Rapport de stage
Analyse de stabilité d'algorithmes distribués probabilistes

encadré par Laurent Fribourg, Richard Lassaigne et Claudine Picaronny

Simon Pinot
écrit en LaTeX

26 septembre 2005

Table des matières

1 Sujet du stage

Le sujet du stage s'intitule **Analyse de stabilité d'algorithmes distribués probabilistes**. Il est nécessaire d'expliquer ce titre en clarifiant chaque mot. On va d'abord expliquer ce qu'on entend par **protocole** et quels cas spécifiques l'on étudie. Puis on expliquera en quoi les protocoles sont **probabilistes**. Et enfin on verra pourquoi l'on s'intéresse plus particulièrement aux protocoles **distribués**.

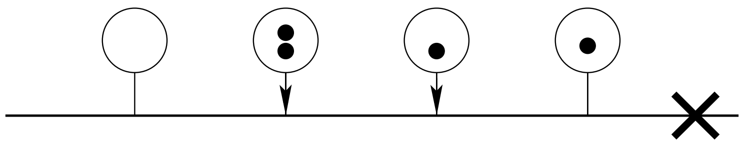
1.1 Problème typique : collisions d'information pendant les échanges inter-ordinateurs

Le développement des ordinateurs a créé naturellement le besoin de les faire communiquer entre eux. Les données transitent par câbles physiques ou par ondes radios L'augmentation du nombre d'utilisateurs entraîne des problèmes de collisions d'informations qui peuvent faire échouer la communication. Une solution radicale serait de donner un canal de communication pour chaque connection possible. Bien sûr cette solution est irréalisable car on veut minimiser les canaux pour des raisons de coût. Il faut alors se résigner à la possibilité qu'il y ait des collisions et donc il faut qu'elles soient sans conséquence sur la communication. Pour que ceci soit possible, il faut qu'une collision soit détectable et sans incidence sur l'intégrité du message. L'utilisation de codes détecteurs d'erreurs dans les messages résoud le problème, un utilisateur garde alors une copie de son message jusqu'à ce qu'il l'ait envoyé avec succès.

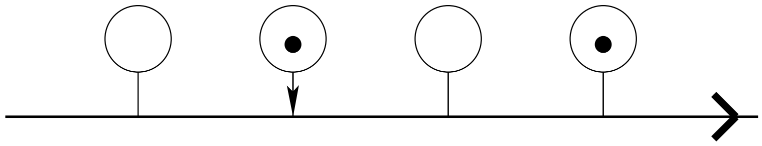
On peut se ramener à n'étudier qu'un seul canal qui servirait pour un certain nombre n d'utilisateurs. Ce nombre peut être variable mais on le supposera constant (par exemple le nombre maximum d'utilisateurs). Soit n utilisateurs tentant d'utiliser un canal ne pouvant supporter qu'un message à chaque fois. Le problème est modélisé par un temps discret, en effet on ne permet aux utilisateurs de ne tenter une transmission qu'aux instants indexés par $\mathbb{N}$. À chaque instant, chaque utilisateur a l'opportunité d'avoir un message de plus à envoyer. Le nombre total d'arrivées sera décrit par une variable aléatoire indépendante du temps dont la moyenne notée λ représente la vitesse d'arrivée des messages. Une transmission est réussie si un seul utilisateur tente d'envoyer son message. Dans les autres cas, soit personne ne tente de transmission et rien ne se passe, soit plusieurs utilisateurs tentent en même temps et alors il y a collision. Les utilisateurs recoivent de nouveaux messages aléatoirement. Ceux-ci sont rajoutés aux messages déjà en attente s'il y en a.

En cas de collision, aucun utilisateur ne peut envoyer son message, ils restent donc tous à envoyer. De plus chaque utilisateur a eu l'opportunité de recevoir de nouveaux messages à envoyer. Donc à l'instant suivant une collision, on se retrouve dans une situation semblable ou pire. C'est pourquoi l'on a besoin d'un protocole de décision.

On illustre ces propos par deux schémas représentant une collision et un envoi. Quatre utilisateurs sont représentés. Chacun peut avoir des messages en attente chez lui, symbolisés par des points noirs. Un utilisateur ayant un message en attente peut décider de tenter l'envoi, ce qui est symbolisé par une flèche.



Ici les deuxième et troisième utilisateurs tentent l'envoi en même temps, donc ils entrent en collision.



Ici seul le deuxième utilisateur tente l'envoi. Donc il réussit.

1.2 Protocole probabiliste et distribué

Un protocole est un algorithme que doit suivre chaque utilisateur pour utiliser le canal. Une première condition est que le protocole soit le même pour chaque utilisateur.

Le protocole trivial qui, une fois qu'un message a subi une collision, retente l'envoi après un laps de temps fixé ne peut fonctionner : comme le protocole est le même pour tous les utilisateurs, tous les messages avec qui il était entré en collision vont réitérer cet échec. D'où la nécessité d'introduire de l'aléatoire dans le protocole. Par exemple on peut demander aux messages entrés en collision de ne retenter la transmission qu'à partir d'un certain temps aléatoire cette fois-ci, afin d'éviter les collisions futures. C'est en cela que le protocole est **probabiliste**.

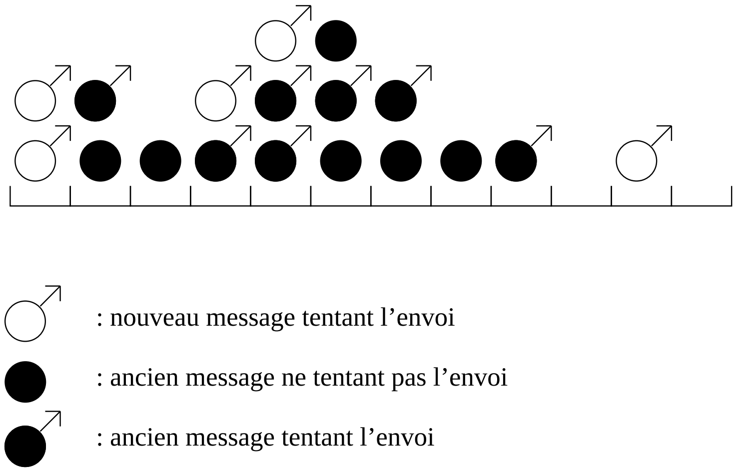
Une autre contrainte va restreindre notre choix sur les protocoles. En effet, les utilisateurs ont connaissance de leurs données personnelles comme leur nombre de messages en attente, du nombre de fois qu'ils ont subi une collision... mais on ne peut leur demander de connaître ni l'état complet du système ni même des informations sur les autres utilisateurs. Ceci parce que l'obtention de ces informations nécessite des communications supplémentaires qui rendraient la communication impossible. Donc un protocole ne peut pas dépendre d'autre chose que des données locales. C'est ce qu'on nomme un protocole **distribué**.

1.3 Exemple de protocole : ALOHA

C'est en 1970 qu'Abram Norman doit résoudre un problème de télécommunications à l'Université d'Hawaii. En effet des communications par ondes hertziennes se font fréquemment et il faut prendre en compte les collisions. Pour cela, c'est le protocole ALOHA qui sera mis en activité.

Les messages arrivent aléatoirement chez chaque utilisateur et y restent tant qu'ils n'ont pas été envoyés. Un message qui arrive tente l'envoi. S'il entre en collision il ne retentera l'envoi qu'avec une certaine probabilité μ. Cette probabilité est constante et est la même pour tout utilisateur. Ceci nous laisse espérer qu'à la prochaine tentative d'envoi tout les messages ne tenteront pas la transmission en même temps.

Voici une illustration du fonctionnement d'ALOHA. À chaque étape, on représente le système en indiquant l'état de chaque message en attente (s'il est nouveau, s'il tente la transmission...).



1.4 Notions de stabilité

Les collisions sont inévitables et entraînent une perte de temps. Cela dit le protocole utilisé peut minimiser la proportion de temps perdu. On veut alors trouver les conditions qu'on demande à un protocole efficace. S'il les satisfsfait, on dira qu'il est **stable**. Il existe en fait plusieurs spécification pour ce problème. Celle que nous choisirons semble être la dominante de nos jours. D'autres spécifications seront exposées à la **p.11**.

On veut qu'un message soit finalement envoyé, c'est-à-dire qu'il y ait une probabilité nulle pour qu'un message ne soit jamais envoyé. Cependant cette condition est souvent satisfaite mais finalement non satisfaisante en pratique. En effet le temps qu'il faut attendre est primordial et savoir qu'il est fini ne suffit pas. Ce qui nous intéresse est sa moyenne. Et cette moyenne peut être infinie quand même, ce qui rendrait le protocole inefficace. Donc on peut demander que le temps moyen d'attente pour qu'un message soit envoyé soit fini.

On demande une condition qui semble plus forte, c'est-à-dire que, à partir d'une configuration où aucun message n'est en attente (qu'on appelle l'état initial), on revient à cette configuration. Cependant, comme dans le cas précédent, on sait que ce temps qui est une variable aléatoire peut avoir une moyenne infinie même si on est sûr de retourner dans l'état initial. Cette situation ne peut nous convenir car on peut avoir un temps de retour de plus en plus grand par exemple. C'est pourquoi notre condition est que le temps de retour à l'état initial doit être de moyenne fini.

Condition pour la stabilité

On ne peut pas demander trop à un protocole. Notamment on pense qu'il est impossible d'envoyer plus d'un message par étape en moyenne. En pratique, les restrictions sont même plus fortes. Ainsi la stabilité ne sera souvent obtenue qu'en fonction de contraintes sur la vitesse d'arrivée mais aussi le nombre d'utilisateurs par exemple. Parfois même les contraintes sur la vitesse d'arrivée dépendront du nombre d'utilisateurs, de telle sorte que plus il y a d'utilisateurs, moins la stabilité est assurée pour des grandes vitesses d'arrivée.

La stabilité est en général dure à obtenir. Par exemple, le protocole ALOHA décrit plus haut est instable quels que soient les paramètres que l'on prend. Les protocoles étudiés par la suite seront moins simples mais seront stables, au moins pour certains paramètres.

1.5 Objectifs du stage

On regarde tout d'abord une démonstration présente dans "Markov Chains, Gibbs fields, Monte-Carlo simulation and queues"par Pierre Brémaud [2]. Plus précisément l'exemple **1.5** intitulé "A collision-resolution protocol".

Ensuite on étudie l'article "An improved stability bound for binary exponential backoff"par Hesham Al-Ammal, Leslie Ann Goldberg et Phil MacKenzie [1] qui montre une preuve très difficile de résultat de stabilité.

Puis on s'intéresse à la simulation des protocoles et aux informations statistiques qu'on peut en tirer.

Enfin, on essaie de conclure sur ce qu'on peut retenir de ces différentes preuves pour ce qui est de l'intérêt des protocoles.

2 Rappels théoriques

2.1 Modèle mathématique

La modélisation mathématique pour étudier le problème est naturellement issue du domaine des probabilités et ce pour deux raisons :

1. le protocole
2. l'environement (l'arrivée des messages est aléatoire)

Les variables aléatoires considérées sont à valeurs dans $\mathbb{N}$.

La probabilité conditionnelle pour que $X = x$ sachant que $Y = y$ est notée $P(X = x|Y = y) = \frac{P(X=x \wedge Y=y)}{P(Y=y)}$.

L'espérance mathématique ou la moyenne de X est notée
$E[X] = \sum_{i \in \mathbb{N}} iP(X = i)$.

L'espérance conditionnelle de X sachant que $Y = y$ est notée
$E[X|Y = y] = \sum_{i \in \mathbb{N}} iP(X = i|Y = y)$.

2.1.1 Chaînes de Markov

Définition

Soit $(X_t)_{t \in \mathbb{N}}$ une suite de variables aléatoires à valeur dans $\mathbb{N}$. On dit que (X_t) est une chaîne de Markov si :

$$P(X_{t+1} = x_{t+1}|X_t = x_t, \cdots, X_0 = x_0) = P(X_{t+1} = x_{t+1}|X_t = x_t)$$

Ce qui signifie que X_{t+1} ne dépend que de X_t. L'indice est en général interprété comme le temps, ce qui nous précise que la probabilité sur la valeur de X à l'instant $t+1$ ne dépend que de sa valeur à l'instant t. Ceci est appelé la propriété de Markov. De plus la chaîne de Markov sera dite homogène si la probabilité $P(X_{t+1} = x_{t+1}|X_t = x_t)$ ne dépend pas de t. Ce sera le cas dans le reste du rapport.

Pour notre problème, X_t caractérise le système à l'instant t : cela peut être le nombre de messages en attente au total par exemple, ou une description plus fine du système. L'état initial est 0.

Une chaîne de Markov est irréductible si, partant d'un état quelconque et considérant un état quelconque, on a une probabilité non nulle pour qu'un jour, le système passe par cet état final. Alors tous les états communiquent. Ce sera toujours le cas pour nous de façon triviale.

Une chaîne de Markov est apériodique si l'on ne peut pas diviser les états en une partition non triviale de telle façon que le système se retrouve périodiquement dans chaque partie. Ce sera aussi le cas pour nos systèmes.

Système de transitions

Du fait de la propriété de Markov, on peut ne s'intéresser qu'aux probabilités du type $P(X_{t+1} = y|X_t = x)$ car elles suffisent à calculer toute autre probabilité. Le système de transitions est formé de la donnée de $P(X_{t+1} = y|X_t = x)$ pour tous les x et y.

Dans le cas où le système ne peut être que dans un nombre fini d'états, le système de transitions est fini et est utilisable sous la forme d'une matrice. Il est alors possible d'apprendre beaucoup d'informations avec cette matrice de transitions qui est la matrice dont le terme de la colonne x et de la ligne y est $P(X_{t+1} = y|X_t = x)$. Cependant notre problème est typiquement infini car le nombre de message en attente n'est pas borné.

2.1.2 Récurrence positive

Le temps de retour τ dans le cas où l'on commence avec $X_0 = 0$ est défini par :

$$\tau = \min\{t \geq 1|X_t = 0\}$$

La chaîne (X_t) est dite alors positive récurrente si et seulement si $E[\tau]$ est fini. La notion de stabilité que l'on veut satisfaire coïncide alors avec la notion de récurrence positive de la chaîne de Markov représentant le système.

2.2 Théorèmes classiques

Voici les théorèmes qui seront utilisés dans le rapport ou les articles cités.

Théorème de Foster

Le théorème de Foster n'est pas en général utilisé directement. C'est en effet un corollaire et une généralisation de celui-ci qui intervient dans le *Brémaud* et l'article.

Théorème 1 (Foster[2]) *Soit (X_t) une chaîne de Markov irréductible, apériodique, homogène sur un ensemble A dénombrable. Alors (X_t) est récurrente positive ssi il existe une fonction $f : A \to \mathbb{R}^+$, un réel $\epsilon > 0$ et un ensemble fini $C \subset A$ tels que*

$$E[f(X_{t+1}) - f(X_t)|X_t = \rho] \leq -\epsilon \quad si\ \rho \notin C \tag{1}$$

$$E[f(X_{t+1})|X_t = \rho] < \infty \quad si\ \rho \in C \tag{2}$$

L'intuition est que l'on cherche une fonction f qui décroît suffisamment vite sur les X_t pour réaliser (1).

Il existe une version de ce théorème qui n'énonce que la condition suffisante. Voici une preuve de la condition nécessaire :

$\square$

On note $\tau_\rho = \inf\{t \geq 1 | X_t \in C \wedge X_0 = \rho\}$ et $p_{ij} = P(X_{t+1} = j | X_t = i)$.

On prend $C = \{\alpha\}$, $\epsilon = 1$ et f $= \begin{cases} E[\tau_\rho] & \text{si } \rho \notin C \\ 0 & \text{sinon} \end{cases}$

Si $\rho \notin C$

$$\begin{aligned} E[\tau_\rho] &= E[\tau_\rho | \tau_\rho = 1] + E[\tau_\rho | \tau_\rho \geq 2] = \sum_{i \in C} p_{\rho i} + \sum_{i \notin C} p_{\rho i} E[\tau_i + 1] \\ &= 1 + \sum_{i \notin C} p_{\rho i} E[\tau_i] \end{aligned}$$

Or on peut réécrire la condition (1) comme ceci :

$$\begin{aligned} E[f(X_{t+1}) - f(X_t) | X_t = \rho] &= E[E[\tau_{X_{t+1}}] | X_t = \rho] - E[\tau_{X_t}] \\ &= \sum_{i \notin C} p_{\rho i} E[\tau_i] - E[\tau_{X_t}] \leq -1 \end{aligned}$$

Donc (1) est satisfait.

D'autre part, (2) s'écrit :

$$\sum_{i \in A} p_{\alpha i} E[\tau_i] < \infty$$

Or on a la récurrence positive de la chaine. Donc :

$$\tau_\alpha < \infty$$

En regardant ce qui se passe à la première étape, on a

$$\tau_\alpha = \sum_{i \in A} p_{\alpha i} E[\tau_i] < \infty$$

$\square$

Restriction du Théorème de Foster

Le résultat suivant est désigné par le Lemme de Pakes. Il est parfois utilisé dans les cas où l'on peut s'autoriser une certaine simplification.

Lemme 1 (Pakes[2]) *Soit* (X_t) *une chaîne de Markov irréductible, homogène sur* $\mathbb{N}$*. Alors pour que* (X_t) *soit récurrente positive , il suffit qu'on ait*

$$\limsup_{\rho\to\infty} E[X_{t+1} - X_n | X_t = \rho] < 0 \quad (1)$$
$$E[X_{t+1} | X_t = \rho] < \infty \quad (2)$$

Il est obtenu par le Théorème de Foster en prenant $f(\rho) = \rho$. On peut trouver un C qui convient grâce à (1).

Généralisation du Théorème de Foster

Voici maintenant le résultat le plus utilisé dans ce rapport. Il s'agit d'une généralisation du Théorème de Foster dans le sens ou il l'implique trivialement et qu'il permet plus de liberté dans une preuve de récurrence positive.

Théorème 2 (Fayolle, Malyshev, Menshikov [4]) *Soit* (X_t) *une chaîne de Markov irréductible, apériodique, homogène sur un ensemble A dénombrable. Alors* (X_t) *est récurrente positive ssi il existe une fonction* $f : A \to \mathbb{R}$*, un réel* $\epsilon > 0$*, une fonction* $k : A \to \mathbb{N}$ *et un ensemble fini* $C \subset A$ *tels que*

$$E[f(X_{t+k(X_t)}) - f(X_t) | X_t = \rho] \leq -\epsilon k(\rho) \quad \textit{si } \rho \notin C \quad (1)$$
$$E[f(X_{t+k(X_t)}) | X_t = \rho] < \infty \quad \textit{si } \rho \in C \quad (2)$$

L'intuition est de s'autoriser une fonction f qui ne vérifie pas toujours le (1) du théorème de Foster mais qui a besoin qu'on attende plus d'une étape pour décroître suffisamment.

3 Les différentes notions de stabilité

Notre notion de stabilité coïncide avec la récurence positive. Mais d'autres notions sont présentes dans certains articles.

- Dans [6], la notion de stabilité inclut la nôtre mais est plus forte. En effet, un protocole est stable si le temps de retour à l'état initial, le temps d'attente d'un message pour être envoyé et le nombre de messages en attente sont tous finis en moyenne. La notion d'instabilité n'est pas évidente car ce n'est pas la négation de la stabilité. Pour être instable, il faut que les trois espérance précédentes divergent.

On s'intéresse aussi dans cet article à la notion de *throughput*, c'est-à-dire le rapport du nombre de messages envoyés et du temps nécéssaire à ces envois. C'est une notion bien différente de la stabilité mais qui peut aussi exprimer l'efficacité d'un protocole.

- Dans [3], la notion retenue est appelée la stabilité d'ordre k. Le système est stable d'ordre k si les moments d'ordre de 1 à k du temps de retour sont finis. Donc notre stabilité correspond à la stabilité d'ordre 1.
- Dans [2], on verra que le résultat n'est pas la stabilité comme on l'entend. Cependant on pourrait tout aussi bien prendre ceci comme définition de la stabilité. Cette notion sera expliquée plus loin.

4 “A collision-resolution protocol”

4.1 Le but

Il s’agit de montrer qu’ALOHA*, une version modifiée d’ALOHA (exposé **p.6**) est stable pour une vitesse d’arrivée λ inférieure à une certaine valeur. Ici, le nombre d’utilisateurs n’est pas précisé. Seule la moyenne de λ arrivées à chaque instant est connue. La répartition de cette variable n’importe pas dans cette démonstration. Ainsi on peut imaginer n utilisateurs ayant chacun la probabilité $\frac{\lambda}{n}$ d’envoyer un message ou toute autre distribution garantissant une moyenne de λ.

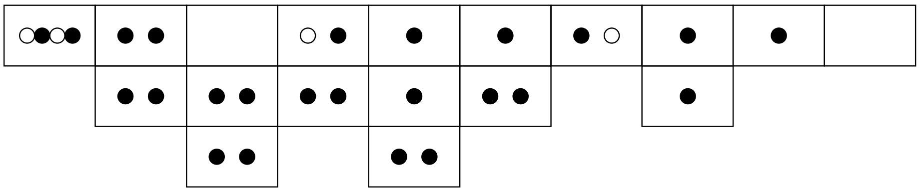
4.2 Le protocole ALOHA*

Ce protocole peut être vu comme une modification d’ALOHA dans le sens où il lui est similaire jusqu’à ce qu’il y ait eu une collision. L’idée est cependant fondamentalement différente. Une fois que des messages sont entrés en collision, le protocole va mettre en attente toute nouvelle arrivée et ne s’occuper de celle-ci qu’une fois les messages impliqués dans la collision envoyés. En effet, les messages entrés en collision se retrouvent dans une boîte (virtuelle, les messages restent bien sûr chez les utilisateurs) et le protocole va devoir en quelque sorte choisir des éléments dans cette boîte pour les envoyer.

Ici le nombre d’utilisateurs n’est pas précisé. On définit a_i comme étant la probabilité de recevoir i nouveaux messages à un instant quelconque. Donc la vitesse d’arrivée est $\lambda = \sum_{i=1}^{\infty} i a_i$.

Tant qu’il n’y a pas de collision, le protocole est similaire à ALOHA. Dès qu’il y a une collision, il se crée alors un CRI (*collision-resolution interval*). Pendant toute la durée de ce CRI, les nouveaux messages sont mis en attente en dehors de celui-ci. Le CRI dispose d’une infinité de couches. Au début du CRI, tous les messages entrés en collision sont dans la première couche. A chaque étape, la première couche essaye de transmettre tous les messages qui y sont. Elle n’y parvient que si elle ne contient qu’un seul message. Sinon la première couche est divisée en deux, chaque message ayant une probabilité 1/2 de ce retrouver dans chacune des deux parties. Une des parties constituera la première couche, l’autre constituera la deuxième puis toutes les autres couches seront décalées. Si toute les couches sont vides, le CRI se finit et tous les messages qui ont été mis en attente tentent la transmission. S’il y en a 0 ou 1, la transmission se fait normalement. Autrement il y a collision et il se crée un nouveau CRI.

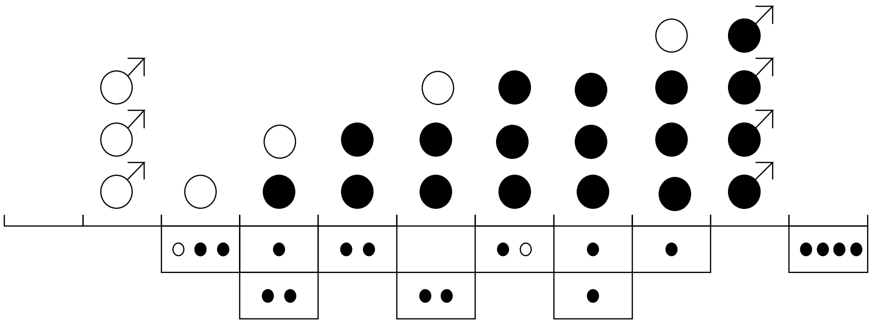
Voilà un exemple de fonctionnement du CRI :



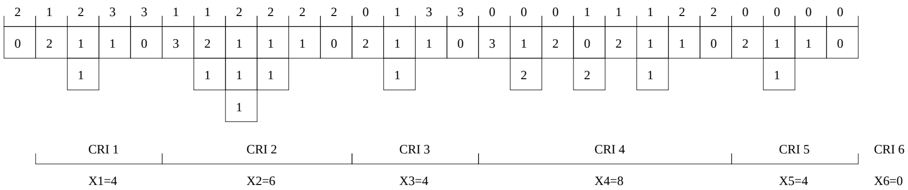
○ : message restant dans la première couche

● : message allant dans la deuxième couche

Et voici une mise en évidence du comportement des nouveaux messages pendant un CRI :



Enfin une exécution complète de plusieurs CRI.



La situation est modélisée par une chaîne de Markov (X_n) où X_n est la longueur du n-ième CRI. Remarquons que l'indice n ne représente pas le temps mais le numéro du CRI considéré. L'état initial étant $X_0 = 0$, le retour à cet état indique qu'il n'y a plus de message en attente d'envoi.

La modélisation dans le *Brémaud* élude ce qui se passe pendant un CRI. Pour en tenir compte, on étudie le système pas à pas :

4.2.1 Modélisation du CRI

Deux modélisations coexistent :
- une fine qui décrit les choses étape par étape (indéxée par t). La chaîne correspondante sera notée Y_t.
- une grossière ne s'intéressant qu'à la description du CRI (indexée par n). La chaîne correspondante sera notée X_n.

La démonstration se base sur la description grossière. La récurence positive de cette chaîne signifie que la longueur du CRI va revenir à 0 en un temps en moyenne fini. On verra à la fin de cette section quel est le rapport avec la stabilité du protocole.

La chaîne de Markov (Y_t) suivante représente la situation :

$$Y_t = (A_t, (S_t^1, \cdots, S_t^k, \cdots))$$

Avec A_t le nombre de messages en attente et S_t^k le nombre de messages dans la couche k du CRI.

On rappelle que a_k est la probabilité pour qu'il y ait k nouveaux messages. On a alors $\lambda = \sum_{i=1}^{\infty} ia_i$.

cas 1

$$(A, (0, \cdots)) \rightarrow \begin{cases} (n, (0, \cdots)) & \text{si } A = 1, P = a_n \\ (n, (A, 0, \cdots)) & \text{si } A \neq 1, P = a_n \end{cases}$$

cas 2

Supposons $S^1 \neq 0$.

$$(A, (S^1, 0, \cdots)) \rightarrow \begin{cases} (A + n, (0, \cdots)) & \text{si } S^1 = 1, P = a_n \\ (A + n, (s, S^1 - s, 0, \cdots)) & P = a_n C_{S^1}^s \left(\frac{1}{2}\right)^{S^1} \end{cases}$$

cas 3

Supposons $S^i \neq 0$.

$$(A, (S^1, \cdots, S^i, 0, \cdots)) \rightarrow \begin{cases} (A + n, (S^2, \cdots, S^i, 0, \cdots)) & \text{si } S^1 \in \{0, 1\}, P = a_n \\ (A + n, (s, S^1 - s, S^2, \cdots, S^i, 0, \cdots)) & P = a_n C_{S^1}^s \left(\frac{1}{2}\right)^{S^1} \end{cases}$$

Le système de transitions, c'est-à-dire $p_{ij} = P(X_{n+1} = j | X_n = i)$, peut alors être déduit.

4.2.2 Système de transitions

Soit Z_n le nombre de messages arrivés durant le n-ième CRI. Alors p_{ij} est calculé suivant les valeurs de Z_n.

$$\begin{aligned}
p_{ij} &= \sum_{k=0}^{\infty} P(X_{n+1} = j \wedge Z_n = k | X_n = i) \\
&= \sum_{k=0}^{\infty} P(X_{n+1} = j | Z_n = k \wedge X_n = i) P(Z_n = k | X_n = i) \\
&= \sum_{k=0}^{\infty} P(X_{n+1} = j | Z_n = k) P(Z_n = k | X_n = i)
\end{aligned}$$

Or on a

$$P(Z_n = k | X_n = i) = \sum_{(q_1, \cdots, q_i) \in \Delta} \prod_{j=1}^{i} a_{q_j}$$

Avec $\Delta = \{(q_1, \cdots, q_i) \in [0, k]^i \mid \sum_{j=1}^{i} q_j = k\}$

Il faut alors calculer $P(X_{n+1} = j | Z_n = k)$. Pour cela, on calcule déjà $P(X_{n+1} = j | Z_n = 0)$, $P(X_{n+1} = j | Z_n = 1)$ et $P(X_{n+1} = j | Z_n = 2)$:

$$P(X_{n+1} = j | Z_n = 0) = 0$$

$$P(X_{n+1} = j | Z_n = 1) = 2$$

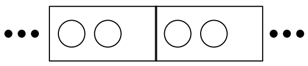
$$P(X_{n+1} = 0 | Z_n = 2) = P(X_{n+1} = 1 | Z_n = 2) = P(X_{n+1} = 3 | Z_n = 2) = 0$$

$$P(X_{n+1} = j | Z_n = 2) = \frac{1}{4} \sum_{\substack{p,q \geq 0 \\ j-4=2p+q}} \frac{Card(\Delta_{p,q})}{4^{p+q}}$$

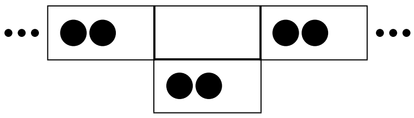
Avec $\Delta_{p,q} = \{(q_1, \cdots, q_{p+q}) \in \{1, 2\}^{p+q} \mid Card\{i \mid q_i = 1\} = q\}$ donc $Card(\Delta_{p,q}) = C_{p+q}^{q}$.

Intuitivement, quand $Z_n = 2$, il y a deux manières de faire augmenter X_{n+1}. Lors de la division :

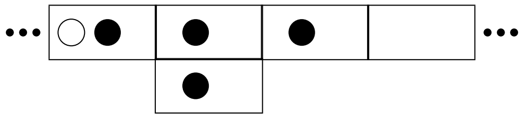
1. soit les deux messages restent dans la première couche et alors on gagne une étape. Ceci arrive avec la probabilité $\frac{1}{4}$.



2. soit les deux messages vont dans la deuxième couche et alors, la première couche étant vide, on gagne deux étapes. Ceci arrive avec la probabilité $\frac{1}{4}$.



q est le nombre d'étapes (1) et p le nombre d'étapes (2). alors la longueur totale j est égale à $2p+q+4$. En effet le 4 correspond à l'envoi des messages et à la première case :



Il reste à trouver toute les configurations possibles, ce que fait $\Delta_{p,q}$. Enfin, la probabilité d'une configuration est $\frac{1}{4^{p+q}}$ et on a $\frac{1}{4}$ devant pour l'envoi des messages.

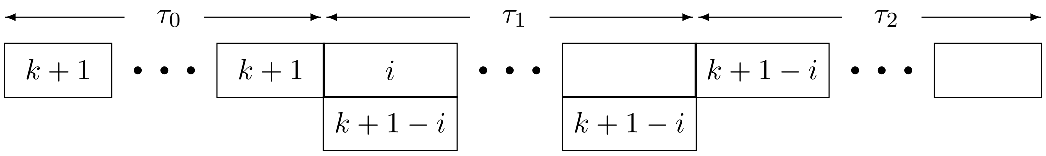
Les termes suivants sont calculés par récurrence :

$$P(X_{n+1}=j|Z_n=k{+}1) = \sum_{\substack{1\leq i\leq k\\ (\tau_0,\tau_1,\tau_2)\in\Gamma_{k,j}}} p'(i,\tau_1)p'(k{+}1{-}i,\tau_2)\frac{1}{4}\sum_{\substack{p,q\geq 0\\ \tau_0-1=2p+q}}\frac{Card(\Delta_{p,q})}{4^{p+q}}$$

Avec $p'(i,j) = P(X_{n+1}=j|Z_n=i)$.

$$\text{et } \Gamma_{k,j} = \left\{ (\tau_0,\tau_1,\tau_2)\in\mathbb{N} \;\middle|\; \begin{array}{l} \tau_0+\tau_1+\tau_2=j \\ 1\leq\tau_0\leq j-(2k-2) \\ 2\leq\tau_1\leq 2k \\ 2\leq\tau_2\leq 2k \\ 2k-2\leq\tau_1+\tau_2\leq j-1 \end{array} \right\}$$

Voici l'intuition de cette formule :



Comme on raisonne par récurrence, on connaît $p'(i,\tau_1)$ et $p'(k{+}1{-}i,\tau_2)$ car $i<k+1$ et $k+1-i<k+1$. On calcule alors la probabilité pour que le début soit de longueur τ_0 comme dans le cas $Z_n=2$ ce qui nous donne la formule. Les conditions sur (τ_0,τ_1,τ_2) viennent du fait qu'un CRI met au minimum $2i$ étapes pour envoyer i messages. Donc on a posé $p'(i,j)=0$ si $j<2i$.

4.3 La méthode

Il s'agit d'une utilisation du lemme de Pakes exposé à la section **p.11**. Les conditions à vérifier sont les suivantes :

$$\limsup_{\rho\to\infty} E[X_{t+1} - X_n | X_t = \rho] < 0 \quad (1)$$
$$E[X_{t+1} | X_t = \rho] < \infty \quad (2)$$

On va détailler la preuve :

- On calcule $E[X_{n+1}|X_n = i]$ en décomposant suivant les valeurs de Z_n :

  $$E[X_{n+1}|X_n = i] = \sum_{k=0}^{\infty} E[X_{n+1}|X_n = i, Z_n = k] P(Z_n = k|X_n = i)$$

  Qu'on peut simplifier car $E[X_{n+1}|X_n = i, Z_n = k] = E[X_{n+1}|Z_n = k]$ puisque la longueur du $(n+1)$-ième CRI ne dépend que des arrivées qu'il aura à traiter, donc de Z_n.
- On montrera que $E[X_{n+1}|Z_n = k] \leq \alpha k - 1$ (a)
  avec $\alpha = 2.886$
- En utilisant (a) on a

  $$E[X_{n+1}|X_n = i] \leq \sum_{k=0}^{\infty} (\alpha k - 1) P(Z_n = k|X_n = i) = \alpha E[Z_n|X_n = i] - 1$$

- On obtient $E[Z_n|X_n = i] = E[B_n]i = \lambda i$, B_n étant le nombre d'arrivées à l'instant n dans la modélisation fine. En effet, dans le cas où $X_n = i$, Z_n est la somme de i résultats B_n indépendant entre eux.
- Ce qui conduit à l'inégalité

  $$E[X_{n+1}|X_n = i] \leq -1 + i\lambda\alpha$$

  $$E[X_{n+1} - X_n|X_n = i] \leq -1 + i(\lambda\alpha - 1)$$

  Donc (2) est toujours vrai et (1) est vrai avec la condition

  $$\lambda < \frac{1}{\alpha} \approx 0.346$$

- Il reste donc (a) à montrer.
  On pose la notation $L_k = E[X_{n+1}|Z_n = k]$. La démonstration utilise certaines propriétés de L_k qu'on prouve au fur et à mesure sans avoir besoin du système de transitions explicitement.

  Si $k = 0, 1$, on a $L_k = 1$ car le CRI n'a rien à faire.

  Supposons $2 \leq k$.

On note $L_{k,i}$ l'espérance de X_{n+1} sachant que sur les k messages du début, i sont restés dans la première couche. Ceci arrive avec la probabilité $q_i(k) = C_k^i\left(\frac{1}{2}\right)^k$. On a une décomposition de L_k suivant les valeurs de $L_{k,i}$:

$$L_k = \sum_{i=0}^{k} q_i(k) L_{k,i} \tag{a1}$$

On remarque que

$$L_{k,i} = 1 + L_i + L_{k-i}$$

En effet, en moyenne on résoud d'abord l'envoi des i messages de tête puis indépendamment celui des $k-i$ messages restant et on prend 1 étape pour passer d'une situation à l'autre.

Reprenant (a1), on a

$$L_k = 1 + \sum_{i=0}^{k} q_i(k)(L_i + L_{k-i})$$

$$= 1 + q_k(k)L_k + q_0(k)L_k + \sum_{i=1}^{k-1} q_i(k)(L_i + L_{k-i}) + q_0(k)L_0 + q_k(k)L_0$$

$$L_k - q_k(k)L_k - q_0(k)L_k = 1 + \sum_{i=1}^{k-1} q_i(k)(L_i + L_{k-i}) + q_0(k)L_0 + q_k(k)L_0$$

$$L_k = \frac{1 + \sum_{i=1}^{k-1} q_i(k)(L_i + L_{k-i}) + q_0(k)L_0 + q_k(k)L_0}{1 - q_k(k) - q_0(k)}$$

$$= \frac{1 + \sum_{i=0}^{k-1}(q_i(k) + q_{k-i}(k))L_i}{1 - q_k(k) - q_0(k)}$$

On suppose l'existence d'une suite α_m tel que :

– $\alpha_m \geq \sup_{j>m} \frac{\sum_{i=0}^{m-1}(L_i + 1)(q_i(j) + q_{j-i}(j))}{\sum_{i=0}^{m-1} i(q_i(j) + q_{j-i}(j))}$ (a2)
– $L_m \leq \alpha_m m - 1$

Alors on va montrer par récurrence que pour tout $n \geq m$ on a

$$L_n \leq \alpha_m n - 1 \tag{a3}$$

Supposons-le pour tout $m < j$ et montrons $L_j \leq \alpha_m j - 1$. On utilise le calcul de L_j :

$$\begin{aligned}
L_j(1-q_j(j)-q_0(j)) &= 1+\sum_{i=0}^{j-1}(q_i(j)+q_{j-i}(j))L_i \\
&= 1+\sum_{i=0}^{m-1}(q_i(j)+q_{j-i}(j))L_i+\sum_{i=m}^{j-1}(q_i(j)+q_{j-i}(j))L_i \\
&\leq 1+\sum_{i=0}^{m-1}(q_i(j)+q_{j-i}(j))L_i+\sum_{i=m}^{j-1}(q_i(j)+q_{j-i}(j))(\alpha_m i-1) \\
&= 1+\sum_{i=0}^{m-1}(q_i(j)+q_{j-i}(j))(L_i-\alpha_m i+1) \\
&\quad +\sum_{i=0}^{j}(q_i(j)+q_{j-i}(j))(\alpha_m i-1)-(q_j(j)+q_0(j))(\alpha_m j-1)
\end{aligned}$$

Or on a :

$$\sum_{i=0}^{j}(q_i(j)+q_{j-i}(j))(\alpha_m i-1)=\sum_{i=0}^{j}2C_j^i\left(\frac{1}{2}\right)^j(\alpha_m i-1)=\alpha_m j-2$$

Car $\sum_{i=0}^{j}\left(C_j^i\right)=2^j$ et $\sum_{i=0}^{j}\left(iC_j^i\right)=j2^{j-1}$
d'ou

$$\begin{aligned}
L_j(1-q_j(j)-q_0(j)) \leq &\sum_{i=0}^{m-1}(q_i(j)+q_{j-i}(j))(L_i-\alpha_m i+1) \\
&+(\alpha_m j-1)(1-q_j(j)-q_0(j))
\end{aligned}$$

$$L_j\leq \alpha_m j-1+\frac{\sum_{i=0}^{m-1}(q_i(j)+q_{j-i}(j))(L_i-\alpha_m i+1)}{1-q_j(j)-q_0(j)}$$

Or on a $\sum_{i=0}^{m-1}(q_i(j)+q_{j-i}(j))(L_i-\alpha_m i+1)\leq 0$ par (a2) :

$$\alpha_m\geq\frac{\sum_{i=0}^{m-1}(L_i+1)(q_i(j)+q_{j-i}(j))}{\sum_{i=0}^{m-1}i(q_i(j)+q_{j-i}(j))}$$

$$\alpha_m\sum_{i=0}^{m-1}i(q_i(j)+q_{j-i}(j))\geq\sum_{i=0}^{m-1}(L_i+1)(q_i(j)+q_{j-i}(j))$$

$$0\geq\sum_{i=0}^{m-1}(L_i+1-\alpha_m i)(q_i(j)+q_{j-i}(j))$$

Donc on a
$L_j\leq\alpha_m j-1$

Enfin on montre par un calcul que si l'on prend $m=6$ et $\alpha_6=2.886$ on a (a2) et pour $n\in\{1,2,3,4,5,6\}$ on a (a3). Pour cela on utilise mapple :

```
L[0] :=1 ;L[1] :=1 ;for k from 2 to 10 do
L[k] :=(1+sum(((k !*(.5)^ k)/(i !*(k-i) !)+
(k !*(.5)^ k)/((k-i) !*(k-(k-i)) !))*L[i],i=0..k-1))/
(1-(k !*(.5)^ k)/(k !*(k-k) !)-(k !*(.5)^ k)/(0 !*(k-0) !)) ;
L[k]/k od ;
```

Ce qui nous donne les valeurs de L_n de 0 à 10. On peut donc vérifier (a3) pour $n \in \{1,2,3,4,5,6\}$.
Puis on calcule (a2)

$$\frac{\sum_{i=0}^{m-1}(L_i+1)(q_i(j)+q_{j-i}(j))}{\sum_{i=0}^{m-1} i(q_i(j)+q_{j-i}(j))} = \frac{\sum_{i=0}^{m-1}(L_i+1)(C_j^i\left(\frac{1}{2}\right)^j + C_j^{j-i}\left(\frac{1}{2}\right)^j)}{\sum_{i=0}^{m-1} i(C_j^i\left(\frac{1}{2}\right)^j + C_j^{j-i}\left(\frac{1}{2}\right)^j)}$$

$$= \frac{\sum_{i=0}^{m-1}(L_i+1)C_j^i}{\sum_{i=0}^{m-1} iC_j^i}$$

Pour $m = 6$ on obtient :

$$\frac{\sum_{i=0}^{5}(L_i+1)C_j^i}{\sum_{i=0}^{5} iC_j^i} = \frac{\frac{j^5}{120}(1+L_5)+\cdots}{\frac{j^5}{120}5+\cdots}$$

D'où :

$$\lim_{j\to\infty} \frac{\sum_{i=0}^{5}(L_i+1)C_j^i}{\sum_{i=0}^{5} iC_j^i} = \frac{1+L_5}{5} \approx 2.883809524$$

On vérifie alors avec maple qu'on est en dessous de $\frac{1+L_5}{5}$ pour des $j > 0$ au début puis on vérifie qu'on reste en dessous en cherchant les zéros du polynôme.

Donc on a (a3) pour tout n, et en ajoutant 2 on a (a), ce qui termine la preuve.

4.4 Lien avec la stabilité du protocole

Pour Pierre Brémaud la récurrence positive de la longueur des X_n s'identifie "naturellement" à la stabilité du protocole. Il semble donc implicite que la récurrence positive de X_n implique celle de Y_n. C'est intuitivement vrai si les X_n ne sont pas trop grands en moyenne car le temps de retour des Y_n (noté τ_Y) sera en quelque sorte le temps de retour de X_n (noté τ_X) multiplié par la moyenne des X_n. Cependant les calculs ne semblent pas directs :

$$\begin{aligned} E[\tau_Y] &= \sum_{j\in\mathbb{N}} jP(\tau_Y = j) = \sum_{j\in\mathbb{N}} j \sum_{i\in\mathbb{N}} P(\tau_Y = j|\tau_X = i)P(\tau_X = i) \\ &= \sum_{i\in\mathbb{N}} P(\tau_X = i) \sum_{j\in\mathbb{N}} jP(\tau_Y = j|\tau_X = i) \\ &= \sum_{i\in\mathbb{N}} P(\tau_X = i)E[\tau_Y|\tau_X = i] \end{aligned}$$

Or on souhaite utiliser l'hypothèse :

$$\sum_{i\in\mathbb{N}} jP(\tau_X = i) = T < \infty$$

Pour ça, il suffirait d'avoir :

$$E[\tau_Y|\tau_X = i] < ki$$

Pour un certain k fixé, ceci pour tout les i à partir d'un certain rang. On peut réécrire ceci grâce à une propriété de τ_Y :

$$\begin{aligned} E[\tau_Y|\tau_X = i] &= E\left[\sum_{n=1}^{\tau_X} X_n \Big| \tau_X = i\right] = E\left[\sum_{n=1}^{i} X_n \Big| \tau_X = i\right] \\ &= \sum_{n=1}^{i} E[X_n|\tau_X = i] \\ &< i \sup_{n\in[1,i]} \{E[X_n|\tau_X = i]\} \end{aligned}$$

Ainsi si $E[X_n|\tau_X = i]$ est borné par une même constante pour tout i, on a le résultat. Mais ceci n'est pas clairement vrai. À ce point, nous avons voulu essayer une simulation pour nous donner une idée de ce qu'on peut espérer prouver. Et les résultats semblent plutôt montrer que $E[X_n|\tau_X = i]$ est croissant en fonction de i. On donne en Annexe **p.38** le résultat de la simulation.

Donc pour l'instant, la stabilité du protocole ne peut pas être déduite de la récurence positive de la longueur des CRI. On parlera alors de ce résultat comme la <u>stabilité de la longueur des CRI</u>.

5 “An improved stability bound for binary exponential backoff”

Le protocole dont s’occupe cet article est **binary exponential backoff**. Il fait partie de l’ensemble des protocoles à backoff que l’on va décrire.

5.1 Les protocoles avec backoff

Les messages entrés en collision ne vont retenter la transmission qu’avec une certaine probabilité. Mais cette probabilité n’est pas constante. Elle est décroissante avec le nombre b de fois que le message a été en collision. De cette façon, plus un message a été en collision, moins il tente la transmission, ce qui régule le réseau. Il y a alors toute une série de protocoles différents par la façon dont dépend cette probabilité. Nous l’appellerons la fonction de backoff. On a par exemple **binary exponential backoff** quand cette fonction est 2^{-b} ou **polynomial backoff** pour une fonction du genre $\frac{1}{1+b^2}$.

Comme l’indique le titre, cet article s’occupe de **binary exponential backoff**. C’est donc un protocole à backoff avec 2^{-b} comme fonction de backoff. On considère un nombre n fixe d’utilisateurs. Il faut maintenant préciser la modélisation utilisée et les notations nécessaires.

5.2 Les notations

Ils considèrent une chaîne de Markov (X_t) représentant le problème de la façon suivante :

$$X_t = (q_1(t), \cdots, q_n(t), b_1(t), \cdots, b_n(t))$$

avec n le nombre d’utilisateurs dans la file, $q_i(t)$ et $b_i(t)$ respectivement la longueur de la file de l’utilisateur i à l’instant t et le nombre de fois que le message de tête de la file de i a été en collision depuis l’instant t.

L’état initial est $X_0 = (0, \cdots, 0, 0, \cdots, 0)$.

La vitesse d’arrivée est appelée λ. Elle est définie par l’espérance du nombre de nouveaux messages reçus à un instant dans les deux protocoles. Chaque utilisateur reçoit à chaque instant un message supplémentaire avec la probabilité λ/n.

Ils seront amenés à utiliser les définitions suivantes :

$$m(X_t) = m = \textit{le nombre de i tels que } q_i(t) > 0 \textit{ et } b_i(t) < \textit{log } \beta + \textit{log } n$$

$$m'(X_t) = m' = \textit{le nombre de i tels que } q_i(t) > 0 \textit{ et } b_i(t) < (1-\eta-\mu)\textit{log } n + 1$$

avec β une constante valant 3.

Voici la description du système de transitions :

Système de transitions

On note $\vec{x} = (x_1, \cdots, x_n)$. On pose $C \subset \{1, \cdots, n\}$.

$$(q_1, \cdots, q_n, b_1, \cdots, b_n) \rightarrow \begin{cases} si\ Card(C) \neq 1\ : \\ \left(q_1 + x_1, \cdots, q_n + x_n, \begin{cases} b_i + 1 & si\ i \in C \\ b_i & si\ i \notin C \end{cases} \right) & P = p_{C,\vec{x}} \\ si\ Card(C) = 1\ : \\ \left(\begin{cases} q_i + x_i & si\ i \neq j \\ q_i - 1 + x_i & si\ i = j \end{cases}, \begin{cases} b_i & si\ i \neq j \\ 0 & si\ i = j \end{cases} \right) & P = p_{\{j\},\vec{x}} \end{cases}$$

$$p_{C,\vec{x}} = \prod_{i=1}^{n} \left(\frac{\lambda}{n}\right)^{x_i} \left(1 - \frac{\lambda}{n}\right)^{1-x_i} \prod_{i \in C} 2^{-b_i} \prod_{i \notin C} \left(1 - 2^{-b_i}\right)$$

5.3 Les résultats antérieurs

Des travaux antérieurs sur le protocole “binary exponential backoff” étudient sa stabilité en fonction de λ.

- *S'il existe $\epsilon > 0$ tel que $\lambda > \frac{1}{2} + \epsilon$, alors, pour des n assez grands, le protocole est instable.*[**6**]
- *Il existe $\alpha > 0$ tel que, pour des n assez grands, si $\lambda < \frac{1}{n^{\alpha log\ n}}$ alors le protocole est stable.*[**5**]

D'autres travaux ont aussi été accomplis sur “polynomial backoff”. C'est le même protocole avec une fonction de retransmission polynomiale. Appelons f la fonction telle que le protocole retransmet avec la probabilité $\frac{1}{f(b)}$un message entré b fois en collision.

- *Si $f(b) = (1+b)^{-\alpha}$ avec $\alpha > 1$, alors pour tout $\lambda < 1$ et n le protocole est stable.*[**6**]
- *Si $f(b) = (1+b)^{-\alpha}$ avec $0 < \alpha \leq 1$, alors pour tout $\lambda < 1$ et n le protocole est instable.*[**6**]

5.4 Le but

Ils veulent améliorer asymptotiquement la borne précédemment trouvée en montrant qu'il existe un α tel que pour tout $\alpha' > \alpha$, tout $\eta \in]0, \frac{1}{4}[$ et pour les n suffisamment grands la stabilité est assurée si

$$\lambda < \frac{1}{\alpha' n^{1-\eta}}$$

Etant donné que le but est de garantir la stabilité du protocole pour des λ les plus grands possibles et cette borne étant asymptotiquement inférieure à la précédente, ce travail a une réelle portée.

5.5 La méthode

Pour cela, ils vont utiliser la généralisation du théorème de Foster décrite **p.11**. On doit donc satisfaire une condition pour avoir la récurence positive :

$$\begin{array}{lll} E[f(X_{t+k(X_t)}) - f(X_t)|X_t = \rho] \leq -\epsilon k(\rho) & \text{si } \rho \notin C & (1) \\ E[f(X_{t+k(X_t)})|X_t = \rho] < \infty & \text{si } \rho \in C & (2) \end{array}$$

Ils définissent une fois pour toute C, $k(C)$, f et ϵ :

$$C = \{(0, \cdots, 0)\} \ et \ k((0, \cdots, 0)) = 1$$

$$f(X_t) = \alpha n^{2-\eta-\mu} \sum_{i=1}^{n} q_i(t) + \sum_{i=1}^{n} 2^{b_i(t)}$$

$$\epsilon = 1 - \frac{2}{\alpha}$$

avec $\mu \in [\eta, \frac{1}{2} - \eta[$

Avec la définition de C et de $k(C)$, on assure déjà le (2) du théorème 2.

Intuitivement, f va augmenter d'une quantité dépendant de façon croissante des tailles des files et du nombre de collisions. Moins il y a de collisions, plus les files baissent, et alors plus f décroît. Cette intuition est utile pour comprendre les choix faits pour les valeurs de k.

Ils vont alors définir la fonction k suivant les valeurs possibles de $\rho = X_t$. Ils font trois cas :

5.5.1 Cas 1 : $m' = 0$ et $m < n^{1-\eta-\mu}$

Le fait que $m' = 0$ dit que les utilisateurs qui ont des messages en attente ont essayé de transmettre leur message de tête un grand nombre de fois. De plus comme $m < n^{1-\eta-\mu}$, on sait intuitivement que la plupart des messages ont été en collision un grand nombre de fois. De ce fait, une collision est peu probable. Donc on peut prendre $k(\rho) = 1$ car on peut supposer que f va décroître suffisamment vite en une étape.

On se ramène par calcul à :

$$E[f(X_{t+1} - f(X_t)|X_t = \rho] \leq g(m, l) - \epsilon$$

Avec l le nombre de i tels que $q_i(t) > 0$. La fonction $g(m, l)$ dépend de m, l, n et α. Il suffit alors de montrer que $g(m, l) \leq 0$ pour tout les $l \geq m$ et les $0 \leq m \leq n^{1-\eta-\mu}$.

Pour cela, on fait trois étapes :

- $g(m, m) \leq 0$ (avec $g(0, 1) \leq 0$ pour cas particulier)
- $g(m, n) \leq 0$
- $\frac{\partial^2}{\partial l^2} g(m, l) > 0$

Pour ces trois étapes, on obtient des sommes de puissances de n dont le terme dominant est négatif quelle que soit la valeur de η, μ et α. Donc, ici le résultat est restreint pour les n suffisamment grands seulement.

5.5.2 Cas 2 : $m \geq n^{1-\eta-\mu}$ ou $m' > n^{\frac{2}{5}}$

Les conditions sur m et m' disent qu'il y a beaucoup de messages qui n'ont pas eu beaucoup de collisions. On pense donc devoir attendre plus de temps pour avoir la bonne condition pour l'application du théorème.

On résume les définitions dans ce tableau :

Nom	Cas $m \geq n^{1-\eta-\mu}$	Cas $m' > n^{\frac{2}{5}}$
r	m	m'
W	$n^{\eta+\mu} \lceil log\ r \rceil 2^{-8}$	$\lceil log\ r \rceil 2^{-8}$
A	W	0
b	$log\ \beta + log\ n$	$(1 - \eta - \mu) log\ n + 1$
v	n	$2 \left\lceil n^{(1-\eta-\mu)} \right\rceil$
$k(\rho)$	$4(r + v) \lceil log\ r \rceil$	$4(r + v) \lceil log\ r \rceil$
B	$\lceil W \rceil + \lceil A \rceil$	$\lceil W \rceil + \lceil A \rceil$
k'	$4r \lceil log\ r \rceil$	$4r \lceil log\ r \rceil$
k''	$4B \lceil log\ B \rceil$	$4B \lceil log\ B \rceil$

Ils définissent S comme étant le nombre de transmissions réussies pendant $\tau = \{t, \cdots, t + k - 1\}$.

Ils définissent aussi plusieurs sous-ensembles de τ :

- $\tau_0 = \{t, \cdots, t + k' - 1\}$
  On a $k' < k$ car $v > 0$, donc $\tau_0 \subset \tau$.

- $\tau'(i)$ les temps t' de τ qui vérifient :
  – soit $b_i(t) = 0$ et $q_i(t) > 0$
  – soit $b_i(t) = 0$ et i reçoit un nouveau message en t'.

- τ_2 est l'ensemble des t' tel qu'il existe au moins B paires (t'', i) tels que $t'' < t'$ et $t'' \in \tau'(i)$.

- τ_1 l'ensemble des $t' \in \tau - \tau_0 - \tau_1 - \tau_2$ tel qu'il existe un i tel que $\tau'(i) \cap [t' - k'' + 1, t'] \neq \emptyset$.

Ils obtiennent la majoration $E[f(X_{t+1}) - f(X_t)|X_t = \rho] \leq -\epsilon k$ à condition d'avoir $\alpha P(W \leq S) \geq 2^{13}$ et $n \geq \epsilon$. Comme α est une variable pouvant être prise aussi grande qu'on veut, il suffit de trouver une minoration de $P(W \leq S)$ indépendante de n. ils trouveront $1 - 5 \times 10^{-5} \leq P(W \leq S)$ ce qui conduit à la condition $\alpha \geq \frac{2^{13}}{1-5\times10^{-5}} \approx 8192$.

Pour montrer $1 - 5 \times 10^{-5} \leq P(W \leq S)$, ils définissent quatre évènements E1, E2, E3 et E4 fortement probables (on a en fait $1 - 10^{-5} \leq P(Ei)$) et tels que $P(S < W|E1 \wedge E2 \wedge E3 \wedge E4) \leq 10^{-5}$. On a alors l'inégalité suffisante.

évènement	description
E1	Il y a au moins A arrivées durant τ
E2	Tout utilisateur i tel que $q_i(t) > 0$ et $b_i(t) < b$ soit envoi avec succès durant τ_0, soit vérifie $b_i(t + k') \geq \lceil log\ r \rceil$
E3	Au moins la moitié des utilisateurs i vérifiant $q_i(t) > 0$ et $b_i(t) < b$ vérifie $b_i(t) \leq b + \lceil log(log\ r) \rceil + 6$ pour tout $t' \in \tau$
E4	Pour tout $t' \in \tau'(i)$ et tout $t'' > t'$ tel que $t'' \in \tau - \tau_0 - \tau_1 - \tau_2$ on a soit $q_i(t'') = 0$, soit $b_i(t'') \geq \lceil log\ B \rceil$

Les preuves de $1 - 10^{-5} \leq P(Ei)$ imposent aussi des conditions sur n.

5.5.3 Cas 3 : $0 < m' \leq n^{\frac{2}{5}}$ et $m < n^{1-\eta-\mu}$

Ce cas est un mélange du **cas 1** et du **cas 2**. En effet, $0 < m' \leq n^{\frac{2}{5}}$ nous dit qu'il peut y avoir un certain nombre d'utilisateurs avec un bon nombre de collisions donc on ne va pas essayer de faire baisser le potentiel en un coup comme dans le **cas 2**. De plus, la condition $m < n^{1-\eta-\mu}$ (qui est aussi vérifiée dans le **cas 1**) nous dit qu'il n'y a pas autant de messages avec un haut compteur de collision comme dans le **cas 2**, donc on ne s'attend pas à ce qu'il y ait beaucoup de transmissions comme dans le **cas 2**.

Puisque $0 < m'$, soit γ un utilisateur tel que $b_\gamma(t) < (1-\eta-\mu)log\ n+1$. On redéfinit plusieurs chose (τ_0 garde sa définition) :

Nom	définition
k	$32m'\lceil log\ m'\rceil + \lceil n^{1-\eta-\mu}\rceil$
k'	$32m'\lceil log\ m'\rceil$
$E1$	Il n'y a pas d'arrivée pendant τ
$E2$	Chaque utilisateur i tel que $q_i(t) > 0$ et $b_i(t) < (1-\eta-\mu)log\ n+1$ soit envoi avec succès durant τ_0 soit vérifie $b_i(t+k') \geq \lceil log\ m'\rceil$
$E3$	pour tout $t' \in \tau$ on a $b_\gamma(t') < (1-\eta-\mu)log\ n+7$

Encore une fois, on montre $1-10^{-5} \leq P(Ei)$ et

$P(S < 1|E1 \wedge E2 \wedge E3) \leq -e^{-\frac{1}{2^{12}}}$, ce qui démontre que

$P(S < 1) \geq 1-3\times 10^{-5} - e^{-\frac{1}{2^{12}}}$, ce qui permet de conclure dans ce cas (comme dans le **cas 2**, une minoration suffit).

6 Simulation de Protocole

Ici, on explique comment simuler les protocoles étudiés. Cela peut servir à se faire une idée du fonctionnement d'un protocole ou de ce qu'on peut espérer de lui. Une fois qu'on a les programmes pour simuler les protocoles, on peut obtenir les informations statistiques qui nous intéressent en modifiant ceux-ci. Les programmes en C sont donnés en annexe **p.39** et **p.42**. Voilà un exemple d'exécution des programmes.

6.1 Exemple d'exécution d'ALOHA*

Le programme demande le nombre de messages au début, c'est l'état initial. Normalement ce nombre est aléatoire mais si l'on veut pouvoir observer des choses intéressantes sans attendre, on a besoin de se placer dans un cas déjà peu probable.

Ensuite on doit entrer λ, le nombre moyen de messages reçus à chaque étape.

La simulation est représentée comme suit :

$$(\mathrm{A},\mathrm{S}_1,\mathrm{S}_2,\cdots,\mathrm{S}_k)$$

Avec A le nombre de messages en attente et S_i le nombre de messages dans la couche i du CRI. Ici, k est la dernière couche du CRI utilisée.

Les étapes successives s'affichent pendant l'exécution.

Le programme fini en affichant le temps mis pour arriver à l'état initial ainsi que le nombre de CRI. Tant qu'on ne revient pas à l'état initial, le programme continue.

```
état initial : 10
lambda : .1
(10)
(0,10)
(1,4,6)
(1,3,1,6)
(2,0,3,1,6)
(3,3,1,6)
(4,2,1,1,6)
(4,1,1,1,1,6)
(4,1,1,1,6)
(4,1,1,6)
(4,1,6)
(4,6)
(4,2,4)
(4,1,1,4)
(4,1,4)
(4,4)
(4,1,3)
(4,3)
(4,1,2)
(4,2)
(4,1,1)
(4,1)
(4)
(0,4)
(0,3,1)
(0,2,1,1)
(0,2,0,1,1)
(0,2,0,0,1,1)
(0,1,1,0,0,1,1)
(0,1,0,0,1,1)
(0,0,0,1,1)
(0,0,1,1)
(0,1,1)
(0,1)
(0)
-------------------------------------
temps : 34
nombre de CRI : 2
```

6.2 Exemple d'exécution d'exponential backoff

Le programme demande le nombre n d'utilisateurs car contrairement à ALOHA*, ce paramètre intervient de façon importante.

Ensuite il demande λ. Chaque utilisateur a alors la probabilité $\frac{\lambda}{n}$ de recevoir un nouveau message. D'où la moyenne de λ nouveaux messages à chaque étape.

Puis on entre le nombre de messages que chacun reçoit au début, ce qui rend la simulation plus intéressante tout de suite.

La présentation est la suivante pour $n = 5$ utilisateurs :

$$\begin{array}{lll} 1: & q_1 -- & b_1 \\ 2: & q_2 -- & b_2 \\ 3: & q_3 -- & b_3 \\ 4: & q_4 -- & b_4 \\ 5: & q_5 -- & b_5 \end{array}$$

Avec les définitions usuelles de q_i (nombre de messages en attente chez i) et b_i (compteur de collisions). Le programme affiche les étapes successives. Si on arrive à l'état initial, le nombre d'étapes mises ainsi que la somme des messages envoyés sont affichés.

```
nombre d'utilisateurs : 5
lamda : .1
arrivée : 2
 1 :   2---   0
 2 :   2---   0
 3 :   2---   0
 4 :   2---   0
 5 :   2---   0

 1 :   2---   1
 2 :   2---   1
 3 :   2---   1
 4 :   2---   1
 5 :   2---   1

 1 :   2---   1
 2 :   3---   1
 3 :   2---   1
 4 :   2---   2
 5 :   2---   2

 1 :   2---   1
 2 :   3---   2
 3 :   2---   2
 4 :   2---   2
 5 :   2---   3

 1 :   2---   1
 2 :   3---   3
 3 :   2---   2
 4 :   2---   2
 5 :   2---   4

 1 :   2---   1
 2 :   3---   3
 3 :   2---   2
 4 :   1---   0
 5 :   2---   4

 1 :   2---   1
 2 :   3---   4
 3 :   2---   2
 4 :   1---   1
 5 :   2---   4

 1 :   2---   1
 2 :   3---   5
 3 :   2---   3
 4 :   1---   1
 5 :   2---   4

 1 :   2---   2
 2 :   3---   5
 3 :   2---   3
 4 :   1---   2
 5 :   2---   4

 1 :   2---   2
 2 :   3---   5
 3 :   2---   3
 4 :   1---   2
 5 :   1---   0

 1 :   2---   3
 2 :   3---   5
 3 :   2---   3
 4 :   1---   2
 5 :   1---   1

 1 :   2---   4
 2 :   3---   5
 3 :   2---   4
 4 :   1---   2
 5 :   1---   1

 1 :   2---   4
 2 :   3---   5
 3 :   2---   4
 4 :   1---   2
 5 :   1---   1

 1 :   2---   4
 2 :   3---   5
 3 :   2---   4
 4 :   1---   2
 5 :   0---   0

 1 :   2---   4
 2 :   3---   5
 3 :   2---   4
 4 :   1---   2
 5 :   0---   0

 1 :   2---   4
 2 :   2---   0
 3 :   2---   4
 4 :   1---   2
 5 :   0---   0

 1 :   2---   4
 2 :   1---   0
 3 :   2---   4
 4 :   1---   2
 5 :   0---   0

 1 :   2---   4
 2 :   1---   1
 3 :   2---   4
 4 :   1---   3
 5 :   0---   0

 1 :   2---   4
 2 :   1---   1
 3 :   2---   4
 4 :   1---   3
 5 :   0---   0

 1 :   2---   4
 2 :   0---   0
 3 :   2---   4
 4 :   1---   3
 5 :   0---   0

 1 :   2---   4
 2 :   0---   0
 3 :   2---   4
 4 :   1---   3
 5 :   0---   0

 1 :   2---   4
 2 :   0---   0
 3 :   2---   4
 4 :   1---   3
 5 :   0---   0

 1 :   2---   4
 2 :   0---   0
 3 :   2---   4
 4 :   1---   3
 5 :   0---   0

 1 :   2---   4
 2 :   0---   0
 3 :   2---   4
 4 :   1---   3
 5 :   0---   0

 1 :   1---   0
 2 :   0---   0
 3 :   2---   4
 4 :   1---   3
 5 :   0---   0

 1 :   0---   0
 2 :   0---   0
 3 :   2---   4
 4 :   1---   3
 5 :   0---   0

 1 :   0---   0
 2 :   0---   0
 3 :   2---   4
 4 :   1---   3
 5 :   0---   0

 1 :   0---   0
 2 :   0---   0
 3 :   1---   0
 4 :   1---   3
 5 :   0---   0

 1 :   0---   0
 2 :   0---   0
 3 :   0---   0
 4 :   1---   3
 5 :   0---   0

 1 :   1---   0
 2 :   0---   0
 3 :   0---   0
 4 :   1---   3
 5 :   0---   0

 1 :   0---   0
 2 :   0---   0
 3 :   0---   0
 4 :   1---   3
 5 :   0---   0

 1 :   0---   0
 2 :   0---   0
 3 :   0---   0
 4 :   1---   3
 5 :   0---   0

 1 :   0---   0
 2 :   0---   0
 3 :   0---   0
 4 :   1---   3
 5 :   0---   0

12 messages envoyés en 33 étapes
```

6.3 Utiliser la simulation

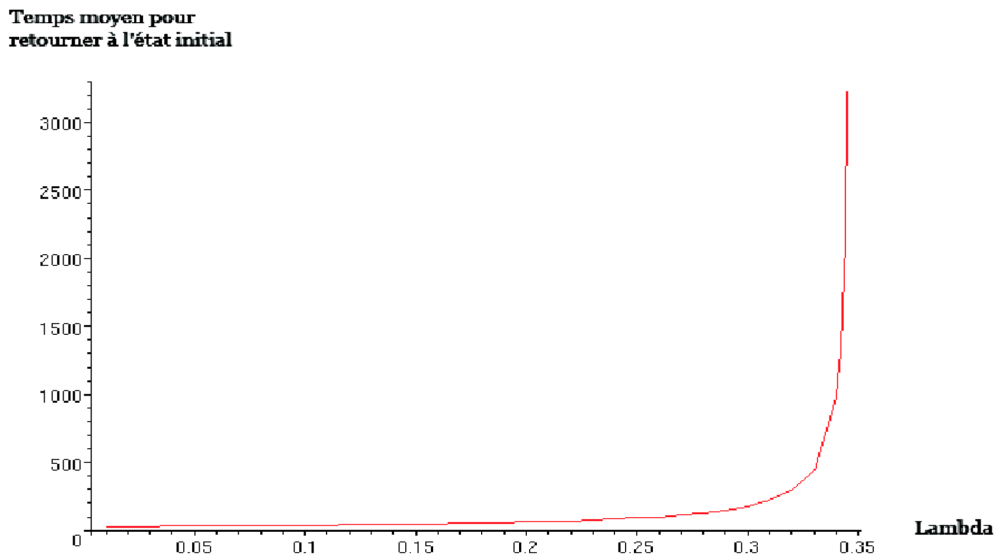
Voici quelques résultats obtenus à l'aide de ces simulations.

6.3.1 Stabilité d'ALOHA*

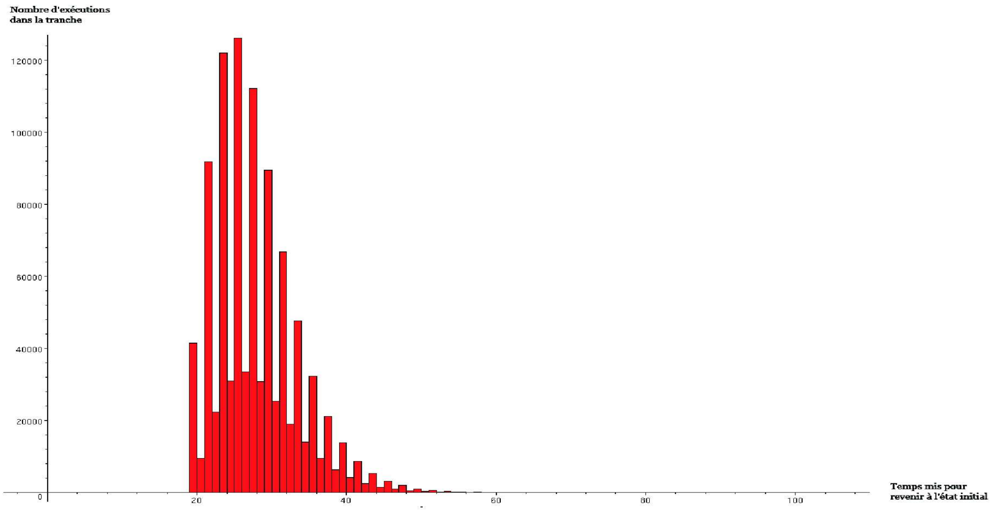
Intuitivement, pour qu'ALOHA* soit stable, il faut et il suffit que le nombre de messages en attente décroisse. Le temps étant divisé en CRI, il faut s'intéresser à la longueur d'un CRI en moyenne en fonction du nombre de messages dont il s'occupe. En effet, cette dépendance va déterminer pour quel λ le protocole sera stable. On se rend compte par la simulation que le temps d'un CRI est en moyenne proportionnel au nombre de messages dont il a à s'occuper (disons A). Pendant le temps T du CRI, le système recevant en moyenne $T\lambda$ messages, on veut $T\lambda < A$. Donc comme $\frac{A}{T}$ est en moyenne constant, cela nous donne une majoration pour $\lambda < \frac{A}{T}$. Expérimentalement on trouve $\lambda < 0.3467$ ce qui est une bonne approximation du résultat théorique.

Une exécution pour étayer ces propos est donnée en annexe **p.45**.

On peut aussi calculer une moyenne du temps de retour à l'état initial et avoir un graphe du temps de retour en fonction de λ. Les résultats sont obtenus sur 10^5 simulations.



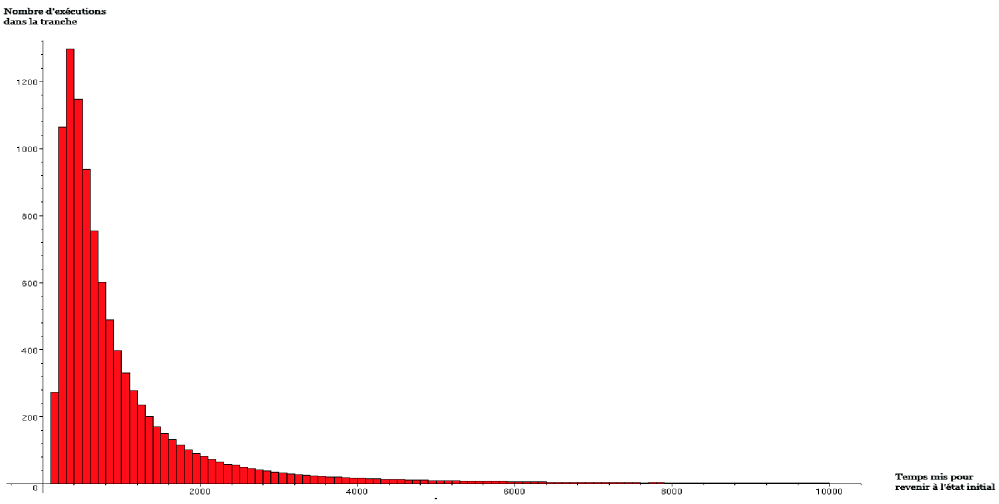
Et enfin un histogramme du temps de retour pour 10^6 exécutions. Il nous indique que pour ALOHA il y a une faible dispersion. Les simulations sont faites avec $\lambda = 0.3$ et 10 messages initiaux.



6.3.2 Autres protocoles avec backoff

Il est facile en modifiant peu le programme de simuler des protocoles utilisant d'autres fonctions de backoff. Par exemple, on peut simuler un polynomial backoff avec $f(b) = \frac{1}{1+b^2}$ comme fonction de backoff. On peut alors comparer celui-ci à exponential backoff pour se rendre compte qu'il semble bien plus rentable. En effet on se rend compte que le temps de retour à l'état initial pour polynomial backoff est plus faible que celui d'exponential backoff. Cela dit, si l'on calcule le rapport entre le nombre de messages envoyés en tout et le temps mis, ces valeurs sont comparables pour les deux protocoles.

On peut noter aussi une grande dispersion pour le temps de retour pour exponential backoff car l'on a souvent de très longues exécutions. L'histogramme suivant est le résultat de 10^6 simulations pour $n = 5$ et $\lambda = 0.3$.



7 Comparaison des résultats

7.1 Résumé du résultat pour exponential backoff

Soit $\alpha' > \alpha$ et $\eta \in]0, \frac{1}{4}[$.
Il existe α et n suffisamment grand pour que l'on ait :

$$\text{si } \lambda < \frac{1}{\alpha' n^{1-\eta}} \text{ alors exponential backoff est stable.}$$

Avec α au moins égale à 8192 et n au moins égale à $\epsilon = 1 - \frac{2}{\alpha} \approx 1$.

7.2 Résumé du résultat pour ALOHA*

Si $\lambda < \frac{1}{\alpha_6} \approx 0.346$ alors la longueur des CRI d'ALOHA* est stable.

7.3 Comparaison

Le résultat pour exponential backoff semble bien faible par rapport à celui d'ALOHA*. En effet, si on fixe un n, ALOHA* peut très bien simuler les arrivées d'exponential backoff en imposant $a_k = C_n^k \left(\frac{\lambda}{n}\right)^k \left(1 - \frac{\lambda}{n}\right)^{n-k}$ et $a_k = 0$ pour $k > n$. Alors si $\lambda < 0.346$ ce protocole est stable mais si on essaye de traiter la même arrivée avec exponential backoff, la grande valeur de α ne nous permet pas de savoir si le protocole est stable pour les même λ. En effet, même pour un seul utilisateur, on a déjà la condition $\lambda < 0.000122$.

On peut alors se demander : pourquoi s'intéresser à exponential backoff ?

En fait ALOHA* n'est pas distribué donc il n'est pas utilisable. En effet, un utilisateur a besoin de savoir s'il y a eu une collision ou un envoi, même s'ils ne le concernent pas. C'est seulement avec ces informations que le CRI peut s'exécuter. Par exemple, un utilisateur dont un message est dans la 2ème couche doit savoir s'il y a eu une collision ou un envoi. En effet il va alors respectivement déplacer son message dans la 3ème couche ou dans la première. Alors qu'un utilisateur d'exponential backoff n'a besoin de connaître que ses propres collisions et ses envois.

Donc exponential backoff est plus utilisable.

Références

[1] Hesham Al-Ammal, Leslie Ann Goldberg, and Philip D. MacKenzie. An improved stability bound for binary exponential backoff. *Theory Comput. Syst.*, 34(3) :229–244, 2001.

[2] Pierre Brémaud. *Markov Chains, Gibbs fields, Monte-Carlo simulation and queues.* Springer-Verlag, New-York, 1999.

[3] J. I. Capetanakis. Tree algorithms for packet broadcast channels. *IEEE Transactions on Information Theory*, IT-25(3) :505–515, 1979.

[4] G. Fayolle, V.A. Malyshev, and M.V. Menshikov. *Topics in the Constructive Theory of Countable Markov Chains.* Cambridge University Press, New-York, 1995.

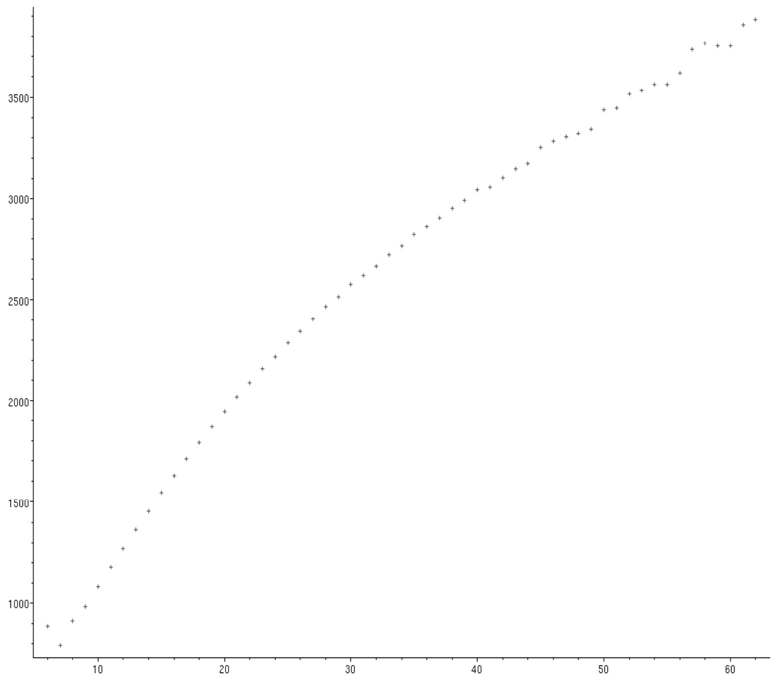
[5] Jonathan Goodman, Albert G. Greenberg, Neal Madras, and Peter March. Stability of binary exponential backoff. *Journal of the ACM*, 35(3) :579–602, 1988.

[6] J. Håstad, T. Leighton, and B. Rogoff. Analysis of backoff protocols for multiple access channels. *SIAM Journal on Computing*, pages 740–774, 1996.

8 Annexe

8.1 Lien avec la stabilité du protocole

Le but était de savoir si $E[X_n|\tau_X = i]$ est borné en fonction de i. On a exécuté alors la simulation d'ALOHA* 10^6 fois avec 100 messages initiaux et $\lambda = 0.3$. On obtient le graphe suivant :



La taille des CRI semble alors augmenter avec le nombre de CRI nécéssaire au retour à l'état initial.

8.2 Programme de simulation d'ALOHA*

```
#include<stdio.h>
#include<stdlib.h>
#include <time.h>

int main (void)
{
        int A,n,init,k1,k2,k3,k4,k5,k99,N,n_CRI;
        /*A est le nombre de messages en attente en dehors du CRI*/
        /*n est le nombre d'utilisateur*/
        /*init est le nombre de messages au début*/
        /*N est le nombre maximum de messages du CRI*/
        /*n_CRI est le numéro de la couche extrême du CRI*/
        N=500;
        int CRI[N],cri_copie[N];
        /*CRI[k] est le nombre de messages dans la couche k du CRI*/
        /*cri_copie[k] va servir à faire une copie de la situation*/
        double l;
        /*l est la moyenne des arrivées*/
        N=500;
        n_CRI=0;
        n=10;
        k4=0;
        k5=0;
        srand((unsigned)time(NULL));
        /*on initialise CRI[N]*/
        for(k1=1;k1<N;k1++)
        CRI[k1]=0;

        printf("état initial : ");
        scanf("%d",&init);
        A=init;
        printf("lambda : ");
        scanf("%lf",&l);
        printf("\n");
        printf("(%d)\n",A);

        /*on ne s'arrête que quand il n'y a plus de messages
        en attente ni dans le CRI, ni en dehors*/
        while(A≠0 || n_CRI≠0)
        {

                /*quant on a vidé le CRI,
```

```
on remet tout ce qui est en dehors dedans*/
if(n_CRI==0)
{
        if(A≤1)A=0;
        else
        {
                CRI[1]=A;
                n_CRI=1;
                A=arrivee(n,l);
        }
}

/*sinon, c'est que le CRI est en cours*/
else
{

        /*le seul moyen de finir un CRI*/
        if(n_CRI==1 & CRI[1]==1)
        {
                CRI[1]=0;
                n_CRI=0;
                A=A+arrivee(n,l);
        }

        /*sinon, c'est le fonctionnement normal du CRI*/
        else
        {

                /*s'il y a collision,
                on divise en deux et on décale tout*/
                if(CRI[1]>1)
                {
                        A=A+arrivee(n,l);
                        for(k1=2;k1≤n_CRI;k1++)
                        cri_copie[k1]=CRI[k1];

                        for(k1=2;k1≤n_CRI;k1++)
                        CRI[k1+1]=cri_copie[k1];

                        n_CRI++;
                        k3=CRI[1];
                        k2=casse_CRI(k3);
                        CRI[1]=k2;
                        CRI[2]=k3-k2;
```

```
                    }

                    /*sinon, il y a soit envoi d'un message,
                    soit aucun envoi*/
                    /*en tout cas, le CRI se décale d'une couche*/
                    else
                    {
                        A=A+arrivee(n,l);
                        for(k1=2;k1<=n_CRI;k1++)
                        cri_copie[k1]=CRI[k1];

                        for(k1=2;k1<=n_CRI;k1++)
                        CRI[k1-1]=cri_copie[k1];

                        n_CRI--;
                    }

                    if(n_CRI>1 && CRI[n_CRI]==0)
                    n_CRI--;
                }
            }
            printf("\n(%d",A);
            for(k1=1;k1<=n_CRI;k1++)
            printf(",%d",CRI[k1]);

            printf(")\n");
            k4++;
            if(n_CRI==0)k5++;
        }
        printf("-------------------------------------\ntemps :
%d \nnombre de CRI : %d\n\n",k4,k5);
}


/*fonction qui simule n_arr utilisateur
pouvant recevoir 1 message avec proba l_arr/n_arr*/
/*arrivee(n,l) est le nombre de messages reçus en tout*/
int arrivee (int n_arr, double l_arr)
{
        int j,c,r;
        c=0;
        for(j=1;j<=n_arr;j++)
        {
            r=rand();
```

```
            if(1.*r/RAND_MAX<l_arr/n_arr)c++;
        }
        return(c);
}


/*divise la première couche du CRI en deux*/
/*casse_CRI(A) donne le nombre
de messages restant dans la première couche*/
int casse_CRI (int A)
{
        int k4, k5;
        k5=0;
        for(k4=1;k4≤A;k4++)
        if(rand()<RAND_MAX/2)k5++;

        return(k5);
}
```

8.3 Programme de simulation d'exponential backoff

```
#include<stdio.h>
#include<stdlib.h>
#include <time.h>
#include<math.h>
int main (void)
{
        int n,init,i,k1,k2,k3,k4,k5,k6,k99,N;
        /*n est le nombre d'utilisateurs*/
        /*init est le nombre que chaque utilisateur a au début*/
        /*N est le nombre maximum d'utilisateurs*/

        N=500;
        int q[N],b[N],a[N],tente[N];
        /*q[k] est le nombre de messages chez k*/
        /*b[k] est le compteur backoff de k*/
        /*a[k] est le nombre d'arrivées chez k*/
        /*tente[k]=1 ssi k tente la transmission*/
        double l;
        /*l est la moyenne des arrivées*/
        srand((unsigned)time(NULL));

        /*initialisation*/
        for(i=1;i<N;i++)
```

```
{q[i]=0;b[i]=0;a[i];tente[i];}

printf("nombre d'utilisateurs : ");
scanf("%d",&n);
printf("lamda : ");
scanf("%lf",&l);
printf("arrivée : ");
scanf("%d",&k1);

for(i=1;i≤n;i++)
{
        a[i]=k1;
        q[i]=q[i]+a[i];

}

/*k1 sera le nombre total de messages en attente*/
k1=0;
for(i=1;i≤n;i++)
k1=k1+q[i];

k5=0;
k6=0;

/*on attend donc que k1=0*/
while(k1≠0)
{
        /*k5 va compter le nombre d'étapes*/
        k5++;

        /*on réalise l'affichage*/
        for(i=1;i≤n;i++)
        printf("%5d : %12d---%12d\n",i,q[i],b[i]);

        /*k3 sera le nombre d'utilisateurs qui tente*/
        k3=0;
        for(i=1;i≤n;i++)
        {
                k2=rand();

                /*ici, on tire au hasard pour savoir
                si i tente sachant son backoff*/

                /*k4 garde en mémoire l'utilisateur
```

```
        qui tente s'il est seul*/
        if(1.*k2/RAND_MAX<pow(2,-b[i]) && q[i]≠0)
        {tente[i]=1;k3++;k4=i;}

        else tente[i]=0;
}
printf("\n");

/*la seule façon de faire diminuer
le nombre de messages en attente*/
if(k3==1)
{
        q[k4]=q[k4]-1;
        b[k4]=0;
        k6++;
}

/*sinon, c'est qu'il y a collision
ou que personne ne tente*/
else
{

        /*on augmente le backoff
        de ceux qui tente */
        for(i=1;i≤n;i++)
        {
                if(tente[i]==1)
                b[i]++;

        }
}

/*ici, on fait venir
les éventuelles nouvelles arrivées*/
for(i=1;i≤n;i++)
{
        k2=rand();
        if(1.*k2/RAND_MAX<l/n)
        a[i]=1;

        else a[i]=0;
        q[i]=q[i]+a[i];
}
```

```
        /*ici on recalcule k1*/
        k1=0;
        for(i=1;i<=n;i++)
        k1=k1+q[i];

    }

    /*affichage du résultat*/
    printf("\n%d messages envoyés en %d étapes\n\n",k6,k5);
}
```

8.4 Rapport $\frac{A}{T}$

Le nombre de messages au début est A et T est le temps que met ALOHA* pour finir son premier CRI, donc pour envoyer les A messages. On obtient une moyenne du nombre T d'étapes pour terminer un CRI commençant avec A messages en faisant une moyenne sur le résultat de 10^5 simulations.

A=1 : A/T=1.000000
A=2 : A/T=0.410090
A=3 : A/T=0.340261
A=4 : A/T=0.373408
A=5 : A/T=0.436583
A=6 : A/T=0.420295
A=7 : A/T=0.359140
A=8 : A/T=0.390307
A=9 : A/T=0.424119
A=10 : A/T=0.394055
A=11 : A/T=0.394926
A=12 : A/T=0.408720
A=13 : A/T=0.349956
A=14 : A/T=0.365851
A=15 : A/T=0.357961
A=16 : A/T=0.367719
A=17 : A/T=0.315595
A=18 : A/T=0.378377
A=19 : A/T=0.366906
A=20 : A/T=0.385634
A=21 : A/T=0.399219
A=22 : A/T=0.346744
A=23 : A/T=0.356600
A=24 : A/T=0.377212
A=25 : A/T=0.371430
A=26 : A/T=0.359376
A=27 : A/T=0.362664
A=28 : A/T=0.361909
A=29 : A/T=0.354217
A=30 : A/T=0.370937
A=31 : A/T=0.362736
A=32 : A/T=0.357547
A=33 : A/T=0.355026
A=34 : A/T=0.358451
A=35 : A/T=0.357996
A=36 : A/T=0.372720
A=37 : A/T=0.354134
A=38 : A/T=0.353163
A=39 : A/T=0.334400
A=40 : A/T=0.347315
A=41 : A/T=0.352805
A=42 : A/T=0.338048
A=43 : A/T=0.349130
A=44 : A/T=0.348524
A=45 : A/T=0.352342
A=46 : A/T=0.351149
A=47 : A/T=0.341856
A=48 : A/T=0.357211
A=49 : A/T=0.364855
A=50 : A/T=0.353289
A=51 : A/T=0.353311
A=52 : A/T=0.367206
A=53 : A/T=0.346424
A=54 : A/T=0.365541
A=55 : A/T=0.351635
A=56 : A/T=0.366532
A=57 : A/T=0.348354
A=58 : A/T=0.368009
A=59 : A/T=0.355498
A=60 : A/T=0.349691
A=61 : A/T=0.346440
A=62 : A/T=0.351895
A=63 : A/T=0.358740
A=64 : A/T=0.354306
A=65 : A/T=0.347826
A=66 : A/T=0.354080
A=67 : A/T=0.353881
A=68 : A/T=0.342097
A=69 : A/T=0.351563
A=70 : A/T=0.346580
A=71 : A/T=0.344906
A=72 : A/T=0.352258
A=73 : A/T=0.353422
A=74 : A/T=0.345198
A=75 : A/T=0.357956
A=76 : A/T=0.354358
A=77 : A/T=0.348385
A=78 : A/T=0.353282
A=79 : A/T=0.361106
A=80 : A/T=0.348385
A=81 : A/T=0.345310
A=82 : A/T=0.343676
A=83 : A/T=0.348510
A=84 : A/T=0.347410
A=85 : A/T=0.346851
A=86 : A/T=0.352308
A=87 : A/T=0.355386
A=88 : A/T=0.352971
A=89 : A/T=0.350452
A=90 : A/T=0.355721
A=91 : A/T=0.351707
A=92 : A/T=0.349297
A=93 : A/T=0.346374
A=94 : A/T=0.351613
A=95 : A/T=0.350967
A=96 : A/T=0.352473
A=97 : A/T=0.344355
A=98 : A/T=0.358190
A=99 : A/T=0.345390
A=100 : A/T=0.345018
A=101 : A/T=0.356855
A=102 : A/T=0.349906
A=103 : A/T=0.352167
A=104 : A/T=0.347050
A=105 : A/T=0.344596
A=106 : A/T=0.349147
A=107 : A/T=0.348646
A=108 : A/T=0.348501
A=109 : A/T=0.348385
A=110 : A/T=0.343757
A=111 : A/T=0.350387
A=112 : A/T=0.349539
A=113 : A/T=0.354398
A=114 : A/T=0.344341
A=115 : A/T=0.352861
A=116 : A/T=0.344952
A=117 : A/T=0.344361
A=118 : A/T=0.347378
A=119 : A/T=0.345716
A=120 : A/T=0.351659
A=121 : A/T=0.348284
A=122 : A/T=0.353771
A=123 : A/T=0.350863
A=124 : A/T=0.346073
A=125 : A/T=0.349710
A=126 : A/T=0.349487
A=127 : A/T=0.352472
A=128 : A/T=0.353961
A=129 : A/T=0.347386
A=130 : A/T=0.347261
A=131 : A/T=0.352903
A=132 : A/T=0.350676
A=133 : A/T=0.345563
A=134 : A/T=0.349061
A=135 : A/T=0.352726
A=136 : A/T=0.349074
A=137 : A/T=0.348352
A=138 : A/T=0.349417
A=139 : A/T=0.353234
A=140 : A/T=0.346100
A=141 : A/T=0.352809
A=142 : A/T=0.350537
A=143 : A/T=0.350779
A=144 : A/T=0.343592
A=145 : A/T=0.353374
A=146 : A/T=0.348993
A=147 : A/T=0.348327
A=148 : A/T=0.348562
A=149 : A/T=0.350404
A=150 : A/T=0.347101
A=151 : A/T=0.351261
A=152 : A/T=0.348109
A=153 : A/T=0.348888
A=154 : A/T=0.347906
A=155 : A/T=0.347420
A=156 : A/T=0.346357
A=157 : A/T=0.348400
A=158 : A/T=0.349860
A=159 : A/T=0.346949
A=160 : A/T=0.346065
A=161 : A/T=0.348836
A=162 : A/T=0.355025
A=163 : A/T=0.350282
A=164 : A/T=0.347602
A=165 : A/T=0.347684
A=166 : A/T=0.346612
A=167 : A/T=0.348741
A=168 : A/T=0.349003
A=169 : A/T=0.347909
A=170 : A/T=0.349528
A=171 : A/T=0.346483
A=172 : A/T=0.349375
A=173 : A/T=0.345895
A=174 : A/T=0.347784
A=175 : A/T=0.346571
A=176 : A/T=0.345072
A=177 : A/T=0.344721
A=178 : A/T=0.348848
A=179 : A/T=0.352099
A=180 : A/T=0.349435
A=181 : A/T=0.346520
A=182 : A/T=0.346760
A=183 : A/T=0.345614
A=184 : A/T=0.347346
A=185 : A/T=0.347312
A=186 : A/T=0.344516
A=187 : A/T=0.348988
A=188 : A/T=0.345910
A=189 : A/T=0.345607
A=190 : A/T=0.349354
A=191 : A/T=0.345688
A=192 : A/T=0.350392
A=193 : A/T=0.351088
A=194 : A/T=0.350061
A=195 : A/T=0.346431
A=196 : A/T=0.350198
A=197 : A/T=0.347994
A=198 : A/T=0.350232
A=199 : A/T=0.345045
A=200 : A/T=0.347513
A=201 : A/T=0.350217
A=202 : A/T=0.350409
A=203 : A/T=0.352333
A=204 : A/T=0.351476
A=205 : A/T=0.347264
A=206 : A/T=0.346657
A=207 : A/T=0.348298
A=208 : A/T=0.346581
A=209 : A/T=0.349276
A=210 : A/T=0.348672
A=211 : A/T=0.349056
A=212 : A/T=0.348360
A=213 : A/T=0.347074
A=214 : A/T=0.348019
A=215 : A/T=0.349920
A=216 : A/T=0.347543
A=217 : A/T=0.346864
A=218 : A/T=0.344951
A=219 : A/T=0.347307
A=220 : A/T=0.347645
A=221 : A/T=0.349261
A=222 : A/T=0.351233
A=223 : A/T=0.348589
A=224 : A/T=0.347400
A=225 : A/T=0.346609
A=226 : A/T=0.348959
A=227 : A/T=0.346714
A=228 : A/T=0.350114
A=229 : A/T=0.347499
A=230 : A/T=0.346019
A=231 : A/T=0.347686

Index