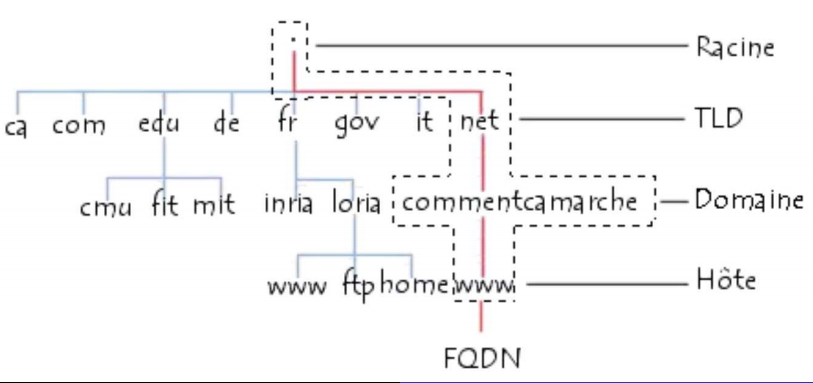
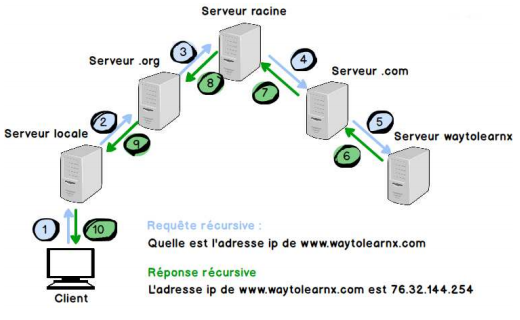
***DNS***

DNS est un service permettant d’établir une correspondance entre un nom de domaine et une adresse IP, se base sur un système de nommage hiérarchique qui garantit l’unicité du nom auquel le **.** représente la racine, Ce système est gérer par l’organisme américain appelé l’ICANN qui est responsable de 13 serveurs DNS qui gère la racine Dns,

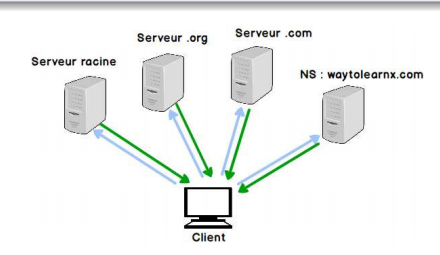


Le mécanisme consistant á trouver l’IP correspondant au nom d’un hôte est appelé la résolution d’un nom de domaine.

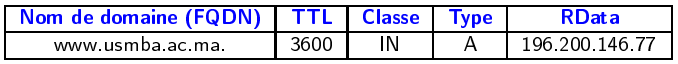
Type de requête DNS :

Récursive : Serveur DNS peut joindre d’autre serveurs pour trouver la bonne réponse, et le résolveur s’attend á ce que le serveur fournisse la réponse finale. 

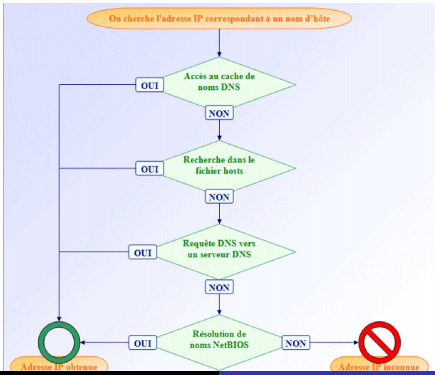
Itérative : Serveur DNS renvois la meilleure réponse qui a peut donner sans contacter les autres serveurs, ou bien le serveur renvoie l’adresse ip de serveur qui selon lui peut résoudre la requête( redirection).

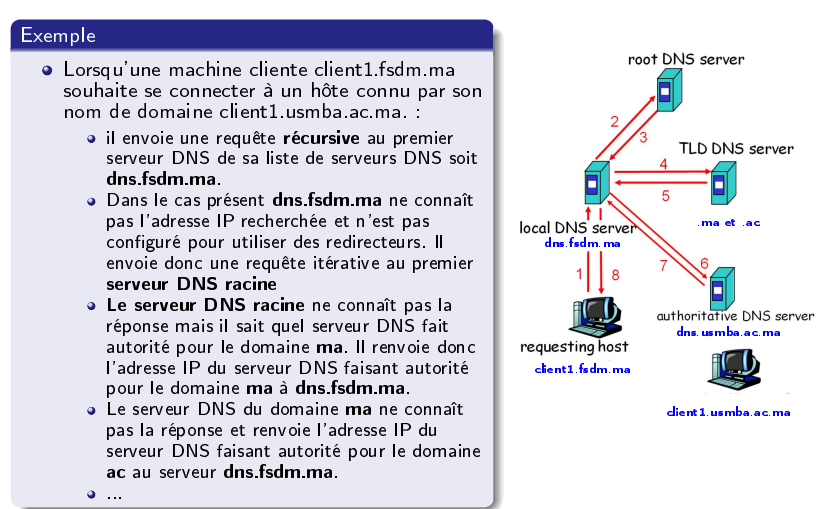


Chaque serveur dns contient une table d’enregistrement qui permet de stocker les résolutions déjà faite, et les plus fréquentes, exemple d’un enregistrement DNS :



Processus de résolution de nom de domaine :





**Nslookup :** permet de savoir le serveur dns utilisé par défaut dans la machine.

**Dig** [**www.google.com**](http://www.google.com) : résoudre le nom de domaine vers adresse ip et affiche des détails sur le serveur et la requête envoyée.

**Dig** [**www.google.com**](http://www.google.com) **+short** : affiche que l’adresse ip correspond a google.com.

Dig [www.google.com](http://www.google.com) +trace : résoudre la requête vers le bas à partir du serveur de noms racine, puis signaler les résultats de chaque étape.

**Dig –x 216.58.211.196 :** résoudre l’adresse ip vers un nom de domaine

Ipconfig /diplaydns : permet d’afficher le cache dns.

Ipconfig /flushdns : permet de vider le cache dns.

Host –t ns . 8.8.8.8 : permet de trouver les 13 serveur dns de la racine á l’aide de serveur 8.8.8.8

Host –t –a [www.siteduzero.com](http://www.siteduzero.com): permet de résoudre le nom en une adresse ip.

***InetAddress***

Pour instancier un objet InetAddress á partir d’un nom de domaine ou bien d’une adresse IP, on utilise la méthode statique getByName(String nomDeDomaine) de la classe InetAddress :

**InetAddress ip = InetAddress.getByName(‘’www.google.com’’) ;**

Ou bien

**InetAddress ip = InetAddress.getByName(‘’127.0.0.1’’) ;**

***Obtenir l’adresse de serveur local :***

InetAddress ip = InetAdress.getLocalHost() ; 🡺 c’est une méthode statique de la classe InetAdresse qui retourne un objet de type InetAdress, avec l’ ip de serveur local: (‘’ renvoie l’adresse ip réseau de la machine exécutant ce code)

***Obtenir le nom de la machine á partir d’un objet InetAddress ip :***

ip.getHostName()  🡺 retourne une chaine de caractère comporte le nom de la machine correspond á l’objet ip ;

***Obtenir l’adresse ip de la machine á partir d’un objet InetAddress ip :***

ip.getHostAdresse() 🡺 retourne une chaine de caractere comporte l’adresse ip de la machine qui correspond á l’objet ip;

***Tester le ping sur un objet de type InetAddress ip :***pour tester le ping on utilise la méthode de l’objet ip isReachable(Time tempReponse)  qui retourne un boolean.

Test de ping sur google : InetAddress ip = InetAddress.getByName(‘’www.google.com’’) ;

If(ip.isReachable(1000)) syso(‘’ping á bien passer’’)

Else syso(‘’ping n’a pas passer’’) ;

***URL***

Pour manipuler les Url on java on utilise un objet prédéfinie s’appelé URL :

Instancier un URL 🡺 URL u1 = new URL(‘’www.googl.com’’) ;

Ou bien 🡺 URL u1 = new URL(‘’http’’,’’google.com’’,80,’’/‘’) ;

La forme du dernier constructeur est URL(‘’protocole’’,’’nomDeDomaine’’,port,’’fichierVoulu’’) ;

Pour afficher le contenu de l’url on utilise la méthode getContent() d’un objet URL ;

u1.getContent() ;

***HttpURLConnection***

Pour afficher le contenu de l’entête d’une requête http, on a besoin d’un objet HttpUrlConnection, et pour instancier cet objet, on utilise la méthode openConnection() de l’objet URL qui contient l’adresse d’un certain siteWeb http, cette méthode retourne un objet de type URLConnction() dont lequel on ajoute le casting (HttpURLConnection) pour avoir l’objet HttpURLConnection, si tout se passe bien, on peut affichier le header de la requête http á l’aide de 2 méthodes , la méthode getHeaderFieldKey(indice i ) pour avoir le nom du champ, et la methode getHeaderField(indice i )  pour avoir sa valeur;

URL u1 = new URL(‘’http://www.google.com’’) ;

URLConnection urlc = u1.openConnection() ;

HttpURLConnection hurlc = (HttpURLConnection)urlc ;

***Les sockets TCP***

***Coté server***

Crée objet ServerSocket ss = new ServerSocket(1030)  , avec 1030 est le port pour accéder á ce serveur.

Pour répondre á un client on utilise un objet socket retourné par la méthode accept() de l’objet ServerSocket 🡺 Socket s = ss.accept() ;

Pour la communication avec le client il y’a 2 mode, le mode INPUT et le mode OUTPUT.

*Pour recevoir l’information á partir du client en utilisant les sockets avec le mode INPUT.*

DataInputStream in = new DataInputStream(s.getInputStream) ;

Suppossant que le contenu est un string et qu’on veut l’afficher :

String ch = in.readUtf() ;syso(ch) ;

*Pour envoyer l’information vers le client on utilise les sockets avec le mode OUTPUT.*

Suppose qu’on veut envoyer une chaine de caractère String ch = ‘’ Bonjour’’, on crée d’abord le flux sortant  :

DataOutputStream out = new DataOutPutStream(s.getOutputStream) ;

out.writeUTF(ch) ;

***Coté client***

On crée un objet de type Socket s = new Socket(‘’serverIp’’,port) ;

Une fois le socket est créé, le reste de la communication est comme les serveur, á l’aide des modes Input et output.

On peut envoyer des différent type á travers les sockets:

|  |  |  |
| --- | --- | --- |
|  | INPUT | OUTPUT |
| Les types primitifs | DataInputStream in = new DatIn…() ;  Int in.readInt() ;  Double in.readDouble() ;  String in.readUTF() ; | DataOutputStream out = new DataOutput() ;  Out.writeInt(int i) ;  Out.writeDouble(double d) ;  Out.writeUTF(String s) ; |
| Les objets | ObjectInputStream in = ObjcectIn..() ;  Object in.readObject() | ObjectOutputStream  out = new ObjectOt..() ;  Out.writeObject() ; |
| Les Fichiers | BufferedReader  Instanciation :  BufferedReader input =  **new**  BufferedReader(  **new** InputStreamReader(  s.getInputStream ()  )  );  String answer = input.readLine(); | PrintWriter  Instanciation :  **new** PrintWriter(socket.getOutputStream(), **true**);  out.println(**new** Date().toString()); |

***Les sockets UDP :***

***Cote client :***

On crée un objet de type DatagramSocket ds = new DatagramSocket() ;

On veut envoyer la chaine String str = ‘’welcome’’ au serveur, pour çela on crée un objet DatagramPacket á l’aide de 4 paramètres :

DatagramPacket = new DatagramPacket(str.getBytes(), str.length, InetAddress.getByName(‘adresse de serveur’ ), serverPort)

DatagramPacket dp = **new** DatagramPacket(str.getBytes(), str.length(), ip, 3000);

Pour envoyer le packet on utilise la méthode send() de l’objet DatagramSocket : ds.send(dp) ;ds.close()

***Cote serveur :***

On crée un objet de type DatagramSocket ds = new DatagramSocket(port) ;

Les données échangées sont de type byte, donc on crée un tableau de type byte : **byte**[] buf = **new byte**[1024]; dans lequel on reçois les information reçus par l’objet DatagramPacket ;

DatagramPacket dp = **new** DatagramPacket(buf, 1024); ds.receive(dp);

Et enfin on convertie le buffer vers une chaine de caractère avec le code suivant :

String str = **new** String(dp.getData(), 0, dp.getLength());

Enfin ds.close