TCP/IP

Le modèle de TCP/IP comporte 4 couches:

Application,

Transport,

Network,

Link.

Link :

C'est le protocole bas niveau utilisé pour communiquer sur le même réseau physique. On utilise le mécanisme d'encapsulation.

Network – Réseau :

La couche réseau permet de transmettre des informations à toutes les machines indépendemment du réseau physique. C'est le mécanisme de l'Internet Protocol (IP).

Transport,

La couche Transport permet le fonctionnement des applications avec deux protocoles Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

Application,

La couche application définit des protocoles pour des applications de l'utilisateur comme HTTP, FTP, ou Telnet.

IP

Internet Protocol – Les données d'Internet circulent sous la forme de paquet IP. Unité de base du protocole. Ce protocole est sans connexion et non fiable. Il n'y a pas d'échange d'informations de contrôle.

IPv4

IPv4 fournit un schéma d'adressage complet appelée adresse IP.

Une addresse IPv4 est sur 32 bits.

Les addresses sont notés sous la forme de 4 nombres de 0 à 255 séparé par des points. Ex :

193.48.38.36

port

Pour faciliter l'utilisation d'internet entre plusieurs applications le mécanisme de port permet d'associer plusieurs communication à une adresse IP.

Sous-Unix les ports inférieurs à 1024 sont reservé à l'administrateur de la machine.

De nombreux numéros de port sont associé à des applications.

rfc 1700 : http://www.faqs.org/rfcs/rfc1700.html

port

21	FTP
23	TELNET
25	SMTP
80	HTTP
111	SunRPC (RPC portmapper)
119	NNTP (News)
194	IRC
389	LDAP

....

TCP

Transmission Control Protocol – Ce protocole est utilisé par de nombreuses applications. Ce protocole est orienté connection, fournit un « stream » continu, est « fiable ».

rfc 793 :http://www.faqs.org/rfcs/rfc793.html

TCP

« Fiable » :

Car les segments seront renvoyé si ils sont perdu ou corrompu.

Orienté connexion : il y a une poignée de main - « handshake » pour établir et stopper la connexion

Continuous-stream – flux continu : lorsque la connexion a été réalisée on a l'apparence d'un flot continu de données.

UDP

User Datagram Protocol – Ce protocole est trés léger. C'est un protocole sans connexion adapté à l'échange de messages simples. Applications NFS (Network File System), NTP (Network Time Protocol)

UDP

« non fiable » : pas de mécanisme pour corriger des erreurs

sans connexion : les données sont envoyés en espérant qu'il y a quelqu'un pour les recevoir

orienté message : datagram udp

Socket

Origine : L'un des apports important de l'Unix de Berkeley est l'interface de communication par Socket.

Au dépard un moyen de communication

entre processus (AF_UNIX).

Puis par extension un point d'accés pour les communications entre processus

Socket

Un socket TCP comporte par exemple

Adresse local IP

Port local de l'application

L'addresse IP distante qui fournit le service

Port distant du service

(Le port local peut être donné dynamiquement)

Il y a deux modes de communication pour les sockets :

Le mode non connecté (SOCK_DGRAM)

Le mode connecté (SOCK_STREAM)

Java Socket

public Socket(String host, int port)

throws UnknownHostException, IOException;

public Socket(InetAddress address, int port) throws IOException;

public Socket(String host, int port, InetAddress localAddr,int localPort)

throws UnknownHostException, IOException;

public Socket(InetAddress address, int port,

InetAddress localAddr, int localPort)

throws UnknownHostException, IOException

Java ServerSocket

ServerSocket permet d'écouter un port pour permettre la mise en place d'une connexion

```
public ServerSocket(int port) throws IOException;

public ServerSocket(int port, int count)
   throws IOException;

public ServerSocket(int port, int count,
   InetAddress localAddr) throws IOException;
```

La méthode accept() permet d'obtenir un Socket pour la communication

Java Stream

Pour obtenir ou envoyer des données sur un socket il faut mettre en place des « stream ». Pour cela on utilisera les méthodes

monSocket.getInputStream()

ou

monSocket.getOutputStream()

Fin