Outils d'analyse des serveurs

Luis D. Pedrosa & Jean-Cédric Chappelier 2017

Le travail de cette semaine est un peu particulier en ce sens que nous vous proposons non pas de développer des composants du cœur du système, mais des outils d'analyse de celui-ci, lesquels devraient vous permettre de plus facilement comprendre se qui se passe en détail et déboguer votre code si nécessaire.

Les outils que nous vous proposons de développer sont `pps-list-nodes` (avec '`s`') et `pps-dump-node` (sans '`s`'). :

I. `pps-list-nodes`

L'outil `pps-list-nodes` (à écrire dans `pps-list-nodes.c`) a pour but de tester tous les nœuds spécifiés dans le fichier *servers.txt* (du répertoire courrant d'où les exécutables sont lancés) et d'indiquer si le serveur correspondant est présent (= répond) ou non.

Cette commande ne prend aucun paramètre en entrée (ne lit rien sur l'entrée standard) et liste tous les nœuds de *servers.txt*, un par ligne, en utilisant le format : **"<IP> <Port> <OK|FAIL>\n"** (le séparateur entre chaque information est ici une seule espace). La commande indique « *OK* » si le serveur correspondant répond et « *FAIL* » sinon. Évidemment, le fichier doit être dans le même dossier que l'exécutable. Pour tester si un serveur répond, envoyez lui simplement un message vide. Le serveur devra alors reconnaître une telle requête simplement par sa taille et répondra alors lui aussi par un message vide.

Voici un exemple typique de ce que la commande `pps-list-nodes` doit donner :

```
$ ./pps-list-nodes
127.0.0.1 1234 OK
127.0.0.1 1235 OK
127.0.0.1 1236 FAIL
```

Pour tester votre outil, vous pouvez suspendre un serveur donné pendant un moment en utilisant `Ctrl-Z` et le faire reprendre avec la commande `fg`. Ouvrez par exemple trois terminaux, un pour chacun de deux serveurs et un pour `pps-list-nodes`, créez un fichier `servers.txt` comprenant deux serveurs, par exemple :

```
127.0.0.1 1234
127.0.0.1 6731
```

puis procédez comme suit :

```
TERM1> head -n 1 servers.txt | ./pps-launch-server

TERM2>tail -n +2 servers.txt | ./pps-launch-server

TERM3>./pps-list-nodes
127.0.0.1 1234 OK
127.0.0.1 6731 OK

TERM1> Ctrl-Z (affiche ^Z
[1]+  Stoppé                 head -n 1 servers.txt | ./pps-launch-server
)

TERM3>./pps-list-nodes
127.0.0.1 1234 FAIL
127.0.0.1 6731 OK

TERM1> fg

TERM2> Ctrl-Z (affiche ^Z
[1]+  Stoppé                 tail -n +2 servers.txt | ./pps-launch-server
)

TERM3>./pps-list-nodes
127.0.0.1 1234 OK
127.0.0.1 6731 FAIL

TERM2> fg

TERM1> Ctrl-C

TERM2> Ctrl-C
```

N'oubliez pas non plus d'utiliser `log-packets.so` et `packets.log` pour vérifier/déboguer vos communications si nécessaire.

II. `pps-dump-node`

L'outil `pps-dump-node` (à écrire dans `pps-dump-node.c`) a pour but d'afficher tout le contenu d'un nœud. Les nœuds à interroger sont donnés sur l'entrée standard, un par ligne au format **"<IP> <Port>\n"** (une seule espace entre

les deux). La commande fournit alors la liste des associations clé-valeur stockées dans le nœud correspondant, au format : **"<key> = <value>\n"**. Par exemple :

```
$ ./pps-dump-node
127.0.0.1 1234
key1 = value1
key2 = value2
```

Pour demander à un nœud de lister son contenu, envoyez lui le message particulier contenant exactement un seul octet contenant le caractère nul ('\0'). Puisque nous n'autorisons pas les clés à être la chaîne vide, ce message ne peut pas être confondu avec un `get` et le serveur peut donc le traiter comme un cas particulier.

En réponse, le serveur va envoyer un ou plusieurs messages composés décrivant son contenu. La première partie de ces messages est le nombre (entier 32 bits) de paires clé-valeur stockées dans ce nœud ; puis viennent ensuite en séquence dans le même message tant que possible (voir après) la liste des paires clé-valeur séparées chacune (clés et valeurs incluses) par un caractère nul ('\0') : **<key1>\0<value1>\0<key2>\0<value2>**.

Rappelez vous (cours de réseaux) que les messages UDP ne peuvent contenir au maximum que 65'507 octets. Il pourra donc être nécessaire d'envoyer plusieurs messages consécutifs. Seul le premier message doit commencer par le nombre total de paires clé-valeur. Il n'est pas nécessaire de le répéter ensuite ; les messages suivants ne contenant que des clés/valeurs. Par contre, une même paire clé-valeur ne doit pas être séparée sur deux messages UDP différents : si une clé et sa valeur ne peuvent plus être ajoutées au message courant, elles devront alors faire, les deux, partie du message UDP suivant. Le nombre total de paires, envoyé dans le tout premier message UDP, vous permettra de savoir quand le contenu aura été complètement reçu.

Si aucune message n'est reçu pendant un temps d'une seconde (et que la transmission est incomplète), `pps-dump-node` affichera alors simplement le message **"FAIL\n"**. (Pour rappel : c'est `get_socket()` qui fixe le temps d'attente.)

Pour créer la liste des paires clé-valeur d'un nœeud donné, il pourrait être utile de compléter et utiliser le type `kv_list_t` proposé dans `hashtable.h`.

III. Conseil et rendu

Le travail de cette semaine ne sera pas évalué séparément, mais devra faire partie du second rendu intermédiaire du projet (délai : le dimanche 13 mai 23:59). Néanmoins, nous vous conseillons comme toujours de continuer à travailler régulièrement et faire systématiquement des tags hebdomadaires (si votre travail correspondant est opérationnel) afin de profiter de nos comptes-rendus de tests automatiques. Le tag correspondant à cette semaine est `week08`.

Pour cette semaine, il faudrait mettre à jour tous les fichiers nécessaires (typiquement `hashtable.c`, `hashtable.h` si vous souhaitez utiliser les `kv_list_t` et `pps-launch-server.c`, sans oublier le `Makefile`, mis à jour pour que les cibles `pps-list-nodes` et `pps-dump-nodes` soient aussi compilées par défaut) et ajouter les fichiers `pps-list-nodes.c` et `pps-dump-nodes.c` au répertoire `done/` de votre dépôt GitHub (de groupe).

Par ailleurs, n'oubliez pas ce dimanche 22 avril de faire le quatrième rendu du semestre (premier rendu sur le projet) correspondant au travail effectué jusqu'à la semaine passée, à rendre sous le tag `projet01`.