TD/TP Utilisation de la programmation multi-threads et du multiplexage pour implémenter un algorithme réparti
Durée estimée : 3h

L'objectif de ce TD/TP est de mettre en oeuvre l'algorithme de la diffusion, dit "fiable", vu dans le premier cours (introduction). Ce travail sera une base pour un prochain TP.

Indications pour la réalisation de ce TP :

- Le graphe d'interconnexion est quelconque et est bidirectionnel. Utiliser votre programme P_{config} mis en oeuvre lors des précédents TP. Si vous n'avez pas un tel programme, commencer par une configuration manuelle du réseau en impliquant jusqu'à 4 processus. Cette alternative ne vous dispense en aucun cas de la nécessité d'avoir un programme automatisant la construction d'un réseau quelconque.
- N'importe quel processus peut diffuser un message à n'importe quel moment. Pour faire simple, un processus P_i effectuera une diffusion d'un message toutes les $intervalle_i$ secondes (donnée passée en paramètre du processus P_i).
- Choisir des structures avec des types simples pour les messages échangés.
- Produire une trace d'exécution pertinente.
- Une fois les premiers tests de votre système effectués, produire une situation d'arrivée de deux messages dans le désordre. Remarque : provoquer un tel test pourrait naturellement nécessiter d'adapter votre programme.
- Si vous souhaitez atteindre un niveau plus avancé, faire des tests en provoquant la panne d'un processus qui aurait commencé la diffusion mais ne l'aurait pas terminée. L'exécution doit pourvoir se poursuivre sans erreur après la disparition de ce processus et tous les processus restants doivent avoir reçu le message que ce dernier a commencé à diffuser.

Important : décomposer la réalisation de ce travail en plusieurs étapes et de manière incrémentale. Cela sera bien plus efficace.

Pour rendre ce TP plus attractif et être plus sensibilisé à un contexte réparti, travailler en groupe impliquant plusieurs personnes, chacune doit implémenter un processus P_i et l'exécuter sur son poste de travail. Une contrainte doit toutefois être respectée : définir un protocole d'échange commun pour que les différents processus se "comprennent".