ESISAR

NE442 - Programmation répartie

Examen de TP – GROUPE 2

Table des matières

1 Consignes générales....1
2 Exercice 1 : Port scanning (7 points)....2
2.1 Présentation générale du problème....2
2.2 Préparation de votre machine....2
2.3 Ecriture du programme....2
3 Exercice 2 : Réalisation d'un pont TCP (13 points)....3
3.1 Présentation générale du problème....3
3.2 Préparation de votre machine....4
3.3 Ecriture du programme PontTCPTaille....4
3.4 Ecriture du programme PontTCP....5
3.5 Détermination de la taille du fichier avec netcat....5

1 Consignes générales

La durée de cet examen de TP est de **1H30.**
Vous devez impérativement travailler sur la station fournie (pas de PC personnel).

Attention : vous devez absolument respecter les consignes suivantes :

- un projet Eclipse par exercice
- le nom du projet Eclipse doit être **examen_exo1_[votrenom]** pour l'exercice 1, **examen_exo2_[votrenom]** pour l'exercice 2, …
- vous devez rendre vos travaux en suivant les instructions données par le serveur d'examen dont l'URL est indiquée au tableau.

L'accès à Internet n'est pas autorisé, tous les documents sont autorisés (documents papiers + une clé USB).

2 Exercice 1 : Port scanning (7 points)

2.1 Présentation générale du problème.

Un serveur UDP est lancé sur une machine, vous savez que ce serveur écoute en UDP sur un port compris entre 30 000 et 32 000.

Ce serveur fonctionne ainsi :

- il écoute sur un port X , X étant compris entre 30 000 et 32 000
- il attend un paquet UDP contenant le texte `hello`
- quand il reçoit ce paquet avec `hello`, il répond `ok`.

L'objectif de cet exercice est de réaliser un programme capable de déterminer ce port X et de l'afficher.

2.2 Préparation de votre machine.

Pour pouvoir tester votre programme, vous allez installer sur votre machine ce serveur UDP.

Pour cela, télécharger le fichier **port-scanning.jar** disponible sur le serveur d'examen.

Lancer ce serveur UDP avec la commande

```
java -jar  port-scanning.jar
```

Vous avez maintenant un serveur UDP écoutant sur un port X.

2.3 Ecriture du programme

Ecrivez maintenant une classe ClientScanner. Si on exécute le main de cette classe, on obtient l'affichage suivant :

```
Début du scanning des ports UDP de 30 000 à 32000 sur la machine
"127.0.0.1"
Le serveur UDP écoute sur le port X = 31 500
Fin du programme
```

Cet affichage devra se faire dans un délai inférieur à 1 minute.

Dans un commentaire tout en haut de cette classe, vous indiquerez

- la valeur de X que vous avez trouvé
- la logique utilisée par votre programme

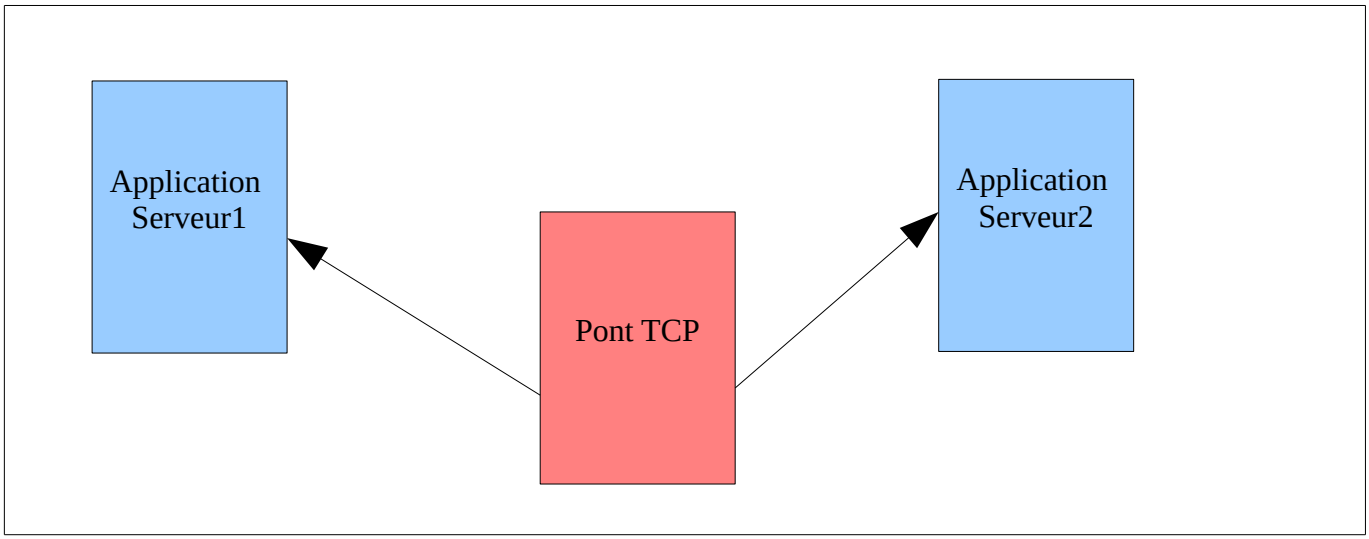
L'adresse IP de la machine scannée (ici `"127.0.0.1"`) sera écrite en dur dans votre programme.

Attention : votre programme doit fonctionner correctement même si le serveur UDP est distant (c'est à dire si on remplace 127.0.0.1 par une autre IP).

3 Exercice 2 : Réalisation d'un pont TCP (13 points)

3.1 Présentation générale du problème.

Vous allez devoir réaliser un pont TCP entre deux applications serveurs.



L'application 1 est un serveur TCP, qui écoute sur le port 8000. Ce serveur 1 fonctionne ainsi :

- quand une demande de connexion arrive, il envoie au client un fichier sous la forme d'un flux d'octets (avec uniquement le contenu du fichier , sans le nom du fichier). Le fichier a une taille d'environ 1 GigaOctet. Quand le serveur arrive à la fin du fichier, il coupe la connexion.

L'application 2 est un serveur TCP, qui écoute sur le port 8200. Ce serveur attend le contenu d'un fichier. Ce serveur 2 fonctionne ainsi :

- il attend les demandes de connexion
- quand une connexion est réalisée, il attend que le client lui envoie le contenu d'un fichier (toujours sous la forme d'un flux d'octets, sans le nom du fichier).
- si le contenu du fichier envoyé par le client est identique à celui envoyé par le serveur 1 : le serveur 2 répond « OK BIEN RECU » puis ferme la connexion.
- si le contenu du fichier par le client n'est pas identique à celui envoyé par le serveur 1 : le serveur 2 répond « ERREUR» puis ferme la connexion

A noter : le client de ce serveur 2 ne doit pas couper la connexion, la connexion est coupée par le serveur 2.

3.2 Préparation de votre machine.

Pour pouvoir tester votre programme, vous allez installer sur votre machine ces 2 serveurs TCP.

Pour cela, télécharger le fichier **pont-tcp.jar** disponible sur le serveur d'examen.

Lancer ces 2 serveurs avec la commande

```
java -jar  pont-tcp.jar
```

Il suffit d'executer une seule fois cette commande, et les deux serveurs seront actifs.

Vous avez maintenant

- un serveur TCP 1 écoutant sur le port 8000 fournissant un fichier
- un serveur TCP 2 écoutant sur le port 8200 attendant un fichier

3.3 Ecriture du programme PontTCPTaille

Ecrivez maintenant une classe PontTCPTaille. Si on exécute le main de cette classe, on obtient la taille du fichier fourni par le serveur 1.

Par exemple, si on exécute le main de cette classe, on obtient l'affichage suivant :

```
Début de la connexion au serveur 1
```

```
La taille du fichier envoyé par le serveur 1 est : 1000222000 octets
Fin du programme
```

Votre programme doit s'exécuter en 10 secondes maximum.

3.4 Ecriture du programme PontTCP

Ecrivez maintenant une classe PontTCP.

Le code de cette classe va permettre de réaliser un premier client qui va se connecter au serveur et un deuxième client qui va se connecter au serveur 2.

Le premier client va aller chercher le fichier du serveur 1 , et le deuxième client va prendre ce fichier pour l'envoyer au serveur 2.

Etant donné que le fichier est très gros (plus de 1Go), il ne sera pas possible d'aller chercher le fichier en entier avant de l'envoyer au serveur 2. Il est nécessaire d'aller chercher le fichier partie par partie et de l'envoyer partie par partie au serveur 2.

Par exemple, si on exécute le main de cette classe, on obtient l'affichage suivant :

```
Début du transfert du fichier depuis serveur 1 vers serveur 2
Fin du transfert.
Réponse du serveur 2 : OK BIEN RECU
```

3.5 Détermination de la taille du fichier avec netcat

Donnez une méthode pour trouver la taille de ce fichier avec netcat, en s'aidant éventuellement d'autres commandes systèmes comme **ls**.

Dans un commentaire tout en haut de la classe PontTCPTaille, vous indiquerez les commandes netcat et les autres commandes systèmes que vous avez utilisées pour déterminer cette taille.