Labo 1 : Implémentation d'un algorithme distribué en utilisant les sockets

04 Octobre 2017

Ce labo se fait par binôme. Chaque binôme choisit un algorithme à implémenter parmi les 7 listés en bas. Pour chaque binôme, le professeur/assistant informera le binôme du langage de programmation à utiliser : java ou python.

L'objectif de ce travail pratique est de programmer l'un des algorithmes suivants :

1. Construction d'un arbre de recouvrement en largeur d'abord
2. Construction d'un arbre de recouvrement en profondeur d'abord
3. Construction d'un arbre de recouvrement sans racine connue à l'avance
4. Algorithme *Test_Connectivity*
5. Algorithme du plus court chemin (version synchrone)
6. Algorithme d'élection (version synchrone)
7. Algorithme Gnutella

Les messages sont transmis entre les nœuds du réseau via les sockets. Chaque nœud est serveur et client en même temps. Il est implémenté sur une machine différente de celles allouées aux autres nœuds.

Un nœud est représenté par :

- Un numéro de port,
- L'ensemble des voisins du nœud

Ces informations sont stockées dans un fichier .txt (texte) ayant comme nom : *voisin-x.txt*. *x* étant le numéro du nœud.

Exemple :

Nom du fichier : *voisin-01.txt*. Ce fichier représente le nœud 1

Contenu du fichier :

129.194.184.101 // identifiant du 1er voisin
129.194.184.110 // identifiant du 2ème voisin
129.194.184.111 // identifiant du 3ème voisin

Par souci de compatibilité, il est ***impératif*** de respecter ce format

Le programme *distributed_alg* à écrire doit être appelé comme suit :

distributed_alg num_port voisin-x.txt INIT

ou

distributed_alg num_port voisin-x.txt WAIT

INIT : signifie que le nœud en question commence l'exécution sans attendre la réception du premier message.

WAIT : signifie que le nœud en question attend la réception du premier message avant de commencer à envoyer des messages.

voisin-x.txt est le fichier qui doit être lu par le nœud concerné.

On vous demande d'écrire le programme *distributed_alg* (qui représente l'un des algorithmes cités en haut) et de le déployer sur un réseau d'ordinateurs en utilisant un script shell.

Le développement de ce programme doit se faire en python ou Java.

Vous devez suivre la chronologie suivante :

Etapes	Livrables
Etape 1 : Conception du protocole de communication entre les nœuds. Conception d'une structure de données distribuée adaptée à l'algorithme distribué à implémenter	Discussion avec l'assistant ou le professeur. Il est nécessaire d'avoir l'accord de l'assistant ou le professeur avant de passer à l'étape 2.
Etape 2 : Ecriture du programme *distributed_alg* et son exécution sur un seul ordinateur. Chaque nœud est représenté par un serveur socket installé sur un port différent.	Une démonstration doit être faite au professeur/assistant avant de passer à l'étape 3. Dans le cas des algorithmes 1, 2 et 3, ces derniers doivent être à 100% opérationnels. Dans le cas des algorithmes 4, 5, 6 et 7, le livrable peut ne pas être complétement finalisé. Mais l'ossature de l'algorithme doit être implémentée (programmée). **Délai : Mercredi 11 Octobre en fin de séance.**
Etape 3 : Déploiement manuel sur plusieurs nœuds (utilisation de la plateforme Cloud *hepiaCloud*)	Une démonstration doit être faite au professeur/assistant avant de passer à l'étape 4. **Délai : Mercredi 18 Octobre en fin de séance.**
Etape 4 : Ecriture et exécution d'un script qui déploie le programme *distributed_alg* sur hepiaCloud. Le script doit permettre : • un login à distance sur les machines du réseau • une exécution de l'algorithme *distributed_alg*	Démonstration finale au professeur/Assistant. Remise, sur cyberlearn, d'un fichier compressé contenant : 1. Le code source de votre programme 2. Les fichiers .txt représentant un exemple de réseau (5 nœuds minimum) **Délai : 1er Novembre**

Critères d'évaluation :

1. Chaque nœud lit correctement le fichier « voisin » tel que décrit dans l'énoncé. Les noms des fichiers voisins doivent respecter l'énoncé. Ce critère sera évalué lors de l'étape 2.
2. L'algorithme est déployé en local. Dans ce cas, chaque nœud utilise un port différent. Ce critère sera évalué lors de l'étape 2.
3. Dans certains algorithmes, les envois de messages doivent se faire en parallèle. Dans ce cas, *distributed_alg* doit utiliser les threads. Ce critère sera évalué lors de l'étape 2.
4. L'algorithme est déployé manuellement sur 5 nœuds minimum (plateforme cible : hepiaCloud). Ce critère sera évalué lors de l'étape 3.
5. Le système est déployé en utilisant un script. Ce critère sera évalué lors de l'étape 4.