UNITÉ DE RECHERCHE
INRIA-SOPHIA ANTIPOLIS

Institut National
de Recherche
en Informatique
et en Automatique

Sophia Antipolis
B.P. 109
06561 Valbonne Cedex
France
Tél.: 93 65 77 77

Rapports de Recherche

N°1576

Programme 1
Architectures parallèles, Bases de données, Réseaux et Systèmes distribués

PROBLÈMES D'ORDONNANCEMENT D'UNE SUPERPOSITION DE FLUX PÉRIODIQUES SOUS CONTRAINTE TEMPS-RÉEL

Sophie LEFEBVRE-BARBAROUX
Alain JEAN-MARIE
Claudine CHAOUIYA

Janvier 1992

Real-time scheduling of the superposition of periodic flows

Sophie Lefebvre-Barbaroux   Alain Jean-Marie   Claudine Chaouiya
INRIA, Centre de Sophia Antipolis
2004, Route des Lucioles
06565 VALBONNE Cedex, France

December 12, 1991

Abstract

We study in this report the properties of queueing systems with periodic arrivals, under a hard real-time constraint. This constraint specifies that each customer must leave the system before the arrival of the next customer of the same type. It must be satisfied for any possible time lag between the customer types.

The properties of a queue, the arrival process of which is the superposition of deterministic processes are analyzed. We then deduce the conditions under which the real-time constraint is satisfied for several classical service policies: first come first served, preemptive and non-preemptive priorities. The case of a network of stations with a ring topology is also considered. We review the optimality results of the policies "shortest recurrence first" and "next deadline first". Finally, we study how the relative performance of these scheduling policies is modified when scheduling overhead is considered.

Keywords Real-time scheduling, Periodic arrivals, Feasibility.

Problèmes d'ordonnancement d'une superposition de flux périodiques sous contrainte temps-réel

Sophie Lefebvre-Barbaroux Alain Jean-Marie Claudine Chaouiya
INRIA, Centre de Sophia Antipolis
2004, Route des Lucioles
06565 VALBONNE Cedex, France

24 avril 1992

Résumé

Nous étudions dans ce rapport les propriétés de systèmes de files d'attente avec arrivées périodiques, et soumis à une contrainte temps-réel forte. Cette contrainte spécifie que tout client doit avoir quitté le système avant l'arrivée du prochain client du même type. Elle doit être satisfaite quelle que soit la manière dont les différents types sont décalés.

Les propriétés d'une file d'attente dont le processus d'arrivée est la superposition de flux déterministes sont d'abord analysées. Nous en déduisons les conditions sous lesquelles la contrainte temps-réel est respectée pour plusieurs politiques de service classiques: premier arrivé premier servi, politiques à priorités, préemptives ou non. Le cas d'un réseau de stations en anneau est également considéré. Nous revenons sur les résultats d'optimalité des politiques "plus petite récurrence d'abord" et "prochaine échéance d'abord". Enfin nous étudions comment la prise en compte d'un temps de latence ("overhead") modifie les performances relatives des ordonnancements considérés.

Mots-clefs Ordonnancement, Temps-réel, Arrivées périodiques, Faisabilité.

1 Introduction

Ce rapport traite d'ordonnancement de flux déterministes sous contrainte temps réel forte.

Il a été motivé initialement par l'étude d'un modèle pour une architecture à bus slotté, proposé dans le cadre du contrat entre le projet MEVAL de l'INRIA Sophia-Antipolis, et Thomson SINTRA ASM Sophia-Antipolis.

L'étude s'est d'abord axée sur l'analyse des propriétés temps-réel d'une file d'attente avec arrivées déterministes et politique de service PAPS (Premier Arrivé-Premier Servi), en isolation puis en réseau. La politique d'ordonnancement PAPS s'avère peu efficace pour le temps réel. Nous nous sommes alors tournés vers l'étude de politiques plus complexes, s'inspirant de la politique "processor sharing", puis à base de priorités.

Dans cette introduction, nous donnons la description du système qui a servi de base à l'étude. Nous exposons ensuite le modèle de files d'attente utilisé dans l'analyse, et faisons un bref survol de la littérature. Enfin, nous détaillons le contenu du rapport et les principaux résultats obtenus. Ces résultats sont en grande partie extraits d'une thèse de doctorat à paraître [8].

1.1 Motivation

L'architecture proposée par Thomson est un système distribué de traitement du signal. Elle est constituée d'un ensemble de stations (capteurs, unités de traitement) connectées par un medium de communication ("artère") en anneau. Cette artère comporte deux bus unidirectionnels, chacun discrétisé en temps ("slotté"), et l'information se propage sous la forme de "godets" de taille fixe, dans lequel chaque station peut prendre puis mettre des données, sans ralentissement. Dans la version considérée, un godet contient 2 octets et passe en 0,1μs: le débit est donc de 160 Mb/s.

Les flux de messages sont produits par des capteurs digitaliseurs qui fournissent des échantillons à intervalles réguliers ("récurrence"), et par des processeurs fonctionnant de manière synchrone. Le réseau est donc parcouru par des "types" de messages, caractérisés par leur origine, leur destination, leur longueur et leur récurrence. Cet ensemble de types de messages est appelé une *messagerie*. Une station peut émettre des messages de récurrences et de destinations différentes. La discipline de service est "premier arrivé, premier servi".

L'aléatoire du système réside dans le fait que les différentes périodes qui se superposent ont des décalages inconnus et non contrôlables. Ces décalages sont modélisés par le *déphasage à l'origine*. Dans une extension du système, il peut se superposer un flux asynchrone à ce flux déterministe, mais cet aspect du problème ne sera pas abordé ici.

Une caractéristique fondamentale de ce système est l'existence de *contraintes* "temps-réel" (TR) et "espace" (ES) très fortes puisqu'il est interdit de perdre un message:

(TR) un message non encore émis est considéré perdu, car obsolète, à l'arrivée du message suivant de même type;

(ES) un message est perdu si, à son arrivée, il n'y a pas assez de place dans le tampon pour le stocker en entier.

Afin d'assurer une certaine équité dans le partage du canal de communication et garantir le temps de réponse des messages, il est nécessaire de restreindre l'accès des stations. On veut en effet éviter que les stations placées juste derrière une station très émettrice soient "étouffées".

Le mécanisme de contrôle de flux choisi par Thomson est à base de "robinets". A chaque station e est attribuée une proportion de la bande passante maximale utilisable X_e. Le mécanisme des robinets peut également être défini par type de message plutôt que par station. Une manière d'implémenter ce mécanisme est de supposer que les godets sont regroupés en "slots". Chaque slot est constitué d'un certain nombre G de godets, et la station e a le droit d'utiliser au plus $G_e = X_e \times G$ godets dans tout slot. Ce partage du bus s'apparente à un multiplexage par division temporelle (TDMA, pour "Time Division Multiplexing"). Une autre manière d'implémenter ce concept de robinets est de forcer les stations à attendre un certain nombre de godets avant de tenter d'émettre leur information. Par exemple, si la station e doit attendre G_e godets avant de pouvoir émettre N_e godets, elle ne pourra effectivement utiliser au maximum que la proportion $X_e = N_e/(N_e + G_e)$ de la bande passante.

Cet outil a été choisi pour sa simplicité d'implémentation du point de vue pratique. Il utilise en effet peu d'information, et le contrôle est local. Il suffit de calculer et communiquer aux stations les valeurs des robinets avant le démarrage du système. D'autres protocoles plus complexes ont été proposés dans la littérature pour résoudre des problèmes similaires. Ils ont en général besoin de plus d'information sur l'état du système (donc plus de mémoire et plus de temps de calcul pour l'émetteur) et parfois un échange d'information entre stations (donc une réduction de la bande passante utilisable pour le transfert des données).

La proportion X_e allouée à chaque station doit être choisie suffisamment faible pour obtenir l'effet voulu, mais pas trop pour ne pas rallonger excessivement la durée d'émission des messages. A la limite, si X_e est trop petit, la station ne peut plus écouler le trafic offert, et elle devient instable.

Le problème est donc de trouver une valeur des paramètres (valeur des robinets, taille des tampons) qui permette de satisfaire les contraintes (ES) et (TR) et ce indépendamment du déphasage à l'origine. La messagerie sera alors dite *faisable*.

Cette question demande d'abord de caractériser la faisabilité d'une messagerie aux niveaux de la station et du bus. Il s'agit ensuite d'optimiser le système du point de vue du temps de réponse et de la taille des tampons.

1.2 Le modèle

Chaque station est modélisée par une file d'attente dont le processus d'arrivée est une superposition de flux déterministes. Bien que le système soit discret par nature, nous considérerons par commodité, et sans perte de généralité, un modèle en temps continu de chaque station en isolation.

Pour chaque flux (ou type de message) la durée de service demandée (longueur du message) et l'inter-arrivée (récurrence ou période du message) sont connues et constantes. Nous supposerons, également sans perte de généralité, que les récurrences sont un multiple entier d'une certaine unité de temps (le slot). L'unité de longueur choisie est également le slot: les quantités d'information ou de durée sont donc considérées équivalentes. Le slot peut correspondre physiquement au godet, mais aussi à un certain nombre fixé de godets.

Dans le modèle que nous considérons, la date limite d'émission d'un message (ou "deadline") correspond à la date de la prochaine occurence (déterminée par la période).

Les variations infinitésimales et aléatoires des récurrences qui se produisent en pratique sont négligées. On propose donc une étude déterministe du système. Les déphasages pourraient être aléatoires (comme dans [4]), mais le contexte nécessite ici une approche du type "cas le pire": il ne s'agit pas d'estimer une probabilité de perte de message, mais bien de déterminer les conditions pour lesquelles il n'y a pas de perte, et ce pour tout déphasage initial.

Le réseau est vu comme un ensemble de stations reliées en anneau. Les deux bus composant l'artère étant indépendants, ils peuvent être étudiés séparément.

Une station qui est uniquement réceptrice peut être agrégée à la première station émettrice en aval. Ainsi, sur le bus, toutes les stations sont supposées émettrices.

La modélisation de l'architecture de Thomson s'apparente à deux classes de problèmes rencontrés dans la littérature: l'analyse des protocoles pour réseaux à haut débit, et l'ordonnancement de tâches cycliques dans (entre autres) les systèmes de production.

Eckberg [2], Gravey & Hébuterne [4,5] et Roberts & Virtamo [13] étudient pour le protocole ATM un modèle de file d'attente avec superposition de flux déterministes. Tous les messages ont même période et même longueur. Par contre, des hypothèses probabilistes sont faites sur les déphasages, et les performances (temps d'attente, probabilités d'occupation) sont calculées.

En ce qui concerne le contrôle de l'accès de stations au réseau, notons que plusieurs protocoles (et de nombreux modèles) ont été proposés dans la littérature pour les (futurs) réseaux à haut débit. Citons par exemple le protocole FDDI [1,7,14] et le protocole DQDB [3,11].

Dans le domaine de l'ordonnancement, notre modèle est semblable à celui étudié par Liu et Layland [10]. Pour un serveur soumis à des arrivées périodiques, il s'agit en effet de caractériser la faisabilité de politiques d'ordonnancement, et même de trouver les politiques "optimales". Le critère d'optimalité est le suivant: un ordonnancement est meilleur qu'un autre, si il rend faisable un plus grand nombre de messageries.

On rencontre dans la littérature des modèles plus généraux où chaque tâche a une date limite ("deadline"). Il s'avère que l'hypothèse que la date limite d'émission de chaque message correspond justement à sa période est importante [9]. C'est elle qui permet en effet de caractériser simplement la faisabilité (par le taux de charge du serveur) pour la politique optimale "première échéance d'abord".

1.3 Notre étude

Nous commençons par étudier dans la section 2 le cas d'une station en isolation soumise à des arrivées périodiques, la seule hypothèse sur la politique de service étant qu'elle est conservatrice. Nous obtenons des résultats sur la périodicité et la borne en trajectoire du processus de charge.

Dans la section 3, nous considérons que la politique de service est PAPS. Les résultats précédents induisent une condition nécessaire et suffisante pour satisfaire la contrainte temps réel pour une station isolée. Cette condition s'écrit: $\Lambda < r$, où Λ est la somme des longueurs des différents types de messages, et r est la plus petite des récurrences. Ce critère est très simple à utiliser, mais malheureusement rarement vérifié dans la pratique.

Ce résultat permet d'obtenir ensuite une condition suffisante de faisabilité pour le bus complet, encore plus contraignante (les échantillons de messageries procurés de Thomson ne la satisfont pas). Nous avons donc cherché à modifier certaines données du problème, et avons étudié la possibilité de rendre une messagerie faisable en modifiant les récurrences. Nous avons également étudié l'allocation de robinet par type de messages, et un nouveau moyen de contrôle d'accès au bus par des "multi-robinets".

Ce dernier mécanisme permet de satisfaire la contrainte temps-réel, à partir de l'hypothèse minimale de stabilité du système. Son principe s'inspire du multiplexage: chaque station est décomposée en plusieurs stations *fictives*, dont l'accès est toujours contrôlé par un robinet. Les flux de messages sont alors redistribués en séparant les messages très récurrents des messages longs. Les performances de cette méthode sont très satisfaisantes, puisque la simple condition de stabilité suffit pour assurer l'existence de multi-robinets rendant faisable une messagerie quelconque sur le bus complet. Nous présentons un algorithme permettant de calculer des valeurs des multi-robinets.

Le critère de faisabilité dans le cadre d'une station en isolation avec une discipline PAPS étant très contraignant, nous nous sommes intéressés à d'autres politiques de priorités.

La section 4 présente le modèle de Liu et Layland [10] mentionné plus haut. Les performances de divers ordonnancements y sont présentées, pour une station en isolation. Les conditions de faisabilité obtenues impliquent l'observation de trajectoires particulières ou bien un test sur le taux de charge du serveur. Comme nous avons essayé d'étendre ce type de condition à des cas plus généraux, nous avons été amenés à les préciser. Ainsi nous reprenons en détail certains résultats de l'article de Liu et Layland.

Ces résultats permettent de déduire une échelle de valeur montrant que la politique "première échéance d'abord" est optimale puisqu'elle rend faisable toute messagerie vérifiant simplement la condition de stabilité. D'une manière générale, les priorités dynamiques sont supérieures aux priorités statiques; de même les priorités préemptives sont supérieures aux priorités non préemptives.

Dans l'architecture de Thomson, il s'avère que le passage d'un message à un autre dans le serveur coûte un temps appelé *overhead*, correspondant à l'établissement d'une nouvelle communication. Pendant ce temps le serveur ne travaille pas, il y a donc un gaspillage de la bande passante du bus.

Cette notion d'overhead est rarement modélisée, bien qu'existant en pratique dans tous les systèmes informatiques. Dans un environnement multiprocesseurs à mémoire partagée, l'overhead correspond au temps nécessaire au processeur pour aller lire en mémoire les données sur lesquelles le programme en attente d'exécution travaillera.

La prise en compte de l'overhead bouleverse les résultats établis. Par exemple, les multi-robinets nécessitent beaucoup de changements de messages occupant le serveur, et donc de nombreux établissements de connexions qui deviennent coûteux.

Les ordonnancements les plus efficaces dans le cas sans overhead sont ceux qui demandent le plus de prises de décision. Ils s'avèrent très coûteux en temps de calcul quand l'overhead est pris en compte, et peuvent redonner l'avantage aux ordonnancements plus "rustiques", à savoir PAPS et priorités statiques non préemptives. Ces derniers conservent l'atout de la simplicité d'implémentation, ce qui n'est pas négligeable en pratique, pour les systèmes de transmission rapide auxquels nous nous intéressons.

Le problème se pose donc de trouver pour une messagerie donnée, un ordonnancement qui la rende faisable, si possible optimal, dans le cas *avec overhead*.

Dans la section 5, nous revenons à l'étude de politiques d'ordonnancement dans une station isolée, en tenant compte de l'existence d'un overhead. Nous avons obtenu des critères de faisabilité dans le cas PAPS et dans le cas non préemptif, qui étendent à double titre les résultats de Liu et Layland qui ne concernaient que les priorités préemptives et sans overhead.

2 Etude d'une station soumise à des arrivées périodiques

On étudie dans cette section le modèle d'une file d'attente dont le serveur travaille en continu à la vitesse 1, et dont le processus d'arrivée est une superposition de M flux déterministes. Nous étudions la périodicité du système et donnons des bornes sur la quantité d'information présente dans le système (maximale et moyenne) ainsi que sur la longueur des périodes d'activité.

Pour chaque flux $i \in I = \{1, .., M\}$, soit:

- L_i la demande constante de service (c'est-à-dire la durée d'émission pour un message, ou encore la durée d'exécution pour une tâche),
- R_i sa période, supposée entière,
- ϕ_i la date de la première arrivée (ou: *déphasage initial*).

On notera $\underline{L}$ le vecteur $(L_1, \ldots, L_M)$, et $\underline{R} = (R_1, \ldots, R_M)$. Le couple $(\underline{L}, \underline{R})$ définit une "messagerie" de taille M qui sera notée $\mathcal{M}$. De plus, soit:

- T le plus petit commun multiple des récurrences. Cet entier est aussi la période du processus d'arrivée,
- $N_i = T/R_i$ le nombre d'arrivées de type i dans une période,
- $S(a, b)$, la quantité d'information à émettre, arrivée dans l'intervalle de temps $[a, b[$,
- $S = S(0, T) = \sum_{i \in I} N_i L_i$, la quantité d'information à émettre par période.
- $\lambda = \sum_{i \in I} (L_i / R_i) = S/T$, la charge moyenne à émettre par unité de temps,
- $\Lambda = \sum_{i \in I} L_i$, la quantité maximale d'information à émettre pouvant arriver instantanément à la station (*i.e.* quand tous les messages ont une occurrence simultanée),

2.1 Périodicité

Dans ce paragraphe, nous étudions la périodicité de W, processus de charge d'une file d'attente, défini par: $W(t)$ est la quantité d'information en attente d'émission à l'instant t^-. La propriété suivante donne une caractéristique du comportement de ce processus. Rappelons qu'une politique de service est dite *conservatrice* si elle ne rend jamais le serveur inactif tant qu'il y a des clients dans la file d'attente.

Proposition 2.1 *Pour toute politique de service conservatrice, W vérifie l'équation fonctionnelle suivante,*

$$b + W(b) = a + W(a) + S(a, b) + \int_a^b \mathbf{1}_{\{W(u)=0\}} du \ , \qquad \forall\ b \geq a \geq 0 \ . \tag{1}$$

Démonstration

La charge à un instant b est égale à la charge à un instant a ($a \leq b$), augmentée de la quantité d'information arrivée dans l'intervalle $[a, b[$, diminuée de celle émise dans le même intervalle. Puisque la vitesse de service est égale à 1, et que la politique est conservatrice, la quantité émise est égale à la durée d'activité du serveur, c'est-à-dire la différence entre la durée totale $b - a$ et le temps où le serveur est vide.

On peut aussi obtenir l'équation fonctionnelle en projetant la partie de la courbe de charge correspondant à l'intervalle $[a, b[$ sur l'axe des temps parallèlement à la deuxième diagonale (figure 1).

#

Les résultats qui suivent sont essentiellement des conséquences de cette proposition 2.1. Ils s'appliquent donc à toute politique conservatrice.

Corollaire 2.2 *W est une fonction continue à gauche.*

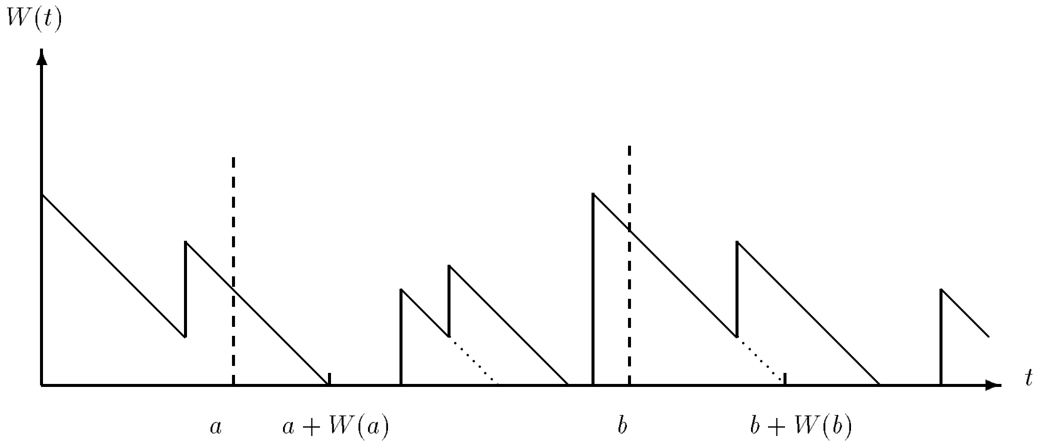


Figure 1 : Courbe de charge $W(t)$

Démonstration Dans l'équation fonctionnelle, la partie intégrale est une fonction continue de b, tandis que $S(a,.)$ est une fonction en escalier, croissante, qui n'est continue qu'à gauche, d'où le résultat. #

Nous introduisons maintenant la définition de la stabilité du système. Une notion couramment utilisée est celle de la finitude du temps d'attente (ou de la longueur de la file). Les résultats de la section 5 nécessitent un sens plus fort de la stabilité, assurant le passage par zéro de la charge de la file.

Définition 2.3 *Une messagerie $\mathcal{M}$ est "stable" pour une discipline de service, si et seulement si quel que soit l'état initial et l'instant t choisis, la file (partant de cet état initial et suivant cette discipline) se vide à un instant fini après t:*

$$\forall W(0),\ \forall \underline{\phi},\ \forall t \geq 0, \quad \exists t' > t,\ t' < \infty \ \ tq \ \ W(t') = 0.$$

L'équation fonctionnelle donne un premier résultat de couplage, qui sera utile pour montrer que le processus de charge est périodique à partir d'un certain temps.

Lemme 2.4 *On considère deux processus de charge W_1 et W_2 avec même processus d'arrivée vérifiant $\lambda < 1$, alors ils coïncident à partir d'un certain instant:*

- *Si $W_1(0)=W_2(0)$, alors $W_1(t)=W_2(t)\ \ \forall t \geq 0$.*
- *Si $W_1(0) > W_2(0)$, alors $W_1(t) > W_2(t)\ \ \forall t < \tau_0$ et $W_1(t) = W_2(t)\ \ \forall t \geq \tau_0$ où l'on note $\tau_0 = inf\{t \mid W_1(t) = 0\}$.*

Démonstration

On définit $V = W_1 - W_2$, et $X = \mathbf{1}_{\{W_1=0\}} - \mathbf{1}_{\{W_2=0\}}$, alors en appliquant la proposition 2.1, on obtient une équation fonctionnelle vérifiée par V: $V(t) = V(0) + \int_0^t X(u)du$, d'où l'on déduit que V est continu sur $\mathbb{R}^+$.

Pour la première assertion, $V(0)=0$, on raisonne par l'absurde en supposant que V n'est pas identiquement nul.

Alors par continuité de V, $\exists t_0 \geq 0,\ a > 0$ vérifiant $V(t_0) = 0$ et $V \neq 0$ sur l'intervalle $]t_0, t_0 + a]$.

Sur cet intervalle, V est donc de signe constant, disons positif sans perte de généralité), on a ainsi $W_1 > W_2 \geq 0$ donc $W_1 > 0$ et $X = -\mathbf{1}_{\{W_2=0\}} \leq 0$, d'où l'on tire $V(t_0+a) = V(t_0+a) - V(t_0) = \int_{t_0}^{t_0+a} X(u)du \leq 0$, ce qui est contradictoire avec $V(t_0 + a) > 0$.

Finalement V est identiquement nul sur $\mathbb{R}^+$, *i.e* $W_1(t) = W_2(t) \ \ \forall t \geq 0$.

Pour la seconde assertion, montrons d'abord que $\tau_0 = inf\{t \mid W_1(t) = 0\}$ est fini. En effet si $W_1 > 0$ sur $[0, kT[$ avec $k \in \mathbb{N}$, alors $\int_0^{kT} \mathbf{1}_{\{W_1(u)=0\}} du = 0$. En utilisant la proposition 2.1, on obtient l'égalité: $kT + W_1(kT) = W_1(0) + kS$. A l'aide de l'hypothèse $\lambda < 1$ *i.e.* $S < T$, on déduit une borne pour k: $k \leq W_1(0)/(T-S)$, puisque $W_1(kT) \geq 0$. La contraposée prouve que $\tau_0 \leq \lceil W_1(0)/(T-S) \rceil T$ où $\lceil x \rceil$ est le plus petit entier supérieur à x.

Dans ce cas, $V(0) > 0$, et on note $b = inf\{t \mid V(t) \leq 0\}$. Alors par continuité de V on a: $V > 0$ sur $[0, b[$ et $V(b) = 0$, d'où l'on déduit tout d'abord, grâce à la première assertion que V est identiquement nul sur $[b, +\infty[$, et ensuite que sur $[0, b[$ on a $W_1 > W_2 \geq 0$ et donc $b \leq \tau_0$.

Montrons maintenant que $\tau_0 \leq b$ grâce à un raisonnement par l'absurde. Supposons $W_1(b) > 0$, alors puisque $V(b) = 0$ on a $W_2(b) > 0$, et par continuité à gauche des W_i, on sait que $\exists a > 0$ tel que $W_i > 0$ sur l'intervalle $[b-a, b]$ et donc X est nul sur cet intervalle. On en déduit que $V(b-a) = V(b-a) - V(b) = \int_b^{b-a} X(u) du = 0$, ce qui est contradictoire avec la définition de b. Finalement $W_1(b) = 0$ et donc $b \geq \tau_0$.

On en déduit que $b = \tau_0$, ce qui prouve la deuxième assertion. #

Quant à la localisation du couplage, nous avons obtenu un encadrement du numéro de la période où il commence.

Corollaire 2.5 *Supposons que $\lambda < 1$ (i.e. $S < T$). Si l'on note $\tau_0 = inf\{t \mid W(t) = 0\}$ et $n_0 = \lfloor \tau_0/T \rfloor$ (où l'on note $\lfloor x \rfloor$ la partie entière de x), alors on peut encadrer n_0 (et donc τ_0):*

$$\frac{W(0) - T}{T - S} < n_0 \leq \frac{W(0)}{T - S}$$

Démonstration Le majorant a déjà été obtenu dans la démonstration de la deuxième assertion du lemme 2.4. Pour le minorant, on applique à l'instant τ_0 l'équation intégrale de la proposition 2.1 : $\tau_0 = W(0) + S(0, \tau_0)$, en remarquant que l'on a: $S(0, \tau_0) \geq n_0 S$ et $\tau_0 < (n_0 + 1)T$. #

Remarque 2.6 Cet encadrement ne peut être réduit, comme le montre l' exemple illustré par la figure 2. Le flux d'arrivée est réduit à un seul type de message, de longueur 2 et de période 3 et la charge initiale est $W(0) = 5$. On obtient pour n_0 les valeurs 3, 4 ou 5, qui correspondent à un déphasage initial a qui vaut respectivement 2, 1 ou 0. L'encadrement obtenu par le corollaire 2.5 est $2 < n_0 \leq 5$.

Les deux théorèmes qui suivent décrivent le régime "stationnaire" de la file, suivant le facteur de charge λ.

Théorème 2.7 *Si $\lambda < 1$, W est périodique, de période T, à partir d'un temps fini τ déterminé suivant les cas:*

- *surcharge: si $W(0) > W(T)$, alors $\tau = inf\{t \mid W(t) = 0\}$,*
- *équicharge: si $W(0) = W(T)$, alors $\tau = 0$,*
- *sous-charge: si $W(0) < W(T)$, alors $\tau = -T + inf\{t \geq T \mid W(t) = 0\}$.*

La figure 3 illustre ces trois cas, avec la valeur de τ pour chacun.

Démonstration On note $W_1 = W(. + T)$, alors W et W_1 ont le même processus d'arrivée, par T-périodicité de celui-ci. La première assertion du lemme règle le cas d'équicharge, tandis que la deuxième règle les cas de surcharge et sous-charge en inversant les rôles de W et W_1. Pour l'expression de τ dans le cas de sous-charge, on remarque que $inf\{t \mid W(t + T) = 0\} = inf\{t \geq T \mid W(t) = 0\} - T$.

#

Dans le cas $\lambda \geq 1$, le régime stationnaire peut dépendre de la charge initiale. Quand $\lambda > 1$, la file est en surcharge. On observe alors une "pseudo périodicité" avec croissance linéaire de la charge.

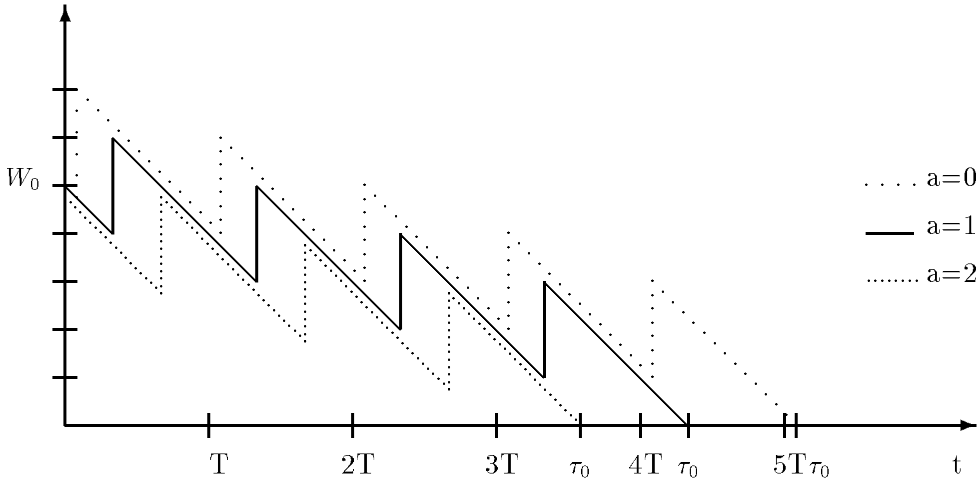


Figure 2 : Exemple de valeurs de τ_0

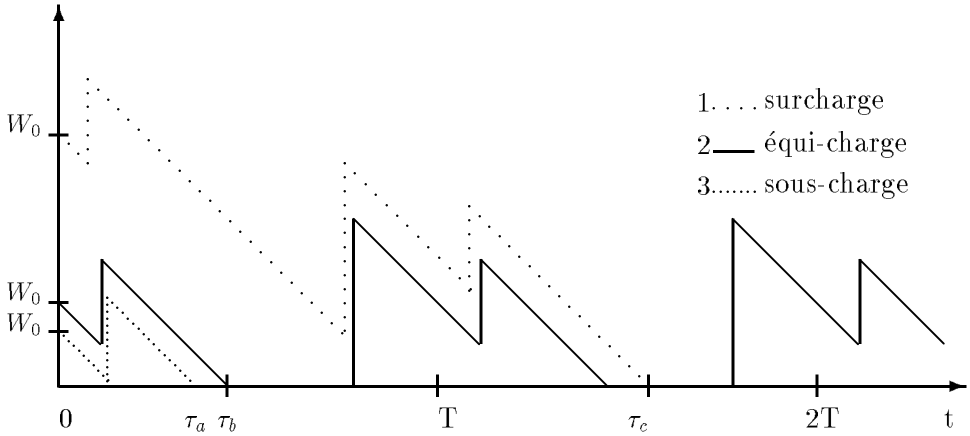


Figure 3 : Exemple de valeurs de τ

Théorème 2.8 *Si $\lambda \geq 1$, alors pour tout $W(0) > 0$, il existe un temps fini $\tau < T$ tel que: $\forall t \geq \tau$,*

$$W(t+T) = W(t) + (\lambda - 1)T \ .$$

Démonstration En appliquant l'équation fonctionelle (1), on obtient que pour tout t, $0 \leq t < T$ et pour tout entier $k \geq 1$,

$$W(t+kT) = W(t) + kS - kT + \int_t^{t+kT} \mathbf{1}_{\{W(u)=0\}} du \ .$$

- Dans le cas $\lambda > 1$, on a $S > T$ et donc: $W(t+kT) \geq W(t) + k(S-T) \geq kT(\lambda-1) > 0$. En particulier, si $t \geq T$, $W(t) > 0$ et par conséquent, pour tout $t \geq T$, $\int_T^t \mathbf{1}_{\{W(u)=0\}} du = 0$.

  Ceci implique que pour tout t, $0 \leq t < T$ et $k \geq 1$:

  $$W(t+kT+T) = W(t) + (k+1)T(\lambda-1) + \int_t^T \mathbf{1}_{\{W(u)=0\}} du = W(t+kT) + T(\lambda-1) \ ,$$

  d'ou le résultat.

- Dans le cas $\lambda = 1$, on a pour tout t:

  $$W(t+T) = W(t) + \int_t^{t+T} \mathbf{1}_{\{W(u)=0\}} du \ . \qquad (2)$$

  Prouvons par l'absurde que $W(2T) = W(T)$, c'est-à-dire que $\int_T^{2T} \mathbf{1}_{\{W(u)=0\}} du = 0$. Par changement de l'origine des temps (comme dans le lemme 2.4), on en déduira facilement la périodicité du système.

  Supposons donc que $\int_T^{2T} \mathbf{1}_{\{W(u)=0\}} du > 0$. Ceci implique, parce que W est continue à gauche, qu'il existe $t < t'$ dans $[T, 2T]$ tels que $\forall x \in]t, t'[$, $W(x) = 0$. Dans l'équation (2), $W(t+T)$ est la somme de deux termes positifs. Si cette somme est nulle, les deux termes sont nuls aussi. On a donc: $\forall y \in]t-T, t'-T[$, $W(y) = 0$ et $\int_y^{y+T} \mathbf{1}_{\{W(u)=0\}} du = 0$.

  Or, $\int_y^{y+T} \mathbf{1}_{\{W(u)=0\}} du = \int_y^{t'-T} dy + \int_{t'-T}^{y+T} \mathbf{1}_{\{W(u)=0\}} du > 0$, et il y a donc contradiction.

Dans les deux cas, la date τ annoncée est donnée par $\tau = inf\{t | W(t+T) = W(t) + (\lambda-1)T\}$, qui est aussi $inf\{t | W(u) > 0, \ \forall u > t\}$ si $\lambda > 1$.

#

On déduit facilement des résultats de périodicité le lemme suivant, utile dans la suite:

Proposition 2.9 *Si $\lambda \leq 1$, alors*

$$\forall t > \tau, \qquad \int_t^{t+T} \mathbf{1}_{\{W(u)=0\}} du = T - S \ .$$

Démonstration Ce résultat s'obtient directement en appliquant l'équation fonctionnelle (1) aux instants t et $t+T$, en remarquant que par définition, $t > \tau$ implique $W(t) = W(t+T)$. #

A partir de ces résultats, on pourrait envisager d'utiliser une simulation pour connaître le régime stationnaire de W. Il faudrait alors se demander comment atteindre et/ou identifier τ le plus tôt possible en jouant sur le choix de la valeur $W(0)$. Cette question est cruciale quand la période T est très grande.

D'après le théorème 2.7, dans les cas d'équi- ou sous-charge, on a $\tau \leq T$. Pour déterminer τ il faut connaître $W(T)$ et parfois $W(t)$ sur $]T, 2T]$. Cela peut demander une longue observation et la mémorisation de nombreuses valeurs.

Par contre, dans le cas de surcharge, le critère de détermination de τ est simple, son identification est immédiate, et ne nécessite aucune mémorisation. Par ailleurs, le corollaire 2.5 montre que la majoration de τ décroît avec $W(0)$. Le meilleur choix de $W(0)$ est donc la plus petite valeur assurant la surcharge, c'est-à-dire Λ (*cf.* théorème 2.13).

On déduit aisément des théorèmes 2.7 et 2.8 une condition nécessaire de stabilité des messageries:

Corollaire 2.10 *La messagerie $\mathcal{M}$ est stable si et seulement si:*

$$\lambda < 1 . \tag{3}$$

Extension des résultats: La proposition 2.1 et son premier corollaire sont valables quel que soit le processus d'arrivée déterministe. Le corollaire 2.5 n'utilise que le fait que la quantité $S(nT, nT+T)$ est indépendante de n: cette hypothèse est plus large que la stricte périodicité du processus d'arrivée. Seuls les théorèmes 2.7 et 2.8 utilisent pleinement cette hypothèse.

2.2 Bornes

Dans ce paragraphe, nous donnons une borne supérieure de $W(t)$. Nous complétons ce résultat par une étude de la longueur des périodes d'activité du serveur et des bornes en moyenne du processus de charge. Dans ce paragraphe, la file sera supposée stable, i.e. $\lambda < 1$.

Définition 2.11 *Une* période d'activité *du serveur est définie comme étant un intervalle de temps $[A, B]$ tel que: $\int_A^B \mathbf{1}_{\{W(u)=0\}} du = 0$, et tel qu'aucun autre intervalle le contenant ne satisfait cette propriété.*

Cela correspond bien à la notion classique de période d'activité (période ou la file ne se vide pas), à la nuance près que les passages de $W(t)$ par 0 qui coïncident avec une arrivée (c'est-à-dire tels que $W(t^+) > 0$) n'interrompent pas une période d'activité. Cela revient à dire que les périodes d'inactivité n'ont jamais une longueur nulle.

Lemme 2.12 *Pout tous t et w positifs,*

$$\forall t \qquad S(t, t+w) \in [s(w), s(w) + \Lambda],$$

où

$$s(w) = \sum_{i \in I} \lfloor w/R_i \rfloor L_i .$$

Démonstration Si l'on note pour chaque flux $n_i = \lfloor w/R_i \rfloor$, alors la contribution de ce flux à la charge totale arrivée dans un intervalle de temps de largeur w est $n_i L_i$ ou $(n_i + 1) L_i$. Comme $\Lambda = \sum_{i \in I} L_i$, on obtient l'encadrement annoncé et la valeur de $s(w)$. #

Théorème 2.13 *La quantité $\Lambda = \sum_{i \in I} L_i$ est une borne supérieure du processus $W(t)$, sur la trajectoire après l'instant τ.*

De plus, si $W(0) = 0$ alors nous avons $W(t) \leq \Lambda$ pour tout $t \geq 0$.

Démonstration Montrons d'abord **(*)**: si t_0 vérifie $W(t_0) = 0$ alors $W(t) \leq \Lambda$ pour tout $t \geq t_0$.

Si $W(t) = 0$, la majoration est triviale. Sinon, $W(t) > 0$, donc t est dans une période d'activité démarrant à un instant $a \geq t_0$. Ainsi $W(a) = 0$, et $W > 0$ sur l'intervalle $]a, t]$. L'équation fonctionnelle (1) donne: $W(t) = S(a, t) - (t - a)$. Or $S(a, t) \leq s(t - a) + \Lambda$, d'après le lemme 2.12. Il ne reste plus qu'à montrer que: $s(w) \leq w \;\; \forall w$. Pour cela on majore $\lfloor w/R_i \rfloor$ par w/R_i dans l'expression explicite de $s(w)$, pour obtenir $s(w) \leq \lambda w$. Finalement, $W(t) \leq (\lambda - 1)(t - a) + \Lambda$, avec $\lambda < 1$ par hypothèse, et $t \geq a$, implique $W(t) \leq \Lambda$.

- le cas $W(0) = 0$ est réglé en considérant $t_0 = 0$ dans **(*)**.
- dans les cas de surcharge et sous-charge, d'après le théorème 2.7, $W(\tau) = 0$. En appliquant **(*)** avec $t_0 = \tau$, on obtient le résultat voulu.
- dans le cas d'équicharge, $\tau = 0$ donc la proposition 2.9 permet d'affirmer qu'il existe un $t_0 \in [0, T]$ tel que $W(t_0) = 0$. Le résultat **(*)** donne alors $W(t) \leq \Lambda$, pour $t \geq t_0$. Pour $t < t_0 < T$, $\quad t + T \geq t_0$ et, par périodicité de W, $W(t) = W(t+T) \leq \Lambda$.

Le dernier cas peut aussi se déduire du précédent qui permet de borner le régime stationnaire de W. #

Remarque 2.14 L'inégalité $W(t) \leq \Lambda$ est clairement satisfaite si il n'y a pas plus d'un message de chaque type dans la station, ce qui est le cas si la contrainte temps-réel est satisfaite. Il est important de remarquer que même dans le cas contraire, cette inégalité est toujours satisfaite.

Corollaire 2.15 *Soit $t \geq \tau$, alors $W(t^+) = \Lambda$ si et seulement si t est une occurrence simultanée de tous les types de messages.*

Démonstration On rappelle que $W(t^+)$ est la charge à l'instant t^+, *i.e.* la somme de $W(t)$ et de la charge arrivée à l'instant t.

Si t est une occurrence simultanée de tous les types de messages, alors $W(t^+) = W(t) + \Lambda \geq \Lambda$. Le théorème 2.13 donne l'inégalité inverse, d'où les égalités: $W(t) = 0$ et $W(t^+) = \Lambda$.

Inversement, si l'on suppose $W(t^+) = \Lambda$, alors en reprenant la démonstration du théorème 2.13, on obtient $s(t-a) = t-a$ avec $s(t-a) \leq \lambda(t-a)$ et $\lambda < 1$, ce qui impose $t - a = 0$, *i.e.* t est le début de la période d'activité. Ainsi $W(t) = 0$, et donc la charge arrivée à t est Λ, ce qui ne se produit que lors d'une occurrence simultanée de tous les messages. #

Les tests de faisabilité présentés dans la section 5.3 étant basés sur l'observation de périodes d'activité (ou d'occupation) du serveur, il sera intéressant d'avoir une majoration de leur durée.

Une première borne facile à obtenir est S, puisque c'est la quantité d'information arrivant dans le serveur dans un intervalle de temps de longueur T.

Nous avons obtenu une autre majoration, indépendante de la première, et qui est, dans la plupart des cas, plus intéressante:

Proposition 2.16 *Si $\lambda < 1$, la durée $\mathcal{D}$ d'une période d'activité quelconque du serveur vérifie:*

$$\mathcal{D} \leq \frac{\Lambda}{1-\lambda} \tag{4}$$

Démonstration Soit une période d'activité du serveur démarrant à un instant t quelconque et de durée $\mathcal{D}$. Par le lemme 2.12:

$$S(t, t+\mathcal{D}) \leq s(\mathcal{D}) + \Lambda \;=\; \sum_{i \in I} \lfloor \frac{\mathcal{D}}{R_i} \rfloor L_i \;+ \Lambda \;\leq\; \mathcal{D} \sum_{i \in I} L_i / R_i \;+ \Lambda.$$

Comme la charge arrivée dans une période d'activité est justement égale à la longueur de cette période, on a $S(t, t+\mathcal{D}) = \mathcal{D}$, et donc:

$$\mathcal{D} \leq \mathcal{D}\lambda + \Lambda \quad \text{ou} \quad \text{encore,} \quad \mathcal{D} \leq \frac{\Lambda}{1-\lambda}.$$

#

Pour conclure cette section, nous obtenons des bornes de la charge en moyenne.

Définition 2.17 *Soit $\overline{W} = \int_\tau^{\tau+T} W(t)dt/T$, la moyenne de la charge à l'état stationnaire.*

On cherche un encadrement de $\overline{W}$, indépendant du déphasage initial, en fonction de λ et Λ.
On note $N = \sum_{i\in I} N_i$ le nombre total de messages arrivant dans une période.

Définition 2.18 *On définit $\beta = N/T$, le taux d'arrivée des messages.*

Théorème 2.19 *On a maintenant des bornes en moyenne:*

$$\lambda^2/(2\beta) \le \overline{W} \le \lambda\Lambda \ .$$

Démonstration Pour simplifier les notations on place l'origine des temps en τ. On obtient le majorant en utilisant d'abord le théorème 2.13:

$$T\overline{W} = \int_0^T W(u)du = \int_0^T W(u)\mathbf{1}_{\{W(u)\neq 0\}}du \le \Lambda \int_0^T \mathbf{1}_{\{W(u)\neq 0\}}du,$$

puis la proposition 2.9:

$$\int_0^T \mathbf{1}_{\{W(u)\neq 0\}}du = T - \int_0^T \mathbf{1}_{\{W(u)=0\}}du = T - (T-S) = S.$$

Finalement $\overline{W} \le \Lambda S/T = \Lambda\lambda$.

Pour le minorant, on remarque d'abord que $\overline{W}$ est toujours plus grand que la charge moyenne obtenue en supposant qu'aucun message ne subit de délai, laquelle est la somme des surfaces des triangles de hauteur L_i divisée par la période, et vaut:

$$\sum_1^M N_i L_i^2/(2T).$$

Ensuite on utilise un argument de convexité: $(x^2+y^2)/2 \ge ((x+y)/2)^2$. On applique ce résultat généralisé à N carrés dont N_i ont pour côté L_i:

$$(\sum_{i\in I} N_i L_i^2)/N \ge ((\sum_{i\in I} N_i L_i)/N)^2 = ((\sum_{i\in I} L_i T/R_i)/N)^2 = (\lambda T/N)^2 = (\lambda/\beta)^2$$

On obtient la minoration: $\overline{W} \ge (\lambda/\beta)^2 N/(2T) = \lambda^2/(2\beta)$. #

Extension des résultats: Ce minorant peut être atteint, comme le montre l'exemple suivant. En prenant $L_i = \Lambda/I$, $R_i = T$, $a_i = iR_i$ $\forall i \in I$, où a_i est l'instant d'arrivée du premier message de type i, alors aucun message ne subit de délai et $N_i = 1$ $\forall i$. Donc $\overline{W} = \sum_{i\in I} L_i/(2T) = \Lambda^2/(2MT) = \lambda^2/(2T)$ puisque ici $\lambda = S/T = \Lambda/T$ et $\beta = M/T$. Le minorant est donc atteint.

Le majorant (très grossier) peut être divisé par 2 dans certains cas particuliers: s'il n'y a qu'un type de message ou si tous les messages ont la même récurrence ($R = T$). Il est alors atteint. En effet, le résultat est immédiat dans le cas d'un seul type de message puisque $\lambda = L/T$, $\Lambda = L$, $\overline{W} = L^2/(2T) = \Lambda\lambda/2$. Dans le cas de plusieurs messages on obtient le résultat grâce à un raisonnement par récurrence sur I, en remarquant que $T\overline{W}$ est augmenté, quand on ajoute un nouveau type de message, d'au plus $(\Lambda - L)L + L^2/2$ car le délai subi par ce nouveau message vaut au plus $\Lambda - L$.

3 Ordonnancement PAPS sous contrainte temps-réel

Nous étudions dans cette section les conditions sous lesquelles les contraintes de temps-réel et d'espace sont satisfaites pour une file d'attente avec discipline de service PAPS. Ces conditions sont ensuite étendues à un réseau de stations connectées en anneau, soumises à un contrôle de flux par robinets. Ces conditions s'avèrent plutôt contraignantes, et nous proposons une politique d'accès au réseau (inspirée du mécanisme des robinets) qui garantit que les contraintes sont satisfaites pourvu que le réseau soit stable.

Définition 3.1 *Une messagerie $\mathcal{M}$ est dite "faisable" par une discipline de service si, appliquant cette discipline et partant d'une charge initiale nulle, la contrainte temps-réel est satisfaite à tout instant, quel que soit le déphasage initial.*

3.1 Faisabilité pour une station en isolation

Théorème 3.2 *Avec pour hypothèse $t \geq \tau$ ou $W(0) = 0$,*

- *une condition nécessaire et suffisante pour que la messagerie $\mathcal{M}$ soit faisable est:*

$$\Lambda < r, \tag{5}$$

  où r est la plus petite récurrence des différents types de messages.

- *une condition nécessaire et suffisante pour satisfaire la contrainte spatiale, indépendamment du déphasage à l'origine, est:*

  $B \geq \Lambda$, où B est la taille du tampon de la station.

Démonstration Puisque la vitesse du serveur est 1, et que la discipline de service est PAPS, la taille $W(t)$ de la file d'attente à l'instant t est aussi le délai avant émission subi par un message arrivant à t. Le théorème 2.13 montre donc que la condition est suffisante , et elle est nécessaire par le corollaire 2.15. #

Dans la suite de ce rapport, nous considèrerons que la condition $B \geq \Lambda$ est toujours satisfaite. Nous ne nous soucierons donc plus que de la contrainte temps-réel.

Remarque 3.3 La condition nécessaire et suffisante (5) pour satisfaire la contrainte temporelle est équivalente à la condition de stabilité, dans le cas particulier où toutes les récurrences sont identiques. En effet, dans ce cas $\lambda = \sum_i L_i/R = \Lambda/R$.

3.2 Faisabilité pour un réseau en anneau

On étudie maintenant un système de N stations connectées en anneau unidirectionnel. Chaque station est considérée comme fonctionnant en continu. Soit:

- $E(p)$ l'ensemble des stations e émettant au moins un message empruntant la section de canal p,
- r_e la plus petite récurrence des messages émis par la tation e,
- $\Lambda_e = \sum L_i$ où l'on somme sur les messages i émis par la station e. C'est la quantité maximale d'information à émettre pouvant arriver instantanément à la station e. Cela se produit si tous les types de messages ont une occurrence simultanée (*cf.* 2.15).

On définit $csect(p) = \sum L_i/R_i$ où l'on somme sur les messages i empruntant la section de canal p (*i.e.* la section précédant la station p). Cette quantité représente le *taux de charge* de la section, c'est-à-dire la proportion moyenne de temps d'occupation de la section.

Dans la suite, nous supposerons donc que l'hypothèse suivante est vérifiée:

$$csect(p) \ < \ 1 \qquad \forall p. \tag{6}$$

Ceci entraine $L_i < R_i$ pour tout i.

L'étude de la faisabilité au niveau du réseau est fondée sur la remarque suivante:

Remarque 3.4 Si les X_e sont choisis tels que $\forall p, \ \sum_{E(p)} X_e \leq 1$, alors il n'y a pas de problème de concurrence entre les stations pour l'accès au canal. Chacune d'elle est assurée de pouvoir disposer de la proportion de bande passante X_e entièrement libre. On peut donc considérer les stations comme des stations isolées ayant le même processus d'arrivée mais un serveur qui ne travaille qu'à vitesse $1/X_e$.

3.2.1 Conditions de faisabilité

Théorème 3.5 *Une condition suffisante pour satisfaire la contrainte temporelle pour le modèle continu du système avec plusieurs stations est:*

$$\forall p, \qquad \sum_{e \ \in \ E(p)} (\Lambda_e/r_e) \leq 1 \ . \tag{7}$$

Démonstration On notera

- $\zeta_p = \sum_{E(p)} \Lambda_e/r_e$, et $\zeta = max \ \zeta_p$.
- $\lambda_e = \sum L_i/R_i$ où l'on somme sur les messages i émis par la station e, le taux moyen de charge émise par la station e.

En faisant la synthèse des résultats précédents, on observe que, si $\zeta \leq 1$, alors en choisissant pour chaque station émettrice e, le "robinet" $X_e = \Lambda_e/r_e$:

- pour chaque section p, $1 \geq \zeta_p = \sum_{E(p)} X_e$, ce qui règle le problème du conflit d'accès au bus (*cf.* la remarque 3.4);
- pour chaque station e, $X_e = \Lambda_e/r_e \geq \lambda_e$ car r_e est plus petit que toutes les récurrences des messages émis par la station e. Cela assure la stabilité de chaque station;
- en prenant pour chaque station e, la taille de tampon: $B_e = \Lambda_e$, on satisfait la contrainte spatiale pour chaque station (*cf.* le théorème 3.2);
- l'attente maximale d'un message i avant émission de la station e est majorée par: $\Lambda_e/X_e = r_e$, puisque la quantité maximale d'information à émettre est Λ_e et la vitesse du serveur $1/X_e$. Comme $r_e \leq R_i$, $\forall i$, la contrainte temps-réel est satisfaite pour chaque message.

#

L'analyse de ce résultat amène les remarques suivantes.

Remarque 3.6 La condition (7) n'est pas nécessaire à l'existence de valeurs des robinets permettant de satisfaire la contrainte temps-réel, comme le montre l'exemple suivant.

Considérons un système avec trois stations se transmettant quatre flux d'information caractérisés par les quadruplets (station émettrice, destinataire, longueur, récurrence) suivants: (1, 3, 3, 6), (1, 2, 2, 6), (2, 3, 2, 6), (3, 1, 3, 6). On a alors $csect(1) = 1/2$, $csect(2) = csect(3) = 5/6$, donc l'hypothèse de départ (6) est vérifiée. D'autre part on a $\zeta_1 = 1/2$, $\zeta_2 = 5/6$, $\zeta_3 = 7/6$, donc la condition (7) n'est pas vérifiée. Et pourtant, même en fixant tous les robinets à la valeur maximale 1, les stations 1 et 3 disposent de chaque slot en entier puisque aucun flux ne les traverse, et la station 2 voit passer 3 slots pleins sur 6, donc le délai maximal subi par un de ses messages est 5, ce qui prouve que la contrainte temps-réel est satisfaite, quel que soit le déphasage à l'origine. Les figures 4 et 5 représentent respectivement la charge moyenne sur le bus due aux différents messages et les "réservations" effectuées sur le bus. On voit bien l'effet de la condition (7) qui fait "comme si" le message 2 allait jusqu'en 3 alors que la bande passante nécessaire au message 3 est en fait disponible.

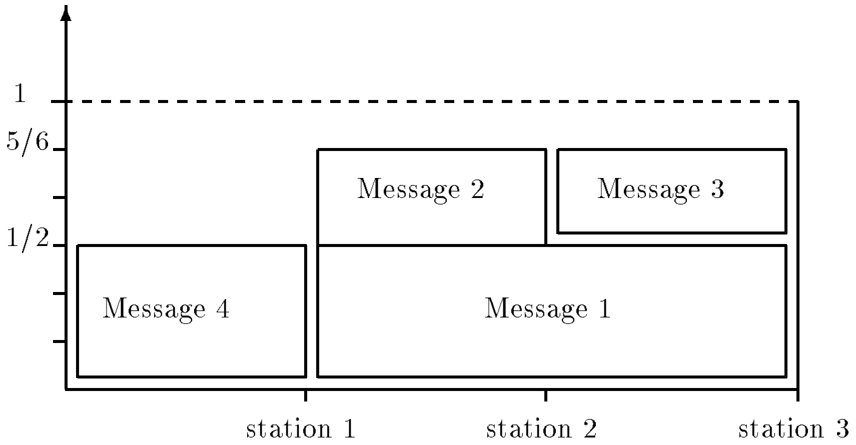


Figure 4 : Occupation moyenne du canal

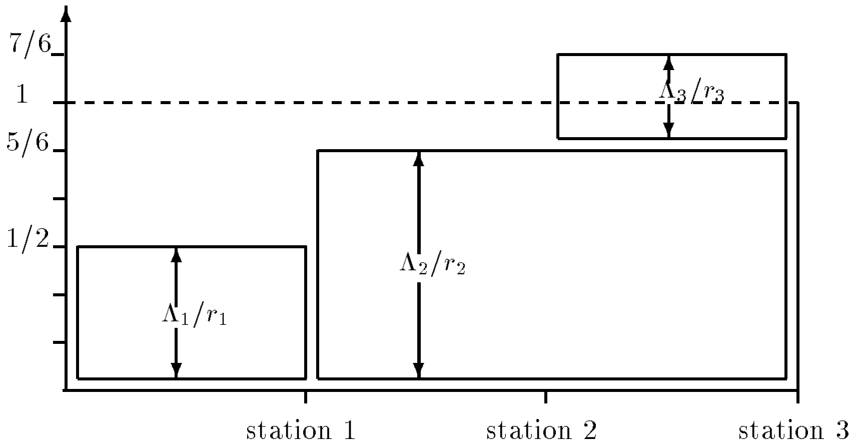


Figure 5 : Réservations nécessaires sur le bus

Remarque 3.7 On a $\zeta_p \geq csect(p)$ quelle que soit la section de canal p, donc la condition (7) implique que l'hypothèse de stabilité (6) est vérifiée. On ne peut pourtant pas lier la condition (7) à un encadrement des charges par section: d'une part elle est vérifiée par des systèmes où la charge sur une section peut être aussi proche de 1 que l'on veut (exemple 1), d'autre part il existe des systèmes avec des charges par section aussi proches de 0 que l'on veut qui ne vérifient pas cette condition (exemple 2).

- exemple 1: on considère un système avec deux stations s'échangeant une information constituée pour chacune d'elles, d'un unique flux de même caractéristiques: $L = l$ et $R = 1$ avec $0 < l < 1$. Alors

$\zeta_1 = \zeta_2 = l \leq 1$, et $csect(1) = csect(2) = l$. Quand l tend vers 1, les charges par section aussi, pourtant la condition (7) reste vérifiée.

- exemple 2: considérons un système avec deux stations dont la première émet une superposition de deux flux caractérisés par: $L_1 = 2l,\ R_1 = 1 + 2l,\ L_2 = 2l^2/(1 + 2l),\ R_2 = l$, avec $0 < l < 1$. La deuxième station n'émet rien (la station 1 est en fait en isolation). Alors $\zeta_2 = 2(1 + l/(1 + 2l)) \geq 2 > 1\ \ \forall l$, donc l'inégalité (7) n'est vérifiée pour aucune valeur de l. Pourtant quand l tend vers 0, $csect(2) = 4l/(1+2l)$ tend aussi vers 0 et $csect(1) = 0$.

Remarque 3.8 Dans le second exemple, aucune valeur des robinets ne pourra permettre de respecter la contrainte temporelle. En effet, même dans le meilleur des cas ($X_1 = 1$), dès qu'un message de type 1 est émis, on est sûr qu'un message de type 2 ratera sa récurrence puisque $L_1 > R_2$. Plus généralement, on a la proposition suivante:

Proposition 3.9 *Il est nécessaire à l'existence de robinets permettant de satisfaire la contrainte temporelle que la condition suivante soit vérifiée par chaque station émettrice:*

$$\Lambda_e \leq r_e \quad \forall e \ . \tag{8}$$

Cette condition correspond à la condition nécessaire et suffisante (5) pour que la contrainte temporelle soit respectée dans le cas d'une station en isolation.

Démonstration Pour que la contrainte temps-réel soit satisfaite, il faut que pour chaque station émettrice e, la plus petite récurrence r_e des flux arrivant soit plus grande que l'attente maximale dans la file avant émission. Le mécanisme des robinets implique que l'attente est elle-même toujours plus grande que la taille de la file. Le maximum de la taille de cette file pour tout instant et quel que soit le déphasage initial est Λ_e d'après le corollaire 2.15. #

Après avoir constaté de plus que les données de Thomson ne remplissaient pas cette condition, nous avons cherché à les modifier pour élargir le domaine d'application des résultats précédents. Trois propositions sont faites dans la suite. La première consiste à augmenter la plus petite des récurrences, la seconde décompose les stations par un mécanisme de "multi-robinets". Enfin, nous étudions l'intérêt d'attribuer des valeurs de robinets par type de message plutôt que par station.

3.2.2 Modification des récurrences

Une idée de modification des données, pour satisfaire la condition (8) consiste à regrouper les messages d'un même flux. Ainsi, par exemple, un flux de message de récurrence R, de longueur L sera transformé en un flux de récurrence $2R$ et longueur $2L$. Cette transformation ne modifie pas les charges par section. En revanche, les valeurs de Λ_e et r_e augmentent. Il n'est pas certain que le rapport Λ_e/r_e diminue lors d'une telle transformation. Mais le résultat suivant prouve que par transformations successives, on finit par obtenir une messagerie faisable: à l'extrême, on est amené à rendre égales toutes les récurrences. Pour certains types de messages, la valeur de la récurrence étant essentielle pour l'application, cette méthode est limitée en pratique.

Nous montrons maintenant que dans le cas où toutes les récurrences des flux de messages sont égales, et si la charge par section reste inférieure à une certaine valeur, la messagerie est faisable. Pour une station en isolation, ce résultat est vrai puisque la condition de faisabilité correspond à celle de stabilité (*cf.* remarque 3.3). Pour le bus complet, il semble clair que le système admet également un régime stationnaire périodique et que sa période vaut R, la période commune à tous les flux. En admettant ce résultat (dont la preuve paraît assez rébarbative), il suffirait que la charge de chaque section soit inférieure à 1, pour que la messagerie soit faisable. Néanmoins, il est possible de prouver facilement un résultat plus faible:

Proposition 3.10 *Si $R_i = R\ \ \forall i$, et $csect(p) \leq 1/2\ \ \forall p$, alors la messagerie est faisable.*

Démonstration Supposons que la messagerie ne soit pas faisable. Notons i, l'indice du premier message pour lequel il y a dépassement, i.e. dont l'émission n'est pas terminée avant l'arrivée du prochain du même type. Ce message a été émis par une station e. Soit t la date du dépassement. Considérons l'occupation de la section $e+1$ durant l'intervalle $[t-R, t[$, en remarquant que $t-R$ est la date d'arrivée du message i. Dans cet intervalle, la file de e ne s'est jamais vidée puisque le message de type i arrivé à $t-R$, n'a pas été complètement émis à t. La section $e+1$ est occupée d'une part par des messages émis par les stations en amont, et d'autre part, par les messages émis par e.

- Le temps d'occupation par un message de type j provenant de l'amont est au plus égal à $2L_j$, sinon il y aurait eu dépassement pour ce flux de messages.

- Le temps d'occupation par les messages émis par e est au plus égal à Λ_e.

Finalement, R, temps d'occupation de la section $e+1$ dans l'intervalle $[t-R, t[$ vérifie: $R \leq 2\sum_j L_j + \Lambda_e \leq 2\sum_k L_k$ où l'on somme sur tous les indices k tels que le message k emprunte la section $e+1$. On obtient $1/2 \leq \sum_k L_k/R = csect(e+1)$, ce qui est contradictoire avec l'hypothèse. #

3.2.3 Les multi-robinets

Un autre moyen de diminuer un des ζ_p et donc ζ, est de "décomposer" une station émettrice $e \in E(p)$, *i.e.* regrouper les différents flux émis par la station e en sous-ensembles notés $J_1, .., J_n$ auxquels on attribue une file et un accès au canal. La station e initiale est donc décomposée en n files ou "stations fictives".

Notons ζ'_p la nouvelle valeur de ζ_p, on a alors:

- $\zeta_p - \zeta'_p = \Lambda_e/r_e - \sum(\Lambda_{J_i}/r_{J_i})$ où l'on somme sur les J_i dont un message emprunte la section p,

- $r_{J_i} \geq r_e$ par restriction de l'ensemble sur lequel le plus petit élément est pris,

- $\sum \Lambda_{J_i} \leq \Lambda_e$ car on restreint l'ensemble des flux concernés quand un des J_i ne regroupe que des messages n'empruntant pas la section p.

Finalement, $\zeta'_p \leq \zeta_p$. Le cas d'égalité se produit quand tous les flux émis par une station ont même récurrence et même destinataire.

Cette dernière remarque nous a amené à introduire la définition d'une *file élémentaire*: c'est un ensemble regroupant tous les flux de messages ayant même chemin et même récurrence. Il n'y a aucun gain à décomposer une telle file.

Les données d'un système peuvent toujours se découper en files élémentaires. Si on décompose les stations d'un système en accordant une station fictive à chaque file élémentaire, alors si l'on note ζ''_p la nouvelle valeur de ζ_p, on a pour chaque section de canal p, $csect(p) = \zeta''_p$. L'hypothèse de départ (6) permet de conclure que $\zeta'' < 1$. Dans ce cas extrême on constate que la condition (7) est vérifiée.

La seule hypothèse de départ (6) assure l'existence d'une partition des données initiales en un réseau de stations fictives qui vérifie la condition (7).

D'un point de vue pratique, ces modifications des *données* reviennent en fait à modifier l'*outil* utilisé. On définit ainsi le mécanisme des *multi-robinets*: les messages à émettre arrivant dans chaque station initiale décomposée sont stockés dans leurs files respectives. Le slot courant est lu par la station puis rempli des données à émettre, par un balayage des différentes files, chacune étant limitée par son propre robinet.

Ce nouveau mécanisme est plus sophistiqué à implémenter que les simples robinets, mais il présente l'énorme avantage de ne se contenter que de l'hypothèse de départ (6) pour assurer son fonctionnement. La condition très restrictive $\Lambda_e < r_e$ nécessaire pour les robinets simples, ne l'est plus pour les multi-robinets (*cf.* la proposition 3.9).

On a cherché à minimiser les complications dues à un trop grand nombre de stations fictives entraînant des valeurs de robinet trop faibles. On présentera dans le paragraphe 3.3 un algorithme pour dédoubler les stations jusqu'à remplir la condition (7).

3.2.4 Robinets par flux

Nous étudions dans cette section l'éventualité d'associer une valeur de robinet à chaque type de message. Au niveau d'une station, cela revient à lui attribuer une proportion de canal dépendante du type de message qui est en cours d'émission. Nous obtenons une condition suffisante de faisabilité. Cette condition est plus faible que (7), mais nous n'avons pas pu caractériser l'existence de valeurs de robinets la satisfaisant, et il faut recourir à des méthodes numériques pour déterminer la valeur de ces robinets.

Soit:

- $M(p)$ l'ensemble des indices de messages qui empruntent la section p,
- $I(e)$ l'ensemble des indices de messages émis par la station e,
- X_i le robinet affecté aux messages de type i.

Pour qu'il n'y ait pas de conflit d'accès au bus, il faut que la somme des proportions maximales attribuées à chaque station soit inférieure à 1. Cette condition s'écrit:

$$\sum_{e \in E(p)} \max_{i \in I(e) \cap M(p)} X_i \quad < \quad 1 \quad . \tag{9}$$

Théorème 3.11 *Une condition suffisante pour satisfaire les contraintes temporelles pour le modèle continu du bus avec plusieurs stations, et le mécanisme de robinets par flux de messages est:*

$$\forall p, \qquad \sum_{e \in E(p)} \max_{i \in I(e) \cap M(p)} X_i \quad < \quad 1 \quad , \tag{10}$$

et

$$\forall e, \quad \sum_{i \in I(e)} L_i / X_i \leq r_e \quad . \tag{11}$$

La condition $\Lambda_e / r_e < 1$ (équation (8)) qui était nécessaire à l'existence de robinets par station doit toujours être vérifiée. Elle correspond à la condition nécessaire et suffisante de faisabilité pour une station en isolation avec un serveur travaillant à vitesse maximale. Cette condition sera nécessaire pour tout mécanisme de restriction de l'accès des stations au bus car cela revient à réduire la vitesse du serveur.

Notons que si (11) est vérifiée alors (8) aussi. En effet,

$$\forall i \ \ L_i \leq L_i / X_i \quad \Rightarrow \quad \sum_{i \in I(e)} L_i = \Lambda_e \leq \sum_{i \in I(e)} L_i / X_i \leq r_e$$

.

Démonstration

- il n'y a pas de conflit sur le bus: la condition (10) correspond à la condition (9).
- puisque $\forall e, \quad \Lambda_e / r_e > \lambda_e (= \sum_{i \in I(e)} L_i / R_i)$ et que $\forall e, \quad \Lambda_e / r_e < 1$, le taux de charge λ_e est bien inférieur à 1 et donc chaque station e est stable (*cf.* corollaire 2.10).
- l'attente maximale d'un message de type $i \in I(e)$ avant son émission est majorée par $\sum_{j \in I(e)} L_j / X_j$. En effet, la stabilité de la station e et l'absence de conflit d'accès au bus étant assurées, la quantité maximale d'information rencontrée par un message à son arrivée dans la station e est bien Λ_e. Et cette quantité sera servie en un temps correspondant à la somme des durées d'émission de chaque type de message $j \in I(e)$, soit $\sum_{j \in I(e)} L_j / X_j$. Cette quantité est inférieure ou égale (par hypothèse) à r_e, et donc a fortiori à R_i. Ainsi, la contrainte temps-réel est satisfaite pour tout type de message.

#

Remarque 3.12 La condition du théorème 3.11 est moins restrictive que celle correspondant à l'attribution d'un robinet par station. En effet, si $\forall i, j \in I(e), \quad X_i = X_j$, on retrouve la condition du théorème 3.5.

Remarque 3.13 Il ne s'agit toujours que d'une condition suffisante. Reprenons l'exemple de la remarque 3.6:

i	e(metteur)	d(estinataire)	L_i	R_i
1	1	3	3	6
2	1	2	2	6
3	2	3	2	6
4	3	1	3	6

Nous nous sommes déjà convaincus de la faisabilité de cette messagerie. Or il n'existe pas de valeurs de robinets par flux satisfaisant les conditions du théorème 3.11. En effet ces conditions s'écrivent:

$$\begin{array}{rcll}
3/X_1 + 2/X_2 & \leq & 6 \quad (11) & \text{pour} \quad e = 1 \\
2/X_3 & \leq & 6 \quad (11) & \text{pour} \quad e = 2 \\
1/X_4 & \leq & 6 \quad (11) & \text{pour} \quad e = 3 \\
max(X_1, X_2) & \leq & 1 \quad (10) & \text{pour} \quad p = 2 \\
X_1 + X_3 & \leq & 1 \quad (10) & \text{pour} \quad p = 3 \\
X_4 & \leq & 1 \quad (10) & \text{pour} \quad p = 1
\end{array}$$

Et l'on vérifie que ce système d'inéquations n'a pas de solution.

3.3 Implémentation des multi-robinets

On revient maintenant au modèle initial, c'est-à-dire au système à N stations où le canal de communication transmet de façon discrète (godet par godet). Nous supposons que la méthode choisie pour implémenter les robinets est celle qui consiste en l'utilisation de "slots" de G godets consécutifs. La valeur du robinet X_e doit correspondre à un nombre entier de godets, c'est-à-dire que X_e doit être un multiple de $1/G$ (*cf.* paragraphe 1.1). On donne tout d'abord la condition suffisante pour trouver des valeurs de robinets permettant de satisfaire la contrainte temporelle. Nous présentons ensuite un algorithme déterminant des multi-robinets qui permettent de satisfaire les contraintes, et nous commentons les résultats obtenus.

3.3.1 Etude pratique

On notera $\langle x \rangle = \lceil Gx \rceil / G$, la valeur de x arrondie au "G-ième" supérieur.

Théorème 3.14 *Une condition suffisante pour satisfaire la contrainte temporelle du système à plusieurs stations avec un bus slotté est:*

$$\forall p \qquad \sum_{e \ \in \ E(p)} \langle \Lambda_e / r_e \rangle \leq 1 \ . \tag{12}$$

La démonstration est similaire à celle du théorème 3.5, en prenant pour valeur des robinets $X_e = \langle \Lambda_e / r_e \rangle$.

On notera $\xi_p = \sum_{E(p)} \langle \Lambda_e / r_e \rangle$, $\xi = max\ \xi_p$.

Cette condition (12) est plus forte que celle du cas continu (7) à cause des arrondis au G-ième supérieur. La perte est plus ou moins grande suivant les données. Plus il y a de termes dans la sommation, plus l'écart avec la condition "idéale" peut être grand.

Afin de tester les algorithmes, Thomson nous a fourni la description de quatre messageries comportant plusieurs dizaines de messages. On constaté que ces messageries ne vérifient pas la condition (7) et donc pas non plus la condition (12). Nous avons donc implémenté un algorithme pour déterminer comment choisir les multi-robinets pour satisfaire la contrainte temporelle.

La convergence de l'algorithme est assurée par l'hypothèse de départ réadaptée:

$$\sum \langle \Lambda_f / R_f \rangle \leq 1 \quad \forall p \ , \tag{13}$$

où l'on somme sur les files élémentaires dont un message passe par p.

A nouveau cette condition (13) est plus forte que l'hypothèse de départ (6) à cause des arrondis au G^{ieme} supérieur.

La condition (13) est réalisée en particulier si, pour chaque section p on a: $csect(p) \leq 1 - K/G$, où K est le nombre total de files élémentaires.

3.3.2 Présentation de l'algorithme

A chaque étape:

- on repère la (ou une) section p ayant le plus grand ξ_p,
- on repère ensuite une station e dont un message emprunte cette section,
- puis on la "dédouble" en créant un nouveau buffer d'émission en parallèle où seront orientés tous les types de messages ayant pour récurrence r_e (si le buffer initial émettait des messages de récurrences différentes) ou (sinon) tous ceux ayant le même destinataire que le premier message du buffer, les autres restant inchangés: cela équivaut à "créer" une nouvelle station fictive, uniquement émettrice,
- enfin, on calcule ξ pour le nouveau réseau fictif, et on le compare à 1: si $\xi > 1$ on itère, sinon l'algorithme est terminé.

Quand la condition (13) est réalisée, l'algorithme converge et son nombre d'itérations est borné par K (lui-même borné par le nombre total de types de messages émis sur le réseau).

On a choisi de dédoubler les stations car cela nous assurait de ne jamais décomposer de file élémentaire, et permettait une manipulation assez simple des données. D'autres choix possibles sont discutés au paragraphe suivant.

Les résultats sont présentés en annexe.

3.3.3 Commentaires

Le fait de dédoubler des stations présente plusieurs inconvénients:

-1- il faut concevoir un aiguillage des types de messages avant leur entrée dans les files d'attente,

-2- pour cela, il faut identifier les différents types, ce qui alourdit le protocole,

-3- l'émission est un peu plus sophistiquée que le simple système des robinets: pour une station e dédoublée en J_1 et J_2, l'émetteur vide le tampon J_1 jusqu'à concurrence de X_{J_1}, puis le tampon J_2 jusqu'à concurrence de X_{J_2}.

L'utilisation des robinets allonge les temps de réponse.

Dans la pratique, ces inconvénients semblent mineurs, dans la mesure où, en observant les résultats obtenus, on constate que:

-1- on modifie très peu de stations. Cela est fortement dû à la "structure" des données: en effet dans les stations à fort Λ/r, la plus petite récurrence est très différente des autres, et donc la diminution de ξ est très grande.

-2- l'identification des types de messages est implicite dans l'outil proposé des robinets par message.

-3- la condition nécessaire (8) n'est pas vérifiée par les données de Thomson. Dans le cas où tous les messages arrivent simultanément à une station, il faut permettre à ceux de plus petite récurrence (les plus contraints) de doubler les autres. Cela ne peut pas se faire avec une file unique et un robinet. Il faut ou bien utiliser une file parallèle (multi-robinet) ou bien interrompre les messages en cours (priorité préemptive), pour pouvoir satisfaire la contrainte temps-réel.

Les intérêts de cette méthode sont considérables:

- En regroupant les stations dédoublées, on remarque que les tailles globales des tampons satisfaisant la contrainte spatiale sont les mêmes que dans le cas optimal où la station disposerait du canal pour elle seule.
- Il n'y a aucune perte de message, que ce soit pour une raison d'espace ou de temps réel.
- Le mécanisme des multi-robinets fait que l'on n'utilise jamais le slot en entier, il reste toujours une proportion au moins égale à $1-\xi$ libre. On pourrait donc utiliser ce reste pour assurer la transmission d'éventuels messages asynchrones sans risque de perturber le flux synchrone.

Plusieurs améliorations de l'algorithme sont envisageables:

- On pourrait poursuivre l'algorithme pour augmenter $1-\xi$, ce qui permettrait d'augmenter la largeur de la bande passante accordée aux messages asynchrones. Un nouveau critère d'arrêt serait alors à déterminer en fonction du coût de l'ajout d'une nouvelle file.
- On pourrait repérer en début d'étape la station qui, une fois transformée, fera diminuer ξ le plus possible.
- On pourrait aussi dédoubler les stations suivant des règles d'implémentation plus délicate, vérifiant certains critères comme minimiser le nombre d'étapes de l'algorithme, minimiser le nombre de stations dédoublées ou le nombre final de stations fictives en tentant d'en regrouper certaines.

Les résultats de l'application de l'algorithme aux messageries fournies par Thomson peuvent être consultés dans [8].

4 Ordonnancement avec préemption

Dans cette section, nous revenons au modèle d'une file simple et nous reprenons les résultats de Liu et Layland [10] sur la faisabilité des politiques à priorités préemptives. Dans la section suivante,nous considérons l'extension de ces résultats (ainsi que de ceux obtenus pour PAPS) quand l'overhead est pris en compte.

Nous commençons par introduire les notations et faire quelques remarques générales qui serviront dans toute la suite de ce rapport.

4.1 Remarques générales

Nous utiliserons souvent le terme de *requête* pour désigner l'arrivée d'un message. Nous dirons alors qu'une requête est insatisfaite, quand l'émission du message dont elle est l'arrivée n'est pas complétée à temps.

Définition 4.1 *Il y a un "dépassement" à l'instant t si t est la date d'une requête et si le message précédant de même type n'a pas encore terminé son émission à l'instant t.*

Cette définition signifie que l'émission a dépassé sa "date limite", cela n'a rien à voir avoir le fait que les messages d'un même type se doublent.

Rappelons que dans le cadre de politiques à priorités, on distingue les priorités suivantes:

- – statiques: un numéro (ou niveau) de priorité est attribué à chaque flux (type de message ou tâches) au départ,
  – ou dynamiques: les numéros de priorité sont fonctions du temps. A chaque instant de décision, le message de plus grande priorité est déterminé.
- – préemptives: l'arrivée d'un message prioritaire interrompt l'émission ou l'overhead d'un moins prioritaire,
  – ou non préemptives: si au moment d'une requête d'un message, le serveur est occupé, alors le message doit attendre la fin de l'émission avant de pouvoir occuper le serveur.

Dans l'étude des systèmes sous contrainte temps-réel se pose la question de la politique à suivre en cas de dépassement: quel(s) message(s) garder, et dans quel ordre les servir. Puisque nous nous intéressons aux systèmes où tout dépassement est interdit (contrainte forte), nous avons choisi de conserver les messages et d'appliquer dans une même classe la discipline PAPS.

Une politique prioritaire est en fait (à chaque instant) un ordre sur l'ensemble des flux de messages $\{1, 2, \ldots, M\}$. Nous noterons $i \succ j$ pour "le flux i est prioritaire sur le flux j", et $\mathbf{P}$ la priorité $1 \succ 2 \succ \ldots \succ M$ c'est-à-dire $i \succ j \Longleftrightarrow i < j$.

Pour résoudre les problèmes de simultanéité des événements arrivée/départ, on considèrera que l'affectation des messages au serveur se fait après la prise en compte des arrivées. Donc si une date de requête coïncide avec la fin d'une émission, alors cette arrivée sera prise en compte *avant* le départ. Ainsi si le message entrant est prioritaire sur tous ceux qui étaient dans la file en attente, c'est lui qui sera servi.

Définition 4.2 *La "trajectoire" d'une messagerie est la fonction f définie par: $f(t) = i$ si le serveur est occupé par un message de type i à l'instant t (bloqué en overhead, ou actif en émission) et $f(t) = 0$ si la file est vide.*

Avec cette définition, la trajectoire dépend de l'ordonnancement choisi et de l'état initial alors que la courbe de charge du système ne dépend pas, elle, de l'ordonnancement.

Dans leur article, Liu et Layland supposent (implicitement) que la charge du système à l'origine est nulle (d'où notre définition de la faisabilité). En général, il faut séparer la période transitoire de la période stationnaire. En effet, on conçoit aisément que si la charge initiale est trop grande, les premières requêtes de certains types vont provoquer des dépassements qui ne se produiront plus par la suite. En fait, l'existence de dépassements en régime stationnaire et faisabilité sont liées par la remarque suivante: si une messagerie stable est faisable au sens de la définition 3.1 , alors la contrainte temps-réel sera respectée pendant le régime stationnaire obtenu en partant d'une charge initiale quelconque.

En effet, considérons une messagerie stable $\mathcal{M}$ et un état initial (charge initiale, déphasage à l'origine) quelconque: $(W(0), \underline{\phi})$. La stabilité donne l'existence d'un instant fini noté t' tel que $W(t') = 0$. Replaçons l'origine des temps à un tel t'. Pour cela définissons un nouveau déphasage initial $\underline{\phi}'$ par: $\forall i \in I \;\; \phi_i' = \phi_i - t'$ modulo R_i.

Alors sur l'intervalle $[0, +\infty[$ en partant de l'état initial $(0, \underline{\phi}')$, on observe la même trajectoire que sur l'intervalle $[t', +\infty[$ en partant de l'état initial $(W(0), \underline{\phi})$.

Dans la suite de ce rapport, nous supposerons donc que la charge initiale $W(0)$ est nulle.

Nous avons choisi, dans cette section, d'utiliser les notations et le vocabulaire de l'article de Liu et Layland, la correspondance avec nos propres notation étant claire.

Cet article traite du modèle d'une station unique, recevant une superposition de flux déterministes. Chaque flux représente une tâche répétitive notée τ_i, $i \in I$, de longueur fixe L_i, et de période R_i. Un ensemble de tâches $\{\tau_1, .., \tau_M\}$ noté τ correspond ce que nous avons appelé à une messagerie.

La même contrainte temps-réel doit être satisfaite, mais ici avec un système à priorités. Les priorités étudiées dans cet article sont toujours *préemptives*, dans un premier cas statiques, et dans un second cas dynamiques. Dans ce modèle il n'y a *pas d'overhead*: le passage d'un message à l'autre est instantané.

Le but est de caractériser la faisabilité de τ, pour les différentes priorités étudiées.

Nous rappelons les résultats principaux obtenus dans cet article, avec un formalisme légèrement différent. Nous redémontrons certains résultats dont les preuves étaient fausses par manque de précision. Dans le cas de priorités statiques, un premier résultat est un *test de faisabilité* par une priorité quelconque, notée $\mathbf{P}$ (condition nécessaire et suffisante). Un second résultat est l'*optimalité* de la priorité notée $\mathbf{R}^+$ qui accorde la plus grande priorité à la tâche la plus récurrente. Pour définir l'optimalité dans ce contexte, nous dirons qu'une politique est meilleure qu'une autre quand elle rend faisable un plus grand nombre de messageries. Un troisième résultat concerne un test de faisabilité qui n'est fonction que de la valeur du taux de charge de l'ensemble de tâches, défini par $U(\tau) = \sum_I L_i/R_i$. Cette valeur était jusqu'ici notée λ.

Enfin dans le cas de priorités dynamiques et toujours préemptives, un dernier résultat montre que la priorité $\mathbf{DL}^+$ est optimale puisqu'elle rend faisable tout ensemble de tâches stable. $\mathbf{DL}^+$ accorde la plus grande priorité, à l'instant t, à la tâche dont la date de la prochaine requête (date limite) est la plus proche de t.

4.2 Cas statique

Ici les priorités sont attribuées dès le départ à chaque flux.

4.2.1 Test par une trajectoire

Définition 4.3 *Le délai subi par la première requête de la tâche τ_i pour le déphasage initial nul $\phi_j = 0 \;\; \forall j \in I$, avec la priorité $\mathbf{P}$, sera noté Δ_i^0.*

Le théorème 1 de Liu et Layland dit que le plus long délai subi par une tâche est le délai correspondant à une requête simultanée de toutes les tâches i.e. Δ_i^0.

Ce résultat est faux. En revanche, il est vrai que:

- si $\Delta_i^0 \leq R_i \;\; \forall i$, alors l'ensemble de tâches est faisable et le plus long délai subi par une tâche de type i est Δ_i^0,
- sinon (i.e. si $\exists i \;\; tq \;\; \Delta_i^0 > R_i$) l'ensemble des tâches n'est pas faisable et *une tâche de type i peut subir un délai plus long que* Δ_i^0, comme le montre l'exemple suivant:

- $M = 3$,
- $\underline{L} = (1, 4, 4)$,
- $\underline{R} = (100, 10, 7)$.

La condition de stabilité est vérifiée puisque $\sum_i L_i/R_i \simeq 0,98$.
La figure 6 montre que $\Delta_3^0 = 9$. L'ensemble de tâches n'est pas faisable puisque $\Delta_3^0 > R_3$. Mais Δ_3^0 est inférieur au délai subi par la troisième requête de type **3** qui vaut 11.

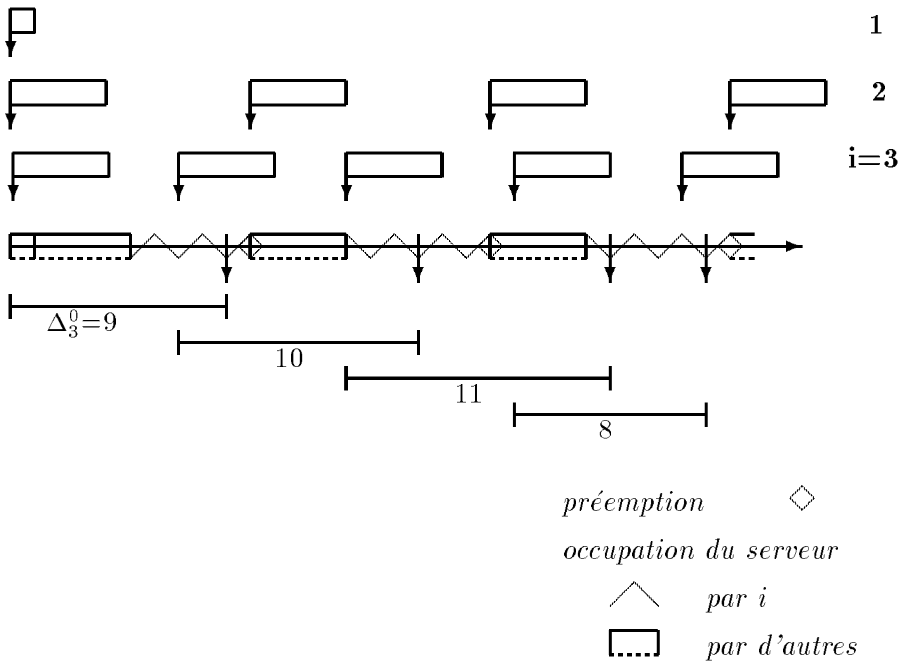


Figure 6 : Δ_i^0 n'est pas le plus long délai.

Dans leur démonstration, Liu et Layland utilisent l'argument suivant: considérons une requête de la tâche la moins prioritaire τ_m à la date t, et une tâche plus prioritaire τ_i dont la première requête après t a lieu à la date t_2. Avançons la date de requête t_2 à l'instant t. Alors "la date de fin d'exécution de la tâche τ_m est ou bien inchangée ou bien retardée par une telle avance". *Cet argument est faux* car il ne tient pas compte de la charge réelle à l'instant t, comme le montre l'exemple suivant:

- $M = 3$,
- $\underline{L} = (5, 4, 4)$,
- $\underline{R} = (100, 11, 100)$.

La condition de stabilité est vérifiée puisque $\sum_i L_i/R_i \simeq 0,45$.

Le schéma du haut de la figure 7 montre qu'avec le déphasage initial $\underline{\phi} = (6, 7, 12)$, la tâche de type **3** dont la requête arrive à la date $t = 12$ subit un délai qui vaut 11. Le schéma du bas montre qu'avec le déphasage initial $\underline{\phi} = (6, 1, 12)$, où la date t_2 de la requête de type **2** initialement à 18 est avancée à 12, la tâche de type **3** subit un délai moindre, qui vaut 8.

Malgré cela, l'intuition de la condition de faisabilité de τ par la priorité préemptive **P** est bonne. Nous la justifions grâce au théorème suivant.

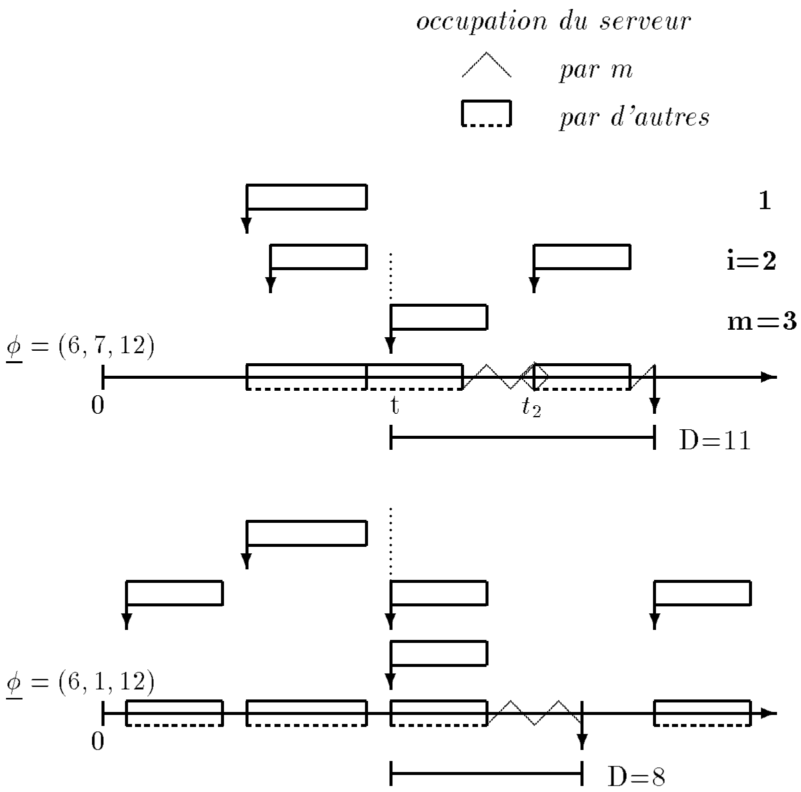


Figure 7 : avancer la requête de **2** peut diminuer le délai de **3**

Théorème 4.4 *Une condition nécessaire et suffisante de faisabilité d'un ensemble de tâches τ par une priorité préemptive* $\mathbf{P}$ *est:*

$$\Delta_i^0 \leq R_i \quad \forall i \in I \iff \tau \textit{ est faisable par } \mathbf{P}. \tag{14}$$

Ce critère est très simple à vérifier dans la mesure où il ne demande l'observation que d'une trajectoire (démarrant par une requête simultanée de toutes les tâches) et pendant un temps fini égal à la plus grande des récurrences.

Démonstration Le sens ($\Leftarrow$) est évident. Si τ est faisable, alors pour tout type de tâche, le délai de toute requête est inférieur à la récurrence, et ce quel que soit le déphasage initial considéré. En particulier le premier délai subi par une tâche de chaque type quand toutes les requêtes sont simultanées est inférieur à sa récurrence i.e. $\Delta_i^0 \leq R_i \quad \forall i \in I$.

Le sens ($\Rightarrow$) est démontré par la contraposée. Supposons que τ n'est pas faisable, c'est-à-dire qu'il existe un déphasage initial noté $\underline{\phi}$, tel qu'une requête n'est pas satisfaite à temps. Considérons la première requête insatisfaite; elle est de type noté i, et arrive à la date t. Son délai est noté D et vérifie $D > R_i$. Nous allons montrer que $\Delta_i^0 > R_i$, en modifiant progressivement le déphasage initial, tout en nous assurant qu'à chaque étape le délai de la requête considérée ne peut qu'augmenter. Dans une première étape nous plaçons la requête insatisfaite au début d'une période d'activité du serveur par des plus prioritaires que i. Dans la deuxième étape nous avançons les dates de toutes les requêtes plus prioritaires que i au même instant.

Etape 1 Puisque la priorité est préemptive, les tâches de type strictement moins prioritaire que i ne seront plus prises en compte dans la suite de cette démonstration.

Soient d_n et f_n les dates respectives de début et de fin de la $n^{ième}$ période d'activité du serveur par des tâches strictement plus prioritaires que i.

Soit $n > 0$ tel que $t \in]f_{n-1}, f_n]$.

Nous distinguons deux cas, illustrés par la figure 8:

cas 1 si $t \in]d_n, f_n]$, alors le serveur ne sert que des tâches strictement plus prioritaires que i sur l'intervalle $[d_n, t]$. Puisque t est la date de la première requête insatisfaite, les précédentes requêtes de type i sont satisfaites avant d_n. Ainsi le cas d'un message qui se retarde lui-même comme dans l'exemple de la figure 6 ne peut pas se produire ici.

En avançant la date de la requête à l'instant d_n (c'est-à-dire en remplaçant ϕ_i par d_n modulo R_i), le délai est donc augmenté de la quantité positive $t - d_n$.

cas 2 si $t \in]f_{n-1}, d_n]$, nous noterons $k \geq n$ l'indice tel que la date $t + D$ de fin d'exécution de la tâche est dans l'intervalle $]f_{k-1}, d_k]$.

Si la date de la requête est retardée à l'instant d_n, deux sous-cas se présentent:

- si $d_n + D \leq d_k$, la tâche ne subit pas d'autre préemption, et son délai est inchangé,
- sinon $d_n + D > d_k$, et la tâche subit au moins une préemption supplémentaire à l'instant d_k. Cela a pour effet d'augmenter son délai d'au moins $f_k - d_k$.

Nous avons donc montré dans tous les cas que le délai ne pouvait qu'augmenter en déplaçant la requête à l'instant d_n. Nous noterons D_1 ce nouveau délai.

Etape 2 Soient $j \succ i$ et t_j la date de la première requête de type j après d_n.

Le nombre de requêtes de type j qui retardent la requête i considérée est

$$\lfloor (D_1 - (t_j - d_n))/R_j \rfloor.$$

En avançant la date t_j à l'instant d_n, ce nombre augmente pour devenir au moins $\lfloor D_1/R_j \rfloor$.

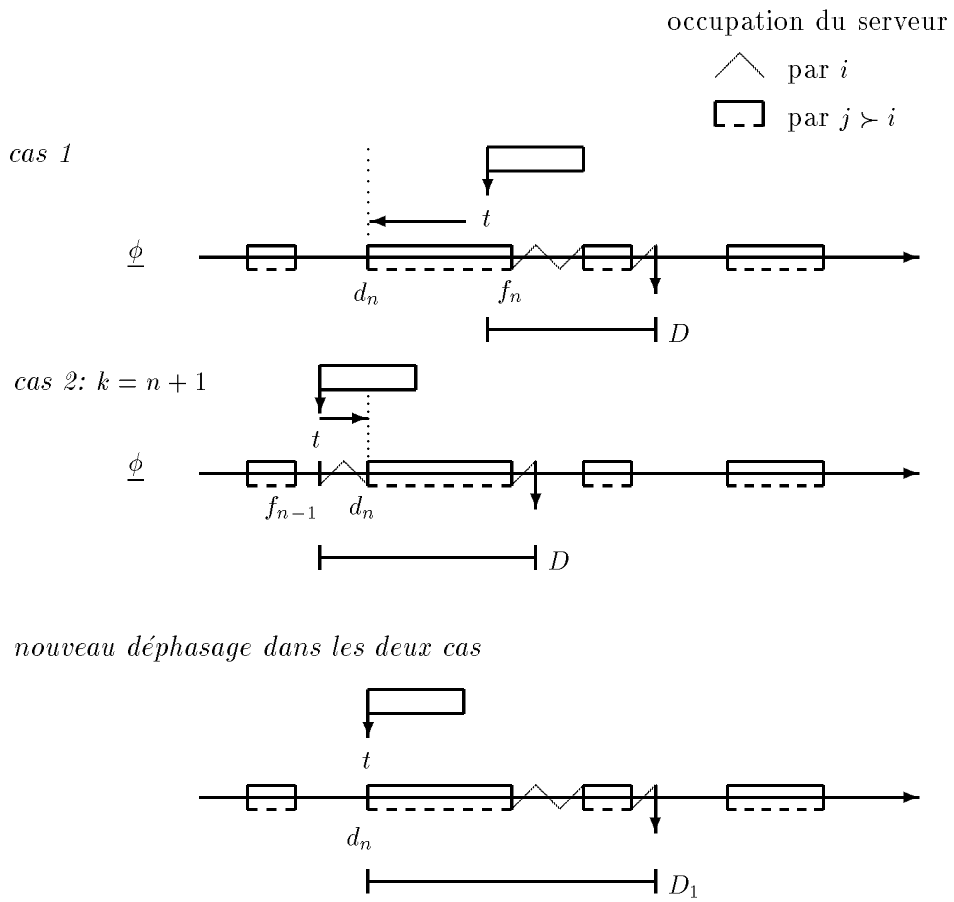


Figure 8 : augmentation du délai dans l'étape 1

L'étape 1 assure que juste avant l'instant de la requête i, la file ne contient aucune tâche plus prioritaire que i. Cela évite de reproduire le cas de la figure 7, et donc le délai subi par la requête i considérée augmente d'au moins

$$L_j(\lfloor D_1/R_j \rfloor - \lfloor (D_1 - (t_j - d_n))/R_j \rfloor).$$

En appliquant ce raisonnement à tous les types strictement plus prioritaires que i, le délai final obtenu est alors égal à Δ_i^0 et vérifie $\Delta_i^0 \geq D_1 \geq D > R_i$, ce qui achève la preuve. #

4.2.2 Priorité optimale

Rappelons que la priorité $\mathbf{R}^+$ accorde, à chaque instant t, la priorité à la tâche de plus petite récurrence parmi les tâches en attente à t.

L'intuition du résultat suivant provient de l'étude du cas $M = 2$. En effet dans le cas de 2 tâches, si l'on suppose $R_1 < R_2$, alors en accordant la priorité à la tâche 1, la faisabilité implique: $L_2 + \lfloor R_2/R_1 \rfloor L_1 \leq R_2$, et en accordant la priorité à la tâche 2, la faisabilité implique $L_2 + L_1 \leq R_1$. La seconde inégalité implique la première, cela fait penser que cet ensemble de tâches sera plus facilement faisable si la priorité est accordée à la tâche 1 qui est la plus récurrente.

Théorème 4.5 (Liu & Layland)
Si un ensemble de tâches τ est faisable par une quelconque priorité statique, alors il est faisable par la priorité $\mathbf{R}^+$.

La démonstration se fait par un argument d'échange.

Ce résultat permet de réduire la recherche d'une priorité statique par laquelle un ensemble donné τ pourrait être faisable, au simple test de faisabilité par $\mathbf{R}^+$.

4.2.3 Test par le taux de charge

La question est de savoir si il existe U_M vérifiant:

$$\forall \tau = (\tau_1, .., \tau_M) \;\; U(\tau) < U_M \Longrightarrow \tau \text{ faisable par } \mathbf{R}^+, \tag{15}$$

et si oui, de trouver le plus grand possible, noté U_M^*.

Théorème 4.6 (Liu & Layland)
$U_M^* = M(2^{1/M} - 1)$ *vérifie: pour tout ensemble de tâches τ de taille M, si $U(\tau) < U_M^*$ alors τ est faisable par $\mathbf{R}^+$, et c'est le plus grand vérifiant cette propriété.*

La suite U_M^* décroît et admet pour équivalent, quand $M \to \infty$, $\ln 2 \approx 0,69$. Nous en déduisons que si un ensemble de tâches a un taux de charge inférieur à ln2, alors il est faisable par la priorité $\mathbf{R}^+$. Ce test est encore plus simple à vérifier que le test par les trajectoires, mais il ne s'agit que d'une condition suffisante.

4.3 Cas dynamique

Ici les priorités restent préemptives, mais elles varient avec le temps.

Un premier résultat permet de tester la faisabilité par $\mathbf{DL}^+$ d'un ensemble de tâches τ à partir de l'observation d'une trajectoire (démarrant par une requête simultanée de toutes les tâches) jusqu'à la fin de la première période d'activité.

Lemme 4.7 (Liu & Layland)
Si τ n'est pas faisable par $\mathbf{DL}^+$, alors la contrainte temps-réel n'est pas satisfaite dès la première période d'activité *du serveur pour un déphasage initial nul, avec la priorité $\mathbf{DL}^+$.*

Un second résultat est un test de faisabilité fonction uniquement du taux de charge. Il montre que $\mathbf{DL}^{+}$ est *optimale* puisque la condition de stabilité suffit à assurer la faisabilité.

Théorème 4.8 (Liu & Layland)

$$\tau \textit{ est faisable par } \mathbf{DL}^{+} \Longleftrightarrow U(\tau) \leq 1 \; . \tag{16}$$

5 Ordonnancement en présence d'overhead

Nous nous proposons ici d'affiner le modèle étudié dans les sections précédentes en tenant compte de l'*overhead*, coût à payer avant chaque émission, qui se traduit par l'indisponibilité du serveur pendant une durée supposée constante, notée δ. Dans le cas PAPS, cette durée s'ajoute simplement à la durée d'émission, mais pour des politiques de service plus générales, elle intervient à chaque fois qu'un nouveau client est affecté au serveur.

Pour le système proposé par Thomson, l'overhead est un temps d'armement du serveur pour chaque nouveau message à émettre, sa durée est d'environ 300 μs.

L'objectif est d'établir des conditions nécessaires et/ou suffisantes à la faisabilité d'une messagerie par diverses disciplines de service (PAPS, priorités statiques ou non, préemptives ou non), quand l'overhead est pris en compte. Comme dans la section 4, nous avons cherché à caractériser la faisabilité par l'observation de trajectoires particulières (par exemple, celle obtenue avec un déphasage nul), ou par un test sur le taux de charge. Dans chaque famille de priorités envisagée, nous avons aussi cherché à obtenir une priorité *optimale* dans le sens de Liu et Layland (*cf.* paragraphe 4.1).

Nous étudions pour commencer le cas PAPS. Les résultats de la section 3 s'adaptent pour donner les conditions nécessaires et suffisantes de stabilité et de faisabilité.

Nous considérons ensuite le cas des priorités statiques, préemptives puis non préemptives. Le cas préemptif est problématique car la quantité totale d'overhead est difficile à caractériser. Nous obtenons cependant une condition suffisante de stabilité, et une caractérisation du plus long délai subi par un message. Nous n'avons pas réussi à caractériser la faisabilité par des trajectoires particulières, et encore moins par le taux de charge.

Par contre, dans le cas non préemptif, la condition de stabilité est la même que dans le cas PAPS. Nous obtenons également une caractérisation de la faisabilité par l'observation d'un nombre fini de trajectoires particulières sur une durée finie. Nous avons aussi montré qu'aucune des priorités classiques n'était optimale, et qu'on ne pouvait pas tester la faisabilité d'une messagerie uniquement par son taux de charge.

5.1 Ordonnancement PAPS avec overhead

Dans le cas de l'ordonnancement PAPS (comme dans tous les cas non préemptifs), les modèles avec overhead ont les mêmes trajectoires (au sens de la définition 4.2) que les modèles sans overhead avec même déphasage initial et même charge à l'origine, mais où l'on inclut la valeur de l'overhead dans la longueur des messages. Ce fait nous permet d'utiliser les résultats de la section 2, quoique les politiques de service ne soient plus conservatrices. Pour simplifier les notations, si l'overhead a pour valeur δ, nous définirons $L'_i = L_i + \delta \ \ \forall i \in I$. La condition de stabilité d'une messagerie dans le cas non préemptif s'écrit alors simplement:

$$\sum_{i \in I} \frac{L'_i}{R_i} < 1 \tag{17}$$

Dans le paragraphe suivant, l'étude de la charge "fictive" nous donne une condition nécessaire et suffisante de faisabilité. Nous montrons ensuite que la charge réelle peut dépasser Λ à cause de l'overhead.

5.1.1 Condition de faisabilité

Le *délai* subi par un message de type i est la durée écoulée entre la date de sa requête et la fin de son émission.

Définition 5.1 *La "charge fictive" $W'(t)$ est la* durée *entre t et la fin de l'émission du dernier des messages présents dans la file à l'instant t ($W'(t) = 0$ si la file est vide à l'instant t).*

Grâce à la remarque faite plus haut, nous pouvons transposer les résultats théoriques obtenus dans l'étude d'une station en isolation (section 2) sur la charge réelle W (dans le cas sans overhead), à la charge fictive W' (dans le cas avec overhead).

- W' est une fonction continue à gauche. Donc $W'(t^-) = W'(t)$, qui peut être différent de $W'(t^+)$.
- La courbe $W'(t)$ est périodique de période T, à partir d'un temps fini, que l'on peut déterminer.
- La charge fictive a une borne supérieure, qui est atteinte pour le déphasage initial nul:

$$W'(t) \leq \Lambda + M\delta \quad \forall t \geq 0 \quad \forall \underline{\phi}$$

De ces résultats on tire une condition nécessaire et suffisante de faisabilité:

Théorème 5.2 *Une messagerie $\mathcal{M}$ de taille M, est faisable par la discipline de service PAPS avec overhead δ si et seulement si*

$$\Lambda + M\delta \leq r,$$

où r est la plus petite des récurrences: $r = \inf_I R_i$.

Démonstration Montrons que cette condition est suffisante. Notons d le délai subi par un message de type i, arrivé à l'instant t. Alors $d = W'(t^+) \leq \Lambda + M\delta \leq r \leq R_i$ qui montre que la contrainte temps réel est respectée à tout instant et quel que soit le déphasage initial. La messagerie $\mathcal{M}$ est donc faisable: cela prouve que la condition est suffisante.

Pour montrer que cette condition est nécessaire, considérons le déphasage initial défini par: $\phi_j = 0^+$ et $\phi_i = 0 \ \forall i \neq j$ où j est tel que $R_j = r$. La première requête de type j est satisfaite après les premières requêtes de tous les autres types. Le délai de cette requête est donc égal à $\Lambda + M\delta$. Il faut que ce délai soit inférieur à $R_j = r$ pour que la messagerie soit faisable, d'où le résultat. #

Cette condition est plus difficile à satisfaire que celle obtenue dans le cas sans overhead ($\Lambda \leq r$), et l'est d'autant plus si l'overhead δ est important où si la taille M de la messagerie est grande.

5.1.2 Borne sur la charge

Rappelons que la charge réelle est la *quantité d'information* en attente d'émission dans la file (comprenant ce qu'il reste à émettre du message en cours d'émission, s'il y en a un).

Dans la section 2, nous avons montré que dans le cas sans overhead, avec une charge initiale nulle, la charge réelle reste inférieure à Λ (théorème 2.13). L'exemple suivant montre que dans le cas avec overhead, même avec une charge initiale nulle, la charge réelle peut dépasser Λ.

- $M = 6$, $\delta = 3$,
- $L_6 = 2$, $L_i = 1$ pour $i \leq 5$,
- $R_6 = 6$, $R_i = 1000$ pour $i \leq 5$,
- $\phi_6 = 0^+$, $\phi_i = 0$ pour $i \leq 5$,

La condition de stabilité est vérifiée puisque $\sum L'_i/R_i \simeq 0,85$. La figure 9 représente les requêtes des divers flux de messages numérotés de 1 à 6, ensuite la trajectoire correspondant au déphasage choisi pour que le message de type 6 soit servi en dernier, et enfin la courbe de charge réelle correspondante. Nous constatons que la charge réelle dépasse Λ pour certains t.

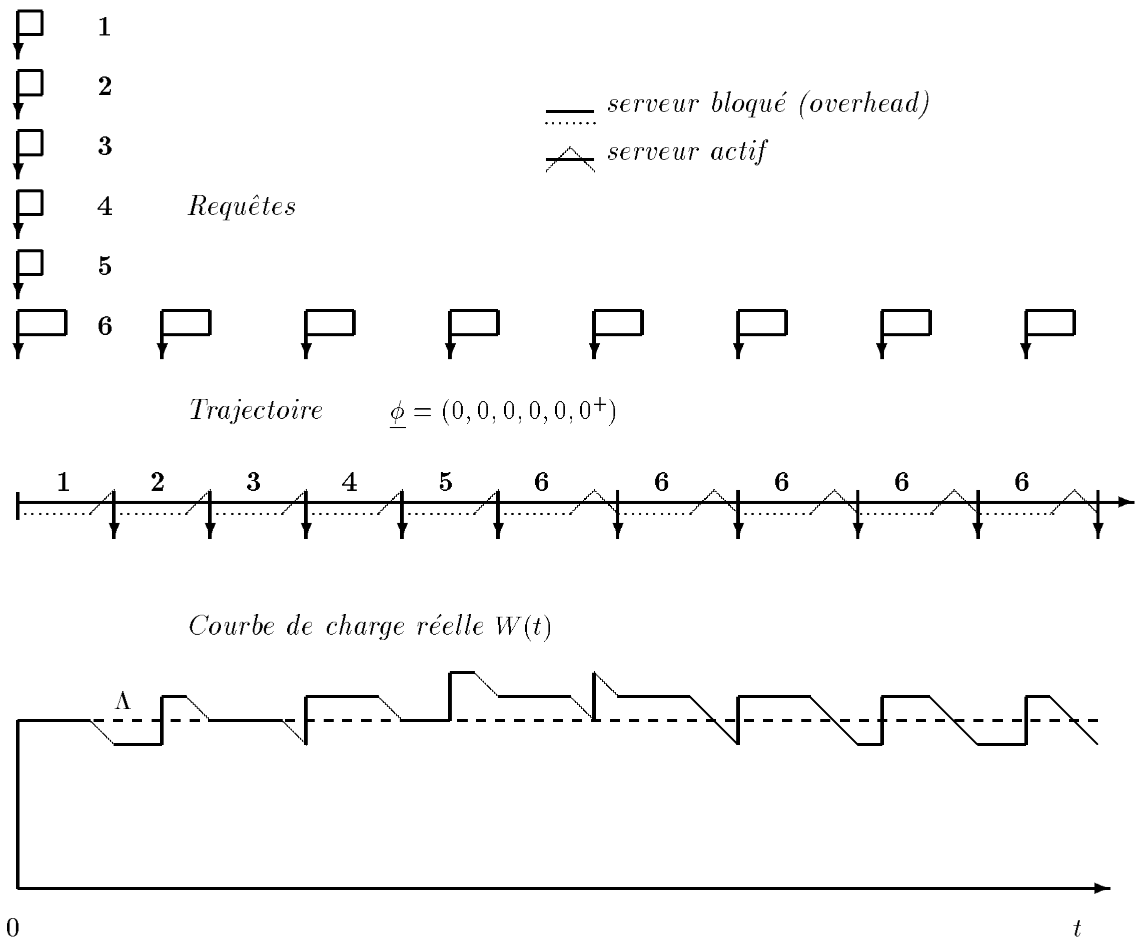


Figure 9 : PAPS avec overhead

5.2 Priorités préemptives

Quand il est le plus prioritaire de la file d'attente, un message de type i occupe le serveur d'abord inactif pendant δ puis actif pendant la durée de l'émission L_i (s'il n'y a eu aucune requête plus prioritaire pendant ce temps) enfin il libère le serveur. Si il y a une requête plus prioritaire avant $\delta + L_i$, que ce soit pendant l'overhead ou pendant l'émission, le message i est préempté: ce qui reste à émettre retourne dans la file d'attente jusqu'à être à nouveau le plus prioritaire.

Nous considérons donc que l'overhead est préemptable. On pourrait également considérer qu'il ne l'est pas. Dans l'exemple d'un environnement multiprocesseurs: si l'instruction de lecture ne peut être interrompue, l'overhead est non préemptable. A l'arrivée d'une tâche plus prioritaire que celle qui occupe le serveur, le processeur termine d'abord sa lecture courante avant de commencer la lecture correspondant à la nouvelle tâche. Il exécute cette tâche, puis reprend au début la lecture des données de la tâche précédente, si aucune tâche plus prioritaire n'est arrivée entre-temps. Inversement, si l'instruction de lecture peut être interrompue, alors l'overhead est préemptable. Dès qu'une tâche plus prioritaire que celle qui occupe le serveur en overhead (ou en exécution) entre dans le système, elle prend le serveur. Elle l'occupe pendant la durée de son overhead et de son service, jusqu'à être elle-même préemptée ou exécutée. Dans ce dernier cas, si aucune tâche plus prioritaire n'est arrivée entre-temps, la tâche précédente reprend le serveur pour la relecture complète de ses données et (la suite de) son exécution (si elle avait été interrompue en cours d'exécution).

L'overhead préemptable est plus difficile à implémenter, mais évite de gaspiller du temps en overhead inutile, et respecte à tout instant la hiérarchie des priorités (contrairement à l'overhead non préemptable). En effet, à tout instant, c'est la tâche la plus prioritaire de la file qui occupe le serveur. C'est pourquoi nous avons choisi ce modèle d'*overhead préemptable*, dans le cas de priorités préemptives.

Un message occupe donc le serveur pendant plusieurs intervalles de temps, dont la durée totale est la somme de sa durée d'émission et des différentes durées d'overhead qu'il subit à chaque début d'utilisation du serveur. Ces durées peuvent être inférieures à δ puisque dans notre modèle, la préemption peut avoir lieu pendant l'overhead.

5.2.1 Condition de stabilité

Un premier résultat est une condition suffisante de *stabilité*:

Proposition 5.3 *Une condition suffisante de stabilité pour une messagerie $\mathcal{M}$ et une priorité statique préemptive avec overhead est:*

$$\frac{L_M + \delta}{R_M} + \sum_{i=1}^{M-1} \frac{L_i + 2\delta}{R_i} < 1 \ . \tag{18}$$

Cette condition qui semble biscornue, s'avère être une condition nécessaire et suffisante pour les messageries de taille M quelconque, dont tous les types ont la même récurrence ($\forall i \in I \quad R_i = T$). Pour s'en convaincre il suffit de considérer le déphasage initial défini par: $\phi_i = (M - i)\delta \quad \forall i \in I$.

Démonstration Rappelons qu'une messagerie est stable si quels que soient l'état initial et l'instant t choisis, la file se vide à un instant fini, après t (*cf.* définition 2.3).

Sans perte de généralité, nous replaçons l'origine des temps à l'instant choisi. Donc à l'instant 0, il y a n_i^0 messages de type i en attente dans la file (dont certains peuvent avoir été préemptés), avec éventuellement un message en cours d'émission ou d'overhead (si $W(0) > 0$).

La démonstration se fait par l'absurde. Nous supposons que l'inégalité (18) est vérifiée et que le serveur ne se vide jamais, i.e. $W(s) > 0 \quad \forall s > 0$, où $W(s)$ est la charge réelle à l'instant s, c'est-à-dire la quantité d'information en attente d'émission à l'instant s.

Nous montrerions de façon analogue à la preuve de la proposition 2.1, en tenant compte maintenant de l'inactivité du serveur due à l'overhead, que la charge réelle vérifie l'équation fonctionnelle suivante:

$$W(t) = W(0) + S(0,t) + \int_0^t 1_{\{W(s)=0\}} ds + \int_0^t v(s) ds - t, \tag{19}$$

où $v(s) = 1$ si le serveur est bloqué en overhead à l'instant s (0 sinon) et $S(0, t)$ est la charge arrivée dans l'intervalle $[0, t[$.

Puisque l'on suppose le serveur toujours actif, $\int_0^t 1_{\{W(s)=0\}} ds = 0$.

L'équation (19) appliquée aux instants nT, avec n entier, donne:

$$W(nT) = W(0) + nS + \int_0^{nT} v(s)ds - nT, \tag{20}$$

où (*cf.* section 2) S est la charge arrivée dans une période, i.e. $S = \sum_i N_i L_i = T \sum_i L_i / R_i$, avec N_i nombre de requêtes de type i dans une période T.

Le terme $\int_0^{nT} v(s)ds$ se majore en remarquant que chaque overhead a une durée inférieure à δ. Il ne reste plus qu'à majorer le nombre d'overheads dans l'intervalle $[0, nT[$. Pour cela nous distinguons les overheads dus à une première occupation du serveur après l'instant 0, des autres, dus à une reprise du serveur après une préemption qui a eu lieu après 0. Les premiers sont en nombre inférieur ou égal à la somme du nombre de messages présents dans la file à l'instant 0 et du nombre de messages arrivés dans l'intervalle $[0, nT[$, c'est-à-dire $\sum_1^M N_i^0 + n \sum_1^M N_i$.

Les seconds sont en nombre inférieur ou égal au nombre total de préemptions, qui est le nombre de messages qui à leur arrivée, sont plus prioritaires que le message présent dans le serveur. Ce nombre est donc majoré par le nombre total d'arrivées de messages de type autre que M (le moins prioritaire), c'est-à-dire $n \sum_1^{M-1} N_i$.

Finalement nous avons une borne supérieure de la charge réelle aux instants nT qui est:

$$W(nT) \leq W(0) + \delta \sum_1^M N_i^0 + n(\delta(N_M + 2 \sum_{i=1}^{M-1} N_i) + S - T) \ . \tag{21}$$

Soit $K = \delta(N_M + 2 \sum_{i=1}^{M-1} N_i) + S - T$. En remplaçant maintenant S et N_i par leurs valeurs, et en factorisant par T, nous obtenons $K < 0$ d'après la condition (18). Puisque la quantité $W(0) + \delta \sum_1^M N_i^0$ est fixe, $W(nT) \longrightarrow -\infty$ quand $n \longrightarrow +\infty$, ce qui est contradictoire. #

5.2.2 Plus long délai subi

Pour caractériser la faisabilité d'une priorité préemptive avec overhead, nous avons cherché à déterminer pour chaque type de message i, un déphasage initial correspondant au cas le "pire".

La proposition 5.4 donne un moyen d'augmenter un délai (la démarche est celle utilisée dans l'étape 1 de la démonstration du théorème 4.4). Les remarques qui suivent montrent pourquoi on ne peut pas adapter la caractérisation de la faisabilité (*cf.* théorème 4.4) au cas avec overhead.

Remarquons tout d'abord que pour un message de type 1 (le plus prioritaire), le délai est constant et égal à $L_1 + \delta$, quels que soient le déphasage initial et la requête considérés. Dans la suite de ce paragraphe, nous ne nous intéresserons donc plus qu'aux délais de messages de type $i \prec 1$. Notons que, la priorité étant préemptive, un message n'a aucune influence sur la trajectoire des messages plus prioritaires que lui. Les messages moins prioritaires que i ne seront donc pas pris en compte.

Le déphasage initial des messages plus prioritaires que i est supposé fixé.

Soient d_n et f_n les dates de début et fin de la $n^{ième}$ période d'occupation du serveur par des messages plus prioritaires que i, avec ce déphasage.

Soient les instants $a_1 = \sup(0, d_1 - \delta)$ et $a_n = d_n - \delta$ pour $n > 1$.

Le délai subi par un message de type i dont la requête est à l'instant t sera noté $\Delta(t)$. Le déphasage initial du message i sera noté ϕ_i, cela impose la relation $\phi_i = t$ modulo R_i.

Proposition 5.4 *Soient $n > 0$ et $t \in]f_{n-1}, f_n]$. Si la contrainte temps-réel est satisfaite sur l'intervalle $[0, t]$ pour le type i, alors $\Delta(t) \leq \Delta(a_n)$.*

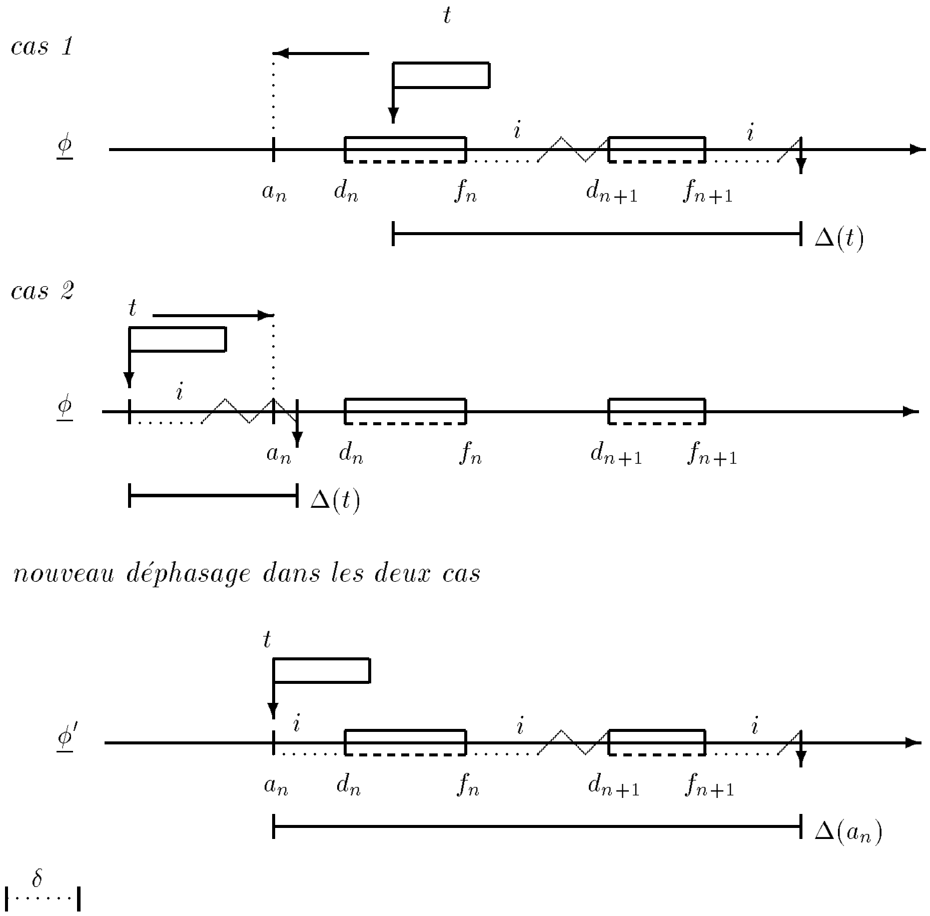


Figure 10 : augmentation du délai

Démonstration Nous allons montrer que si t n'est pas l'instant a_n, alors le délai $\Delta(t)$ peut être augmenté. Les différents cas sont illustrés sur la figure 10:

cas 1 si $t \in]a_n, f_n]$, alors puisque la contrainte temps-réel est satisfaite sur $[0, t]$, il n'y a aucun message de type i dans la file à l'instant d_n (sinon ce message ne pourrait pas avoir fini son émission à l'instant t).

En remplaçant ϕ_i par a_n modulo R_i, la date de la requête est alors avancée à l'instant a_n, et la trajectoire des plus prioritaires que i n'est pas perturbée. Le délai de la requête augmente donc de $t - a_n$, ce qui donne $\Delta(a_n) > \Delta(t)$.

cas 2 si $t \in]f_{n-1}, a_n[$, nous noterons $k \geq n$ l'indice tel que la date de fin d'émission du message $(t + \Delta(t))$ est dans l'intervalle $]f_{k-1}, d_k]$.

Si la date de la requête est retardée à l'instant a_n, deux sous-cas se présentent:

- si $d_n + \Delta(t) \leq d_k$, le message ne subit pas d'autre préemption, et le délai est inchangé: $\Delta(a_n) = \Delta(t)$.
- sinon (en particulier si le message n'était pas preempté), $d_n + \Delta(t) > d_k$, le message subit au moins une préemption supplémentaire à l'instant d_k. Cela a pour effet d'augmenter son délai d'au moins $f_k - d_k + \delta$, ce qui donne $\Delta(a_n) > \Delta(t)$.

Dans tous les cas si t n'est pas l'instant a_n, le délai peut toujours être augmenté. Et cela achève la preuve.
#

Remarque 5.5 En revanche, l'argument utilisé dans l'étape 2 de la démonstration du théorème 4.4 n'est plus vrai dans ce cas, à cause de l'overhead: *le délai d'un message n'est plus systématiquement augmenté quand on avance la première requête d'un message plus prioritaire*, pour la synchroniser. Ce délai peut même être diminué comme le montre l'exemple suivant:

- $M = 3, \quad \delta = 1$
- $\underline{L} = (\ 2,\ 2,\ 3)$
- $\underline{R} = (10, 10, 30)$

La figure 11 montre qu'avec le déphasage initial $\underline{\phi} = (1, 5, 0)$, le premier délai de type **3** vaut 20, tandis qu'avec le déphasage initial $\underline{\phi} = (1, 1, 0)$ qui avance la requête de type **2** (plus prioritaire) au début de la période d'activité des plus prioritaires, le délai de type **3** est diminué à 11.

Remarque 5.6 La proposition 5.4 fait penser que le premier délai, noté D_i^0, subi par un message de type i, pour un déphasage initial $\phi_i = (M - i)\delta$, pourrait être un bon candidat pour tester la faisabilité d'une messagerie de taille M par la priorité $1 \succ 2 \succ .. \succ M$. Il n'en est rien, comme le montre l'exemple suivant:

- $M = 3, \quad \delta = 1$
- $\underline{L} = (\ 1,\ 1,\ 1)$
- $\underline{R} = (\ 6, 10, 10)$

La condition suffisante de stabilité est bien vérifiée.

Sur le schéma du haut de la figure 12, nous avons représenté les requêtes et la trajectoire correspondant au déphasage initial $\underline{\phi} = (2, 1, 0)$. Cela permet de calculer $\underline{D}^0 = (\ 2,\ 5,\ 8)$.

Sur le schéma du bas, nous constatons que la contrainte temps-réel n'est pourtant pas satisfaite pour le type 3 à $t = 17$, quand on prend pour déphasage initial $\underline{\phi} = (2, 1, 7)$.

Ces deux remarques montrent que la faisabilité d'une messagerie par une priorité est difficile à caractériser dans le cas avec overhead, et ce à cause de la préemption. Cela nous amène à étudier le cas non préemptif.

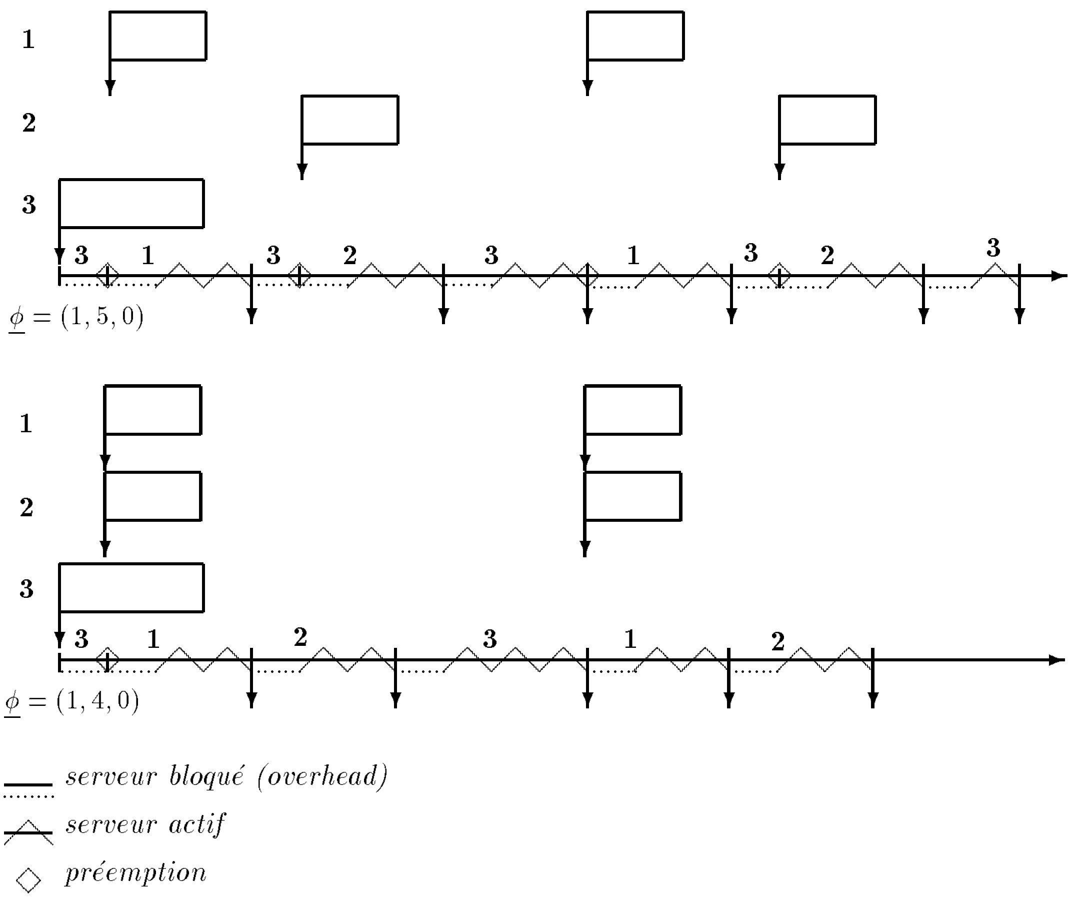


Figure 11 : synchronisation des plus prioritaires

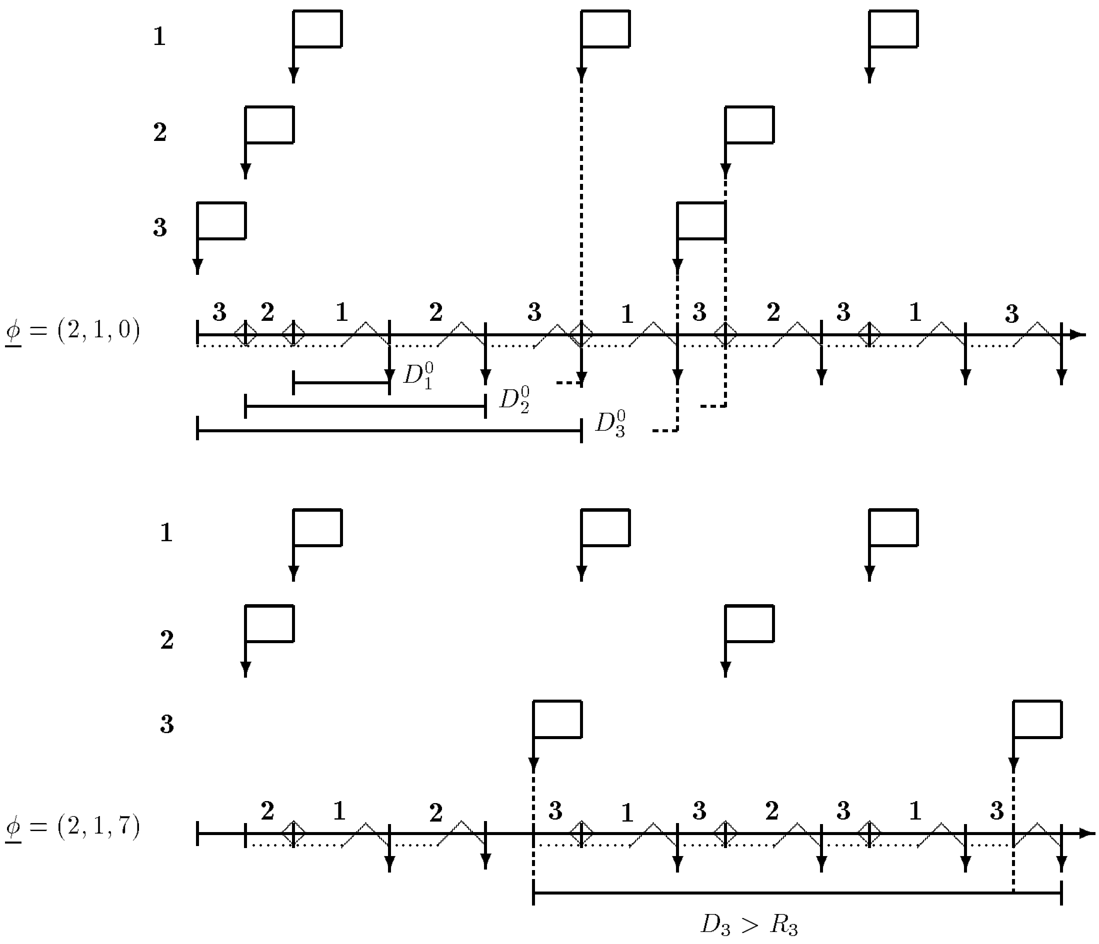


Figure 12 : le plus long délai n'est pas D^0

5.3 Priorités non préemptives

Ici, les services ne sont plus interrompus, et donc

- contrairement à ce qui se passait dans le cas préemptif, un message peut être retardé par un message moins prioritaire, si ce dernier est déjà en train d'occuper le serveur lors d'une requête du premier;
- quand un message commence à occuper le serveur, c'est pour une durée ininterrompue qui est la somme de l'overhead et de sa longueur.

Un premier résultat est un *test de faisabilité* pour la priorité **P**, utilisant des débuts de trajectoires spéciales. Un second résultat, plutôt négatif, concerne la recherche d'une *priorité optimale*: nous montrons qu'aucune des priorités classiques n'est optimale pour notre critère. Un troisième résultat, lui aussi négatif, est qu'il n'existe pas de borne U_M^* pour le taux de charge d'une messagerie, en deçà de laquelle on serait assuré de la faisabilité de cette messagerie par une priorité statique non préemptive.

Comme dans le cas PAPS (paragraphe 5.1), les trajectoires du système se ramènent à celles d'un système sans overhead dont les messages ont des longueurs modifiées. La condition de stabilité est donc (*cf.* (17)): $\sum_I L_i'/R_i < 1$.

5.3.1 Test par des trajectoires

Considérons la priorité **P** définie par $1 \succ 2 \succ .. \succ M$ (i.e. 1 est le type de flux le plus prioritaire). Nous inspirant des résultats de Liu et Layland, nous cherchons ici à obtenir un test de faisabilité qui se base sur l'observation de certaines trajectoires, définies par des déphasages initiaux bien particuliers.

Nous utiliserons par la suite la notation t^+ pour désigner un instant strictement postérieur à t mais aussi proche que l'on veut de t. Cet artifice de notation sera utilisé pour définir des déphasages particuliers avec des arrivées quasi simultanées. Ces déphasages correspondent aux cas les "pires", et sont utilisés pour tester la faisabilité des messageries.

Définition 5.7 *Pour le type de message i:*

- *le déphasage initial $\underline{\phi}^i$ est défini pour $i < M$, par: si j est le plus petit indice du plus long message moins prioritaire que i (i.e. $j = \inf\{l \prec i \ tq \ L_l = \sup_{\{k \prec i\}} L_k\}$), $\phi_j^i = 0$ et pour les autres $k \neq j$ $\phi_k^i = 0^+$. Pour $i = M$, $\underline{\phi}^M = (0, .., 0, 0)$.*
- *T_i le premier instant où la file ne contient plus que des messages moins prioritaires que i, avec le déphasage initial $\underline{\phi}^i$, et la priorité **P**. La file étant stable, cet instant est fini. Il est en fait borné par T la période du système, car $W(0) = 0$.*
- *$\Delta_i = \sup_n \ D_i^n$ où D_i^n est le délai subi par la $n^{ième}$ requête du type i dans l'intervalle $[0, T_i]$.*

Les vecteurs de dimension M et de $i^{ième}$ coordonnées T_i et Δ_i seront notés respectivement $\underline{T}$ et $\underline{\Delta}$.

Pour fixer ces notations, prenons l'exemple suivant:

- $M = 4$, $\delta = 0$,
- $\underline{L} = (\ 2,\ 2,\ 5,\ \ 5)$,
- $\underline{R} = (\ 7, 17, 16, 100)$,

La condition de stabilité est vérifiée puisque $\sum_i L_i'/R_i \simeq 0,77$.

Pour la priorité **P**, les différents déphasages initiaux sont:

- $\underline{\phi}^1 = (0^+, 0^+, 0\ , 0^+)$
- $\underline{\phi}^2 = (0^+, 0^+, 0\ , 0^+)$

- $\underline{\phi}^3 = (0^+, 0^+, 0^+, 0\ )$
- $\underline{\phi}^4 = (0\ , 0\ , 0\ , 0\ )$

La figure 13 représente pour chaque type i les requêtes successives et le début de la trajectoire du système correspondant au déphasage initial $\underline{\phi}^i$ avec la priorité $\mathbf{P}$.

Cela permet de calculer le vecteur $\underline{T} = (\ 7, 11, 27, 27)$.

Seul le message $i = 3$ a deux requêtes dans l'intervalle $[0, T_i]$, et les valeurs des délais sont $D_1^1 = 7$; $D_2^1 = 11$; $D_3^1 = 16$, $D_3^2 = 9$; $D_4^1 = 16$ et donc le vecteur$\underline{\Delta}$ vaut $(\ 7, 11, 16, 16)$.

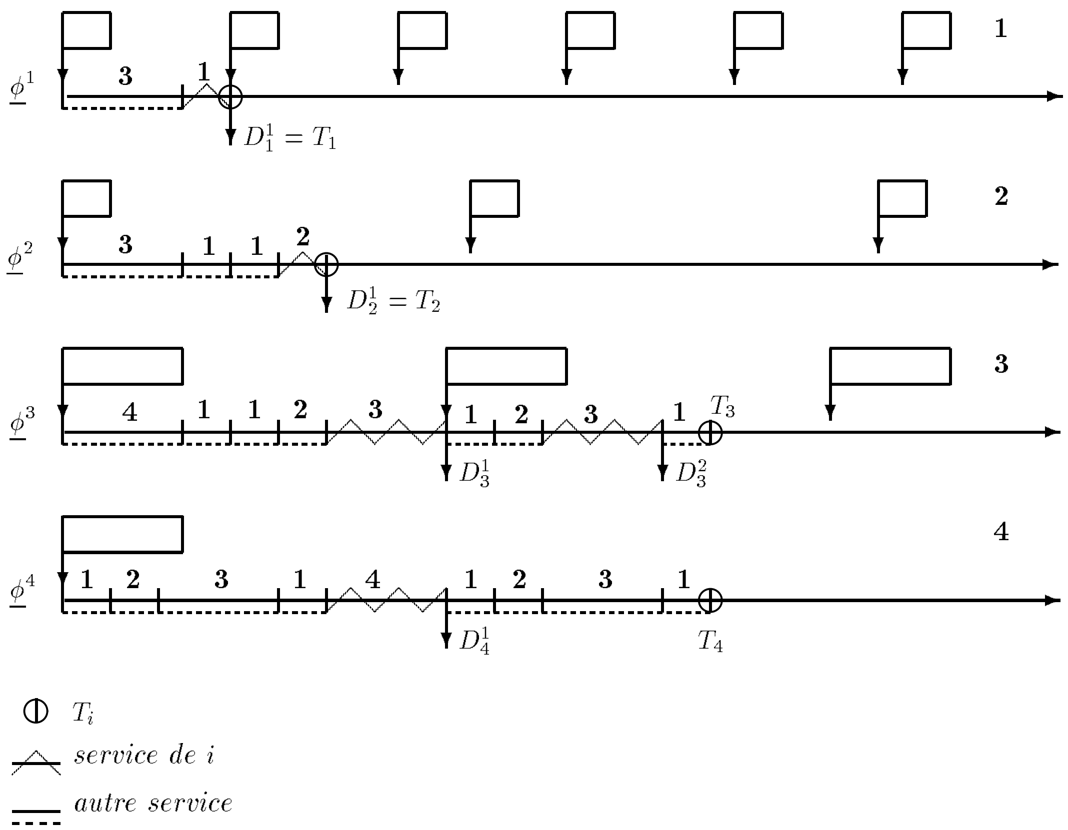


Figure 13 : calcul de $\underline{T}$ et $\underline{\Delta}$

Le théorème 5.9 permettra de montrer que cette messagerie est faisable.

Nous présentons d'abord une condition nécessaire à la faisabilité d'une messagerie.

Proposition 5.8 *Une condition nécessaire à la faisabilité d'une messagerie $\mathcal{M}$ par la priorité non préemptive* $\mathbf{P}$ *est:* $Y_i \leq R_i \quad \forall i \in I$, *où*

$$Y_i = 1_{\{i \neq M\}} \sup_{k > i} L'_k + \sum_{j \leq i} L'_i$$

Démonstration En effet Y_i est la somme des longueurs des messages plus prioritaires que i, du message i, et éventuellement du plus long des messages moins prioritaires que i (si $i < M$).

Le premier délai subi par un message de type i quand on applique le déphasage initial $\underline{\phi}^i$, est donc minoré par Y_i ($D_i^1 \geq Y_i$). Par ailleurs, il faut vérifier $D_i^1 \leq R_i$ pour que la messagerie soit faisable par $\mathbf{P}$, d'où le résultat. #

Théorème 5.9 *Une messagerie $\mathcal{M}$ est faisable par la priorité statique, non préemptive* $\mathbf{P}$ *si et seulement si les délais Δ_i calculés en partant des déphasages initiaux $\underline{\phi}^i$ et en appliquant* $\mathbf{P}$ *sont inférieurs aux récurrences R_i:*

$$\Delta_i \leq R_i \quad \forall i \in I \iff \textit{la messagerie } \mathcal{M} \textit{ est faisable par } \mathbf{P} \tag{22}$$

Démonstration Le sens ($\Leftarrow$) est assez simple à démontrer. Si $\mathcal{M}$ est faisable, alors pour tout type, le délai de toute requête est inférieur à la récurrence quel que soit le déphasage initial considéré. Donc $D_i^n \leq R_i$, $\forall i \in I$ et $\forall n \geq 1$, et par extension $\Delta_i \leq R_i$, $\forall i \in I$.

Nous démontrons le sens ($\Rightarrow$) en utilisant la contraposée. Supposons qu'il existe un déphasage initial $\underline{\phi}$ tel que la contrainte temps-réel n'est pas respectée pour une requête de type i, et montrons qu'elle n'est alors pas non plus respectée pour une requête de ce type dans l'intervalle $[0, T_i]$ avec le déphasage $\underline{\phi}^i$. Pour cela on passe progressivement d'une trajectoire à l'autre en s'assurant à chaque étape de conserver une requête insatisfaite. Dans une première étape nous nous ramenons à la première période d'activité du serveur. La deuxième étape ramène les déphasages initiaux de tous les types sauf i à leur valeur pour $\underline{\phi}^i$. Enfin le déphasage initial du type i est ramené à ϕ_i^i dans la troisième étape.

On présente en annexe un exemple illustrant les différentes étapes de cette démonstration.

Etape 1 Pour se ramener à une première période d'activité du serveur, on va replacer l'origine des temps à un instant t_0 où aucun message de type $k \succeq i$ n'est en attente dans la file.

On note t l'instant du "dépassement" avec $\underline{\phi}$: l'émission du message de type i arrivé à la date $t - R_i$ n'est pas terminée à l'instant t, date de la prochaine requête de type i. Soit d le début de la période d'occupation du serveur par des messages de type $k \succeq i$, qui contient t. Plusieurs cas de figure se présentent:

cas 1: $i \succ M$ et juste avant d, un message de type $k \succeq i$ est en attente. Ce message est nécessairement arrivé pendant l'émission d'un message strictement moins prioritaire (de type noté j) qui se termine à d.

cas 2: $i \succ M$ et la file ne contient aucun message de type $k \succeq i$ juste avant d,

cas 3: $i = M$, le type i est le moins prioritaire et donc d est le début d'une période d'activité du serveur: la file est vide juste avant d.

Pour qu'à l'instant t_0 aucun message plus prioritaire que i ne soit en attente, t_0 est défini par:

cas 1: $t_0 = d - L'_j$. On sait qu'aucun message de type $k \succeq i$ n'est en attente dans la file juste avant t_0, puisqu'à cet instant c'est un message de type $j \prec i$ qui commence son service,

sinon: $t_0 = d$. Il a la même propriété que d qui est le début de la période d'occupation du serveur par des messages plus prioritaires que i.

Sur l'intervalle $[t_0, t]$ avec $\underline{\phi}$, le serveur n'est jamais vide et de plus aucun message moins prioritaire que i ne l'occupe, sauf le premier message dans le cas 1. On notera cette propriété (*).

On définit maintenant le déphasage initial $\underline{\psi}^1$ par:

- pour $k \succeq i$, $\psi_k^1 = \phi_k - t_0$ modulo R_k, dans tous les cas (on remarque que $\psi_k^1 > 0$ $\forall k$ dans le cas 1, et $\exists k$ tq $\psi_k^1 = 0$ dans les autres cas),
- pour $l \prec i$
  - dans le cas 1: $\psi_j^1 = 0$ et $\psi_l^1 = 0^+ \quad \forall l \neq j$,

– dans le cas 2: $\psi_l^1 = 0^+ \quad \forall l$.

On a choisi $\underline{\psi}^1$ pour que le serveur ait exactement la même *trajectoire* (même suite de services réalisés) sur l'intervalle $[0, A]$ en partant du déphasage $\underline{\psi}^1$ avec $A = t - t_o$, que sur l'intervalle $[t_o, t]$ en partant de $\underline{\phi}$.

La requête de date $A - R_i$ avec $\underline{\psi}^1$ n'est donc toujours pas satisfaite à temps.

De plus sur l'intervalle $[0, A]$ avec $\underline{\psi}^1$ la propriété (*) est conservée (le serveur n'est jamais vide, ni occupé par un message moins prioritaire que i, sauf dans le cas 1). Cette propriété s'écrit aussi dans ce cas:

$$A \leq \inf\{x > 0 \mid S_1(0, x) - x + \epsilon L'_j \leq 0\} \tag{23}$$

où $\epsilon = 1$ dans le cas 1 (0 sinon), et $S_1(0, x)$ est la charge fictive due aux messages de type $k \succeq i$ arrivés dans l'intervalle $[0, x[$ avec le déphasage $\underline{\psi}^1$, i.e.

$$S_1(0, x) = \sum_{k \succeq i} \left\lceil \frac{x - \psi_k^1}{R_k} \right\rceil L'_k$$

Par ailleurs, il est équivalent de dire:

(i) que la requête d'un message de type i, de date $a = A - R_i$ n'est pas satisfaite,

(ii) que le message n'a pas encore fini son émission à la date A,

(iii) que le message n'a pas encore commencé son émission à la date $b = A - L'_i$,

(iv) que sur l'intervalle $[a, b]$ le message i ne s'est jamais trouvé seul dans la file,

(v) que, d'après la propriété (*):

$$\forall x \in [a, b],\ S_1(0, x) - x + \epsilon L'_j > L'_i\ . \tag{24}$$

Etape 2 Nous allons faire arriver à l'instant 0 le message de type i_0 défini par:

cas 3: $i_0 = 1$,

sinon: i_0 est l'indice d'un message strictement moins prioritaire que i, tel que $L_{i_0} = \sup_{j \prec i} L_j$.

et tous les autres messages sont ramenés le plus tôt possible en s'assurant que ce soit bien le message de type i_0 qui occupe le serveur en premier. Le déphasage du message de type i est inchangé s'il est non nul, mais il est mis à 0^+ sinon, faute de quoi il prendrait le premier service à la place du message de type i_0.

Le déphasage $\underline{\psi}^2$ est défini par

- $\psi_{i_0}^2 = 0$,
- $\psi_i^2 = \sup(\psi_i^1, 0^+)$,
- pour $k \neq i,\ \neq i_0$,

cas 3: $\psi_k^2 = 0$,
sinon: $\psi_k^2 = 0^+$.

Un message de type i_0 est ainsi présent dans la file à l'instant 0. Dans le cas 3, il y est le plus prioritaire et dans les autres cas, il y est seul: dans tous les cas c'est donc bien lui le premier servi.

On vérifie que $\forall k \preceq i \ \ \psi_k^2 = 0$ ou 0^+alors que $\psi_k^1 \geq 0$. On en déduit que si l'on définit $S_2(0, x)$ avec ψ^2 de la même façon que $S_1(0, x)$ avec ψ^1, alors $S_2(0, x) \geq S_1(0, x)$. Par ailleurs on a $L'_{i_0} \geq \epsilon L'_j$. Les inéquations (23) et (24) restent donc vérifiées dans cette étape, ce qui permet de conclure que l'intervalle $[0, A]$ avec $\underline{\psi}^2$ conserve la propriété (*), et que la requête de type i de date a n'est toujours pas satisfaite.

Etape 3 On considère enfin le déphasage $\underline{\phi}^i$, qui revient simplement à ramener la première requête de type i à 0 dans le cas 3, et à 0^+ dans les autres cas. On montre que ce changement ne perturbe pas la *trajectoire* sur $[0, A]$. Pour cela on fait un raisonnement par récurrence sur l'indice des instants de début de service (notés d_n).

Pour le premier ($d_0 = 0$), avec $\underline{\phi}^i$ comme avec $\underline{\psi}^2$, c'est un message de type i_0 qui est le premier servi. Pour les autres, si la trajectoire est la même jusqu'à d_n, alors à cet instant on retrouve en attente dans la file pour $\underline{\phi}^i$ les mêmes messages de type $k \succ i$ et peut-être un message de type i (dont la date de requête a été avancée) de plus que pour $\underline{\psi}^2$. Mais le choix du plus prioritaire présent reste le même puisque l'éventuel message supplémentaire est le moins prioritaire, et que la file n'est jamais vide sur $[0, A]$ avec $\underline{\psi}^2$ d'après la propriété (*).

On conclut non sans peine que l'intervalle $[0, A]$ conserve la propriété (*) avec $\underline{\phi}^i$, ce qui peut aussi s'écrire: $A \leq T_i$. De plus la requête (cette fois de date $a - \psi_i^2$) n'est toujours pas satisfaite à l'instant A. Finalement une requête de type i dont la date est dans $[0, T_i]$ avec $\underline{\phi}^i$ n'a pas été satisfaite à temps.

On a ainsi montré que dans tous les cas, si une requête de type i n'est pas satisfaite pour un déphasage quelconque $\underline{\phi}$ alors $\Delta_i > R_i$, ce qui achève la preuve du théorème. #

Remarque 5.10 Ce test nécessite l'observation de M trajectoires, sur une durée qui est inférieure à la longueur de la première période d'activité du système. D'après la proposition 2.16, cette période est majorée par $\Lambda/(1-\lambda)$. Si la charge du système est trop proche de 1, cette borne est moins bonne que S, qui est alors de l'ordre de la période T qui peut être elle même très grande. Nous nous sommes donc demandés s'il était possible de restreindre cette observation. La proposition 5.8 montre qu'une messagerie n'est pas faisable sitôt que $\exists i \in I$ tel que $Y_i > R_i$. Plus généralement, si au cours de l'observation d'une trajectoire partant de $\underline{\phi}^i$ on constate un dépassement ($\exists n$ tel que $D_i^n > R_i$), on peut conclure directement que la messagerie n'est pas faisable, sans poursuivre l'observation jusqu'à T_i.

Par contre, on ne peut plus se contenter d'observer uniquement le délai de la première requête pour chaque type, comme c'est le cas pour le théorème 4.4. Pour s'en convaincre, observons l'exemple suivant:

- $M = 5$, $\delta = 1$
- $\underline{L} = (\ 1,\ 1,\ 1,\ 1,\ 5,\ 1)$
- $\underline{R} = (17, 15, 13, 11, 16, 1000)$

La condition de stabilité est vérifiée: $\sum_i \ L'_i/R_i \simeq 0,96$. Sur la figure 14, on voit qu'avec le déphasage initial $\underline{\phi}^5$, la première requête du message 5 est satisfaite à temps, ($D_5^1 = 16 \leq R_5$), mais pas la deuxième ($D_5^2 = 18 > R_5$).

L'étude exhaustive du cas $M = 2$ nous a permis d'obtenir une condition nécessaire et suffisante, explicite:

Proposition 5.11 *Une messagerie $\mathcal{M}$ de taille 2 est faisable par une priorité statique, non préemptive* **P** *si et seulement si l'inégalité suivante est vérifiée:*

$$L'_1 + L'_2 \leq \inf(R_1, R_2) \tag{25}$$

Démonstration On notera $L = L'_1 + L'_2$ et $r = \inf(R_1, R_2)$.

On se donne $i \in \{1, 2\}$. Avec le déphasage initial $\underline{\phi}^i$ qui est défini ici par $\phi_i = 0^+$ et $\phi_j = 0$ pour $j \neq i$, le serveur est d'abord occupé par le message j pendant L'_j unités de temps, puis par le message i pendant L'_i unités de temps. Partant de l'hypothèse $L \leq r$, on sait qu'il n'y a pas d'autre requête du message i ou j avant L, et donc à l'instant L, la file se vide pour la première fois. On en déduit que $T_i = L$ et donc $\Delta_i = D_i^1 = L$. L'hypothèse $L \leq r$ sert de nouveau pour obtenir l'inégalité $\Delta_i \leq R_i$ et ce $\forall i \in \{1, 2\}$, d'où l'on déduit la faisabilité de $\mathcal{M}$ par **P** grâce au théorème 5.9.

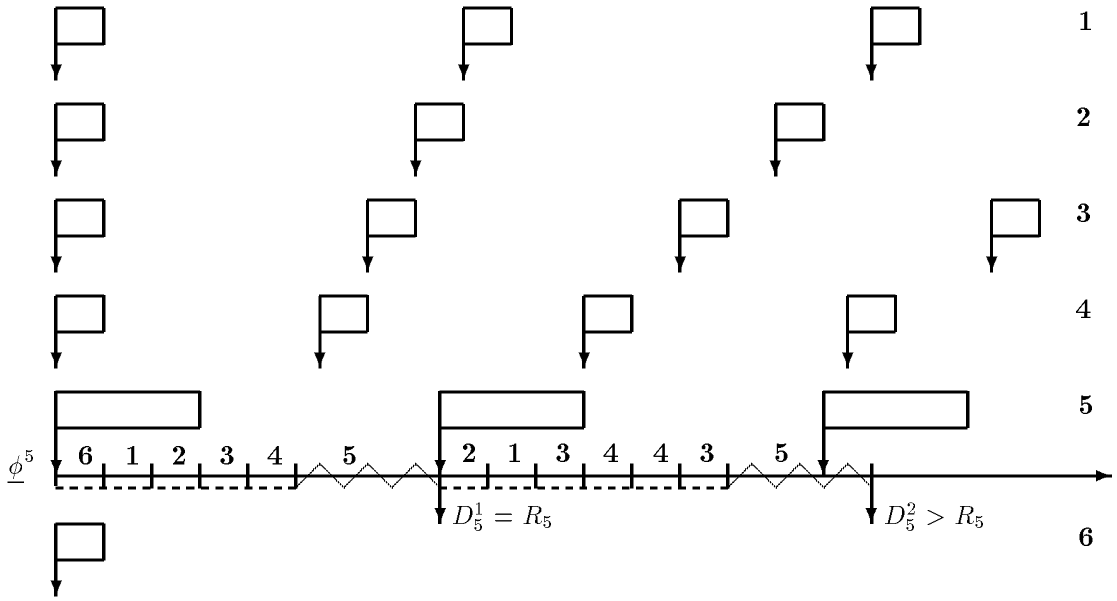


Figure 14 : la $1^{ère}$ requête ne permet pas de conclure

Réciproquement, si $\mathcal{M}$ est faisable par $\mathbf{P}$, on a $\Delta_i \leq R_i \quad \forall i$. Par ailleurs, la proposition 5.8 donne $\Delta_i \geq L'_1 + L'_2$, ce qui permet de conclure que $L \leq r$. #

Remarque 5.12 Dans le cas d'une messagerie de taille 2, les ordonnancements PAPS, $\mathbf{1} \succ \mathbf{2}$, $\mathbf{2} \succ \mathbf{1}$ non préemptifs sont équivalents, au sens où ces priorités donnent les mêmes conditions de stabilité et de faisabilité.

5.3.2 Priorité optimale

Dans ce paragraphe nous montrons qu'aucune des priorités statiques classiques n'est optimale selon notre critère. Nous avons étudié les priorités statiques les plus classiques, "optimales" pour d'autres critères, définis à partir de fonctions de coût [12,6].

- $\mathbf{R}^+$, qui accorde la plus grande priorité au message le plus récurrent,
- $\mathbf{L}^+$, qui accorde la plus grande priorité au message le plus court,
- $(\mathbf{R}-\mathbf{L})^+$, qui accorde la plus grande priorité au message pour lequel la valeur de $R - L$ est la plus faible.

Proposition 5.13 *Ni* $\mathbf{R}^+$*, ni* $\mathbf{L}^+$*, ni* $(\mathbf{R}-\mathbf{L})^+$ *n'est optimale, dans le sens où pour chacune d'elles on peut trouver une messagerie infaisable par cette priorité, et pourtant faisable par une autre priorité statique et non préemptive.*

Démonstration
Les priorités $\mathbf{R}^+$ et $(\mathbf{R}-\mathbf{L})^+$ coïncident sur la messagerie illustrée par la figure 13. Le théorème 5.9 permet de montrer que cette messagerie est faisable par $\mathbf{P}$.

Si on lui applique maintenant la priorité $\mathbf{R}^+$ ou $(\mathbf{R}-\mathbf{L})^+$, i.e. $\mathbf{1} \succ \mathbf{3} \succ \mathbf{2} \succ \mathbf{4}$, la figure 15 montre qu'avec le déphasage initial $\underline{\phi}^2 = (0^+, 0^+, 0^+, 0)$, le délai $D_2^1 = 25 > 17 = R_2$. La contrainte temps-réel n'est donc pas satisfaite pour le message de type $\mathbf{2}$.

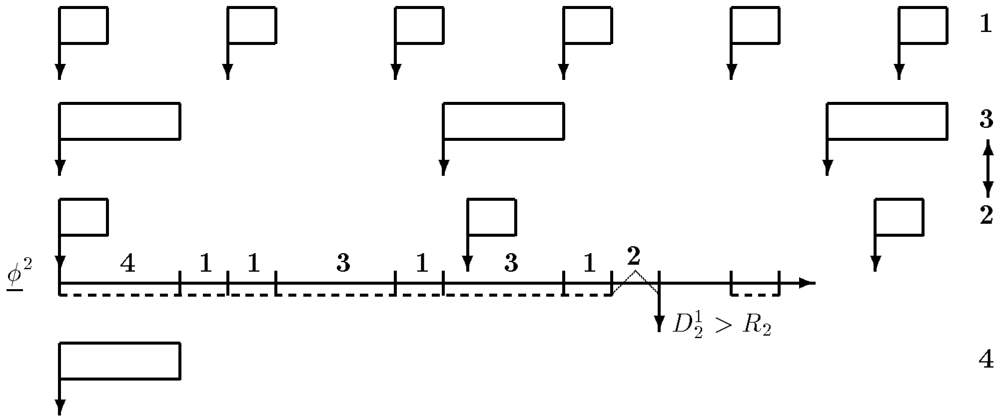


Figure 15 : non-faisabilité par $\mathbf{R}^+$

Le second exemple concerne la priorité $\mathbf{L}^+$:

- $M = 4$, $\delta = 0$,
- $\underline{L} = (\ 3,\ 2,\ 5,\ \ 5)$,
- $\underline{R} = (\ 8, 14, 20, 100)$,

La condition de stabilité est vérifiée puisque $\sum_i \ L'_i/R_i \simeq 0,82$.

Les vecteurs $\underline{T} = (\ 8, 13, 36, 36)$ et $\underline{\Delta} = (\ 8, 13, 18, 18)$ obtenus pour la priorité $\mathbf{P}$ sont calculables à l'aide de la figure 16. Cela prouve que cette messagerie est faisable par $\mathbf{P}$.

En appliquant maintenant la priorité $\mathbf{L}^+$, i.e. $\mathbf{2} \succ \mathbf{1} \succ \mathbf{3} \succ \mathbf{4}$, la figure 16 montre qu'avec le déphasage initial $\underline{\phi}^1 = (0^+, 0^+, 0, 0^+)$, on obtient $D_1^1 = 10 > 8 = R_1$. La contrainte temps-réel n'est donc pas satisfaite pour le message de type **1**. #

5.3.3 Test par le taux de charge

Nous montrons ici que dans le cas qui nous intéresse (non-préemptif), on ne peut avoir un test de faisabilité portant sur le seul taux de charge (contrairement au cas préemptif sans overhead).

Proposition 5.14

$$\forall\ 0 < u < 1 \ \ \forall M > 1 \ \ \ \exists \mathcal{M} \ \ \textit{de taille } M \ \ \textit{telle que } U(\mathcal{M}) \le u$$

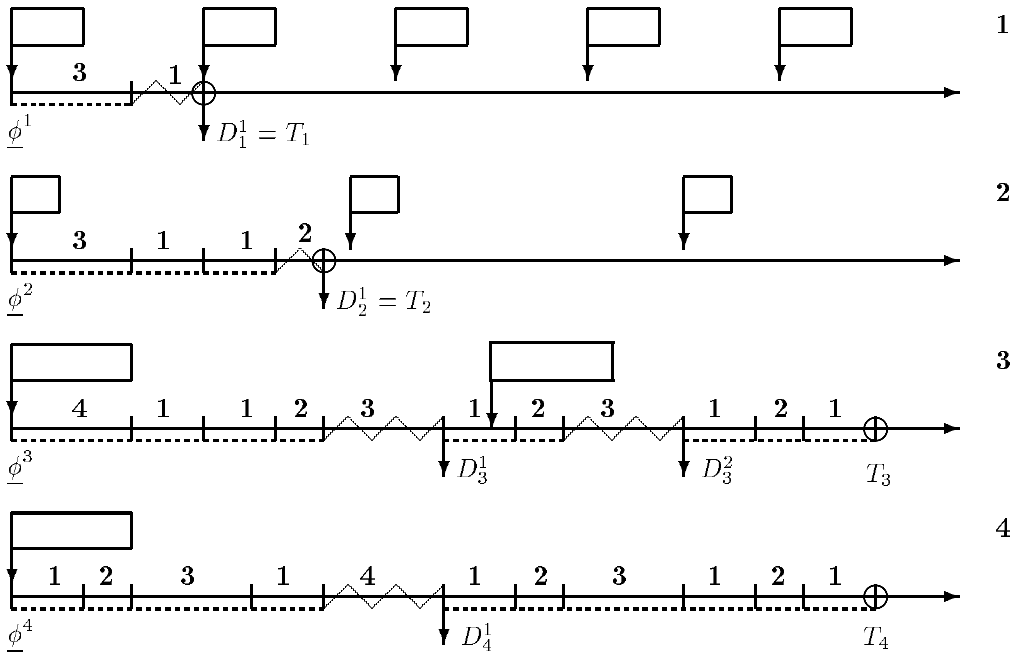
et $\mathcal{M}$ n'est faisable par aucune priorité statique non préemptive.

Démonstration Soient $0 < u < 1$ et $M > 1$, on définit $n = \lceil M/u \rceil$, et la messagerie $\mathcal{M}_n$ par:

- M, $\delta = 0$
- $\underline{L} = (\ 1,\ 2n,\ 1, ..)$
- $\underline{R} = (\ n, 2n^2,\ n, ..)$

On vérifie que $U(\mathcal{M}_n) = M/n \le u$. Pourtant, la messagerie $\mathcal{M}_n$ n'est faisable par aucune priorité non préemptive puisqu'une des récurrences est inférieure à une des longueurs ($R_1 < L_2$). Avec le déphasage initial $(0^+, 0, 1, ..)$, le délai d'un message de type **1** est toujours supérieur à sa récurrence. #

faisabilité par **P**



non-faisabilité par $\mathbf{L}^+$

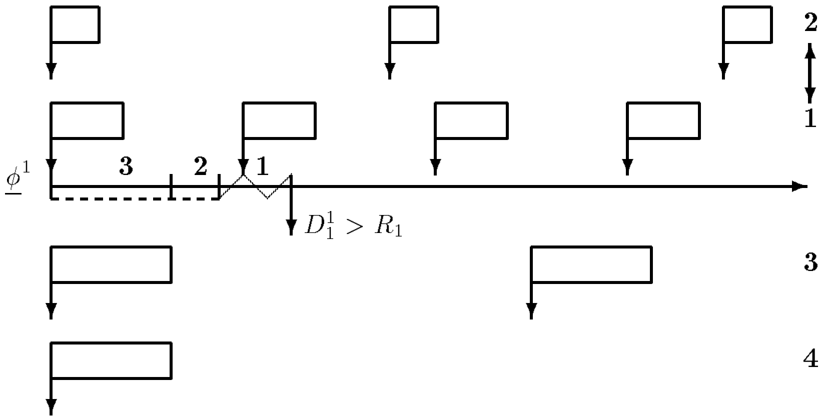


Figure 16 : contre-exemple à l'optimalité de $\mathbf{L}^+$

5.4 Conclusion

Analyse des résultats

Pour savoir si une messagerie donnée est faisable, la définition 3.1 est difficilement exploitable directement.

Il faudrait en effet *a priori* pouvoir observer autant de trajectoires que de déphasages possibles, et ce sur une durée qui est le double de la période du système. Ce n'est en effet qu'à partir de $2T$ que l'on est sûr d'avoir atteint l'état stationnaire du système depuis au moins une période (*cf.* les résultats de la section 2, sur la périodicité et τ). Or, cette période T peut être en pratique gigantesque, si on ne prend pas garde au choix des récurrences R_i. Par exemple, dans le cas des messageries-test de Thomson, T est de l'ordre de 8×10^{23} godets, bien que le maximum des récurrences ne soit que de 170700 godets.

Il est donc important de trouver des "caractérisations" de la faisabilité d'une messagerie qui puissent être facilement vérifiées: par exemple en réduisant l'ensemble des trajectoires à observer. C'est cette démarche qui est adoptée dans la section 4, et que nous avons tenté de reproduire en tenant compte de l'overhead, pour diverses disciplines de service.

- Avec la discipline PAPS, la caractérisation obtenue (*cf.* théorème 5.2), est sous forme d'inéquation. Ce résultat est très simple à appliquer puisqu'il ne demande aucune observation de trajectoire. Cette inéquation est difficilement réalisée:

  - elle n'est pas homogène, puisque les paramètres R_i n'y sont représentés que par leur plus petit élément. Cet inconvénient est en fait propre à cette discipline, nous l'avions déjà rencontré dans la section 3;
  - elle est d'autant plus difficile à vérifier que la taille de la messagerie est grande ou que l'overhead est important. Ce deuxième point sera évidemment commun à toutes les disciplines envisagées.

  Nous en déduisons que la discipline PAPS est très contraignante.

- Dans le cas statique préemptif, nous n'avons obtenu qu'un début de caractérisation du plus long délai, à cause des difficultés suivantes:

  - la durée totale d'overhead subie par un message non prioritaire est difficilement modélisable,
  - en faisant jouer le déphasage initial, cette durée peut devenir très élevée; les majorations a priori grossières utilisées pour obtenir une condition suffisante de stabilité (*cf.* paragraphe 5.2.1) s'avèrent atteintes par certaines messageries.

  C'est pourquoi une caractérisation de la faisabilité d'une messagerie étendant intuitivement les résultats de Liu et Layland n'est pas valable.

- En revanche dans le cas non préemptif, la durée totale d'overhead subie par un message quelconque est constante égale à δ. Cela nous a permis d'obtenir une condition nécessaire et suffisante à la faisabilité d'une messagerie. Ce résultat étend à double titre celui obtenu dans [10], puisque les priorités non préemptives n'y sont pas traitées.

  Le critère obtenu est très satisfaisant puisqu'il ne réclame plus que l'observation de M trajectoires, sur une durée inférieure à $\Lambda/(1-\lambda)$, ce qui est raisonnable si la charge n'est pas trop importante.

Perspectives

Dans toute notre étude, nous nous sommes restreints aux priorités statiques. Ce choix a été motivé par la facilité d'implémentation de telles priorités comparées aux priorités dynamiques.

De plus les priorités dynamiques demandent de prendre une décision à chaque nouvelle arrivée quand le serveur est occupé. Ces décisions sont d'autant plus nombreuses que le taux de charge $\sum_i L_i/R_i$ ou la taille M de la messagerie sont grands. Elles peuvent alors devenir très gourmandes en temps, ce qui est un grand handicap pour la rapidité des applications envisagées. Il est pourtant reconnu que ces priorités sont en échange moins restrictives pour la faisabilité des messageries [10].

Une extension à envisager est donc l'étude des priorités dynamiques avec overhead, préemptives ou non.

Une autre dimension à explorer est l'application au cas du système proposé par Thomson, des résultats obtenus dans l'étude théorique d'une station en isolation. Il faudrait en effet caractériser la faisabilité d'une messagerie pour le bus complet.

A Annexe

Nous illustrons ici la démonstration du théorème 5.9 par un exemple où l'on détaille les différentes étapes.

- $M = 4, \quad \delta = 0,$
- $\underline{L} = (3, 5, 5, 6),$
- $\underline{R} = (9, 9, 1000, 1000),$

On vérifie que cette messagerie est stable pour la priorité non préemptive **P**, puisque $\sum_I L_i'/R_i \simeq 0,90$.

Sur la figure 17 on représente les requêtes et la trajectoire correspondant aux déphasages initiaux des différentes étapes.

Pour le déphasage initial $\underline{\phi} = (3, 0, 10, 11)$, on constate qu'il y a un dépassement à l'instant $t = 27$: la quatrième requête de type $i = \mathbf{2}$ a lieu alors que la troisième n'a pas encore été satisfaite.

Etape 1 Pour déterminer t_0 la nouvelle origine des temps, on observe qu'à l'instant $d = 22$ (début de la période d'occupation du serveur par des messages de type 1 ou 2, qui contient t) un message de type 1 et un de type 2 sont en attente d'émission, et un message de type $j = 3$ termine son émission. On est dans le *cas 1*, ce qui donne $t_0 = d - L_j = 17$. On peut donc calculer le nouveau déphasage initial: $\underline{\psi}^1 = (4, 1, 0, 0^+)$.

On a représenté sur la même figure les requêtes et la trajectoire correspondant à ce nouveau déphasage, sur l'intervalle $[0, A]$. La valeur de $A = t - t_0$ est 10.

Etape 2 Le plus long message strictement moins prioritaire que $i = 2$ est de type $i_0 = 4$, on considère donc maintenant le déphasage initial $\underline{\psi}^2 = (0^+, 1, 0^+, 0)$.

Etape 3 On considère enfin le déphasage $\underline{\phi}^2 = (0^+, 0^+, 0^+, 0)$, dont on a représenté les requêtes et la trajectoire sur l'intervalle $[0, A]$. On constate qu'il y a un dépassement à l'instant 9, la première requête du message de type 2 n'est pas satisfaite à temps. On aurait pu ici utiliser simplement la proposition 5.8, puisque $X_2 = L_1 + L_2 + L_4 = 14 > R_2 = 9$, pour conclure directement que la messagerie n'est pas faisable par la priorité **P**.

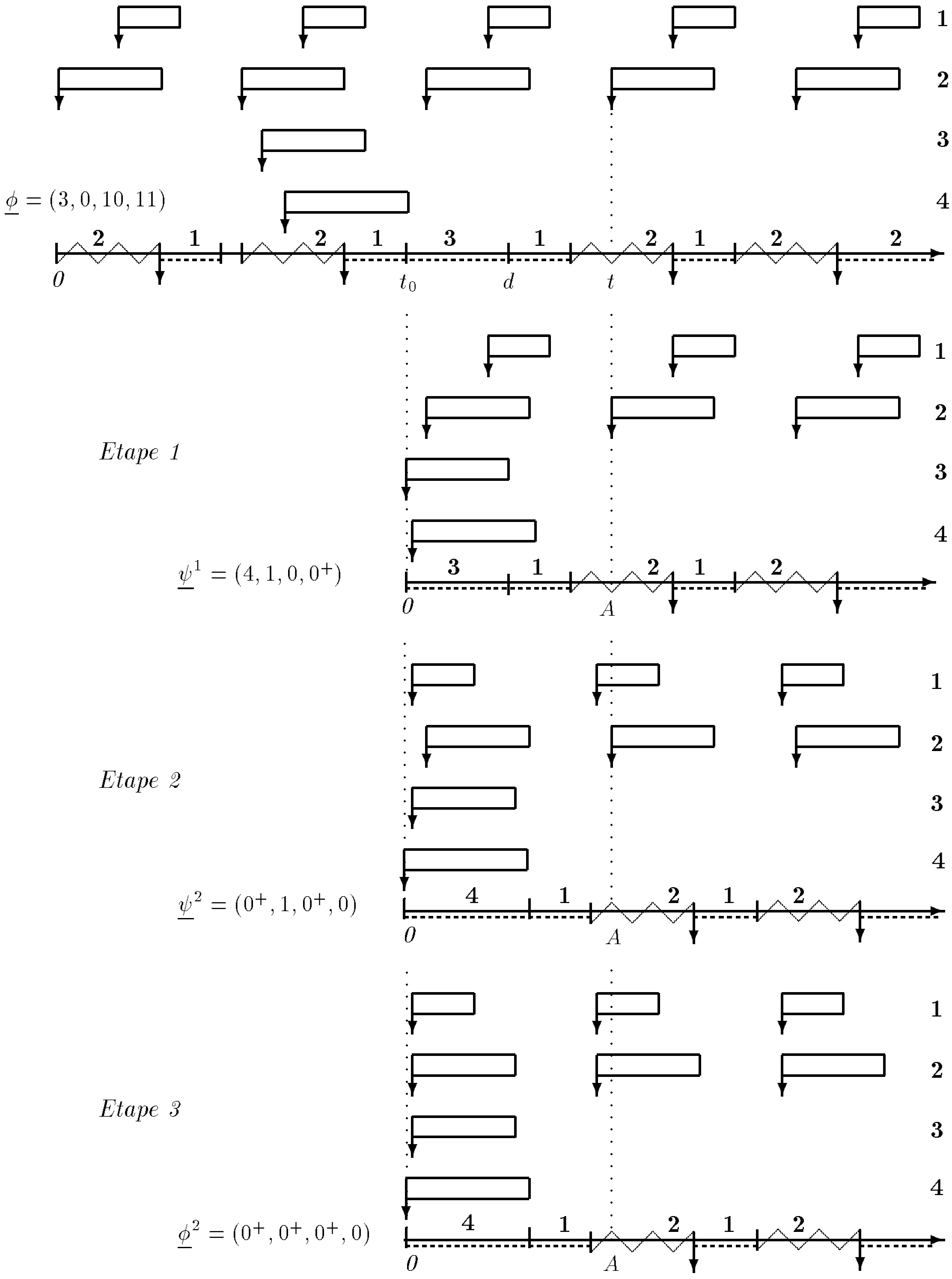


Figure 17 : illustration des étapes

Bibliographie

[1] D. DYKEMAN and W. BUX, Analysis and Tuning of the FDDI Media Access Control Protocol, *IEEE Journal on Selected Areas in Communications*, **6**, No 6, July 1988, pp. 997-1010

[2] A.E. ECKBERG, The Single Server Queue with Periodic Arrival Process and Deterministic Service Times, *IEEE Trans. Comm.*, **Com-27** (3), Mar 1979.

[3] S. FDIDA and H. SANTOSO, Performance Issues of the DQDB Protocol, *Proc. 4th Int. Conf. on Data Communication Systems*, Barcelone, Juin 1990.

[4] A. GRAVEY, Temps d'Attente et Nombre de Clients dans une file $nD/D/1$, *Ann. Inst. Henri Poincaré*, **20**, 1984.

[5] A. GRAVEY and G. HEBUTERNE, Mixing Time and Loss Priorities in a Single Server Queue, *Proc. ITC 13*, 1990.

[6] T. HIRAYAMA, M. KIJIMA and S. NISHIMURA, Further Results for Dynamic Scheduling of Multiclass G/G/1 Queues, *Journal of Applied Probabilities*, **26** (1989), pp. 595-603.

[7] M.J. JOHNSON, Proof that timing Requirements of the FDDI Token Ring Protocol are Satisfied, *IEEE Transactions on Communications*, **COM-35**, No 6, June 1987, pp. 620-627

[8] S. LEFEBVRE-BARBAROUX, *Files d'Attente avec Arrivées Atypiques: Environnement Aléatoire et Superposition de Flux Périodiques*, Thèse de doctorat, Université Paris XI, Orsay, à paraître, février 1992.

[9] J. Y.-T. LEUNG et M.L. MERRIL, "A note on preemptive scheduling of periodic, real-time tasks", *Information Processing Letters*, **11**, 3 (1980), pp. 115–118.

[10] C.L. LIU and J.W. LAYLAND, Scheduling Algorithms for Multiprogramming in Hard-Real-Time Environment, *Journal of the Association for Computing Machinery*, **20** , No 1, January 1973, pp. 46-61.

[11] L.F.M. de MORAES, Message Delay Analysis of the DQDB Protocol, *Proc. NATO workshop on High Speed Networks*, Sophia-Antipolis, 1990.

[12] P. NAIN and K.W. ROSS, Optimal Priority Assignment with Hard Constraint, *IEEE Transactions on Automatic Control*, **AC-31**, No 10, October 1986, pp. 883-888

[13] J.W. ROBERTS and J.T. VIRTAMO, The Superposition of Periodic Cell Arrival Streams in an ATM Multiplexer, *IEEE Trans. Comm.* 1991.

[14] K.C. SEVCIK and M.J. JOHNSON, Cycle Time Properties of the FDDI Token Ring Protocol, *IEEE Transactions on Software Engineering*, **SE-13**, No 3, March 1987, pp. 376-385

Table des matières