau ordonné des tâches par ordre de priorité (de la plus forte à la plus faible) dans lequel on met tous les évènements et leurs tâches de traitement associées avec leurs différents paramètres:

ei : identifiant de l'évènement
T : période d'arrivée de cet évènement
ti : identifiant de la tâche de traitement de l'évènement
C : temps d'exécution de la tâche de calcul (cas pis)
P : priorité de la tâche (qui est celle de traitement de l'évènement)
B : délai de blocage de la tâche (on prend le plus long des différents blocages dû à des tâches de plus faible priorité, voir cours pour justification)
D : échéance

Les règles de construction à partir de ces informations constituent un processus itératif décrit ci-après.

Etape 1

Sélectionner la tâche de plus forte priorité avec un terme de blocage non nul

pour laquelle le diagramme n'a pas encore été tracé. Si elle existe, on va tracer un nouveau diagramme en ignorant toutes les tâches de plus faibles priorités. Si elle n'existe pas, on va immédiatement à l'étape 2 en prenant toutes les tâches à la fois.

Etape 2

Tracer le graphe d'exécution pour toutes les tâches à prendre en compte en se plaçant dans l'hypothèse la plus défavorable où elles demandent toutes le CPU simultanément.

Dessiner la trace de l'exécution de toutes les tâches retenues jusqu'à la fin de leur *période d'activité*. La *période d'activité* d'une tâche se définit comme la période qui démarre à la première demande d'exécution et finit dans la première configuration où l'exécution déclenchée se termine avant la fin de la période dans laquelle elle a été initiée (cette période d'activité peut impliquer plusieurs exécutions sur plusieurs périodes d'activation successives).

Dans le graphe temporel on prendra soin de marquer (repérer):
- les dates des demandes d'exécution
- les dates des échéances correspondantes

On fera attention à ce qu'une seule tâche (la plus prioritaire) occupe le CPU à chaque instant.
A chaque itération du processus de construction du graphe **on ne prendra en compte le blocage que pour la tâche la moins prioritaire**. On le fera intervenir comme une prolongation de la durée d'exécution à inclure avant que la tâche ne commence son exécution réelle.

Etape 3

A la fin de chaque itération on vérifie que toutes les tâches ont toujours terminé leur exécution avant leur échéance.

Etape 4

Si le graphe que l'on vient de faire inclut la tâche la moins prioritaire du système, on a fini la vérification. Sinon on repart à l'étape 1 pour tracer un nouveau graphe en incluant de nouvelles tâches.

2.0 - Calculs exacts des temps de réponses

2.1 Cas où toutes les échéances sont inférieures à la période

2.1.1 Formule de base

M. Joseph et P. Pandaya (voir ref) ont développé une analyse exacte de l'ordonnancement étudié par Liu et Layland (RMA) afin d'être en mesure de calculer précisément le délai de terminaison en cas pire de toutes les tâches d'une configuration temps réel entre le moment où cette tâche est invoquée et celui où elle a fini de s'exécuter.

Ils ont démontré que le délai de terminaison de la iéme tâche (par ordre de priorités décroissantes) d'une configuration TR pouvait être calculée par un calcul itératif convergeant vers un point fixe qui est la valeur de ce délai. Ils ont établit la formule récurrente suivante pour la tâche i:

$$R_i^{n+1} = C_i + ? \lceil R_i^n / T_j \rceil C_j$$

$$\forall j \in hp(i)$$

Avec:

R_i^n : délai de terminaison de la f^{éme} tâche à la n^{éme} itération du calcul de ce délai

hp(i) : ensemble des tâches plus prioritaires que la tâche i

La fonction $\lceil x \rceil$ est la «fonction plafond» correspondant au plus petit entier supérieur ou égal à x (5.1 = 6, 5.9 = 6, 5 = 5).

Ci : temps d'exécution de la iéme tâche

Ti : période de la ième tâche

Il a été démontré que ce calcul itératif converge vers (R_i^{n+1}=R_i^n) par valeurs strictement croissantes en **commençant avec n'importe quelle valeur R^0_i inférieure à la valeur finale.**

1.2 Eléments de démonstration

Le délai minimum de terminaison de toute tâche est sa durée d'exécution C_i (dans l'hypothèse où elle est seule).

Mais nous nous intéressons au cas pis de ce délai où cette tâche sera interrompue sans doute plusieurs fois par les tâches plus prioritaires. Il nous faut donc évaluer le temps d'exécution de ces tâches plus prioritaires pendant ce délai en cas pis. Nous connaissons le taux d'occupation moyen du CPU par chacune de ces tâches j = (C_j/T_j). Le temps d'occupation du CPU par une tâche j pendant un délai R est $\lceil R/T_j \rceil C_j$. Le délai de terminaison de la tâche i est donc égale au temps que la tâche a mis à s'exécuter C_i plus le temps pendant lequel les autres tâches plus prioritaires ont occupé le CPU (terme somme du temps d'occupation par chacune d'entre elles dans l'équation).

On démarre donc le calcul du délai d'exécution d'une tâche avec un calcul optimiste dans lequel on ne prend que le temps d'exécution de la tâche seule. Puis on calcule le temps pendant lequel les tâches plus prioritaires auraient occupé le CPU au cours de cette durée. On l'ajoute le temps obtenu au temps d'exécution de la tâche seule et on obtient une première évaluation optimiste du délai d'exécution.

Cette évaluation n'est pas cohérente. En effet on a calculé le temps d'exécution des autres tâches uniquement sur la base d'un délai égal à la durée d'exécution de la tâche i seule alors que dans le résultat final du calcul on doit ajouter le délai d'exécution des autres tâches plus prioritaires. On doit donc recommencer le même calcul en se basant sur l'évaluation optimiste du délai qu'on vient de faire pour le calcul du temps d'exécution des tâches plus prioritaires. On va obtenir un nouveau délai plus proche (toujours de façon optimiste) du délai réel. Les auteurs ont démontré qu'au bout d'un certain nombre d'itérations, ce processus itératif converge vers le délai réel qui, lorsqu'il est utilisé en entrée du

calcul de délai, donne la même valeur de délai en sortie.

On verra dans la méthodologie proposée ci-après qu'on utilise une autre valeur de durée pour démarrer le calcul itératif. En fait il a été démontrer qu'on peut utiliser n'importe quelle valeur pour autant qu'elle soit inférieure à la valeur finale.

2.1.3 Méthodologie (programmable)

2.1.3.1 Etape 1

Calculer la première approximation du délai de terminaison R0 en faisant la somme des durées d'exécution de la tâche i et de toutes les tâches plus prioritaires. Par exemple :

$$R_0 = \sum_{j=1}^{n} C_j$$

2.1.3.2 Etape 2

Calculer l'itération suivante avec la formule de base du paragraphe 2.1.1

2.1.3.3 Etape 3

Analyser le résultat:

Si R_{n+1} (qu'on vient de calculer) <= D_i et $R_{n+1} < R_n$ recommencer à l'étape 2.

Si $R_{n+1} > D_i$ la tâche a loupé son échéance, la configuration n'est donc pas ordonnançable.

Si $R_{n+1} = R_n$ nous avons obtenu le délai de terminaison de la tâche i.

2.1.4 Exemple simple

Soit la configuration de tâches suivante(une priorité est d'autant plus forte que sa valeur est petite).

tâches	T	D	C	prio
A	100	100	40	1
B	150	150	40	2
C	350	350	100	3

On constate que les priorités ont été affectées selon le principe RM.

Evaluons le délai d'exécution de la tâche C la moins prioritaire.

hp(C) contient les tâches A et B.

Etape 1 calcul de la première itération

$$R_0 = \sum_{j=1}^{n} C_j$$

$R_0 = 100 + 40 + 40 = 180$

Etape 2 calcul de la première itération

$$R_i^{n+1} = C_i + \sum_{\forall j \in hp(i)} \lceil R_i^n / T_j \rceil C_j$$

$R_1 = 100 + \lceil 180 / 100 \rceil 40 + \lceil 180 / 100 \rceil 40 = 260$

Etape 3 analyser le résultat

$R_1 <= 350$ et $R_1 != R_0$ alors on réitère l'étape 2.

Etape 2 calcul de la seconde itération

$$R_i^{n+1} = C_i + \sum_{\forall j \in hp(i)} \lceil R_i^n / T_j \rceil C_j$$

$R_2 = 100 + \lceil 260 / 100 \rceil 40 + \lceil 260 / 100 \rceil 40 = 300$

Etape 3 analyser le résultat

$R_2 <= 350$ et $R_2 != R_1$ alors on réitère l'étape 2.

Etape 2 calcul de la troisième itération

$$R_i^{n+1} = C_i + \sum_{\forall j \in hp(i)} \lceil R_i^n / T_j \rceil C_j$$

$R_3 = 100 + \lceil 300 / 100 \rceil 40 + \lceil 300 / 100 \rceil 40 = 300$

Etape 3 analyser le résultat

$R_3 <= 350$ et $R_3 = R_2$

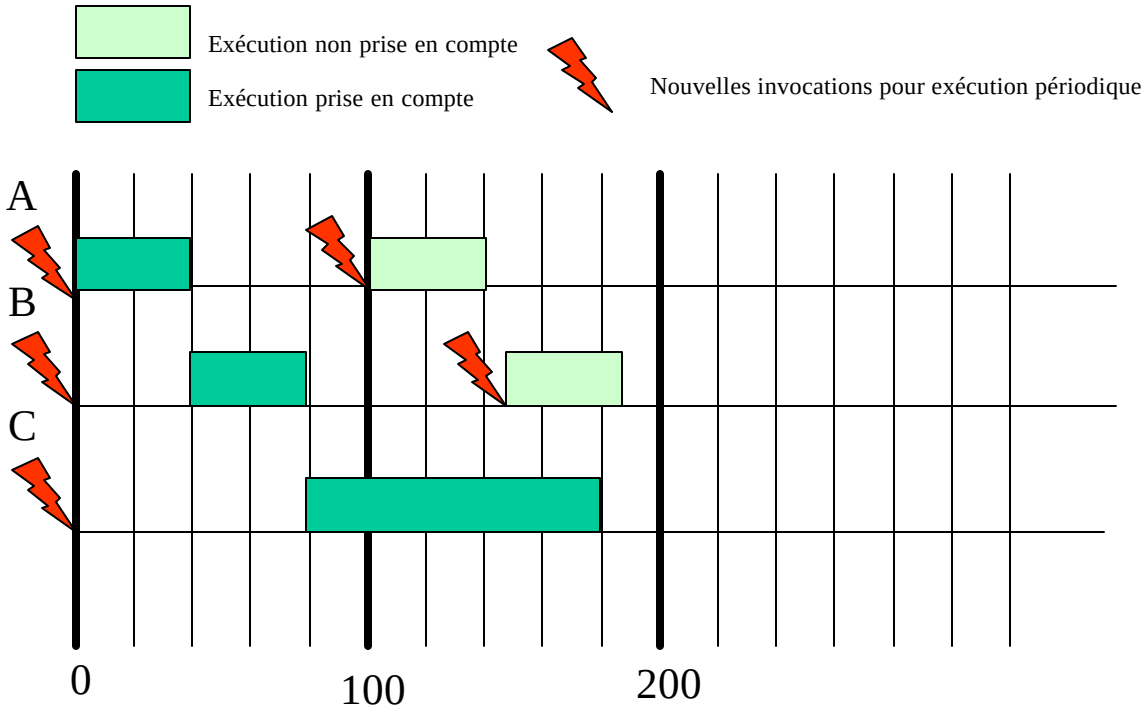
Nous avons atteint le résultat, la durée de terminaison en cas pis est de 300 et est bien inférieure à l'échéance.

ATTENTION: Il ne faut pas en conclure que la configuration est ordonnançable. Il faudrait pour cela vérifier **toutes les tâches** car il est possible qu'une tâche moins prioritaire respecte son délai alors qu'une tâche plus prioritaire ne le fait pas.

2.1.5 Explication graphique

Le calcul de R_0 est la simple somme des temps d'exécution des tâches sans tenir compte du fait que certaines sont exécutées plusieurs fois dans la période d'analyse.

$R_0 = 100 + 40 + 40 = 180$



Calcul de la seconde itération

$R_1 = 100 + \lceil 180 / 100 \rceil 40 + \lceil 180 / 150 \rceil 40 = 260$

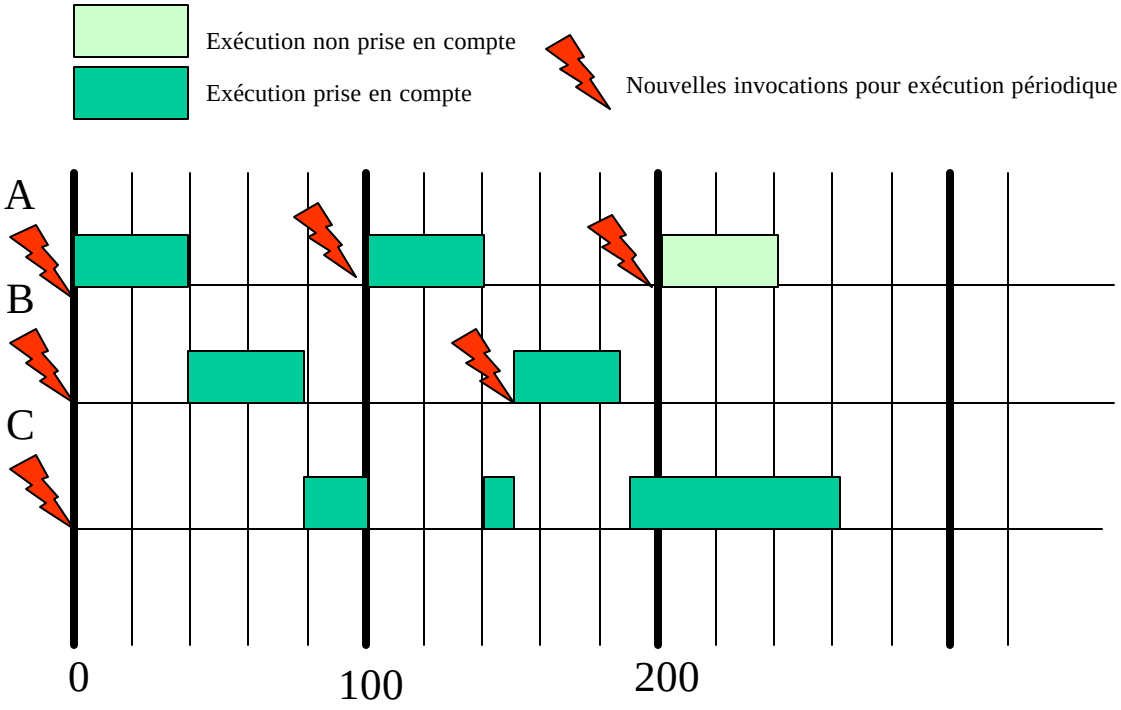
Cette seconde itération tient compte des préemptions A à 100ms et de B à 150ms mais ne tient pas encore compte de celle de A à 200ms. On notera que $40 * 180/200$ représente le temps d'exécution consommé par A entre 0 et 180 ms et que plus généralement $Ci * Rn/Ti$ représente le temps total d'exécution de la tâche i de la date 0 à la date Rn.

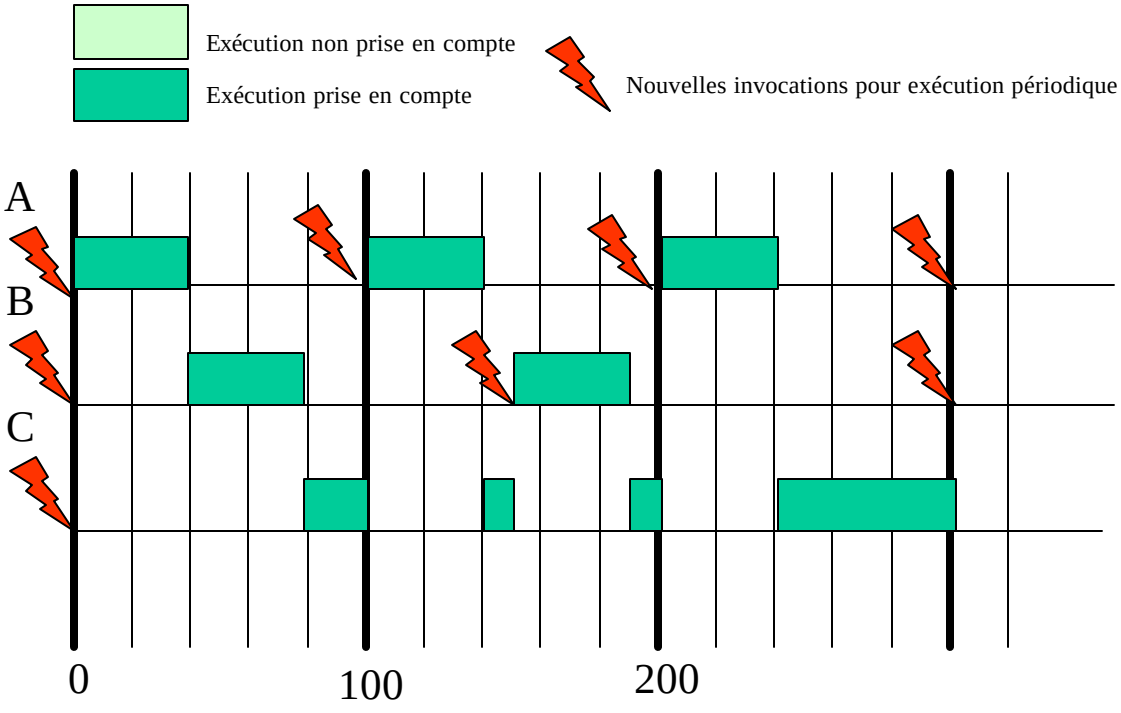
Dans notre cas nous voyons que la période d'exécution de C doit se poursuivre au-delà de 180ms, ont doit donc réitérer.

Troisième et dernière itération

$R_2 = 100 + \lceil 260 / 100 \rceil 40 + \lceil 260 / 150 \rceil 40 = 300$

$R_3 = 100 + \lceil 300 / 100 \rceil 40 + \lceil 300 / 100 \rceil 40 = 300$





Ces dernières itérations prennent en compte la préemption à 200ms et on

constate qu'il n'y en a pas d'autre avant la fin du délai d'exécution de la tâche C.

1.6 Ajout du facteur de blocage (synchronisation entre tâches)

Lorsque les tâches ne sont pas indépendantes il y a des synchronisations (sémaphores) qui introduisent des blocages de celles-ci. Ce phénomène est pris en compte grâce au « facteur de blocage ».

On tient compte du facteur de blocage de la tâche i en l'ajoutant simplement à la formule de calcul comme une prolongation du temps d'exécution de la tâche dont on vérifie la durée d'exécution.

$$R_i^{n+1} = C_i + B_i + \sum_{\forall j \in hp(i)} \lceil R_i^n / T_j \rceil C_j$$

On retiendra que le facteur de blocage B_i est la plus longue durée de blocage de la tâche par une autre tâche de plus faible priorité. Les deux algorithmes vus en cours qui utilisent le concept de plafond de priorité garantissent en effet qu'une tâche ne peut-être bloquée pendant une période d'exécution que par au plus une tâche de plus faible priorité. On se place donc dans le cas pis où ce serait celle qui la bloque le plus longtemps.

3.0 Cas général avec des échéances quelconques et prise en compte du facteur de blocage

Cette technique est une généralisation de la précédente.

3.1 Méthodologie (programmable)

3.1.1 Etape 1

On calcule la première approximation du délai de terminaison.

$$R_0 = B_i + \sum_{j=1}^{n} C_j$$

3.1.2 Etape 2

On définit la *période d'activité* d'une tâche comme la durée qui part de l'invocation de cette tâche, jusqu'à la terminaison de la première de ses exécutions successives qui pourra se terminer avant la fin de la période dans laquelle elle a été demandée. Nous appellerons *traitements* les exécutions successives de la tâche.

On initialise un compteur k qui variera de 1 au nombre de *traitements* qui sont inclus dans la *période d'activité*.

3.1.3 Etape 3

Appliquer la formule itérative ci-dessous étudiée par Tindell et Lehoczky (voir ref) pour calculer le délai de terminaison de la tâche en cas pis.

$$R_i^{n+1} = kC_i + Bi + \sum_{j=1}^{n} \lceil R_i^n / T_j \rceil C_j$$

3.1.4 Etape 4

Analyser le résultat du $k^{éme}$ traitement:

Si $R_{n+1} > ((k-1)*T_i+D_i)$ l'échéance du traitement qu'on vient de calculer ne respecte pas la contrainte d'échéance de la tâche. On arrête le calcul.

Sinon si $R_{n+1} != R_n$ on réitére à l'étape 3

3.1.5 Etape 5

$R_{n+1}=R_n$ on a trouvé la date fin du $k^{ième}$ traitement. Il faut calculer le délai de terminaison de ce $k^{ième}$ traitement.

$$E_{i,k} = R_n - T_i\ (k-i)$$

3.1.6 Etape 6

On regarde si on est à la fin de la période d'activité, c'est à dire si le délai de terminaison de ce traitement est inférieur à la période de la tâche, si:

$$E_{i,k} <= T_i$$

Si ce n'est pas le cas on incrémente k de 1 (passe au traitement suivant) et on reprend à l'étape 3.

Sinon on passe à l'étape suivante.

3.1.7 Etape 7

Calcul pire temps de réponse.

Le pire temps de réponse pour la tâche est la plus grande valeur des $E_{i,b}$ ($1<=b<=k$) calculés a l'étape 5.

3.2 Exemple simple

Soit la configuration de tâches suivante(une priorité est d'autant plus forte que sa valeur est petite).

tâches	T	D	C	prio	B
A	100	100	40	1	0
B	150	160	60	2	20
C	350	350	60	3	0

Calculons le délai d'exécution de la tâche B.

Etape 1 Calcul de la première itération

$$R_0 = B_i + \sum_{j=1}^{n} C_j$$

$R_0 = B_2 + C_1 + C_2 = 20 + 40 + 60 = 120$

Etape 2 Initialisation du compteur

k = 1

Etape 3 Calcul de l'approximation suivante

$$R_i^{n+1} = kC_i + Bi + \sum_{j=1}^{n} \lceil R_i^n / T_j \rceil C_j$$

$R_1 = B_2 + 1*C_2 + R_0/T_1 * C_1 = 20 + 1(60) + 120/100 * 40 = 160$

Etape 4 Vérifier si cette approximation correspond à la fin de la k^{ième} exécution

Non car 120 != 160 , nous devons donc réitérer

Etape 3 Calcul de l'approximation suivante

R2 = 20 + 1(60) + 160/100 *40 = 160

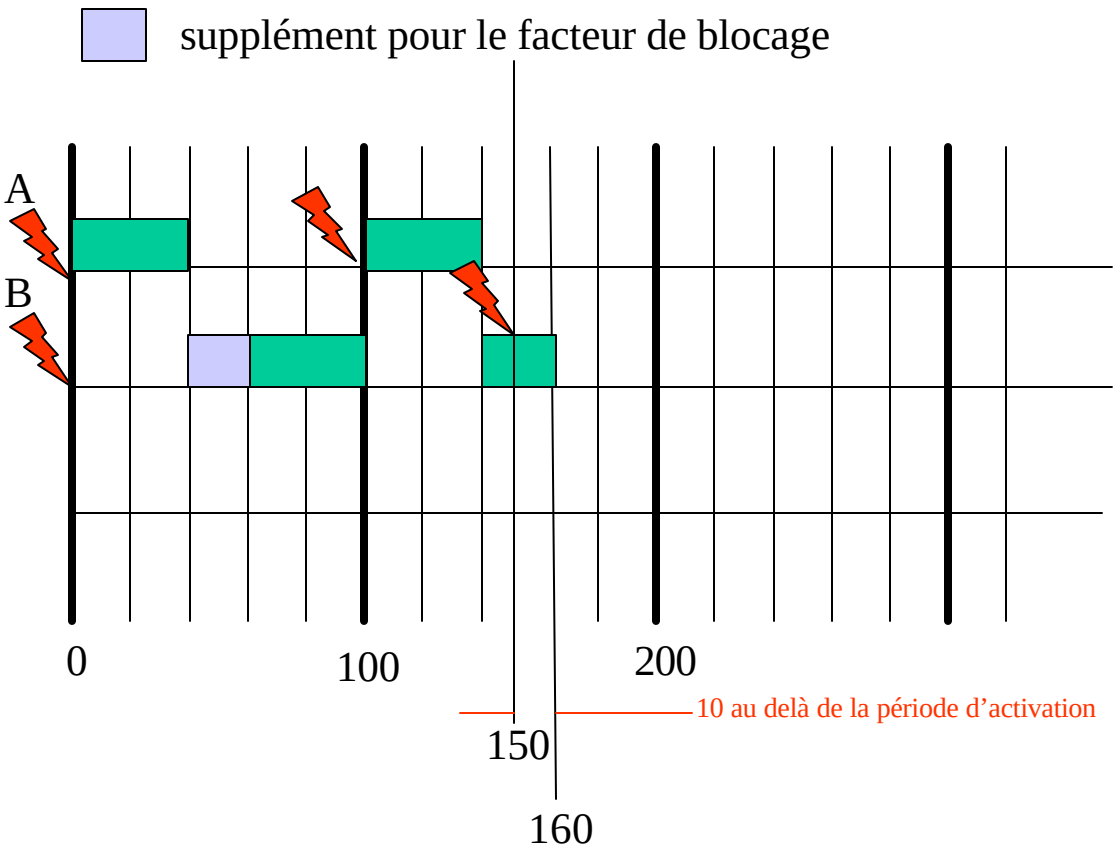
Etape 4 Vérifier si cette approximation correspond à la fin de la k^{ième} exécution

Oui car 160 = 160 , nous passons à l'étape 5

Etape 5 Calcul du temps de réponse de la k^{ième}exécution

$$E_{i,k} = R_n - T_i (k-i)$$

$E_{2,1} = 160 - 150 (0) = 160$ délai d'exécution de la première activation



Etape 6 Test si c'est la fin de la période d'activité

160 est égal à l'échéance donc la contrainte de temps est satisfaite mais la valeur est supérieur à la période de 150, nous ne sommes donc pas a la fin de la période d'activité.

Nous fixons $R_2 = 160 + 60$ et réitérons le calcul.

Etape 3 Calcul de l'approximation suivante

$$R_i^{n+1} = kC_i + Bi + \sum_{j=1}^{n} \lceil R_i^n / T_j \rceil C_j$$

$R_3 = 20 + 2(60) + \lceil 220/100 \rceil * 40 = 260$

Etape 4 Vérifier si cette approximation correspond à la fin de la k^{ième} exécution

Non car 160 != 260 , nous devons donc réitérer

Etape 3 Calcul de l'approximation suivante

R4 = 20 + 2(60) + ⌈260/100⌉ *40 = 260

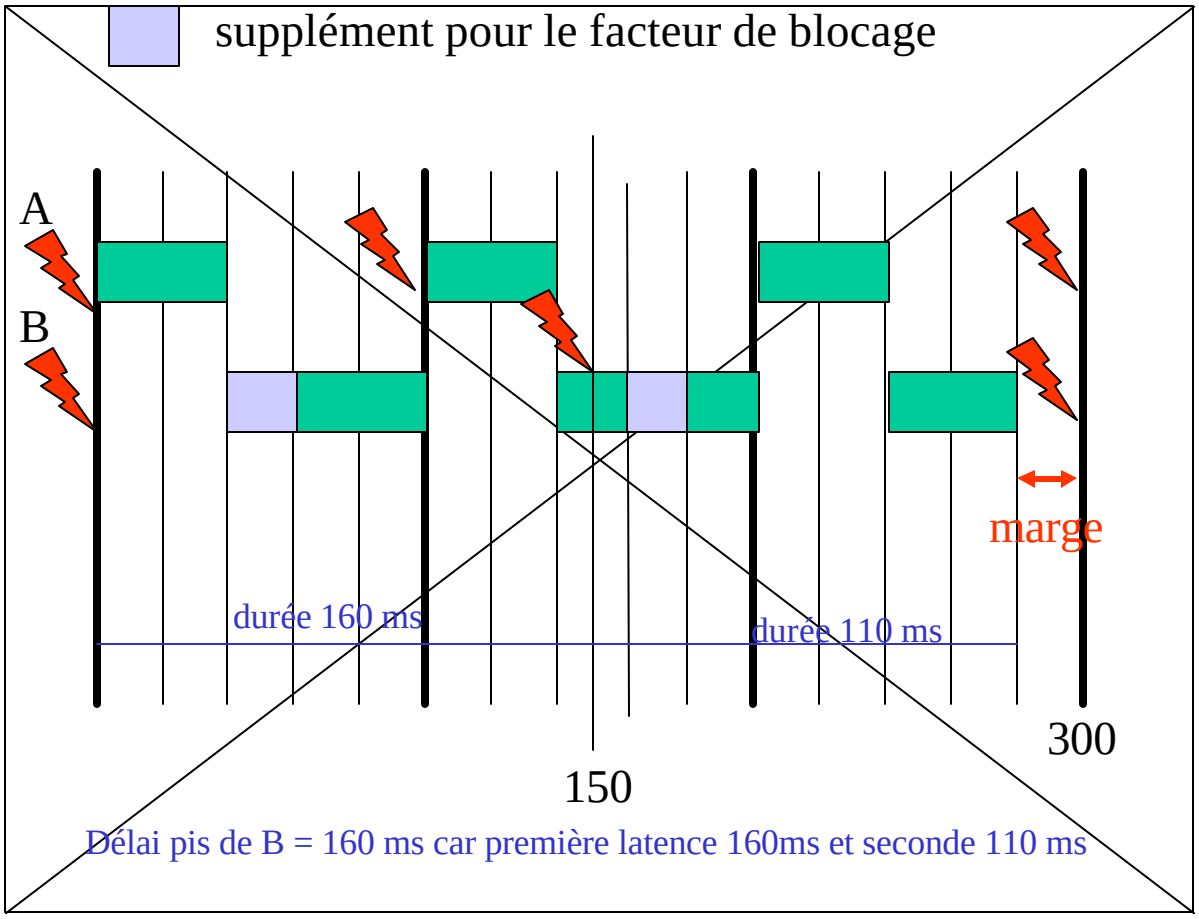
Etape 4 Vérifier si cette approximation correspond à la fin de la k^{ième} exécution

Oui car 260 = 260 , nous passons à l'étape 5

Etape 5 Calcul du temps de réponse de la k^{ième}exécution

$$E_{i,k} = R_n - T_i\,(k-i)$$

$E_{2,2}$ = 260 - 150 (2-1) = 110 délai d'exécution de la seconde activation



Etape 6 Test si c'est la fin de la période d'activité

Comme 110, délai d'exécution, est inférieur à la fin de la période d'activation, nous sommes à la fin de la période d'activité.

Etape 7 Détermination du plus long temps de réponse

Les deux exécutions analysées ont duré 160 et 110 ms. La plus longue de 160 ms est inférieure ou égale à l'échéance. La tâche 2 est donc ordonnançable dans cette configuration.

5.0 Application à l'ordonnancement des messages sur un réseau

Il y a une analogie évidente entre l'allocation du CPU à des tâches en compétition pour un certain temps de calcul et l'allocation d'un réseau à des messages en compétition pour la durée de transmission du message. Dans les réseaux temps réel les messages ont des priorités comme les tâches qui s'exécutent.

Il y a toutefois quelques différences importantes :

- le nombre de priorités disponibles généralement beaucoup plus faible dans un réseau que dans un exécutif temps réel ce qui va obliger plusieurs messages à partager une même priorité.

- la non préemptabilité d'un message implique qu'une fois qu'il a obtenu l'accès au réseau, un message l'occupera jusqu'à la fin de sa transmission même si un message plus prioritaire arrive avant.

Prise en compte de la non préemptabilité avec un niveau de priorité et un seul par message (exemple CAN).

$$R_i^{n+1} = C_i + kB_i + ? \lceil R_i^n / T_j \rceil C_j$$

$$\forall j \in hp(i)$$

C est la durée de transmission du message i

T la période d'émission du message i

B la durée de transmission du plus long des messages dans le système.

Prise en compte en plus de l'usage d'un même niveau de priorité par plusieurs messages (exemple Ethernet voir cours Y.Q. Song).

5.0 Quelques outils d'analyse RMA

5.1 Produits du marché

RAPID RMA de TRI-PACIFIC software, inc (www.tripac.com) distribué en France par Antycip

(info site WEB Price per Single Developer seat: $3,500 Price Floating License seat: $4,900)

TimeWiz de TimeSys (www.timesys.com) distribué en France par Aonix

Basé sur le projet PERTS: A Prototyping Environment for Real-Time Systems (1993)

(info mail Developer License is USD 20,000. avec 80% discount pour les projets de recherche)

Ces outils offrent beaucoup plus que l'analyse RMA qui n'est qu'un de leurs services. En particulier **TimeWiz** est un outils de conception d'architecture avec toutes sortes de possibilités pour allouer des fonctions/tâches á des nœuds interconnectés par des réseaux et pouvoir simuler le fonctionnement du système.

5.2 Produits universitaires et gratuits

XRMA **http://computing.unn.ac.uk/staff/CGWH1/xrma/xrma.html**

gRMA **http://www.tregar.com/gRMA/**

6.0 Sites web utiles

Carnegie Mellon http://computing.unn.ac.uk/staff/CGWH1/xrma/xrma.html

Divers pointeurs http://www.eeglossary.com/rma.htm

7.0 Références

Joseph,M..;&Pandaya,P.*''Finding Response Times in a Real-Time System''* The Computer Journal (British Computing Society) Volume 29, Number 5, pages 309-395, 1986

Klein M.H.; Ralya T.; Pollak B.; Obenza R.; Gonzales Harbour M. *''A Practitioner's Handbook for Real-Time Ananlysis: Guide to Rate Monotonic Analysis for Real-Time Systems''* (Kluwer Academic Publishers)

Lehoczky,J.P.; Sha,L.& Strosnider,J.K. *''Enhanced Aperiodic Scheduling in Hard-Real Time Environement''*,261-270 Proceeding of IEEE Real-Time Systems Symposium, Los Alamitos,CA: IEEE Computer Society Press, 1987

Audsley,N.C.; Burns,A.; Richardson,M.F.; &Wellings,A.J. *''Hard Real Time Scheduling: The Deadline Monotonic Approach''* 133-137 Proceedings- of the 8th IEEE Workshop on Real Time Operating Systems and Software. Oxford; UK: Pergamon, May 1991

8.0 Livres

Ordonnancement temps reel (cours et exercises corriges)

F. Cottet, J. Delacroix, C. Kaiser, Z. Mammeri

Hermes ISBN 2-7462-0099-6

http://www.hermes-sciences.com

A Practitioner's Handbook fir Real-Time Analysis

M. H. Klein, T. Ralya, B. Pollak, R. Obenza, M.G. Harbour

Kluwer Academic Publishers ISBN 0-7923-9361-9