Rapport de modal INF474N : Internet over DNS

Raphaël Dang-Nhu et Anouk Paradis

Juin 2017

1 Documentation utilisateur

Côté client et côté serveur : se placer dans `bin` du dossier dezippé. Compiler le fichier .c avec :

```
g++ -fPIC -shared -I /usr/lib/jvm/java-1.8.0-openjdk-amd64/include/
 -I /usr/lib/jvm/java-1.8.0-openjdk-amd64/include/linux -o libtest2.so
 TestOuvertureFichier.c
```

en remplaçant

`/usr/lib/jvm/java-1.8.0-openjdk-amd64/`

par le chemin correspondant sur votre machine.

1.1 Côté client

Corriger le script client `configClient.sh` en remplaçant `enp2s0` par le nom de l'interface reliée au serveur. Lancer le script.
Compiler le code java. Le lancer avec :

```
java -cp "/home../lib/*:." -Djava.library.path=/home.../bin/ DNS_Client
```

en remplaçant `home..` par le chemin jusqu'au dossier dezippé. Pour enlever la configuration des interfaces, lancer le script `stopClient.sh`.

1.2 Côté serveur

Corriger le script serveur `configServer.sh`en remplaçant `enp1s0` par le nom de l'interface reliée au client et `wlp2s0` par l'interface connectée à internet. Lancer le script. Compiler le code java. Le lancer avec :

```
java -cp "/home../lib/*:." -Djava.library.path=/home.../bin/
DNSServerComplet
```

en remplaçant `home..` par le chemin jusqu'au dossier dezippé. Pour enlever la configuration des interfaces, lancer le script `stopServer.sh`.

2 Documentation développeur

Dans ce projet nous avons utilisé la librairie DNSjava (http://www.xbill.org/dnsjava/) pour envoyer des messages DNS.

2.1 Chemin d'un paquet IP

Un paquet IP venant d'une application du client est d'abord redirigé sur tun0, où il est lu par notre programme `DNS_Client`, "caché" dans une ou plusieurs requêtes DNS qui sont envoyées au serveur, sur son port 5959. Il est reçu par le programme `DNSServeurComplet` qui le retransforme en paquet IP, l'écrit sur tun1 d'où il est routé vers le net. Lorsqu'un paquet IP à destination du client revient vers le serveur, celui-ci est routé sur tun1, où notre programme le lit, le "cache" dans une ou plusieurs réponses aux requêtes DNS du client. Ces réponses sont reçues par le notre programme sur le client, retransformée en paquets IP et écrites sur tun0, d'où elles sont retransmises à la bonne application.

2.2 Architecture générale du programme

2.2.1 Côté client

Trois threads sont lancés :

- `ReaderThread` : ce thread lit les paquets IP, venant du client, sur tun0, et les met, sous forme de tableaux de bytes, sur la file bloquante `ipToSend`;
- `WriterThread` : ce thread récupère les paquets IP, reçus du serveur et traduits par le programme, sur la file bloquante `ipReceived` et les écrit sur tun0;
- `DNS_Client` : ce thread lance les deux threads précédents et leur donne les files partagées `ipReceived` et `ipToSend` en paramètre. Il choisit ensuite le serveur DNS. Ensuite, pour chaque paquet IP récupéré dans `ipToSend`, il le transforme en une ou plusieurs requêtes DNS et le envoie au serveur. Pour chacune des requêtes qu'il envoie, il récupère les réponses du serveur, les re-traduit en paquets IP et place ceux-ci dans `ipReceived`.

2.2.2 Côté Serveur

Quatre threads sont lancés :

- `ReaderThread` : ce thread, le même que celui côté client, lit les paquets IP, venant d'internet à destination du client, sur tun1, et les met, sous forme de tableaux de bytes, sur la file bloquante `ipPReceivedFromInt`;
- `WriterThread` : ce thread, le même que celui côté client, récupère les paquets IP, reçus du client sous forme de requête DNS et traduits par le programme, sur la file bloquante `ipPReceivedFromClient` et les écrit sur tun1;
- `ClientCommThread` : ce thread reçoit les requêtes DNS du client, traduit en paquets IP et les place sur la file `ipPReceivedFromClient`. S'il y a des paquets IP à envoyer sur `ipPReceivedFromInt` il les traduit alors en réponse aux requêtes du client et les lui envoie. S'il n'y a rien à envoyer, il envoie une réponse vide;
- `DNS_Server` : ce thread lance les trois threads précédents et leur donne les files partagées `ipPReceivedFromInt` et `ipPReceivedFromClient` en paramètre.

2.2.3 Classes annexes

Le code contient trois classes de plus. La classe `Encoder` contient les fonctions permettant de convertir entre les base 256 et 52 et entre les bases 10 et 52. La classe `TransfoDNSIP` contient des méthodes statiques qui permettent de convertir les paquets IP en requêtes DNS,

d'envoyer celles-ci et de traduire leurs réponses en paquets IP. Elle contient aussi des méthodes qui permettent de récupérer les paquets IP "cachés" dans une requête DNS et de répondre à cette requête, en cachant d'autres paquets dans la réponse. Enfin la classe `IncompleteIPStr` définit la structure de données permettant de stocker des morceaux de paquets IP incomplets reçus, afin de pouvoir les reconstruire quand l'ensemble du paquet a été reçu.

2.3 Transformation paquets IP - messages DNS

2.3.1 Vers des messages DNS

Les paquets IP sont tout d'abord transformés en chaîne de caractères a-z, A-Z. Pour cela, on considère le tableau d'octets qui les représente comme un nombre en base 256, on convertit ce nombre en base 52, puis l'on représente ce dernier avec les "chiffres" a-z, A-Z. Cela fait disparaître les éventuels octets à zéro au début du paquet, on rajoute donc 2 lettres au début de la chaîne, représentant le nombre de zéros en tête de paquet.
Si la chaîne de caractères est trop longue pour être transmise en un seule requête ou réponse DNS, on la découpe en plusieurs chaînes plus courte. On préface ensuite chaque chaîne par 6 lettres :

- 2 lettres pour l'identifiant du paquet;
- 2 lettres pour la taille du paquet (en nombre de chaînes de caractère, ie de requêtes/réponses DNS envoyées);
- 2 lettres pour l'identifiant du morceau de paquet.

(A chaque fois, on utilise 2 lettres pour représenter le nombre en base 52).
Ensuite, les chaînes obtenues sont mises au format d'une adresse (ajout de . et nom de domaine du serveur) ou laissées telles quelles et mises dans un record de type TXT de la section réponse d'un paquet DNS.

2.3.2 Vers des paquets IP

On récupère les chaînes de caractères placées dans le message DNS. On les parse ensuite pour obtenir un `IncompleteIPStr`. Celui-ci contient l'identifiant du paquet, sa taille (en nombre de messages DNS), l'identifiant du morceau de paquet et la chaîne de caractère (sans les 6 caractères préfixes). On stocke ces `IncompleteIPStr` dans une `Map<Integer,ArrayList<IncompleteIPStr>>`. La clé est l'identifiant du paquet. Lorsque la liste associée à un identifiant de paquet contient tous les morceaux, on la retire de la map et on reconstruit le paquet IP complet (en repassant la chaîne de caractère de base 52 en base 256).