Examen final rattrapage applications réseaux 2015

M1 MIAGE apprentissage (F. Sikora)

—1h30. Uniquement la javadoc est autorisée.
Inspiré d'E. Duris ou M. Chilowicz.—

▸ **Exercice 1.**

Le but de l'exercice est de proposer un serveur TCP de sondage pouvant gérer plusieurs clients à la fois. Des clients se connectent au serveur de sondage. Pour simplifier, on suppose que le serveur ne pose qu'une seule question concernant l'âge du client. Le protocole est donc le suivant :

```
Client connecté
S>C: Quel age ?
C>S: 42 ans
S>C: Merci, au revoir.
```

Ainsi le serveur doit simplement gérer une ligne terminée par un retour à la ligne au format UTF8.

Le serveur choisit de stocker les réponses de tous les clients à la suite dans un unique fichier (dont le chemin est donné en argument de la méthode main). Les réponses sont sauvegardées ligne par ligne en mode texte en utilisant l'encodage UTF-8. On pourra utiliser `new OutputStreamWriter(new FileOutputStream(chemin, true), "UTF-8")` pour ouvrir le fichier en mode ajout et y ajouter une nouvelle ligne de réponse. Implémenter un tel serveur pouvant gérer plusieurs clients simultanément (on ne demande pas de limite sur ce nombre) (si le serveur de peut gérer qu'un seul client à la fois, il y aura moins de points).

La NSA souhaite également pouvoir lire en temps réel les réponses. Pour cela, à l'aide du mot clé NSA en réponse à la question, un client accède à ce mode. Lorsque ce mode est activé, le client reçoit en temps réel une copie des réponses fournies par les clients connectés au serveur (jusqu'à fermeture de la connexion). Bien sûr, la NSA possède plusieurs employés voulant utiliser ce mode simultanément.

Créer une nouvelle classe pour le serveur possédant le mode satisfaisant la NSA. Vous pouvez tester avec netcat.

▸ **Exercice 2.**

On cherche à réaliser un serveur UDP permettant de faire des résolutions DNS. On testera le serveur en utilisant netcat (`nc -u ip port`).

1. On va dans un premier temps créer un serveur UDP réceptionnant des paquets et accusant réception. Chaque paquet reçu par ce serveur doit être de la forme `id:message` où `id` est un identifiant unique utilisé par la suite, et `message` est le message proprement dit, dans notre cas des noms de domaines dont on souhaite connaitre l'adresse IP. Ce premier serveur devra uniquement répondre `id:OK` dès qu'il reçoit un message dans le bon format (pas de résolution DNS donc, et avec `id` l'id envoyé par le client). On devra pouvoir exécuter le serveur avec la commande `java UDPServer1 port`. L'écrire. Ajouter également le code permettant de fermer proprement (fermeture de socket et terminaison propre du programme java) le serveur après un certain timeout (de votre choix).
2. On souhaite modifier le serveur (pour les besoins de la correction, créer une nouvelle classe). On veut qu'il serve à quelque chose et qu'il réponde au client avec des adresses IP plutôt que par OK. Ainsi, s'il reçoit un message `1:www.lamsade.dauphine.fr`, ce serveur devra répondre par `1:216.58.211.992` (il réutilise l'identifiant afin que le client puisse s'organiser). Vous pouvez utiliser la classe `InetAddress`.
3. Créer encore une nouvelle classe. Cette fois-ci, on veut pouvoir gérer les cas où le nom de domaine possède plusieurs adresses IP associées (`www.google.com` par exemple). De plus, on souhaite avoir un protocole plus économe en bande passante : au lieu d'envoyer les résultats en mode textuel, il indique les adresses IP sous forme binaire. L'identifiant id reste sous forme textuelle. De plus, après l'identifiant, on trouve le caractère ' :' utilisé comme séparararateur puis, avant l'adresse IP en elle même, on utilise un octet indiquant le format de l'IP qui suit (IPv4 ou IPv6). Pour résumer, le paquet est de la forme suivante : `OctetsIdentifiantTextuel OctetCaractere: OctetFormatIP1 Octet1IP1 ... OctetNIP1 OctetFormatIP2 Octet1IP2 ... OctetNIP2`. Implémenter ce protocole permettant de répondre au client avec l'ensemble des adresses associées. Attention, vous ne pourrez plus vérifier la validité de l'implémentation avec netcat puisque ce dernier n'affiche qu'en mode texte ce qu'il reçoit. En option, vous pouvez donc écrire un client UDP pour ce protocole.