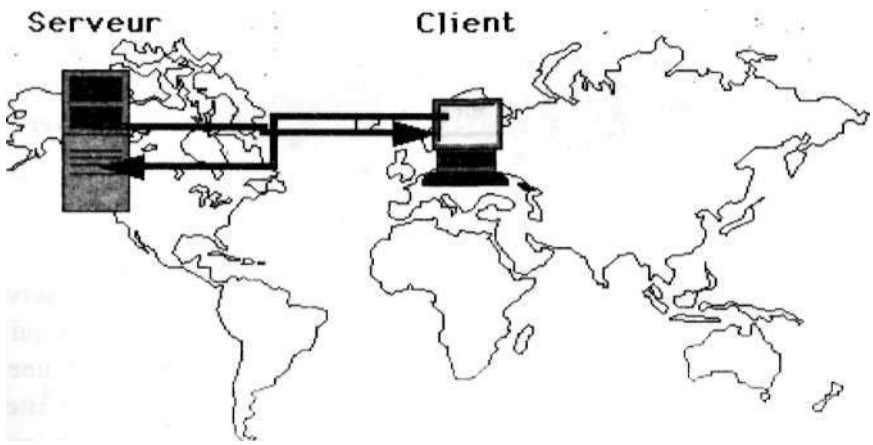
Java et la programmation Client/Serveur

Lorsque l'on écrit une application réseau, il est courant de parler de clients et de serveurs. La distinction est de plus en plus vague, mais celui qui initie la communication est généralement le client. De l'autre côté, celui qui accepte la requête est le serveur.

Dans une acception plus restrictive, est appelé client/serveur un modèle de fonctionnement logiciel dans lequel plusieurs programmes autonomes communiquent entre eux par échange de messages. Par extension, l'architecture client/ terminaux passifs.

En ce qui nous concerne, la différence la plus importante entre un client et un serveur, est que le client peut créer une Socket pour initier la communication avec une application serveur n'importe quand, alors qu'un serveur doit être en permanence à l'écoute et se tenir prêt à accepter une communication.



Dans Java, les applets ont la particularité de manipuler directement des URLs* (Uniforrn Resource Locator), ce qui représente une très grande facilité car on peut gérer des objets a travers le réseau depuis un haut niveau d'abstraction. L'objectif de cette communication est surtout l'accès à une ressource Internet. Charger une image revient à charger une URL

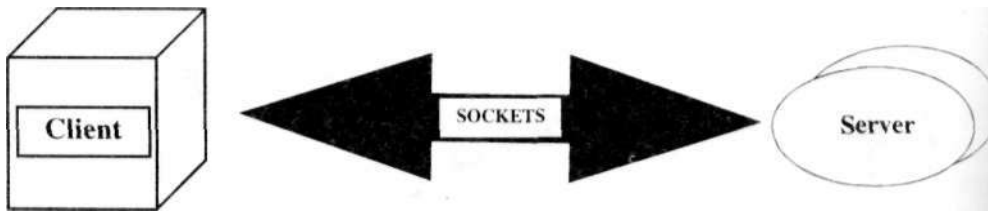
et non pas aller chercher un fichier d'un type donné sur une machine donnée avec un protocole de communication donné. Mais parfois, certaines applications nécessitent une communication de bas niveau : c'est le cas des applications Client/Serveur.

Dans l'architecture Client/Serveur, le serveur fournit un service tel que le traitement des requêtes de bases de données et le client utilise ce service fourni pour une fin donnée : Traiter les résultats des requêtes. La communication qui se produit entre le client et le serveur doit être fiable (pas de perte de données) et les données doivent être reçues par le client dans le même ordre dans lequel elles ont été transmises. Le protocole TCP fournit une telle communication à travers laquelle les applications Client/Serveur sur Internet se communiquent.

Le **package java.net** fournit deux classes nécessaires permettant de développer des applications Client/Serveur. Ce sont les classes **Socket** et **SocketServer.**

Les Sockets

Les sockets sont une interface de programmation de bas niveau pour la communication réseau. Une Socket est une liaison point à point entre un serveur et un client (le code des programmes est légèrement différent). La communication est full-duplex (en fait, les communications sont probablement bufférisées avant d'être envoyées, doù l'apparence du full-duplex là où TCP/IP ne sait faire que du half-duplex), le protocole d'échange d'informations est laissé à la charge du programmeur.



Les numéros de port

Un client a besoin de deux informations pour trouver et se connecter à un serveur: un nom de machine et un néméro de port. Le numéro de port est un idenficateur qui permet de distinguer les différents clients et les serveurs qui fonctionnent sur une même machine. Une application serveur écoute sur un numéro de port fixé et attend des connexions, ensuite pour chaque client connecté, elle attribue un nouveau numéro de port afin d'établir un dialogue.

Si nous comparons un oridnateur avec un Hôtel et les applications se connectant au serveur comme des Clients (ce qu'ils sont d'ailleurs), alors les ports correspondent aux numéros de chambres.

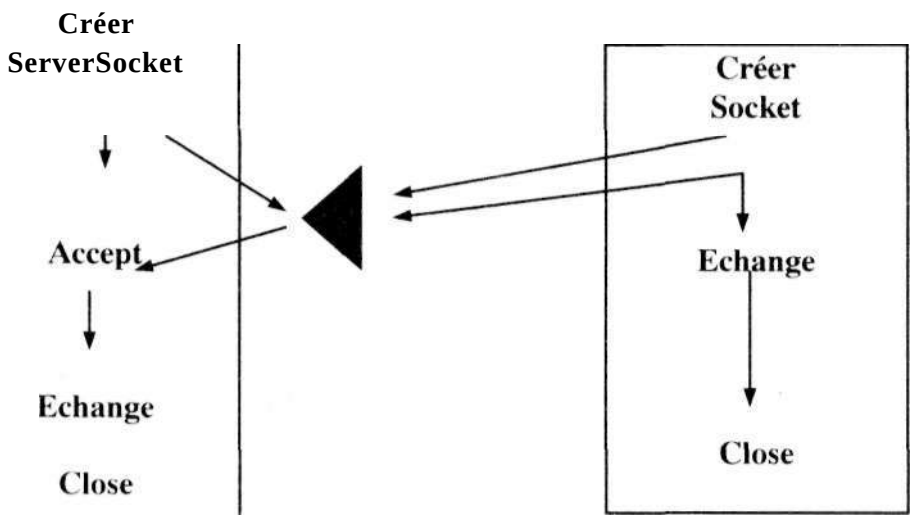
Le client

Le client est un logiciel installé sur l'ordinateur local qui permet d'une part une utilisation optimisée des ressources locales (souris, écran, etc), et d'autre part une communication avec un serveur du réseau. C'est lui qui fait le premier pas lors d'une connexion avec un serveur en établissant une Socket. Pour créer une socket dans un programme Java, le programmeur doit spécifier le nom de machine où se trouve le serveur et le port où l'on doit se connecter.

ma-socket = new Socket ("machine.dz", 1999);

Le Serveur

Un serveur est un logiciel exécuté sur un certain ordinateur du réseau, 24 heures sur 24. Il accepte commandes, questions, demandes et envoie une réponse automatiquement. 11 attend la connexion d'un client sur un port de l'ordinateur sur lequel il est installé et renvoie des données demandées. Dans un programme Java, le serveur doit créer une ServerSocket en indiquant son numéro de port puis attendre qu'un client demande une connexion qui peut alors être acceptée. Des flux d'entrées sont ensuite définis et les échanges peuvent commencer selon le protocole choisi. En fin de programme, toutes les sockets doivent être libérées.



Applets et la programmation Client/Serveur

L'utilisation de stockets avec une applet pose un certain nombre de problèmes particuliers. En effet, l'applet impose des restrictions de sécurité sur les machines accessibles : on ne peut communiquer qu'entre la machine où tourne l'applet et celle d'où provient le code.

Le Serveur multi-clients

Généralement, on désire pouvoir servir plusieurs applications clientes à partir du même serveur comme c'est le cas pour FTP et HTTP. Il est possible de créer un serveur mutithread qui crée un processus par client et pourra donc communiquer avec plusieurs clients à la fois.

Références Bibliographiques '

JDKl.1.3 Documentation
"The Java Tutorial : Object Oriented Programming for the Internet"
M. Campione, K. Walrath 1977
http ://java.sun.com/nav/read/Tutorial