Projet : Simulation des algorithmes des tickets et des estampilles de Lamport dans les systèmes distribués

Yves Caniou, *caniou@loria.fr*

23 janvier 2004

1 Introduction

Les besoins informatiques nécessitent de plus en plus l'utilisation de ressources réparties. Elles sont de plusieurs natures : ressources de calcul, de mémoire, interfaces, périphériques, données. On peut par exemple utiliser des grappes de machines, ou *clusters*, composées de machines reliées par un réseau local pour exécuter une application parallèle. Mais cela implique de nouvelles contraintes quant à la gestion des ressources utilisées. En effet, une application tournant sur un cluster, par exemple celui de l'ENSEM auquel vous avez accès par les machines présentes dans les salles machines, va nécessiter quelques précautions afin que l'exploitation des machines et/ou des données soient possibles de manière cohérente : des problèmes de synchronisation apparaissent.

Nous avons vu en cours quelques-uns des problèmes engendrés par la distribution des processus. Nous les avons expliqués et résolus grâce à des outils et des algorithmes précis.

2 Objectifs

Le travail demandé est l'application du cours pour des problèmes de synchronisation réduits à l'accès à une section critique.

Le but est de simuler des problèmes de concurrence à des données (*m*utex) et de les résoudre par les algorithmes des tickets et des estampilles de Lamport ([2], [1]). Votre travail devra prendre en compte le fait que les processus simulant les sites ne sont pas nécessairement sur une même machine.

Le rapport, que vous êtes invités à composer sous LaTeX ([4]), devra comprendre la présentation du problème, les moyens mis en oeuvre (ne refaites pas le cours ! C'est plutôt son complément qui est intéressant) et des détails sur leurs implantations. Il devra surtout rendre compte des avantages et/ou des inconvénients des deux algorithmes, appuyés par des observations faites grâce aux quelques scénarii d'exécution, et des traces conséquentes du comportement du système que vous commenterez. Attention, ce rapport ne devra pas excéder une vingtaine de pages !

Pour simplifier la mise en œuvre et l'utilisation des résultats obtenus, vous pourrez par exemple utiliser un processus relais `R` qui simule le réseau, et donc la communication entre les sites (Fig. 1). Ainsi, il sera plus 'naturel' de modifier les paramètres de votre réseau, d'une exécution à l'autre ou pour simuler une évolution du système. Cependant, n'oubliez pas dans ce cas d'expliquer les hypothèses ainsi faites.

De plus, lorsqu'un site souhaite entrer en section critique (ce qui peut se faire à la pression d'une touche. . .), le processus du site doit envoyer des messages aux autres. Il pourra le fera par l'intermédiaire d'une unique fonction `envoiMessage()` à laquelle vous passerez tous les arguments nécessaires (dont naturellement **la** cible, le message. . .). Le but de cette fonction est de contacter le processus `R` qui fera ensuite selon votre scénario.

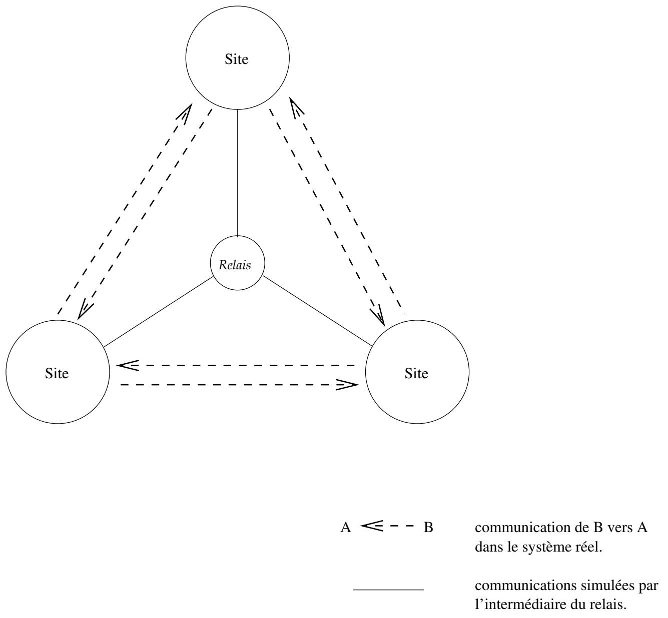

FIG. 1 – Un relais assure la communication entre les sites

Vous pouvez vous appuyer sur le TP sur TCP que vous avez fait en 2ème année pour la gestion de l'envoi des messages ainsi que sur le livre [3] disponible à la bibliothèque universitaire ou encore sur les documentations que vous pouvez obtenir à l'URL : `http://www.advancedlinuxprogramming.com/`.

Bon Projet !

Références

[1] A.K Agrawala G. Ricart. An optimal algorithm for mutual exclusion in computer networks. In *Comm. ACM, vol.24,1*, jan 1981.

[2] L. Lamport. Time, clocks and the ordering of events in a distributed system. In *Comm. ACM, vol.21,7*, july 1978.

[3] N. Matthew and al. *Programmation Linux*. Broché, 2000.

[4] T. Oetiker, H. Partl, I. Hyna, and E. Schlegl. Une courte (?) introduction a latex2e. http ://tex.loria.fr/english/general.html.