DNS - Résolution des noms - Filius

Michel Billaud (michel.billaud@u-bordeaux.fr, michel.billaud@laposte.net)

5 juillet 2020

Table des matières

Objectif

Expliquer quelques bases sur les DNS, avec des manips faisables avec Filius et ses limitations

TODO : trouver de meilleurs noms pour les machines de l'exemple "iut".

1 Objectif : résolution des noms.

1.1 Adresses, noms et résolution

Sur Internet on désigne des machines (qu'on appelle historiquement des **domaines**) par les adresses IP de leurs interfaces : adresses sur 32 bits pour IPv4, sur 128 bits pour IPv6, mais en pratique, on utilise des noms de domaines comme `www.iut.u-bordeaux.fr`.

La **résolution d'adresses** consiste à déterminer la ou les adresses IP qui correspondent à un nom (`147.210.94.18` pour `www.iut.u-bordeaux.fr`)

Les programmes qui utilisent des adresses IP font appel à un module du système d'exploitation, le **resolver**, qui se charge de la résolution. Voir annexe : utilisation de la fonction `getaddrinfo()`.

1.2 Noms qualifiés ou pas.

Quand on travaille avec les machines d'un réseau, il est commode de les désigner par un "petit nom" (*hostname*) comme `pc-alice`, `console-bob`, etc. Ces noms font implicitement référence à un **suffixe par défaut** (exemple `home.org`).

Le nom complet (ou FQDN = fully qualified domain name) s'obtient en combinant les deux, exemple `pc-alice.home.org`.

1.3 Un système hiérarchisé

L'espace des noms est découpé en **zones**, qui forment une arborescence arborescente :

- `u-bordeaux.fr` fait partie de la zone `.fr`,
- `iut.u-bordeaux.fr` fait partie de la zone `.u-bordeaux.fr`,
- `www.iut.u-bordeaux.fr` est un nom qui fait partie de la zone `.iut.u-bordeaux.fr`.

Convention : on mettra un point devant quand on parle d'une zone.

Tout en haut, les noms `fr`, `com`, `edu` etc. (les TLD = Top Level Domains) font partie de la zone "racine" (.).

Les informations de chaque zone sont maintenues par des serveurs de noms. qui connaissent les informations pour les noms qui y sont *directement* rattachés. Par exemple, dans `.fr` on a des informations sur `u-bordeaux.fr` et `gouv.fr` etc.

1.4 Délégation

Par contre, le serveur de noms de `.fr` ne connaît rien sur le contenu de la zone `.u-bordeaux.fr`. Tout ce qu'il sait c'est que

- `u-bordeaux.fr` est le nom d'une zone,
- les serveurs qui s'en occupent sont `ns2.nic.fr`, `bxnms.u-bordeaux.fr` et `cnudns.cines.fr`.

Le système de noms est donc décentralisé, avec un mécanisme de **délégation** de l'administration des informations. L'administrateur de `.fr` ne s'occupe pas

de mettre à jour le contenu de `.iut.u-bordeaux.fr` dont il ne connaît même pas l'existence.

1.5 Mécanisme de résolution

Pour résoudre le nom `www.u-bordeaux.fr`, il faudra donc en principe

- interroger un serveur du domaine racine pour trouver un serveur de noms pour `.fr`,
- interroger le serveur de `.fr` pour trouver un serveur de noms pour `.u-bordeaux.fr`,
- interroger ce serveur de noms pour obtenir l'adresse de `www.u-bordeaux.fr`.

C'est un processus itératif, une boucle pour interroger divers serveurs.

Heureusement, on ne le fait pas à chaque fois : le *resolver* réemploie les informations qu'il a déjà obtenues, dans la limite de leur **durée de vie**.

1.6 Interrogation d'un serveur

En pratique, les machines reliés à internet sont configurées pour interroger un serveur DNS (Domain Name SERVER) ou plusieurs, qui prennent en charge tout ou partie du travail.

On peut distinguer plusieurs rôles pour les serveurs DNS

- **relayer** les interrogations venant des machines d'un réseau interne. Il a un rôle de **mandataire**. En mémorisant dans un **cache** les informations qu'il obtient, il contribue à réduire la charge réseau et améliorer les temps de réponse.
- **rendre disponibles** les informations à propos d'un ou plusieurs domaines dont il est **autorité**

Dans le premier cas, un serveur mandataire prend en charge la requête envoyée par le client, et lui fournit une réponse complète. On parle de **requête récursive** (pour résoudre, le client demande à quelqu'un de résoudre). Les réponses ne font pas autorité (non-authoritative answers) parce qu'elles ont été obtenues indirectement, et risquent de ne pas être parfaitement à jour.

Dans le second cas, le serveur public ne donne de réponse complète que pour les domaines dont il est **autorité**. Pour le reste, il retourne au client l'adresse d'un autre serveur à qui s'adresser. On parle alors de **requête itérative**.

2 Pratique

2.1 Un serveur pour une zone

3 machines sur un réseau local

2.1.1 Config réseau

- Alice = 10.1.1.1, Bob = 10.1.1.2, Server = 10.1.1.100.
- vérifier pings de l'une à l'autre.

2.1.2 Config DNS

- installer logiciel DNS sur NS.
- Sur Serveur, ajouter un enregistrement de type "A"

```
bob.home.org    10.1.1.2
```

2.1.3 Premier test

- démarrer le serveur
- configurer Alice pour employer le serveur de noms.
- tester avec la commande `host bob.home.org`

2.1.4 Suite

- ajoutez les autres noms
- testez depuis les autres machines

2.1.5 Complément

Ajoutez aussi un enregistrement A pour une machine qui n'existe pas (charlie, 10.1.1.4).

Quelles conclusions tirer des commandes

```
ping charlie.home.org
host charlie.home.org
```

2.2 Domaine / sous-domaine

Dans cette partie on regarde l'articulation entre plusieurs serveurs qui sont autorités pour des zones subordonnées (`iut`, `iut.info`, `iut.gc`…)

2.2.1 Création du réseau sous FILIUS

On se contente d'un réseau "plat", avec tous les machines connectées au même switch.

- lancer Filius, créer un nouveau projet
- positionner un switch reliant 4 machines : `CLIENT`, `SIUT`, `SINFO`, `SGC`.
- affecter des adresses à ces 4 machines (10.1.1.1, ….)

2.2.2 Configuration du client

- sur le client, installez le logiciel "Ligne de Commande",
- vérifier par `ping` depuis le client qu'il peut communiquer avec les serveurs.

Lien avec un serveur de noms :

- Essayez la commande `host client.iut`. Interprétez la réponse (CHEAT : `non résolu, aucun serveur disponible`, le client n'a contacté aucun serveur, et n'a donc pas pu trouver de réponse)
- Dans la configuration réseau, indiquez l'adresse de `SIUT` comme serveur de noms.
- Essayez de nouveau. Interprétation ? (CHEAT : `délai écoulé, aucune réponse` : le client a tenté de contacter un serveur, qui tarde à répondre)

2.2.3 Configuration d'un serveur DNS (serveur IUT)

- Sur le serveur SIUT, installez et démarrez le logiciel DNS.
- Depuis le client, essayez la commande `host client.iut`. Interprétez la réponse (CHEAT : `nom d'hôte non résolu` le client a contacté un serveur, qui répond qu'il ne sait pas).
- Enregistrez maintenant le nom de domaine `client.iut` avec l'adresse qui va bien.
- Testez de nouveau. CHEAT : `client.iut a pour adresse IP 10.1.1.1` .

Maintenant que ça marche (si tout va bien),

- Enregistrez les noms `siut.iut`, `sinfo.iut`.
- Vérifiez que le client peut les résoudre (commande `host`).

Note : apparemment le DNS simulé n'est pas capable de fournir la résolution inverse.

2.2.4 Serveur DNS pour un sous-domaine

On veut maintenant que le serveur SINFO soit autoritatif pour le sous-domaine `info.iut`.

- installez et démarrez le serveur DNS sur SINFO
- enregistrez-y un nom comme `jj.info.iut` (adresse IP quelconque).

Depuis le client,

- on essaie host `jj.info.iut`. Réponse : `nom d'hôte non résolu`
- interprétation : le client a interrogé SIUT, qui n'a pas pu lui donner la réponse voulue.

Pour que ça fonctionne, il faut que le serveur SIUT sache que la gestion de la zone `.info.iut` est **déléguée** au serveur de noms `sinfo.iut`.

- À ajouter dans l'onglet NS (name servers).

Ainsi, le serveur SIUT va renvoyer le client qui l'interroge vers l'autre serveur.

2.2.5 Autre cas de figure

Imaginons maintenant qu'une machine cliente soit configurée pour interroger le serveur SINFO.

- tester la résolution de `jj.info.iut` (ok)

- tester avec `client.iut` (ko)

Le problème, c'est que SINFO ne sait pas qui gère la zone .iut Une solution simple est de déclarer que le domaine .iut est généré par le serveur SIUT.

- dans l'onglet NS du serveur SINFO, déclarer que `sinfo.iut` est serveur DNS de `.iut`.
- tester : ça ne marche toujours pas.

La raison, c'est que le serveur DNS ne connaît pas l'adresse IP de `siut.iut`.

- **rafistolage :** ajouter une déclaration d'adresse pour `siut.iut`.

Cette déclaration est appelée "glue record", parce qu'elle sert à recoller les morceaux. elle est dans le serveur pour `siut.iut`, mais concerne une machine qui est en dehors de cette zone.

2.2.6 Glue records pour les serveurs racine

Mais ce rafistolage ne "passe pas à l'échelle" dans des situations réalistes : en effet, on ne va pas enregistrer (et maintenir) sur chaque serveur des déclarations pour toutes les autres zones. Ca irait totalement à l'encontre du principe d'administration décentralisée de l'espace des noms.

La **solution** est de déclarer simplement les adresses des serveurs "racine", à partir desquels on peut atteindre (itérativement) tous les serveurs des domaines enregistrés.

On déclarera a donc un enregistrement NS indiquant que le domaine racine "." est desservi par un serveur

```
.                    NS  a.root-servers.net.   ; le serveur de noms pour "."
a.root-servers.net.  A   xx.yy.zz.tt           ; glue record pour son adresse IP
```

La liste (root hints file) des 13 serveurs racine (de A à L) est publiée et mise à jour officiellement. https ://www.iana.org/domains/root/files

2.3 Your own dark Internet

Les noms de domaines d'Internet forment une arborescence, déterminée par les serveurs racines.

Mais rien n'empêche de construire un serveur racine non-officiel. Exemple, dans leur projet pour gouverner le monde, les chats veulent des noms

```
abyssan
persan
...
slaves.humans
slaves.dogs
```

Le serveur racine connaîtra donc les TLD `abyssan`, `european`, etc. avec les serveurs qui font autorité.

Construire un réseau avec quelques TLD et sous domaines, avec un ou 2 noms dans chaque.

3 Annexes

3.1 Terminologie

- **Domaine réseau** (network domain) : groupement administratif de réseaux d'ordinateurs ou hôtes. Les domaines qui doivent être accessible depuis l'Internet public ont un nom unique dans le Domain Name System.
- Le **Domain Name System** (DNS) est un système hiérarchique et décentralisé pour associer des informations diverses à des *noms de domaines*. En particulier, il permet de traduire des *noms de domaines* en adresses IP numériques. Attention, le sigle est aussi utilisé pour **Domain Name Server**.
- Un **nom de domaine** est une chaîne qui identifie une entité sur Internet. C'est une suite d'étiquettes séparées par des points. Il identifie un *domaine réseau*, ou une ressource comme un ordinateur. Tout nom enregistré dans le DNS est un nom de domaine. Les noms sont organisés en sous-domaines.
- un **nom de domaine complètement qualifié** (FQDN = fully qualified domain name) est un nom de domaine complet. On le termine généralement par un point. Exemple : `www.iut.u-bordeaux.fr.`est le FQDN de `www.iut` dans le domaine réseau `u-bordeaux.fr`.
- le **domaine racine** n'a pas de nom. En dessous, on trouve les "Top Level Domains", comme `fr.` , `com.`, etc.
- un **serveur de noms** (Domain Name Server) est une machine capable de fournir à d'autres des informations à propos de certains noms de domaines.
- **la résolution d'adresse** consiste à obtenir des adresses (ou plus généralement des informations a propos d'un nom.

3.2 Exemple résolution en C

Le programme qui suit

- prend en paramètres une suite de noms de domaines
- affiche les adresses IP correspondantes (IPv4 et IPv6).

Exemple d'utilisation

```
$ cc -std=c11 -Wall -Wextra     resolve.c   -o resolv
$ ./resolve  www.google.fr www.u-bordeaux.fr
www.google.fr
    IPv6: 2a00:1450:4007:807::2003
    IPv4: 216.58.209.227
www.u-bordeaux.fr
    IPv4: 147.210.215.26
```

Code source (fichier `resolve.c`)

```
/**
 * Displays ipv4 and ipv6 addresses of domain names
 * given as command line parameters.
 *
```

```c
 * m billaud, 2020
 */

#define _XOPEN_SOURCE 700

#include <stdio.h>

#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <arpa/inet.h>

void print_addresses(char *hostname);

int main(int argc, char *argv[])
{
    for (int i = 1; i < argc; i++) {
        print_addresses(argv[i]);
    }
    return 0;
}

void print_addresses(char *hostname)
{
    // get list of results
    struct addrinfo hints = {
        .ai_family   = AF_UNSPEC,       // ipv4 or IPv6
        .ai_socktype = SOCK_STREAM
    };

    struct addrinfo *results;
    printf("%s\n", hostname);
    int  s = getaddrinfo(hostname, NULL, &hints, &results);
    if (s != 0) {
        printf ("\tinconnu\n");
        return;
    }
    // display results
    for (struct addrinfo *ptr = results; ptr != NULL; ptr = ptr->ai_next) {
        char * family_name = "unknown";
        void * addr = NULL;

        switch (ptr->ai_family ) {
        case AF_INET :
            family_name = "IPv4";
            addr = &(((struct sockaddr_in *)(ptr->ai_addr)) ->sin_addr);
            break;
        case AF_INET6 :
            family_name = "IPv6";
            addr = &(((struct sockaddr_in6 *)(ptr->ai_addr)) ->sin6_addr);
```

```
            break;
        }
        // inet_ntop converts address to textual numeric form
        char ip_as_string[100];
        inet_ntop(ptr->ai_family, addr, ip_as_string, sizeof(ip_as_string));
        printf("\t%s: %s\n", family_name, ip_as_string);

    }
    freeaddrinfo(results);
}
```

3.3 Exemple résolution en Java

Le programme suivant, écrit en Java, fait presque la même chose :

```
$ java resolution.Resolution  www.google.fr www.labri.fr
www.google.fr
        IPv4: 216.58.215.35
        IPv6: 2a00:1450:4007:808:0:0:0:2003
www.labri.fr
        IPv4: 147.210.8.59
        IPv6: 2001:660:6101:404:0:0:0:80
```

Source

```
package resolution;

/**
 * Displays ipv4 and ipv6 addresses of domain names
 * given as command line parameters.
 *
 * m billaud, 2020
 */

import java.net.InetAddress;
import java.net.Inet4Address;
import java.net.Inet6Address;

import java.net.UnknownHostException;

public class Resolution {

    public static void main(String[] args) {
        for (String domainname : args) {
            print_addresses(domainname);
        }
    }

    private static void print_addresses(String name) {
        System.out.println(name);
        try {
```

```
            for (InetAddress address : InetAddress.getAllByName(name)) {
                String family
                        = address instanceof Inet4Address ? "IPv4"
                                : address instanceof Inet6Address ? "IPv6"
                                        : "other";
                System.out.format("\t%s: s\n",
                                  family, address.getHostAddress());
            }
        } catch (UnknownHostException ex) {
            System.out.format("\tnot found");
        }
    }
}
```

3.4 Annexe : configurer un serveur DNS

Avec le simulateur FILIUS

1. Déterminer si il doit être récursif ou pas (en général : oui si serveur "mandataire", non pour serveur accessible de l'extérieur).
2. Enregistrer au moins un "serveur racine" (serveur pour la zone ".", avec un *glue record* pour son adresse).
3. Enregistrer les adresses IP (A) des hôtes connus (et gérés) par ce serveur
4. Enregistrer les serveurs de noms (NS) pour les sous-domaines connus, et leurs adresses (A).

Il s'y ajoutera ensuite les déclarations pour les échangeurs de courrier (MX).

Sur de vrais serveurs de noms, il y aura davantage de choses à gérer, notamment

- la résolution inverse (à partir d'un numéro IP comme `147.210.8.59`, retrouver le nom de domaine `www3.labri.fr`)
- les transferts d'informations pour mettre à jour les serveurs secondaires d'une zone, qui doivent avoir une copie des informations d'une zone détenues par le serveur primaire, au cas où ce dernier serait indisponible).

Ainsi qu'une grande quantité de types de Resource Records

- SOA : Start of authority, déclaration de zone
- HINFO : hardware information
- TEXT : information
- CNAME : canonical name, synonyme
- AAAA : adresses IPv6
- PTR : pointers, résolution inverse

Liste plus complète (plus d'une centaine) dans https ://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-4