Manuel d'utilisation du programme `Pconfig`
HAI721I

Pablo LAVIRON
Faculté des Sciences
Université de Montpellier

18 octobre 2023

Résumé

Ce document a pour objectif d'expliquer comment utiliser le programme `Pconfig`, permettant la création d'un graphe quelconque de processus communiquant entre eux via des sockets TCP, et construit de façon centralisée en utilisant UDP. Des pistes d'implémentation avec un programme `P` sont également mises à disposition.

1 Utilisation du programme `Pconfig`

1.1 Exécution

Le programme `Pconfig` est compilé pour fonctionner sur les machines Linux de l'université. Pour exécuter le programme `Pconfig`, il suffit de lancer la commande suivante :

`./Pconfig <fileName> <port>`

Où `fileName` correspond au chemin vers le fichier décrivant le graphe à construire, et où `port` désigne le port sur lequel lancer le serveur Pconfig (pas besoin de préciser l'adresse IP : la socket UDP de Pconfig est configurée pour écouter sur toutes les adresses de la machine).

De nombreuses traces d'exécution sont affichées lors de l'exécution de `Pconfig`. De plus, pour éviter tout problème, le programme terminera si une erreur se produit.

Un exemple d'exécution du programme `Pconfig` est disponible en annexe.

1.2 Fichier de graphe

Le format utilisé pour décrire le graphe à construire est celui proposé sur le site http://cedric.cnam.fr/~porumbed/graphs/. Voici un exemple de graphe simple que vous pouvez utiliser pour effectuer vos tests, ainsi que son fichier associé :

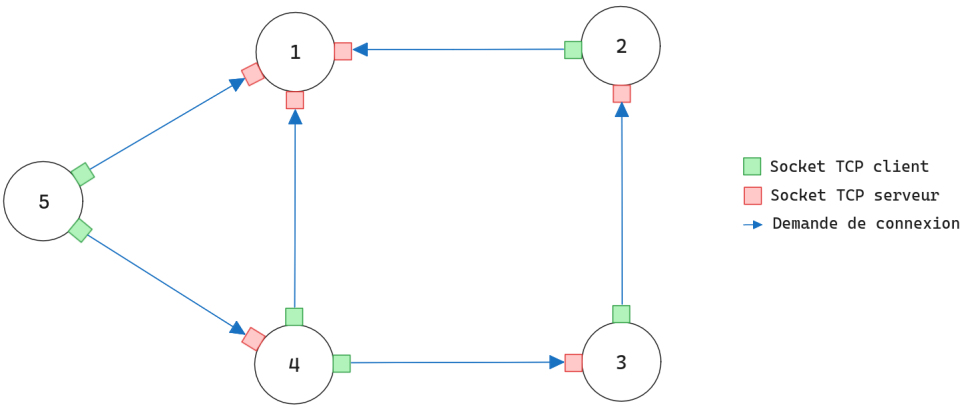

FIGURE 1 – Graphe d'exemple

```
c FILE: TEST
c
p edge 5 6
e 2 1
e 3 2
e 4 3
e 4 1
e 5 4
e 5 1
```

FIGURE 2 – Fichier décrivant le graphe d'exemple

Comme vous pouvez le constater, les lignes commençant par le caractère «c» sont des commentaires, l'unique ligne commençant par «p» contient des paramètres (nombre de sommets et nombre d'arêtes), et celles de la forme «e i j» sont les arêtes qui relient les sommets i et j.

Une fois le programme `Pconfig` lancé, ce dernier va attendre un message de la part de tous les sites P_i contenant l'adresse TCP de leur socket serveur en écoute (permettant d'accepter des clients). Ensuite, `Pconfig` envoie pour chaque ligne de fichier «e i j», l'adresse de la socket de P_j à P_i (P_j est le serveur et P_i le client). Le client P_i devra alors créer une socket et envoyer une demande de connexion vers P_j (comme illustré dans le schéma du graphe). Lorsqu'il n'y a plus de sites P_j auxquels un site P_i doit se connecter, P_i en est notifié (c'est comme ça qu'il sait qu'il doit arrêter les réceptions). Chaque site P_i recevra par la suite le nombre de sites P_j qui vont se connecter à lui, et devra donc accepter ces sites et ainsi obtenir les sockets TCP correspondantes (en rouge sur le schéma).

L'ordre dans lequel les sites P_i envoient leur adresse à $Pconfig$ n'est pas important.

2 Eléments d'implémentation avec un programme `P`

Le programme `P` correspond au comportement adopté par chaque site (ou sommet du graphe) P_i. Il s'agit d'un seul programme, dont l'indice i sera donné en paramètre.

2.1 Arguments

Votre programme `P` devra au minimum contenir les paramètres suivants :

— Indice i du site P_i

— Adresse IP du serveur exécutant $Pconfig$

— Port du serveur exécutant $Pconfig$

Par exemple, `./P 1 127.0.0.1 9999` lance le site P_1 et se connectera à $Pconfig$ par l'adresse `127.0.0.1`, sur le port `9999`. Vous pourrez par la suite ajouter de nouveaux arguments en fonction de l'algorithme distribué que vous souhaiterez implémenter (par exemple l'intervalle de temps dans le TP 2).

2.2 Structures de données

La structure utilisée par `Pconfig` et `P` pour communiquer les couples (Identifiant, Adresse) des sites P_i est la suivante :

```
struct site {
    int id;
    struct sockaddr_in addr;
};
```

FIGURE 3 – Structure `site`

L'entier `id` correspond à l'identifiant du site P_i, c'est-à-dire à i. Le champ `addr` correspond à une adresse de socket. Celle-ci sera utilisée lors de l'envoi de l'adresse de la socket serveur d'un P_i à $Pconfig$,

ainsi que lors de la réception d'un couple <Identifiant, Adresse> (décrivant P_j) à qui P_i devra se connecter. Pour communiquer à P_i le fait qu'il n'y ait plus de sites P_j auxquels il doit se connecter, nous utiliserons un `id` avec une valeur de -1.

Un tableau de descripteurs de fichiers pourra être utilisé pour stocker les descripteurs des sockets des sites voisins. Comme la taille de ce tableau n'est pas connue à l'avance, il faudra l'initialiser à une taille fixe.

2.3 Algorithme

Voici une idée d'algorithme pour établir une communication entre un site P_i et $Pconfig$:

Algorithme 1 : Création d'un graphe d'interconnexion quelconque, site P_i

$dsPconfig \leftarrow$ créer la socket UDP pour communiquer avec Pconfig (en utilisant les paramètres passés en arguments du programme);
$dsServ \leftarrow$ créer une socket serveur TCP permettant d'accepter les demandes de connexion des sites voisins;
$adServ \leftarrow$ nommer la socket $dsServ$;
passer la socket $dsServ$ en mode écoute;
envoyer $< i, adServ >$ à Pconfig (en utilisant la structure `site`);
$voisins \leftarrow$ créer un tableau vide (contiendra les sockets des voisins);
$id \leftarrow 0$;
tant que $id \neq -1$ **faire**
    $< id, adr > \leftarrow$ recevoir une structure `site` de Pconfig;
    si $id = -1$ **alors**
        sortir de la boucle;
    sinon
        $ds \leftarrow$ créer une socket TCP;
        insérer ds dans $voisins$;
        envoyer une demande de connexion à adr sur ds;
    fin
fin
$nbClients \leftarrow$ recevoir un entier depuis Pconfig;
pour $n \leftarrow 0$ **à** $nbClients$ **faire**
    $ds \leftarrow$ accepter une connexion depuis $dsServ$;
    insérer ds dans $voisins$;
fin

Après cet algorithme, le graphe sera opérationnel. Pensez à ajouter une gestion des valeurs de retour pour les erreurs, et à libérer les descripteurs non utilisés (avec la fonction `free()`).

3 Annexes

```
$ ./Pconfig test.col 9999
Je suis le processus Pconfig
Nom du fichier graphe : test.col
Creation de la socket reussie
Nommage de la socket reussie
Ouverture du fichier reussie
Nombre de sites : 5, nombre d'aretes : 6
Debut de la reception des messages de tous les sites
Attente d'un message...
Message recu du site P1 : 127.0.0.1:36145
Attente d'un message...
Message recu du site P3 : 127.0.0.1:57711
Attente d'un message...
Message recu du site P4 : 127.0.0.1:51025
Attente d'un message...
Message recu du site P2 : 127.0.0.1:33731
Attente d'un message...
Message recu du site P5 : 127.0.0.1:45335
Tous les messages ont ete recus
Envoi de FIN (-1) a 1...
Message envoye
Envoi de l'adresse de P1 a P2...
Message envoye
Envoi de FIN (-1) a 2...
Message envoye
Envoi de l'adresse de P2 a P3...
Message envoye
Envoi de FIN (-1) a 3...
Message envoye
Envoi de l'adresse de P3 a P4...
Message envoye
Envoi de l'adresse de P1 a P4...
Message envoye
Envoi de FIN (-1) a 4...
Message envoye
Envoi de l'adresse de P4 a P5...
Message envoye
Envoi de l'adresse de P1 a P5...
Message envoye
Envoi de FIN (-1) a P1...
Message envoye
Fichier lu en entier
Envoi du nombre de clients a 127.0.0.1:40352 (3)...
Message envoye
Envoi du nombre de clients a 127.0.0.1:39030 (1)...
Message envoye
Envoi du nombre de clients a 127.0.0.1:40602 (1)...
Message envoye
Envoi du nombre de clients a 127.0.0.1:52026 (1)...
Message envoye
Envoi du nombre de clients a 127.0.0.1:45029 (0)...
Message envoye
Fermeture de ma socket reussie
Je termine
```