

Résolution de noms

1 Description

On désire concevoir une architecture hiérarchique de résolution de noms. Notre architecture est inspirée du service DNS. L'objectif est de traduire un nom de machine en une IP.

Notre architecture est composée de serveurs racines qui possèdent les noms des serveurs de domaines ainsi que l'information nécessaire afin de s'y connecter. Ces serveurs de domaines pointent eux-mêmes vers d'autres serveurs de domaines et permettent la résolution de noms de machines, ainsi de suite, pour les serveurs de l'étage suivant. Tous les serveurs de cette hiérarchie sont appelés serveurs de noms. Ils possèdent les mêmes fonctionnalités.

Les domaines sont hiérarchiques. Par exemple, le nom `zombie.example.fr` se compose du nom de machine `zombie`, ensuite du domaine `example` qui est un sous-domaine de `.fr`.

Pour résoudre le nom `zombie.example.fr` de notre exemple, le client se connecte d'abord à un serveur racine, pour obtenir le nom des serveurs du domaine `.fr` (nom + numéro de port). Le client contacte ensuite l'un de ces serveurs afin d'obtenir l'information des serveurs du sous-domaine `example.fr`. Finalement, le client contacte l'un de ces derniers serveurs afin d'obtenir l'IP du serveur `zombie.example.fr`.

Les noms à résoudre sont de la forme `<nom de machine>. <sous-domaine>. <domaine>`.

Au démarrage, le client charge un fichier avec la liste des serveurs racines. Ce fichier est au format :

```
<IP1> | <port1>
<IP2> | <port2>
...
<IPn> | <portn>
```

Vous ferez des tests avec plusieurs serveurs racines. Les adresses de ces serveurs sont un mélange d'adresses IPv4 et IPv6.

Les serveurs de noms chargent un fichier au format suivant :

```
<domaine1> | <IP1> | <port1>
<domaine2> | <IP2> | <port2>
...
```

Le champ `domaine` indique le domaine servi par le serveur dont l'IP est indiquée en deuxième colonne. Le port sur lequel écoute ce serveur est listé dans la troisième colonne. Plusieurs serveurs servent le même domaine. Les adresses sont un mélange d'adresses IPv4 et IPv6.

Une requête client est de la forme :

```
<identifiant de transaction> | <horodatage> | <nom>
```

La réponse du serveur de nom comprend les résolutions pour IPv4 et IPv6.

```
<identifiant de transaction> | <horodatage> | <nom> | \
<code> | <domaine>, <IP1>, <port1> | ... | <domaine>, <IPn>, <portn>
```

L'identifiant de transaction et l'horodatage sont identiques aux valeurs contenues dans la requête. Ils sont suivis du nom indiqué dans la requête, d'un code indiquant le succès (≥ 0) ou l'échec de la

requête (= -1) ainsi que d'une suite de tuples (domaine, IP, ports) contenant l'information de machines servant le domaine indiqué.

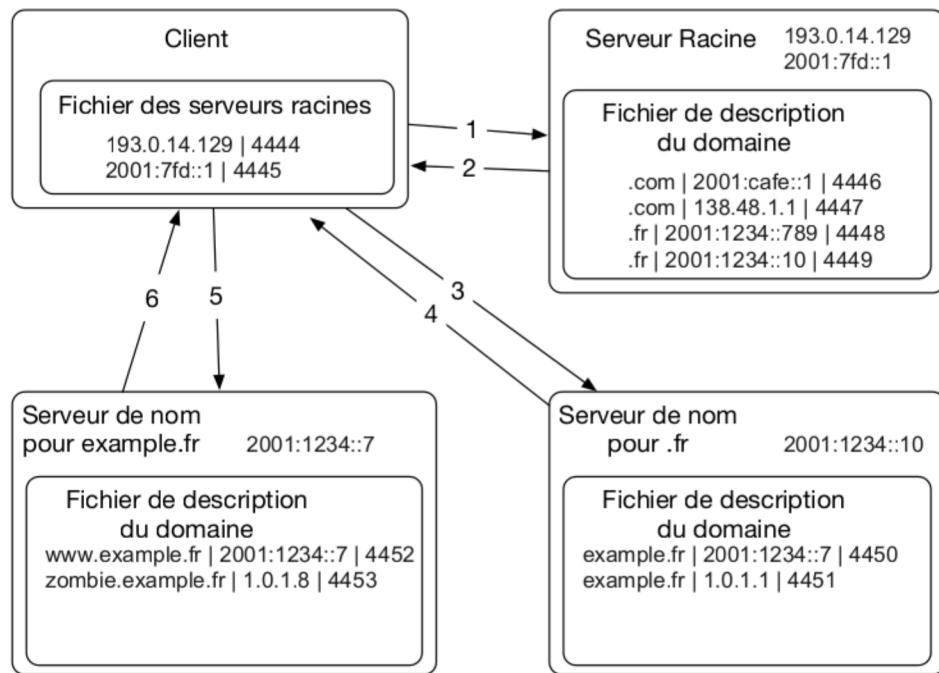
Le triplet identifiant, horodatage et nom est utilisé comme identifiant unique par le client. Si le client ne reçoit pas de réponse dans un temps donné, le client renvoie la requête. Lorsque plusieurs serveurs sont disponibles pour un même nom, la requête est envoyée à un autre serveur.

Le programme client prend en paramètre le chemin et nom vers le fichier contenant la liste des serveurs racines. Les serveurs de noms quant à eux reçoivent en paramètre la chemin vers un fichier contenant la liste des serveurs de sous-domaines et des noms de machines de leur domaine.

De plus, le client lira une liste de noms à résoudre depuis l'entrée standard. Vous permettrez également de charger les requêtes à résoudre depuis un fichier passé en second argument.

Dans le client, vous utiliserez l'horodatage pour mesurer la performance des différents serveurs contactés. Le programme client envoie les requêtes pour un même domaine sur différents serveurs en suivant la méthode du tourniquet. Si le délai aller-retour vers un serveur est grand, on éliminera ce serveur pour les requêtes suivantes. Il ne s'agit pas d'envoyer une requête sur tous les serveurs pour chaque résolution de nom. Par contre, des résolutions de noms successives sont envoyées sur des serveurs différents afin d'équilibrer la charge sur les serveurs et obtenir des mesures de performances. Vous décrirez dans le rapport comment vous avez choisi de calculer les performances, détecter les pannes de serveurs, équilibrer la charge et tester des serveurs nouvellement accessibles.

Figure 1, décrit un exemple de déploiement avec échange pour la résolution du nom `zombie.example.fr`



1. 1 | 1601674366 | www.zombie.example.fr
2. 1 | 1601674366 | www.zombie.example.fr |1|.fr, 2001:1234::789, 4448 | .fr, 2001:1234::10, 4449
3. 2 | 1601675432 | www.zombie.example.fr
4. 2 | 1601675432 | www.zombie.example.fr |1| example.fr, 2001:1234::7, 4450 | example.fr, 1.0.1.1, 4451
5. 3 | 1601675435 | www.zombie.example.fr
6. 3 | 1601675435 | www.zombie.example.fr |1| zombie.example.fr, 1.0.1.8, 4453

FIGURE 1 – Exemple d'échange de messages pour la résolution d'un nom de machine

2 Implémentation

Le projet est à réaliser sur les machines de la salle T40 (*Ubuntu18.04*) en langage C à l'aide des *sockets* (vous n'utiliserez pas d'autre mécanisme de programmation réseau).

Pour éviter les délais imposés par TCP lorsqu'un serveur est en panne, vous utiliserez exclusivement UDP et vous lancerez certaines requêtes en parallèle. Comme vous utiliserez UDP, vous prendrez soin de contrôler que les données sont correctement transférées.

Vous fournirez une série de tests afin de vérifier chacune des fonctionnalités demandées. Ces tests doivent notamment, mais pas exclusivement, comprendre le test de la méthode du tourniquet, l'élimination d'un serveur pour cause de délai important et la panne de serveurs de noms. Vous implémenterez ces tests au fur et à mesure afin de tester chaque fonctionnalité directement après son implémentation.

L'application doit fonctionner en IPv4 et IPv6. Il ne s'agit pas seulement de résoudre des noms de machines vers des adresses IPv6 et IPv4. Les serveurs de noms seront aussi joignables en IPv6 et IPv4.

3 Travail demandé

Il vous est demandé :

- de décrire votre implémentation et de justifier vos choix;
- de réaliser une documentation utilisateur, en décrivant les commandes du programme client et les arguments de lancement des serveurs de noms;
- de programmer les composants (client, serveurs de noms) de l'application en langage C sous Unix, avec les sockets;
- d'automatiser le lancement de la hiérarchie de serveurs de noms;
- d'automatiser les tests sur cette hiérarchie, de fournir les programmes de tests et de décrire les tests dans votre rapport;
- d'indenter et de commenter votre code.

4 Modalités de remise

Ce projet est à réaliser en **binôme ou individuellement**.

Vous déposerez sur Moodle, dans l'espace de devoirs prévus à cet effet, votre projet (documentation, sources, Makefile, ainsi que les fichiers et les programmes de tests) sous forme d'une archive au format « *login.tar.gz* » où *login* est votre nom de login sur Turing. Vous prendrez soin de supprimer tous les fichiers binaires (exécutables, objets).

Date limite : le lundi 16 novembre 2020 à 23 heures.

Sauf contre-ordre, une soutenance aura lieu les 30 novembre et 1er décembre, en fonction des groupes.