UE INF4003 : TP 2
Parcours parallèle de graphe

Frédéric Raimbault

Rappel : ce TP doit être réalisé avant la séance, il sera testé et noté en début de séance. Au début de la séance, un travail complémentaire vous sera remis et vous serez évalué en temps limité sur ce nouveau travail. Les TP sont testés sous linux, sur les PC des salles de TP et sont exécutés sur le cluster *pedago*. Votre travail doit rester personnel et tout code partagé, repris d'un autre étudiant ou copié sur le web sera considéré comme une tentative de fraude et traité en conséquence.

1 Construction d'une table de routage

L'objectif de ce TP est de construire une table de routage pour la diffusion dans un graphe. Cette table permettra ensuite à chaque nœud de déterminer vers quels voisins il doit propager le message de diffusion qu'il a reçu. Dans un premier temps on se place dans l'hypothèse où les messages de diffusion qui circulent dans le réseau ont toujours la même origine, le nœud racine d'identifiant 0. La figure 1 illustre la propagation d'un message en trois phases diffusé à partir du nœud racine, une fois la table de routage de chaque nœud construite. L'arbre recouvrant qui a été exploité est mis en évidence par des arêtes doubles. A noter que cet arbre recouvrant n'est pas unique.

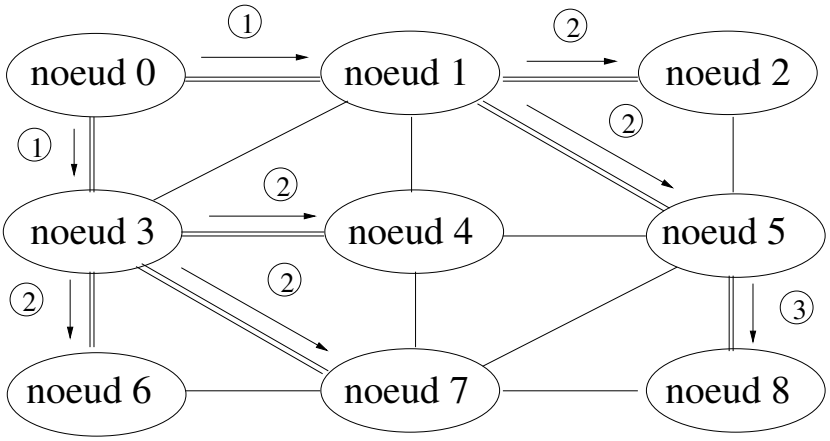

Fig. 1: Exemple de propagation d'un message diffusé à partir du nœud racine

2 Mise en œuvre

L'objectif présenté au paragraphe précédent se ramène au calcul d'un arbre recouvrant à partir du nœud racine. Pour construire la table de routage sur chaque nœud vous vous appuierez donc sur l'algorithme de parcours parallèle vu en cours et que vous mettrez en œuvre à l'aide de la bibliothèque de communication réalisée au précédant TP [1].

3 Travail à réaliser

Vous trouverez sur l'ENT les éléments suivants à compléter.

— La javadoc de la classe `spanning_tree.SpanningTree` qui construit les tables de routage ; vous écrirez la totalité du code correspondant. Notez que tous les nœuds doivent avoir terminé leur initialisation (méthode `SpanningTree`) avant que le premier message soit diffusé dans le réseau ; sinon certains nœuds échoueront sur la réception du message pour lequel ils n'ont pas encore ajouté de handler. La solution est de mettre en place une barrière de synchronisation entre tous les noeuds à l'aide de la méthode `process.waitNeighbouring()` au début de la méthode `SpanningTree.make()`.

— Le source du programme test `spanning_tree.Broadcast` qui lit la configuration d'un réseau, appelle votre programme de construction des tables de routage, et pour le nœud racine, diffuse un message dont le contenu est affiché par chaque nœud. Vous adapterez le script développé au précédent TP pour lancer ce programme sur le cluster.

1. Si votre bibliothèque de communication ne fonctionne pas vous pourrez utiliser la mienne qui est disponible sur l'ENT.