Cours n°4

Programmation réseau et sockets

Sommaire

Bibliographie

Jan Graba, An Introduction to Network Programming with Java, Springer-Verlag London, 2006

E.-R. Harold, Java Network Programming, O'Reilly, 2004

G. Roussel, E. Doris, N. Bedon, R. Forax, Java et Internet, Vuibert, 2002

34 classes et 12 types d'exception

Ensemble de classes permettant une interaction avec le réseau pour recevoir et envoyer des données.
Développement de programmes établissant des communications avec d'autres applications distantes

Communication réseau

Gestion des protocoles des couches réseaux et transport

Encapsulation de la notion de Socket

Accélération par l'utilisation de caches

Utilisation simplifiée par rapport aux bibliothèques systèmes (Unix et Windows)

Programmation internet

Gestion des protocoles de la couche application (ssh, sftp, http, …)

Sécurisation par l'utilisation de mots de passe

Classe InetAdress

Représentation d'une adresse internet avec deux attributs

hostName représentant le nom sous forme d'une chaîne de caractères

Address représentant le numéro IP sous forme d'un tableau d'octets

Méthodes statiques pour obtenir une InetAddress

Pas de constructeur,

InetAddress InetAddress.**getLocalHost**()
obtention de l'InetAdress correspondant à la machine locale

InetAddress InetAddress.**getByName**(String host)
obtention de l'InetAdress d'une machine d'un nom donné

InetAddress InetAddress.**getByAdress**(byte[] addr)
obtention de l'InetAdress d'une machine d'un numéro IP donné

InetAddress InetAddress.**getByAdress**(String host, byte[] addr)
obtention de l'InetAdress d'une machine d'un nom et d'un numéro IP donné

Levée de l'exception UnknownHostException en cas de machine inconnue

Méthodes (1/2)

Obtenir le nom d'une machine

String **getHostName**() String **getHostAddress**() Nom de la machine

Obtenir le numéro IP d'une machine

byte[ ] **getAddress**() Adresse IP sous forme de 4 octets

Caractéristiques

boolean **isAnyLocalAddress**()	Adresse non spécifiée (0.0.0.0)
boolean **isLoopbackAddress**()	Adresse locale de test (127.0.0.0) localhost
boolean **isLinkLocalAddress**()	Adresse locale (169.254.0.0)
boolean **isSiteLocalAddress**()	Adresse privée (127.0.0.0)
boolean **isMulticastAddress**()	Adresse multicast (224.0.0.0)
boolean **isReachableAddress**(int timeout)	Test de connectivité

Méthodes (2/2)

getInetAdress.java

```
try{
InetAddress  monAdresse = InetAddress.getByName("smtp.laposte.net");
System.out.println(monAdresse.getHostName());
System.out.println( monAdresse.getHostAddress());
System.out.print( monAdresse.isAnyLocalAddress() + " ");
System.out.print( monAdresse.isLoopbackAddress() + " ");
System.out.print( monAdresse.isLinkLocalAddress() + " ");
System.out.println( monAdresse.isReachable(100));

} catch (UnknownHostException exp){
      System.out.println("machine inconnue");
} catch (IOException e) {
      System.out.println("machine non atteignable");
      }
```

```
smtp.laposte.net
81.255.54.9
false false false false
```


Principes

Lien de communication inter-processus

Développé initialement sous UNIX (1983)

Extension de la notion de tube nommé (pipe) pour des machines distantes

Utilisation des mécanismes classiques d 'E/S (java.io)

Restriction au domaine AF_INET pour les sockets java (couche réseau)

Une adresse (nom, numéro IP) et un numéro de port

Diagramme de communication

Un réseau et deux sockets

Type de services IP (Internet Protocol)

Socket STREAM

Protocole TCP (Transmission Control Protocol)

Mode connecté avec contrôle de flux

Connexion établie entre les machines distantes

Fiabilité de la transmission des données

Comparaison avec une communication téléphonique

Socket DATAGRAM

Protocole UDP (User Datagram Protocol)

Mode non connecté avec transmission par paquet sans contrôle de flux

Pas de connexion entre le client et le serveur

Pas de sécurité de transmission des données

Comparaison à une communication par lettre

Port d'entrée-sortie

Canal de communication accessibles à travers un réseau

Identification par un entier (16 bits),

Correspondance avec un service spécifique

Services standard internet
- terminal sécurisé (SSH port 22),
- courrier sortant (SMTP port 25),
- serveur web (HTTP port 80),
- courrier entrant (POP3 port 110),
- transfert de fichiers sécurisé (SFTP port 115),
- courrier entrant (IMAP3 port 220),
- authentification (LDAP, port 389),
- serveur web sécurisé (HTTPS port 443),
- accès à la base de données Mysql (port 3306)

Ports utilisateurs (entre 30 000 et 60 000)
- consultation de la table des services
- enregistrement à l'IANA

Classe InetSocketAdress

Représentation d'une adresse de socket avec deux attributs

Adresse IP sous la forme d'une InetAdress

Port d'entrée/sortie sous la forme d'un entier

Constructeurs

InetSocketAddress(String hostName, int port)

InetSocketAddress(InetAddress addr, int port)

InetSocketAddress(int port) adresse IP non spécifiée (utilisée en écoute)

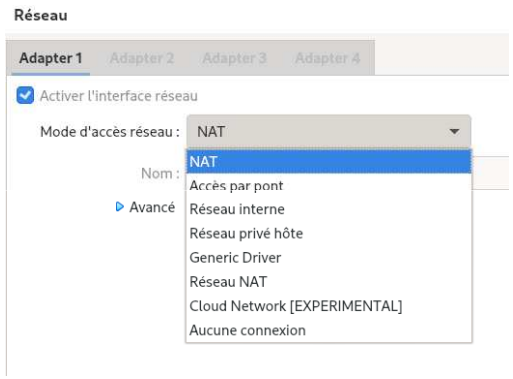
Accesseurs

InetAddress **getAddress**() accès à l'InetAddress

String **getHostName**() accès au nom

int **getPort**() accès au numéro de port

Configuration Réseau sous VirtualBox (1/5)



Configuration Réseau sous VirtualBox (2/5)

Routeur logiciel entre la machine virtuelle (MV) et l'accès réseau de de la machine hôte

NAT (Network Address Translation)

Attribution à chaque MV d'un adresse IP en 10.0.x.y.
Accès de la MV à l'extérieur (dont l'hôte),
Pas d'accès par défaut de l'extérieur (dont l'hôte) à la MV sauf redirection de port

Redirection de port

Un paquet (TCP ou UDP) adressé à l'adresse IP de l'hôte sur le port hôte sera envoyé à l'adresse IP de la MV sur le port invité.

Configuration Réseau sous VirtualBox (3/5)

Routeur logiciel entre la machine virtuelle (MV) et l'accès réseau de de la machine hôte

Accès par pont (Bridged Network)

Création d'une nouvelle interface réseau sur la machine hôte connectée à la MV.
Utilisation du pilote réseau de la machine hôte pour l'obtention d'une adresse.
Filtrage des données de la carte réseau physique.
Demande au serveur DHCP d'une adresse IP (publique ou privée)
refus possible

En cas d'acceptation
accès dans les deux sens entre la MV et l'extérieur (dont l'hôte)

Réseau interne

Communication possible en plusieurs MV sans passer par la machine hôte

Configuration Réseau sous VirtualBox (4/5)

Réseau privé hôte

Intermédiaire entre un accès par pont et un réseau interne
Communication possible en plusieurs MV et la machine hôte
Création d'une nouvelle interface logicielle sur l'hôte

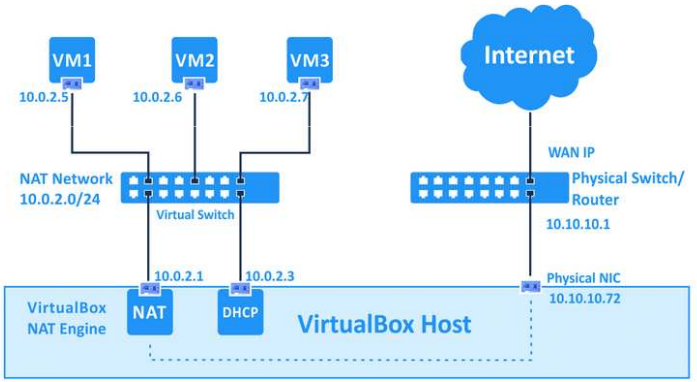
Generic Driver

Création d'un extranet
Interconnexion par un tunnel UDP de MV s'exécutant sur différents hôtes.

Configuration Réseau sous VirtualBox (5/5)

Réseau NAT

Création d'un réseau interne complet (DHCP virtuel, switch virtuel)



[www.nakivo.com]

Principes

Protocole fiable de la couche réseau (RFC 793)

Flot découpé en paquets (~536 octets)

Mécanismes de contrôle de flot (contrôle de congestion)

Mécanisme d'acquittement des paquets
- alarme activée à l'émission d'un paquet,
- désactivation en cas de réception de l'acquittement,
- réémission après un temps donné

Ajout à chaque paquet d'un numéro de séquence
- préservation de l'ordre,
- élimination des doublons

Demande de connexion (1/2)

Choix de l'adresse des sockets

Adresse de la socket locale : saddr1
- addr1 : numéro IP de la machine locale
- port1 : numéro de port choisie de la machine locale

Adresse de la socket distante : saddr2
- addr2 : numéro IP de la machine distante
- port2 : numéro de port choisie de la machine distante

Trois étapes

1. Construction d'une socket de communication — Socket s = **new** Socket()
2. Attachement du port de connexion — s.**bind** (saddr1)
3. Demande de connexion à une socket distante — s.**connect** (saddr2)

Constructeur alternatif

Socket s = **new** Socket(addr2, port2, addr1, port 1)

Levée de l'exception IOException en cas de refus de connexion

Demande de connexion (2/2)

DemandeConnexion.java

```
// création de la socket
Socket s = new Socket(); InetAddress addr1 = null;
int port1 = 32506, port2 = 32504;
try { addr1 = InetAddress.getLocalHost(); }
catch (UnknownHostException exp){ }
try {
      // attachement
      InetSocketAddress saddr1 = new InetSocketAddress(addr1, port1);
      s.bind(saddr1);
      // adresse de la machine distante
      InetAddress addr2 = InetAddress.getByName("192.168.1.102");
      InetSocketAddress saddr2 = new InetSocketAddress(addr2, port2);
      // demande de connexion
      s.connect(saddr2);
      System.out.println("Connexion établie entre " +
                  s.getLocalSocketAddress() + " et " +
                  s.getRemoteSocketAddress());
```

```
Connexion etablie entre /192.168.1.75:32506 et /192.168.1.75:32504
fin de la communication TCP
```


Acceptation de la connexion (1/2)

Choix des caractéristiques du service

Adresse de la socket de service : saddr
- addr : numéro IP de la machine locale
- port : numéro de port choisie pour le service

max : nombre maximum de connexions

Trois étapes

1. Construction d'un écouteur de connexion — ServerSocket ss = **new** ServerSocket()
2. Attachement du port de service — ss.**bind** (addr, max)
3. Acceptation de la connexion et et récupération de la socket — Socket s = ss.**accept** ()

AcceptionConnexion.java

Acceptation de la connexion (2/2)

```
int port = 32504;
InetSocketAddress saddr = null;
try {
      InetAddress addr = InetAddress.getLocalHost();
      saddr = new InetSocketAddress(addr, port);
} catch (UnknownHostException exp){ }
try {
      // création d'un écouteur de connexion
      ServerSocket ss = new ServerSocket();
      // attachement
      ss.bind(saddr);
      // acceptation de la connexion
      Socket s = ss.accept();
      System.out.println("Connexion établie entre " +
                  s.getLocalSocketAddress() + " et " +
                  s.getRemoteSocketAddress());
} catch (IOException exp){ }
```

```
Connexion etablie entre /10.0.2.15:32504 et /10.0.2.2:32506
début de la communication TCP
```


Gestion des flots

Canal de communication à double sens (full duplex)

Fourniture d'un flot de lecture des données arrivant sur la connexion

InputStream **getInputStream**()

Fourniture d'un flot d'écriture des données sortant de la connexion

OutputStream **getOutputStream**()

Fermeture du flot de lecture	**shutdownInput**()	boolean **isOutputShutdown**()
Fermeture du flot d'écriture	**shutdownOutput**()	boolean **isInputShutdown**()
Fermeture du canal	**Close**()	boolean **isClosed**()

Lecture/Ecriture

Méthodes read/write

Composition de classes new **InputStreamReader**(s.**getInputStream**()));

DemandeConnexion.java

Flots (1/2)

```
// Création des flots
BufferedReader br = null; PrintStream ps = null;
try {
br = new BufferedReader(new InputStreamReader(s.getInputStream()));
ps = new PrintStream(s.getOutputStream());
} catch (IOException exp){
      System.out.println("erreur de création des flots");
}

// Communication
try {
ps.println("début de la communication TCP");
String ligne = br.readLine();
System.out.println(ligne);
      }
catch (IOException exp){
      System.out.println("erreur d'entée-sortie");
}
```

```
fin de la communication TCP
```


AcceptionConnexion.java

Flots (2/2)

```
// Création des flots
BufferedReader br = null; PrintStream ps = null;
try {
br = new BufferedReader(new InputStreamReader(s.getInputStream()));
ps = new PrintStream(s.getOutputStream());
} catch (IOException exp){
      System.out.println("erreur de création des flots");
}

// Communication
try {
String ligne = br.readLine();
System.out.println(ligne);
ps.println("fin de la communication TCP");
br.readLine();
} catch (IOException exp){
      System.out.println("erreur d'entée-sortie");
}
```

```
début de la communication TCP
```

Introduction

Protocole non fiable de la couche réseau (RFC 768)

Pas de flot,
Acheminement de bloc de données (datagramme)
Taille max des données transportées: ~64Ko
Checksum optionnel en IP v4, obligatoire en IP v6

Classe DatagramSocket

Socket permettant un service UDP
Création d'une socket sur chaque machine

Classe DatagramPacket

Empaquetage des données envoyées ou reçues
Adresse de la socket de destination

Principes

Adresse de la socket distante : saddr

addr : numéro IP de la machine distante

port : numéro de port choisie de la machine distante

Quatre étapes

1. Construction d'une socket de communication
   DatagramSocket ds = **new** DatagramSocket()
2. Empaquetage des données dans un tableau de bytes : b
3. Construction du datagramme à transmettre
   DatagramPacket dp = **new** DatagramPacket(b, b.length, saddr)
4. Envoi du datagramme ds.**send** (dp)

Levée des exceptions SocketException et IOException

void **setBroadcast**(boolean on)
autorisation d'envoi à toutes les machines du sous-réseau
void **setSendBufferSize**(int size)
modification de la taille du tampon

Exemple

EnvoiDatagramme.java

```
// création de la socket
DatagramSocket ds = null; int port = 32505;
try { ds = new DatagramSocket();
}
catch (SocketException exp){ }

// adresse de la machine distante
InetSocketAddress saddr = null;
try { InetAddress addr = InetAddress.getByName("192.168.1.75");
      saddr = new InetSocketAddress(addr, port);
} catch (UnknownHostException exp){ }

// construction et envoi du datagramme
try { String s = "début de la communication UDP";
      byte[] b = s.getBytes();
      DatagramPacket dp = new DatagramPacket(b, b.length, saddr );
      ds.send(dp);
} catch (SocketException exp){ } catch (IOException exp){ }
```


Principes

Quatre étapes

1. Construction d'une socket de communication écoutant le port p
   DatagramSocket ds = **new** DatagramSocket(p)
2. Construction d'un datagramme contenant un tableau de bytes vide : b
   DatagramPacket dp = **new** DatagramPacket(b, b.length)
3. Reception du datagramme ds.**receive** (dp)
4. Extraction des données du paquet

Levée des exceptions SocketException et IOException

byte[] **getData**()
tableau de byte contenant les données
int **getLength**()
taille en octet des données reçues
SocketAddress **getSocketAddress**()
adresse de la socket émettrice

Exemple

ReceptionDatagramme.java

```
DatagramSocket ds = null; int port = 32505;
try { ds = new DatagramSocket(port);
} catch (SocketException exp){ }

// réception du datagramme
try {
byte b[] = new byte [1024];
DatagramPacket dp = new DatagramPacket(b, b.length);

ds.receive(dp);
InetSocketAddress saddr = (InetSocketAddress)dp.getSocketAddress();
System.out.println("datagramme en provenance de " + saddr.toString());

System.out.println(new String(b, 0, dp.getLength()));
} catch (SocketException exp){} catch (IOException exp){ }
```

```
datagramme en provenance de /10.0.2.2:48336
debut de la communication UDP
```