UPMC
SORBONNE UNIVERSITÉS

AR (MI048) - Écrit réparti - Deuxième épreuve

Les deux exercices doivent être traités sur des copies séparées.

Exercice(s)

Exercice 1 – Algorithme par vague

On considère un réseau connexe dans lequel les liens entre nœuds sont bidirectionnels. On considère également que les communications sont fiables, c'est-à-dire que tout message émis est délivré au bout d'un temps arbitraire mais fini.

Au départ chaque nœud ne connait que lui-même et ses voisins directs, et exécute l'algorithme de Finn :

Ensemble Inc_p de processus
    init à $\{p\}$
Ensemble $Ninc_p$ de processus
    init à $\emptyset$
Tableau $rec_p[q]$ de booléens
    inits à $faux$

BEGIN
    Si p est initiateur alors
        $\forall r \in Voisins_p$: Envoyer (Inc_p, $Ninc_p$) à r
    TantQue ($Inc_p \neq Ninc_p$)
        Recevoir(Inc, $Ninc$) de q
        $Inc_p \leftarrow Inc_p \cup Inc$
        $Ninc_p \leftarrow Ninc_p \cup Ninc$
        $rec_p[q] \leftarrow vrai$
        Si ($\forall q \in Voisins_p : rec_p[q]$) alors
            $Ninc_p \leftarrow Ninc_p \cup \{p\}$
        Si (Inc_p ou $Ninc_p$ a changé) alors
            $\forall r \in Voisins_p$: Envoyer (Inc_p, $Ninc_p$) à r
    FinTantQue
    $Decision$
END

Question 1

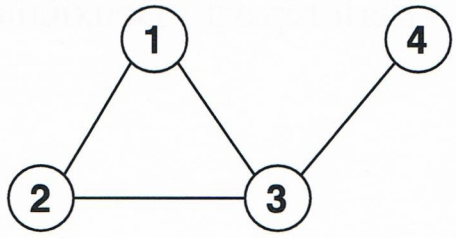

FIGURE 1 – Réseau pour la trace d'exécution

On désire effectuer une trace d'exécution de l'algorithme de Finn dans le graphe représenté en figure 1. Complétez le tableau suivant à cet effet.

Site	Opération	Rec_{Site}	Inc_{Site}	$Ninc_{Site}$	Décision (O/N)
1	Send[2, 3]({1}, ∅)	2F 3F	1	∅	N
2	Recv[1]({1}, ∅)	1V 3F	1, 2	∅	N
3	Recv[1]({1}, ∅)	1V 2F 4F	1, 3	∅	N
2	Send[1, 3]({1, 2}, ∅)	1V 3F	1, 2	∅	N

Question 2

Dans l'exemple de la figure 1, quels sont les nœuds qui peuvent être initiateurs ? Est-ce vrai pour n'importe quel autre graphe non orienté connexe ?

Question 3

Que se passe-t-il si aucun nœud n'est initiateur ? Déduisez-en une hypothèse pour le bon fonctionnement de cet algorithme.

Question 4

Cet algorithme fonctionne-t-il dans un graphe orienté ? Si oui, à quelle(s) condition(s) ?

Question 5

Quelle est la complexité de cet algorithme en nombre de messages ? Justifiez votre réponse.

Question 6

Sans aller jusqu'à le démontrer, expliquez pourquoi tous les nœuds du réseau finiront par décider.

Exercice 2 – Pair à Pair

Pour chaque exercice, une série d'affirmations sont proposées. Entourer toutes les affirmations vraies et barrer toutes les affirmations fausses. Motiver brièvement chaque choix. Une affirmation qui n'est ni entourée, ni barrée n'ajoute ni ne retire aucun point.

1. Les architecture P2P actuelles sont plus largement utilisées pour implanter des applications de :
   - Traitement de texte
   - Messagerie instantanée
   - Systèmes de fichiers distribués
   - Télévision en direct
2. Pour minimiser le nombre de messages utilisés pour chaque requête de recherche, il vaut mieux utiliser :
   - CAN que Gnutella ;
   - Gnutella que Chord ;
   - Chord que CAN ;
3. Parmi les architectures suivantes, quelle(s) est(sont) celle(s) qui est(sont) la(les) plus adaptée(s) pour le stockage de grands fichiers (plusieurs giga octets) sur les pairs :
   - CAN
   - Chord
   - Freenet
   - Gnutella
   - Skipe
4. Parmi les architectures suivantes, quelle(s) est(sont) celle(s) qui est(sont) la(les) plus adaptée(s) pour les applications de découverte de services :
   - Pastry
   - Chord
   - Skype
   - Bittorent
   - Gnutella
5. Considérons le système Chord vu en cours. Le système repose sur une topologie en anneau. Un noeud Chord a la connaissance de son prédécesseur et du noeud suivant. Une fonction de hachage génère une clé pour chaque noeud à partir de son adresse IP. Ensuite, chaque noeud est placé dans l'anneau de manière à ordonner les clés par ordre croissant. La table de routage maintenue par chaque noeud N s'exprime en O(log(N)). Toutefois, la simple connaissance de son prédécesseur et de son suivant présente des performances médiocres (recherche linaire en nombre de noeuds à parcourir pour acheminer une requête). Afin de palier ce problème chaque noeud p se voit doté d'une entrée vers les noeuds (appelés fingers) de clé $suivant(p + 2^i)$ où i varie de 0 à m (pour un espace de clés compris dans l'intervalle $[0, 2^m - 1]$).

   Nous considérons l'implantation des fonctions d'insertion, de recherche de noeud et de départ. Pour simplifier le problème nous suposons que tout noeud notifie ses voisins avant de sortir du système (pas de panne brutale). Ecrire le pseudocode de ces fonctions. Reprendre la fonction de départ et proposer un pseudocode de réparation de la structure Chord au cas où un départ imprévu est détecté.