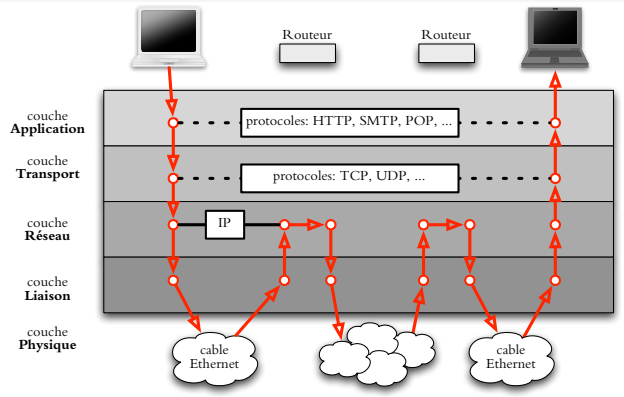
IFT1025 été 2010

Miklós Csűrös

28 juillet 2010

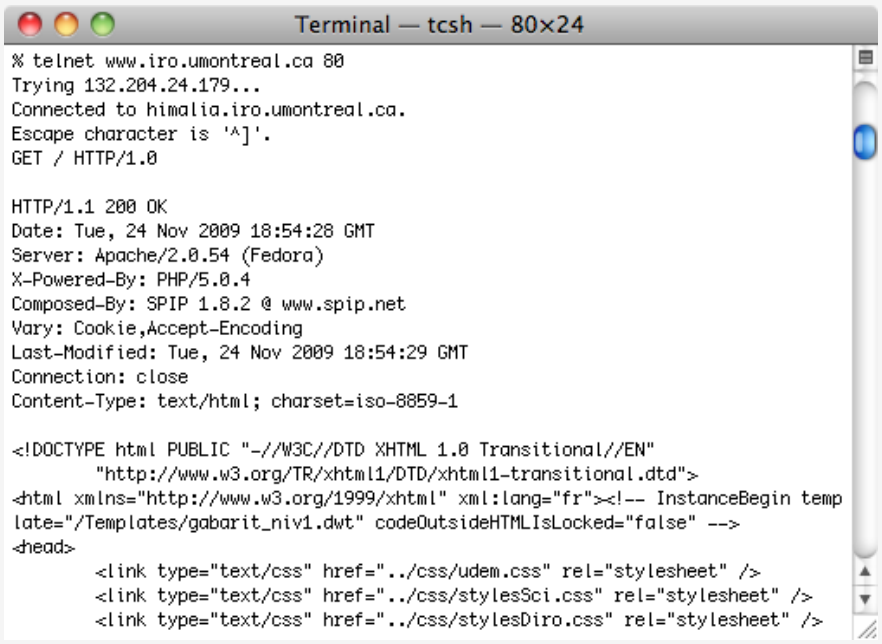
15 Java et l'Internet

15.1 Suite TCP/IP



La **suite de protocoles d'Internet** est définie à l'aide de l'abstraction par «**couches**», organisées en une hiérarchie. Chaque couche correspond à un aspect séparé de la communication, allant de fonctionnalité de base jusqu'au niveau des applications sur les ordinateurs liés. Conceptuellement, les nœuds dans une couche échangent de l'information à l'aide des protocoles standardisés. Physiquement, la connexion dans chaque couche est établie par communication avec le nœud dans la couche sous-jacente.

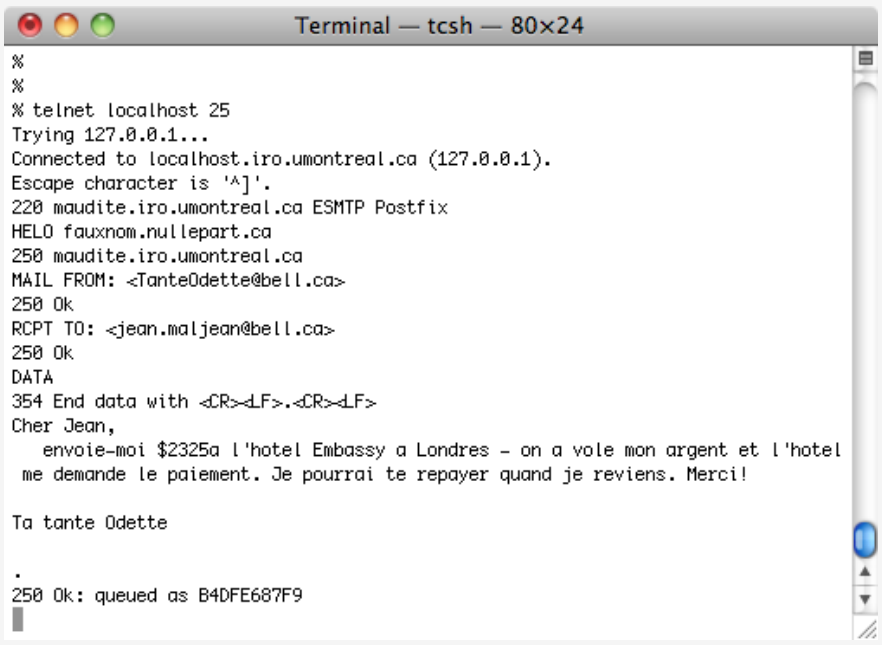
- ⋆ couche physique : caractéristiques physiques liés à la médium (encodage par signaux, synchronisation, etc.)
- ⋆ couche liaison : transportation de paquets dans la couche physique (p.e., protocole Ethernet)
- ⋆ couche réseau : acheminement de paquets à travers un seul réseau. Exemple : *Internet Protocol* définit l'adressage IPv4 (32 bits en format xxx.xxx.xxx.xxx) où IPv6 (128 bits), et la transmission de messages : (1) passerelles entre réseaux [*next hop*], (2) couche liaison → couche transport [reçu de message], (3) couche transport → couche liaison [envoi de message].
- ⋆ couche transport : assure la fiabilité de la communication (assemblage de paquets, requêtes de retransmission). Exemple : *Transmission Control Protocol* (TCP) assure l'assemblage et livraison propre de messages, ainsi que le multiplexage par **port**s.
- ⋆ couche application : définit le syntaxe de la communication sur un port. Exemples : *Hypertext Transfer Protocol* (http), *File Transfer Protocol* (ftp), *Simple Mail Transfer Protocol* (smtp), *Post Office Protocol* (pop), *Internet Message Access Protocol* (imap), *Internet Relay Chat Protocol* (irc), etc.



On peut établir une connexion TCP/IP par le logiciel `telnet` :

`telnet` *adresse port*

(*adresse* est l'adresse IP ou le nom du serveur.) Le protocole **HTTP** définit par exemple le syntaxe du message `GET` : c'est la requête d'un document.



Le protocole **SMTP** définit les messages pour envoyer un courriel : `HELO` pour se présenter, `MAIL FROM` pour l'expéditeur, `RCPT TO` pour le destinataire, et `DATA` pour le texte du courriel.

15.2 Sockets

L'accès à la suite TCP/IP se fait par *sockets*. Un socket correspond à une connexion dans la couche de transport (adresse+port au serveur).

Socket au client

```
/** Code d'un client */
import java.net.Socket;
import java.io.*;
...
    Socket s = new Socket("www.iro.umontreal.ca", 80); // adresse et port
    InputStream instream = s.getInputStream();
    OutputStream outstream = s.getOutputStream();
    PrintWriter out = new PrintWriter(outstream);
    BufferedReader in = new BufferedReader(new InputStreamReader(instream));
    String command = "GET / HTTP/1.0\n\n";
    out.print(command);
    out.flush(); // pour assurer l'envoi
    while(true) // et on lit ce qui arrive à l'input
    {
        String line = in.readLine();
        if (in == null) break;
        ...
    }
    s.close(); // libère la ressource Socket
```

Sockets au serveur

```
/** Code du serveur */
import java.net.Socket;
import java.net.ServerSocket;
import java.io.*;
...
    int port = 8888; // port écouté sur cette machine
    SocketServer server = new SocketServer(port); // lance le serveur
    while (true)
    {
        Socket s = server.accept(); // attente de connexion [sommeil du procéssus]
        Runnable service = new ClientAssistant(s);
        Thread t = new Thread(service); // traiter les connections en Threads séparés
        t.start(); // et on peut continuer la boucle
    }
```

Ici, la classe `ClientAssistant` utilise les méthodes **getInputStream** et **.getOutputStream** comme le client dans la communication.

15.3 Connexion URL

L'API de Java définit des classes explicitement conçues pour la communication par HTTP et FTP.

- ⋆ classe `URL` : définit le format *Unified Resource Locator* (URL), comme `http://www.iro.umontreal.ca/index.html`.
- ⋆ classe `URLConnection` : encapsule la communication par sockets et la construction de messages HTTP/FTP.
- ⋆ classe `HttpURLConnection` : extension de `URLConnection` avec d'autres fonctionnalités de HTTP.

⇒ il n'est pas nécessaire de connaître les détails du protocole http, ou les sockets.

```
import java.net.URL;
import java.net.URLConnection;
import java.net.HttpURLConnection;
...
    String urlString = "http://www.iro.umontreal.ca/";
    URL u = new URL(urlString);
    URLConnection connection = u.openConnection(); // throws IOException
    HttpURLConnection httpConnection
        = (HttpURLConnection) connection; // conversion OK à cause de "http://"
    int code = httpConnection.getResponseCode(); // 200=OK, 404=Not Found
    if (code != HttpURLConnection.HTTP_OK)
    {
        System.err.println(httpConnection.getResponseMessage());// p.e., "Not Found"
    } else
    {
        InputStream instream = connection.getInputStream();
        ... // lecture du texte
    }
```

15.4 Applets

Imbrication d'une application Java dans un page Web : **applet**. La classe `javax.swing.JApplet` définit un containeur de plus haut niveau comme `JFrame`. Un applet n'a pas d'accès aux fichiers locaux et ne peux télécharger que des document au serveur (par `URL`). Inclusion de l'applet dans un document HTML :

```
<applet code="TumbleItem"
        archive="tumbleClasses.jar, tumbleImages.jar"
        width="600" height="95">
    <param name="maxwidth" value="120">
    <param name="nimgs" value="17">
    <param name="offset" value="-57">
    <param name="img" value="images/tumble">

Your browser is completely ignoring the <applet> tag!
</applet>
```

Extension de `JApplet` : redéfinition des méthodes suivantes.

- ⋆ **init**() : c'est comme un constructeur, appellé une fois quand l'applet est affiché par le fureteur (p.e., lancer des threads)
- ⋆ **start**() : appellée après **init**, chaque fois que la page est affiché de nouveau (p.e., lancer une animation avec `Timer`)
- ⋆ **stop**() : appellée avant de déstruction quand l'usager quitte le page (p.e., arrêter l'animation)
- ⋆ **destroy**() : appellé pour la libération de ressources (p.e., arrêter les threads)

Graphisme : redéfinir **paint**(`Graphics` g) où attacher des composantes Swing à **getContentPane**().

Paramètres : utiliser la méthode `String` **getParameter**(`String` nom).