INF4032 – RESEAUX IP

Programmation Réseau

ESIEA – 2021–2022 – Haidar/Ollivier/Radi/Teillac

Résumé

Notions abordées :
— Socket, flux
— JAVA
— TCP, UDP, HTTP

Documents fournis :
— Sujet : TP3_Prog_Res.pdf

Notation
— Si non précisé, chaque question vaut 1 point.
— Il y a une validation en salle de chaque programme.
— Un rapport doit être rendu, en utilisant le moodle.
— Si aucune consigne spécifique (donnée par l'enseignant ou le chargé de TP), vous disposez de 24h à partir de la fin du TP pour rendre le rapport. Les soumissions sont clôturées automatiquement sur le moodle.
— Vous devez soumettre dans l'espace de soumission qui correspond à votre groupe.
— Aucune soumission ne sera acceptée par mail, aucun retard ne sera toléré.

Matériel à disposition pour chaque groupe : :
— Groupes de 2.
— Un poste par groupe.
— Chaque groupe d'étudiants gère son propre sous-réseau.
— L'adressage des groupes est défini par le chargé de TP.

Eléments de compréhension :
— Dans cet énoncé, lorsque des commandes système sont illustrées, nous avons deux types de prompt :
  — % signifie que les commandes sont tapées en tant que simple utilisateur (user)
  — # signifie que les commandes sont tapées en tant qu'administrateur (root)

Eléments spécifiques à ce TP :
— Pour programmer vous pouvez utiliser Gvim, Eclipse, ou autre environnement d'édition.
— La documentation de Java peut être consultée à l'aide de votre navigateur Web à partir du fichier suivant : /usr/share/doc/openjdk-7-doc/api/index.html
— Si vous n'êtes pas administrateur de votre machine, les ports TCP #1 à #1023 ne sont pas utilisables. Votre serveur Web HTTP devra donc fonctionner en utilisant un port supérieur à #1024
— Pour se connecter à votre serveur à partir d'un client HTTP, on utilisera une URL du type http ://[adresse IP de la machine sur laquelle se trouve votre serveur] :[numéro de port sur lequel est installé le serveur]/

I Remarques générales

Ce TP est volontairement long. Il a été fait ainsi pour que chacun puisse comprendre les principes de la programmation Réseau. Nous attacherons une attention toute particulière à la qualité des codes que vous nous fournirez. Le rendu pour ce TP consiste à mettre les sources de vos fichiers, correctement nommées, sur le Moodle. Ces sources devront être stockées dans un fichier ZIP avec les noms des 2 membres du groupe.

La notation est simple : un point par fonctionnalité attendue. Nous attendons l'utilisation des fonctions de bas niveau présentées en cours pour tout ce qui concerne la communication réseau et le traitement des fichiers. Si vous utilisez d'autres fonctions, vous aurez 0 à l'exercice concerné. Pour la partie serveur HTTP, vous devez avec telnet, un client HTTP en mode texte et avec un navigateur classique (Firefox, Chrome ou autre).

Attention : certains éléments sont volontairement omis, mais ils sont présents dans les cours qui sont sur le réseau de Vésale (ou sur le Moodle).

2 Client-Serveur

2.1 Exercice 2 : Mon premier programme en réseau

Objectif : Vous allez devoir créer des scanners de port pour trouver les ports ouverts localement (i.e : sur votre ordinateur) ainsi que sur des ordinateurs distants.

Vous devez réaliser des scanners de ports en Java (utilisation de Eclipse autorisée). Vous programmerez 3 versions de ces scanners : une version TCP pour scanner les ports localement, une seconde version TCP pour scanner les ports à distance et une version UDP pour scanner les ports localement, ces programmes étant nommés respectivement :

— ScannerTCPv1.java
— ScannerTCPv2.java
— ScannerUDPv1.java

2.2 Fonctionnalités attendues

1. Le mécanisme des exceptions doit être pris en charge.
2. Chaque scanner doit afficher uniquement la liste des ports ouverts.
3. Pour le scanner à distance, le programme prend en ligne de commande l'adresse IP de l'ordinateur cible.
4. Tous les programmes doivent prendre en ligne de commande les arguments permettant de tester une plage de ports donnée.

2.3 Algorithmes par programme

1. ScannerTCPv1
   — Pour **chaque port TCP** à tester, ce programme va tenter **d'installer un serveur TCP** et ainsi vérifier si le port est libre. En cas d'erreur, il affichera un message à la console indiquant qu'un serveur TCP est à l'écoute sur le port testé. *Cette version de scanner est un programme qui s'exécute sur l'ordinateur local.*
2. ScannerTCPv2
   — Pour **chaque port TCP** à tester, ce programme va tenter **d'ouvrir un socket vers le port distant** et ainsi vérifier si un serveur y est à l'écoute. En cas de succès, il affichera un message à la console indiquant qu'un serveur est à l'écoute sur le port testé. *Cette version de scanner est un programme qui s'exécute sur un ordinateur local et qui scanne un autre ordinateur.*
3. ScannerUDPv1
   — Pour **chaque port UDP** à tester, ce programme va tenter **d'installer un serveur UDP** et ainsi vérifier si le port est libre. En cas d'erreur, il affichera un message à la console indiquant qu'un serveur est à l'écoute sur le port testé. *Cette version de scanner est un programme qui s'exécute sur l'ordinateur serveur.*

Avant de passer à la suite de ce TP, vous devez faire valider les trois programmes par votre chargé de TP.

3 Serveur HTTP qui renvoie du code HTML

Exercice 3 :

Objectif : Vous devez créer un serveur Web en Java qui respecte le protocole HTTP. Ce serveur Web doit produire une réponse HTTP qui contient un code HTML.

Vous devez créer un serveur HTTP en Java. Ce serveur HTTP doit respecter le protocole HTTP tel que décrit précédemment. Ce serveur HTTP doit produire une réponse dont le corps est composé de code HTML statique. Le fichier Java sera nommé :— HTTPdServerV1.java

3.1 Fonctionnalités attendues

1. Les exceptions Java doivent être prises en charge.
2. Le programme prend en entrée le numéro du port TCP d'écoute.
3. Le programme doit accepter les connexions de clients HTTP.
4. Le programme produit une réponse HTTP qui contient du code HTML et déconnecte ensuite le client Web.
5. Le client Web doit pouvoir afficher le code HTML contenu dans la réponse HTTP.
6. Le programme ne s'arrête pas après la déconnexion d'un client, il se remet en attente du client suivant.

Un code HTML commence par une déclaration de type de document qui indique la norme HTML employée. Exemple :

```
<!DOCTYPE html>
```

Un code HTML contient ensuite impérativement une "balise" qui délimite le début et la fin du document. Exemple :

```
<html></html>
```

La "balise" qui délimite la début et la fin du document contient une "balise" qui délimite la zone d'entête du document (head) et une "balise" qui délimite le corps du document (body).

```
<html><head></head><body></body></html>
```

Les balises d'entête ou de corps de document peuvent à leur tour contenir des balises pour décrire des contenus comme par exemple un paragraphe de texte. Exemple :

```
<html><head></head><body><p>Paragraphe de texte</p></body></
    html
```

4 Serveur HTTP qui renvoie du code HTML contenu dans un fichier

Exercice 4 :

Objectif : Vous devez créer un serveur Web en Java qui respecte le protocole HTTP. Ce serveur Web doit produire une réponse HTTP qui contient le contenu d'un fichier HTML.

Vous devez créer un serveur HTTP en Java. Ce serveur HTTP doit respecter le protocole HTTP tel que décrit précédemment (voir le TP précédent où vous avez observé les entêtes HTTP). Ce serveur Web HTTP doit produire une réponse dont le corps est composé du contenu d'un fichier HTML statique : index.html. Le fichier Java sera nommé :

— HTTPdServerV2.java

4.1 Fonctionnalités attendues

1. Les exceptions Java doivent être prises en charge.
2. Le programme prend en entrée le numéro du port TCP d'écoute.
3. Le programme doit accepter les connexions de clients HTTP.
4. Le programme produit une réponse HTTP qui contient le contenu d'un fichier HTML et déconnecte le client HTTP.
5. Le client Web doit pouvoir afficher le code HTML contenu dans la réponse HTTP.
6. Le programme ne s'arrête pas après la déconnexion d'un client. Il se remet en attente du client suivant.

Comment lire le contenu d'un fichier HTML ?

— On utilisera un objet du type *java.io.FileReader* pour accéder au contenu d'un fichier HTML sous la forme d'un flux de caractères.
— On utilisera un objet du type *java.io.BufferedReader* pour lire le flux de caractères.
— On utilisera un objet du type *java.io.PrintStream* pour écrire des caractères ou des chaînes de caractères dans l'OutputStream du Socket.

5 Serveur HTTP qui affiche la requête à la console

Exercice 5 :

Objectif : Vous devez créer un serveur Web en Java qui respecte le protocole HTTP. Ce serveur Web doit afficher le contenu des requêtes HTTP reçues dans la console système.

Vous devez créer un serveur HTTP en Java. Ce serveur HTTP doit respecter le protocole HTTP tel que décrit précédemment. Ce serveur HTTP doit afficher dans la console les requêtes HTTP reçues en provenance de clients HTTP. Le fichier Java sera nommé :

— HTTPdServerV3.java

5.1 Fonctionnalités attendues

1. Les exceptions Java doivent être prises en charge.
2. Le programme prend en entrée le numéro du port TCP d'écoute.
3. Le programme doit accepter les connexions de clients HTTP.
4. Le programme produit une réponse HTTP dont le corps est vide.
5. Le programme affiche le contenu de la requête HTTP dans la console et déconnecte le client HTTP.
6. Le programme ne s'arrête pas après la déconnexion d'un client. Il se remet en attente du client suivant.

Comment lire le contenu de la requête HTTP :

— On utilisera un objet du type *java.net.BufferedReader* pour lire le contenu de l' InputStream du Socket sous la forme d'un flux de caractères

6 Serveur HTTP qui renvoie le fichier HTML demandé

Exercice 6 :

Objectif : Vous devez créer un serveur Web en Java qui respecte le protocole HTTP. Ce serveur Web doit produire une réponse HTTP qui dépend du contenu de la requête HTTP faite

Vous devez créer un serveur Web en Java. Ce serveur Web doit respecter le protocole HTTP tel que décrit précédemment. Ce serveur Web HTTP doit produire une réponse dont le corps est composé soit du contenu du fichier index.html soit du contenu du fichier page.html, en fonction de l'URL Le fichier Java sera nommé : HTTPdServerV4.java

Fonctionnalités attendues

1. Les exceptions Java doivent être prises en charge.
2. Le programme prend en entrée le numéro du port TCP d'écoute.
3. Le programme doit accepter les connexions de clients HTTP.
4. Le programme analyse le contenu de la requête HTTP pour obtenir le nom fichier HTML à renvoyer.
5. Si la requête HTTP ne contient pas de nom de fichier, le programme choisi par défaut le fichier index.html.
6. Le programme produit une réponse HTTP qui contient le contenu d'un fichier HTML et déconnecte le client HTTP.
7. Le client Web doit pouvoir afficher le code HTML contenu dans la réponse HTTP.
8. Le programme ne s'arrête pas après la déconnexion d'un client. Il se remet en attente du client suivant.

Comment lire et analyser le contenu de la requête HTTP :

— On utilisera un objet du type *java.net.BufferedReader* pour lire le contenu de l' InputStream du Socket sous la forme d'un flux de caractères.

— On utilisera un objet du type *java.util.StringTokenizer* pour analyser le flux de caractères.

7 Serveur HTTP qui renvoie un fichier HTML différent selon le client

Exercice 7 :

Objectif : Vous devez créer un serveur en Java qui respecte le protocole HTTP. Ce serveur doit produire une réponse HTTP qui dépend du contenu de la requête HTTP.

Vous devez créer un serveur HTTP en Java. Ce serveur doit respecter le protocole HTTP tel que décrit précédemment. Ce serveur HTTP doit produire une réponse dont le corps est composé soit du contenu du fichier iceweasel.html soit du contenu du fichier chromium.html. Le fichier Java sera nommé :

— HTTPdServerV5.java

7.1 Fonctionnalités attendues

1. Les exceptions Java doivent être prises en charge.
2. Le programme prend en entrée le numéro du port TCP d'écoute.
3. Le programme doit accepter les connexions de clients Web.
4. Le programme analyse le contenu de la requête HTTP pour déterminer le nom fichier HTML à renvoyer.
5. Si la requête HTTP contient un nom de client Web non pris en charge, le programme choisi par défaut un des deux fichiers.

6. Le programme produit une réponse HTTP qui contient le contenu d'un fichier HTML et déconnecte le client.
7. Le client doit pouvoir afficher le code HTML contenu dans la réponse HTTP.
8. Le programme ne s'arrête pas après la déconnexion d'un client. Il se remet en attente du client suivant

Comment lire et analyser le contenu de la requête HTTP :

— On utilisera un objet du type *java.net.BufferedReader* pour lire le contenu de l' InputStream du Socket sous la forme d'un flux de caractères.
— On utilisera un objet du type *java.util.StringTokenizer* pour analyser le flux de caractères.