Programmation répartie

HEIG-VD / PRR / Labo 3

Temps à disposition : 8 périodes
Travail à réaliser par groupe de 2 étudiants.
La présence aux labos est obligatoire. En cas d'absences répétées, l'étudiant sera pénalisé.
En cas de copie manifeste entre les rendus de deux labos, tous les étudiants des deux groupes se verront affecter la note de 1.
Distribué le mercredi 4 décembre 2019 à 14h55.
A rendre le mercredi 18 décembre 2019 à 16h30. *Une démonstration pourra être demandée.*

Forme du rendu : impression des fichiers sources à remettre en séance, archive du projet par email à l'assistant et à l'enseignant. Readme indiquant comment l'utiliser et précisant ce qui a été réalisé, ce qui fonctionne et ce qui reste à faire.

Objectifs

- Comprendre le fonctionnement d'un algorithme d'élection avec pannes.
- Utiliser l'algorithme de Chang et Roberts avec pannes de processus pour déterminer le processus de meilleure aptitude parmi un ensemble de processus.
- Réaliser des communications UDP ou TCP en langage Go

Critères d'évaluation

La note du labo sera constituée de 2 notes au demi-point prêt, chacune étant comprise entre 0 et 2,5 points : une pour le bon fonctionnement du programme et une pour la qualité du rendu. La note du labo (sur 6 points) sera obtenue par : 1 + la somme de ces 2 notes.

Le bon fonctionnement du programme est obtenu s'il couvre toutes les fonctionnalités de l'énoncé du labo, s'exécute de façon performante et ne présente pas de bugs.

La qualité du rendu sera évaluée en tenant compte des points suivants :

- Facilité et rapidité de la mise en œuvre du programme (activation, aide, paramétrage), en particulier si on utilise un seul PC pour le tester (options par défaut bien choisies).
- Facilité et rapidité de la vérification de son fonctionnement : traces éventuellement paramétrables, datées et signées, mais en tout cas claires et adéquates.
- Possibilités de paramétrage pour simuler des conditions réseau réelles (uniquement les délais de transmission car le réseau est supposé sans pannes) dans les limites de l'énoncé.
- Réalisation de tests automatisés avec simulation de processus ou d'une partie de l'application (mocking).
- Conception du code source (structure et décomposition). Possibilité de réutilisation d'une partie du code avec un autre énoncé (autre couche de transport réseau, …). Cependant, il ne doit pas faire plus que ce qui est demandé, ni être prévu pour des extensions hypothétiques.
- Qualité, simplicité, concision et lisibilité du code source. Conformité au format de code source et aux bonnes pratiques préconisées pour le langage.
- Documentation des entêtes de modules et de fonctions. Commentaires adéquats des instructions : un commentaire trivial ou inutile est aussi pénalisant que l'absence d'un commentaire nécessaire à la compréhension.
- Lisibilité du code imprimé : pagination adéquate (fonction sur une seule page ou bien découpée), entêtes et fin de pages, titres des modules et fonctions apparents, police adéquate et indentation bien choisie (pas de retour ligne automatique).

Énoncé du problème

Réalisez un programme en langage GO qui implémente l'algorithme d'élection de Chang et Roberts avec pannes possibles des processus que nous avons vu en classe. Pour ce faire, nous avons des processus qui, de temps en temps, interrogent le dernier site élu, et si celui-ci n'est plus opérationnel, une élection est démarrée. Un processus en panne doit pouvoir se réinsérer au sein de l'anneau lors de sa reprise.

Hypothèses

1. Le réseau qui interconnecte les sites est entièrement fiable, seuls les processus peuvent tomber en panne.
2. La communication entre les sites se fait uniquement par protocole UDP (point-à-point).
3. Afin de simplifier la construction de l'anneau, vous êtes libre de prendre toutes les hypothèses qui simplifient le problème.
4. Les adresses IP, numéro de port et aptitudes des processus sont configurés avant le démarrage de l'application. Les aptitudes sont invariables. Le site devant être élu est le site possédant l'aptitude la plus grande. En cas d'égalité de deux aptitudes, le site ayant l'adresse IP la plus grande devient l'élu.
5. Pour simuler une panne, il vous suffit de terminer un processus.
6. Un processus se composera de 2 taches : un gestionnaire d'élections connaissant l'élu et les processus de l'environnement réparti, et une tache applicative ayant pour rôle d'obtenir l'élu de son gestionnaire pour interroger périodiquement le processus élu avec un simple écho. C'est cette tâche qui lance une élection auprès de son gestionnaire local s'il constate la panne de l'élu.