

Programmation sockets

Séminaire 2005-2006

Notions préalables (1/2)

- Réseau
 - deux machines connectées
 - plusieurs machines connectées localement, et puis globalement
- Internet
 - réseau de réseaux
- Adresse IP
 - adresse identifiant un ordinateur dans le réseau Internet
 - ex.: 129.192.70.36
- Protocole
 - ensemble de règles défini pour un type de communication
- Datagramme
 - bloc de données indépendant transitant sur un réseau, et contenant toutes les informations nécessaires à son routage

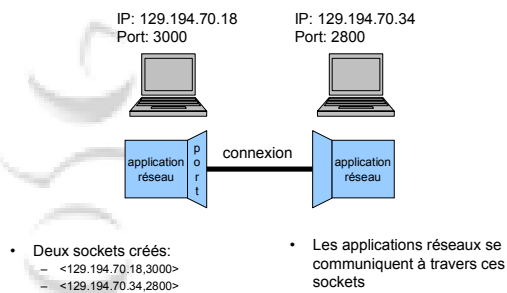
Notions préalables (2/2)

- TCP (Transmission Control Protocol)
 - protocole de transfert de données sur un réseau
 - fiable: garantie de livraison
 - travaille en mode connecté
- UDP (User Datagram Protocol)
 - protocole de transfert de données sur un réseau
 - non fiable: aucune garantie de livraison
 - utilise des datagrammes
 - travaille en mode non connecté
- Processus
 - programme qui tourne sur un ordinateur à un moment donné
 - un processus contient
 - le code du programme
 - les données du programme
 - les ressources utilisées par le programme (fichier, connexions réseau, etc.)
 - ex.: serveur http, MSN, etc.

Sockets

- API permettant de programmer des applications réseau
- Développé à Berkley, maintenant un standard
- Socket
 - point de jointure entre un processus et le réseau
 - défini par la paire <adresse IP, port>
 - un socket donne une identification unique à une application
- Port
 - abstraction offerte par le système d'exploitation
 - adresse de 16 bits utilisée par les applications
 - associe une connexion à un processus
 - ports 0-1023 réservés pour le système (FTP port 21, HTTP port 80, etc.)
 - ports 1024-65535 disponibles à l'utilisateur

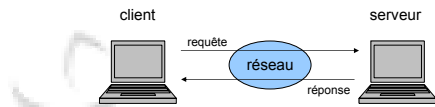
Exemple de sockets



Quelques ports déjà utilisés

Port	Service/Application
21	FTP (File Transfer Protocol)
22	SSH (Secure Shell)
23	Telnet
25	SMTP (Send Mail Transfer Protocol)
53	DNS (Dynamic Name Server)
68	DHCP (Dynamic Host Control Protocol)
80	HTTP (Hyper Text Transfer Protocol)
110	POP3 (Post Office Protocol v3)
115	SFTP (Secure FTP)
3724	World of Warcraft

Modèle Client – Serveur (1/2)

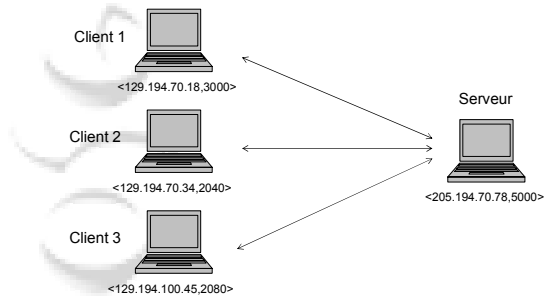


- Modèle largement répandu parmi les applications réseau
- Composé d'un serveur et d'un ou plusieurs client(s)
- Scénario
 - client envoie une requête au serveur
 - serveur exécute une tâche ou recherche des données
 - serveur envoie une réponse au client
- Exemples
 - sites web
 - lire/écrire de fichiers (samba)
 - heure du système
 - etc.

Infrastructure de Communication - Séminaire

7

Modèle Client – Serveur (2/2)



Infrastructure de Communication - Séminaire

8

Types de serveurs



- Serveurs itératifs
 - quand les requêtes peuvent être traitées dans un temps court et connu d'avance
 - un seul processus s'encharge d'écouter et de traiter les requêtes
 - ex.: heure du système
- Serveurs concurrents
 - quand le temps de traitement d'une requête dépend de la requête elle-même
 - un processus est dédié à écouter les requêtes des clients
 - un processus différent est créé pour chaque nouvelle requête reçue
 - ex.: lire un fichier très grand

Infrastructure de Communication - Séminaire

9

Modes de communication

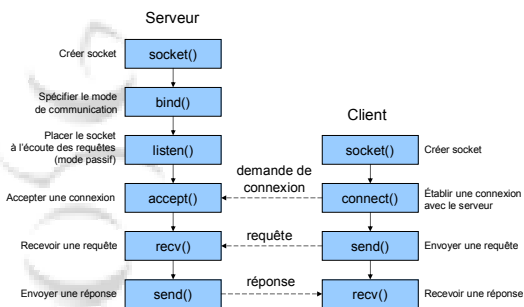


- Mode connecté
 - un canal de communication est créé avant d'envoyer les données
 - l'arrivée des données au serveur est assurée
 - protocole de transport utilisé: TCP
 - ex.: SSH, Telnet et FTP
- Mode non connecté
 - il n'y a pas besoin de créer un canal de communication
 - les données sont envoyées utilisant des datagrammes
 - l'arrivée des datagrammes n'est pas assurée
 - protocole de transport utilisé: UDP
 - ex.: heure système

Infrastructure de Communication - Séminaire

10

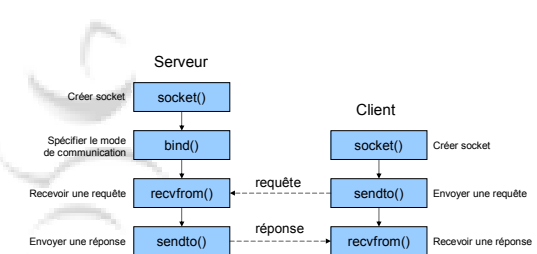
Communication en mode connecté



Infrastructure de Communication - Séminaire

11

Communication en mode non connecté



Infrastructure de Communication - Séminaire

12

Implémentation d'un serveur



1. Créer un socket serveur:

```
ServerSocket server;  
DataOutputStream os;  
DataInputStream is;  
server = new ServerSocket( PORT );
```
2. Attendre à qu'un client demande une connexion:

```
Socket client = server.accept();
```
3. Créer des I/O streams pour se communiquer avec le client:

```
is = new DataInputStream( client.getInputStream() );  
os = new DataOutputStream( client.getOutputStream() );
```
4. Envoi et réception de données:
Recevoir des données du client: `int x = is.readInt();`
Envoyer des données au client : `os.writeInt(ENTIER);`
5. Fermer le socket:

```
client.close();
```

Implémentation d'un client



1. Créer un socket client

```
client = new Socket( SERVER, PORT);
```
2. Créer des I/O streams pour se communiquer avec le client:

```
is = new DataInputStream(client.getInputStream() );  
os = new DataOutputStream( client.getOutputStream() );
```
3. Envoi et réception de données:
Recevoir des données du client: `int x = is.readInt();`
Envoyer des données au client : `os.writeInt(ENTIER);`
4. Fermer le socket:

```
client.close();
```

Exemple: un serveur simple



```
// SimpleServer.java: a simple server program  
import java.net.*;  
import java.io.*;  
public class SimpleServer {  
    public static void main(String args[]) throws IOException {  
        // Register service on port 1234  
        ServerSocket s = new ServerSocket(1234);  
        Socket s1=s.accept(); // Wait and accept a connection  
        // Get a communication stream associated with the socket  
        OutputStream slout = s1.getOutputStream();  
        DataOutputStream dos = new DataOutputStream (slout);  
        // Send a string!  
        dos.writeUTF("Hi there");  
        // Close the connection, but not the server socket  
        dos.close();  
        slout.close();  
        s1.close();  
    }  
}
```

Exemple: un client simple



```
// SimpleClient.java: a simple client program  
import java.net.*;  
import java.io.*;  
public class SimpleClient {  
    public static void main(String args[]) throws IOException {  
        // Open your connection to a server, at port 1234  
        Socket s1 = new Socket("mundroo.cs.mu.oz.au",1234);  
        // Get an input file handle from the socket and read the input  
        InputStream s1In = s1.getInputStream();  
        DataInputStream dis = new DataInputStream(s1In);  
        String st = new String (dis.readUTF());  
        System.out.println(st);  
        // When done, just close the connection and exit  
        dis.close();  
        s1In.close();  
        s1.close();  
    }  
}
```

Socket Exceptions



```
try {  
    Socket client = new Socket(host, port);  
    .....  
    ....  
}  
  
catch(UnknownHostException uhe) {  
    System.out.println("Unknown host: " + host);  
    uhe.printStackTrace();  
}  
  
catch(IOException ioe) {  
    System.out.println("IOException: " + ioe);  
    ioe.printStackTrace();  
}
```

Exemple: un serveur qui tourne tout le temps

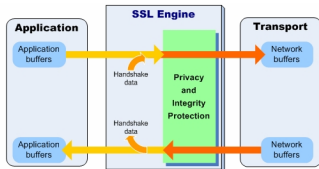


```
// SimpleServerLoop.java: a simple server program that runs forever in a single thread  
import java.net.*;  
import java.io.*;  
public class SimpleServerLoop {  
    public static void main(String args[]) throws IOException {  
        // Register service on port 1234  
        ServerSocket s = new ServerSocket(1234);  
        while(true)  
        {  
            Socket s1=s.accept(); // Wait and accept a connection  
            // Get a communication stream associated with the socket  
            OutputStream s1out = s1.getOutputStream();  
            DataOutputStream dos = new DataOutputStream (s1out);  
            // Send a string!  
            dos.writeUTF("Hi there");  
            // Close the connection, but not the server socket  
            dos.close();  
            s1out.close();  
            s1.close();  
        }  
    }  
}
```

Secure Socket Layer (SSL)

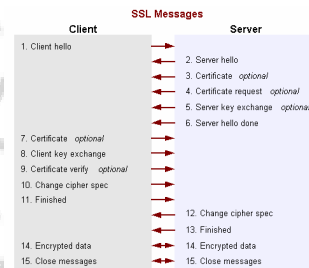


javax.net.ssl.*
java.security.*



- Authentification
- Confidentialité
- Intégrité

SSL messages



Implémentation d'un serveur sécurisé



- Créer un socket serveur sécurisé:

```
DataOutputStream os;  
DataInputStream is;  
SSLServerSocketFactory sf = (SSLServerSocketFactory)  
    SSLServerSocketFactory.getDefault();  
SSLServerSocket server =  
    (SSLServerSocket) sf.createServerSocket(PORT);
```
- Attendre à qu'un client demande une connexion:

```
SSLSocket client = (SSLSocket) server.accept();
```
- Créer des I/O streams pour se communiquer avec le client:

```
is = new DataInputStream( client.getInputStream() );  
os = new DataOutputStream( client.getOutputStream() );
```
- Envoi et réception de données:

```
Recevoir des données du client: int x = is.readInt();  
Envoyer des données au client : os.writeInt(ENTIER);
```
- Fermer le socket: `client.close();`

Implémentation d'un client sécurisé



- Créer un socket client

```
DataOutputStream os;  
DataInputStream is;  
SSLSocketFactory sf =  
    SSLSocketFactory.getDefault();  
SSLSocket client =  
    (SSLSocket) sf.createSocket(SERVER, PORT);
```
- Créer des I/O streams pour se communiquer avec le client:

```
is = new DataInputStream( client.getInputStream() );  
os = new DataOutputStream( client.getOutputStream() );
```
- Envoi et réception de données:

```
Recevoir des données du client: int x = is.readInt();  
Envoyer des données au client : os.writeInt(ENTIER);
```
- Fermer le socket:

```
client.close();
```

Authentification et confidentialité



- L'authentification est réalisée au moyen d'une paire de clés (une clé publique/une clé privée) et d'un certificat.
- On peut créer ces clés et un certificat auto-signé au moyen de keytool, un utilitaire disponible avec JDK 1.4 et en saisissant un certain nombre d'informations, dont le mot de passe "keystore". Ici est utilisé l'algorithme RSA:
 - keytool -genkey -v -keyalg RSA -keystore keystore
- On invoquera ensuite le client Java permettant la communication avec le serveur de la manière suivante:
 - java -Djavax.net.ssl.trustStore=keystore -Djavax.net.ssl.keyStore=keystore -Djavax.net.ssl.keyStorePassword="motdepasse" nomClient

Références



- Tutoriaux Java
<http://java.sun.com/learning/tutorial/index.html>
- Spécifications du API Java
<http://java.sun.com/j2se/1.4.2/docs/api/>
- Java™ Secure Socket Extension (JSSE)
<http://java.sun.com/j2se/1.5.0/docs/guide/security/jsse/JSSERefGuide.html>
- KeyStore Explorer
<http://www.lazgosoftware.com/kse/>