Année 2020-2021

Rapport de projet

Projet de Résolution de noms

Projet réalisé par

Yassine MALIKI
Abdelhak BENBASSOU

Projet encadré par

Cristel Pelsser

Introduction :

Dans le cadre de notre troisième année en licence informatique à l'UFR de Strasbourg, il nous est proposé un projet de 1 mois nous permettant de mettre en pratique nos connaissances et nos compétences au travers d'un projet de résolution des noms de domaines (DNS) en langage C.

Nous avons saisi donc cette opportunité pour soumettre l'ébauche d'un projet personnel à notre professeur Cristel Pelsser.

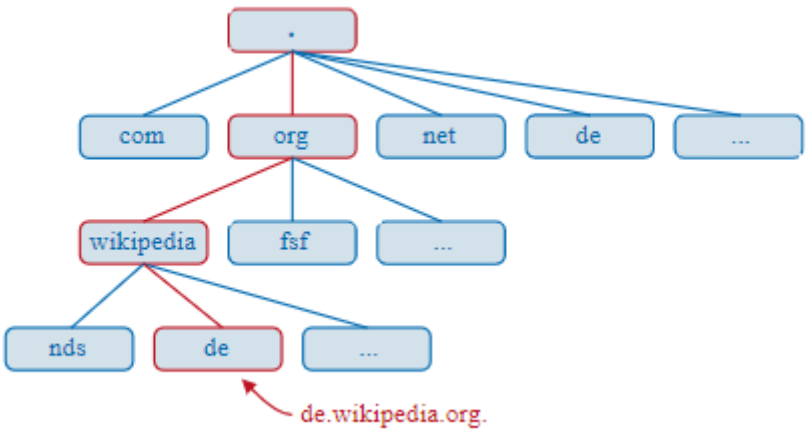
Ce projet étant personnel, notre première mission fut de définir nous-mêmes une stratégie ainsi que les objectifs à atteindre. Bien entendu, cette stratégie a évolué au cours du temps afin de satisfaire nos exigences, mais aussi les contraintes auxquelles nous avons fait face. Nous avons ainsi tissé notre réflexion sur la façon d'architecture et d'implémentation la plus adaptée, après avoir bien assimilé le principe du DNS qui peut se résumer ainsi :

1. Présentation du Principe DNS (Domain Name System) :

Pour faciliter la recherche d'un site donné sur Internet, le système de noms de domaine (DNS) a été inventé. Le DNS permet d'associer un nom compréhensible, à une adresse IP. On associe donc une adresse logique, le nom de domaine, à une adresse physique l'adresse IP.

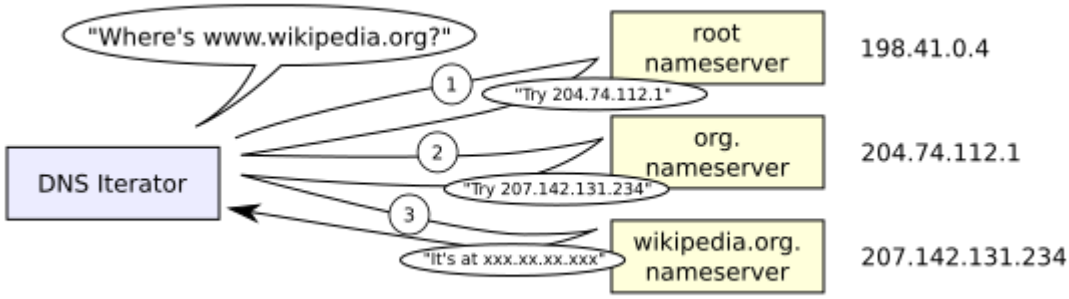
Le nom de domaine et l'adresse IP sont uniques. Le DNS permet au message d'un internaute d'atteindre son destinataire et non quelqu'un d'autre possédant un nom de domaine similaire. Il permet également de taper « www.google.com » sans avoir à saisir une longue adresse IP pour pouvoir accéder au site web approprié.

Résoudre un nom de domaine consiste donc à trouver l'adresse IP qui lui est associée.



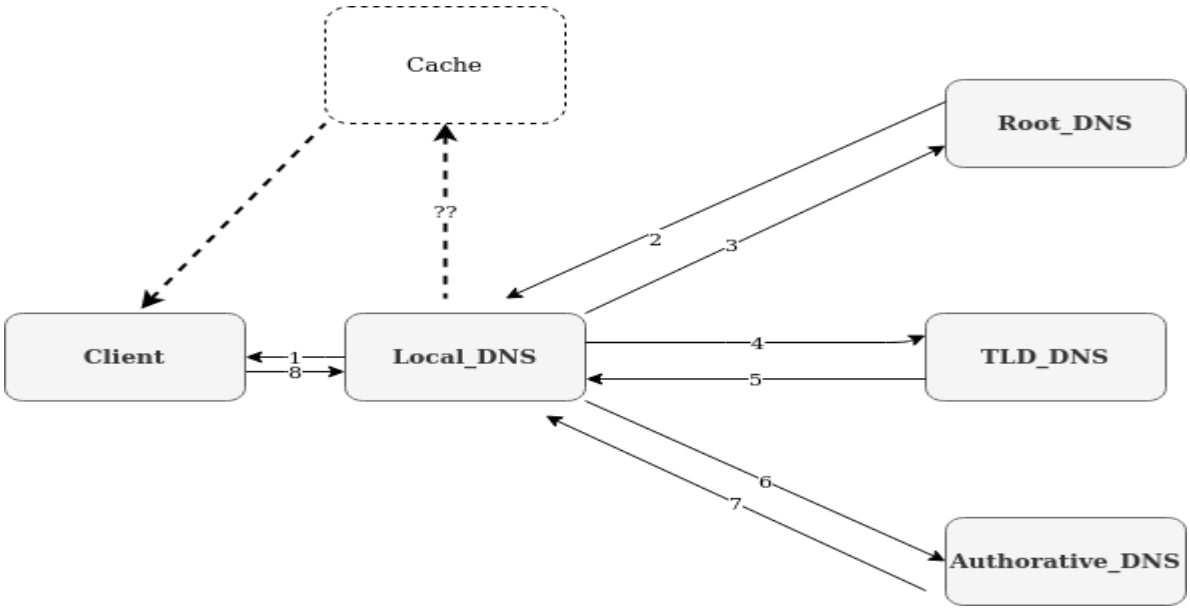
Le système des noms de domaines est sous la forme d'une hiérarchie dont le sommet est appelé la racine. On représente cette

dernière par un point. Dans un domaine, on peut créer un ou plusieurs sous-domaines ainsi qu'une délégation pour ceux-ci, c'est-à-dire une indication que les informations relatives à ce sous-domaine sont enregistrées sur un autre serveur. Ces sous-domaines peuvent à leur tour déléguer des sous-domaines vers d'autres serveurs.



2. Architecture générale :

En s’inspirant de l’architecture du système DNS, notre projet suivra l’architecture suivante pour la résolution de noms (domaine ‘.com’) :



En effet, nous avons ajouté un système de cache comme plus-value pour notre projet. Nous avons alors opté pour la séparation du Client en un serveur client et un serveur Local_DNS pour la clarté du code.

3. Fonctionnement général du projet

Tout commence par le client qui charge en paramètre le ou les noms de domaine depuis un fichier texte qu'on a nommé "domain.txt", puis construit la forme de requête comme demandé dans le sujet, et l'envoie au Local_DNS. Ce dernier vérifie l'existence du nom du domaine dans le cache. Dans le cas de réussite, il renvoie directement l'IP au Client, ce qui permet une résolution rapide et presque instantanée. Dans le cas où il ne le retrouve pas, il communique alors de manière itérative avec les serveurs Root_DNS, ensuite pour le domaine '.com' TLD_DNS, puis Authoritative_DNS et pour le domaine '.in' TLD_in, puis Authoritative_in.

Plus de détails:

2.1. Client :

Le client se connecte, à l'aide de sockets, au serveur local qui sera le premier serveur de la hiérarchie, après avoir récupéré le nom de domaine. Puis reste en attente de réponse, soit de la part du cache, si le domaine est déjà dans le cache , ou du Local_serveur dans le cas contraire. Bien évidemment cette attente n'est pas active. Après une durée déterminée sans recevoir de réponse, un timeout est déclenché grâce à la fonction **`setsockopt`**.

2.2. Local Server :

Ce serveur récupère le nom de domaine du client. Il vérifie si le domaine existe ou pas dans le fichier cache "localcache.txt". Ce dernier est de la forme `<domaine 1>|<IP1>`. Si le domaine en question existe déjà dans ce fichier, l'IP est directement retourné au Client. Il nous

permet ainsi de retrouver l'IP plus rapidement (principe d'anticipation) la prochaine fois, sans avoir à reconsulter le reste de la hiérarchie. Dans le cas où le nom de domaine n'existe pas, le serveur construit la première requête de la forme **<identifiant de transaction>|<horodatage>|<nom>.** Puis, il adresse cette requête au serveur Root et attend sa réponse de la forme :

<identifiant_de_transaction>|<horodatage>|<nom>|\<code>|<domaine>,<IP1>,<port1>|...|<domaine>,<IPn>,<portn>.

Puis il contacte le serveur TLD et termine par le serveur authoritative. Ce dernier retourne l'adresse IP. Le serveur Local finit par envoyer cette IP au Client.

2.3. Root Server :

Il possède 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 suivants.

Dans notre cas, ce serveur charge le fichier "rootdnfile". Ce dernier est de la forme "extension IP port". Ainsi, en fonction de la requête reçue, le serveur extrait l'extension. Par exemple, si l'on demande www.abcdef.com au serveur du domaine .com, il ne pourra retourner que l'adresse du DNS de la zone abcdef.com. Si l'extension n'est pas présente dans le fichier chargé, l'adresse IP "1.0.0.0" est renvoyée au serveur local pour témoigner de présence d'erreur. Si l'extension existe, le serveur renvoie une réponse au serveur local de la forme:

<identifiant de transaction> | <horodatage> |<nom>|\<code>|<extension>,<IP>

Pour plus de sécurité et clarté, le serveur renvoie aussi le numéro du port et aussi l'IP du sous-serveur correspondant.

2.4. TLD Server :

Après la réception de la réponse du serveur Root, le serveur local **interroge le serveur de domaine de premier niveau (TLD)** et ne reçoit pas encore de réponse définitive, mais la demande est à nouveau

transmise : les serveurs de domaine de premier niveau n'ont qu'une fonction de transfert. Vous informez le serveur demandeur sur quel serveur DNS autoritaire le nom de domaine recherché est consigné. Et pour ce faire, le serveur TLD charge le fichier "tldcom" ou "tldin". Cela dépend évidemment de la réponse du serveur Root. Ainsi, ce fichier est de la forme : *sousdomaine.domaine IP PORT*. Et comme le serveur Root, le serveur TLD renvoie une réponse au serveur local de la forme: **<identifiant de transaction> | <horodatage> |<nom>|\<code>| <sous-domaine.domaine>,<IP, <port>.**

Pour plus de sécurité et clarté, le serveur renvoie aussi le numéro du port et aussi l'IP du sous-serveur correspondant. Ceci peut servir dans le cas où la première requête rencontre des problèmes et n'arrive pas au serveur Local.

Ceci est dans le cas où l'extension est '.com' dans les cas d'un domaine '.in' on utilisera le server TLD_in qui exécute le même principe.

2.5. Authoritative Server :

A ce stade, le serveur local se tourne désormais vers le **serveur faisant autorité qui est responsable du nom de domaine** et obtient finalement l'adresse IP désirée. Ce serveur enregistre aussi l'IP recherchée dans le fichier cache pour une utilisation future.

Ceci est dans le cas où l'extension est '.com' dans les cas d'un domaine '.in' on utilisera le server Authoritative_in qui exécute le même principe.

Implémentation supplémentaire et plus-value :

Comme dans le cas d'un DNS réel on a ajouté un système de cache, pour réduire la 'charge' sur les serveurs du système de noms de domaine en mettant les résultats en cache localement.

Les résultats obtenus à partir d'une requête DNS sont toujours associés à la durée de vie (TTL), un délai d'expiration après lequel les résultats doivent être supprimés ou actualisés. Le TTL est défini par l'administrateur du serveur DNS. La période de validité peut varier de quelques secondes à quelques jours voire semaines.

Pour faciliter l'implémentation, le cache sera effacé manuellement avec la commande : make cache_reset
Et il est aussi effacé implicitement à chaque compilation des codes du projet avec la commande : make.

Makefile :

Pour simplifier la procédure de compilation, et dans le but d'automatiser le projet et par conséquent automatiser la résolution de noms, nous avons mis en place un fichier Makefile. Ce dernier compile les fichiers C, supprime les fichiers exécutables générés après la compilation, vide le cache, lance les terminaux des différents serveurs d'un seul coup et lance automatiquement la résolution des noms de domaines présents dans le fichier "domain.txt".

Fonctionnement du programme :

Pour lancer le programme, Il faut se placer dans le répertoire du projet, lancer la commande "make" qui va compiler les différents codes sources, supprimer le cache si déjà présent et va générer les exécutables des différents serveurs. Puis grâce à la commande "make open" les différents terminaux associés aux serveurs seront lancés automatiquement. La commande "make clean" permet de supprimer les exécutables.

Un fichier "domain.txt" est mis à disposition. Il contient les différents noms de domaine dont on veut la résolution. Il contient déjà deux noms de domaine qu'on veut résoudre. Ils sont suffisants pour tester, mais si vous en ajoutez un, pensez à ajouter le nom de domaine

saisi et son IP dans le fichier “authdns.txt” si l’extension est “com”, ou “authin.txt” si l’extension est “in” sous la forme <domaine> <IP>

Pensez aussi à vider le cache (fichier localcache.txt), afin de voir l'exécution de tous les serveurs. Si vous laissez le cache rempli, le serveur Local renvoie directement l’IP au serveur Client sans consulter les différents serveurs !

Une fois l'exécution faite (Rappel: make -> make open), les différents terminaux s’ouvrent et lancent les serveurs. Vous pouvez savoir chaque terminal représente quel serveur à travers la première ligne de chaque terminal. Ainsi, le serveur client lit le fichier “domain.txt” qui renferme les noms de domaines qu’on veut résoudre. Pour chaque nom de domaine, on attend la réponse du serveur Local avant de passer à la ligne suivante. Les autres serveurs restent en attente de requêtes de la part du serveur local. Cette attente est conditionnée par un timeout qui déclenche une interruption si jamais le serveur trop longtemps en attente. A chaque requête/réponse, la taille du message échangé est affichée afin de voir l’origine d’une éventuelle perte de données.

Nous avons veillé à ce que le code soit clair et lisible afin de comprendre davantage le fonctionnement du système. Des commentaires avant chaque début de bloc sont mis à disposition pour cerner la composition du programme.

Ainsi, comme demandé, des fichiers de tests sont mis à disposition dans le répertoire tests. Ils permettent de voir si la compilation est réussie, si les fichiers txt sont de la forme demandée et si les résultats reçus par le client sont corrects.

Références :

https://openclassrooms.com/fr/courses/857447-apprenez-le-fonctionnement-des-reseaux-tcp-ip/857163-le-service-dns

https://www.pofilo.fr/post/20180630-dns-unbound/

https://en.wikipedia.org/wiki/Domain_Name_System

https://www.geeksforgeeks.org/udp-client-server-using-connect-c-implementation/

https://assiste.com/Principe_d_anticipation.html

C. Pelsser, J.-R. Luttringer, and N. Pierre, “Cours, tp et td d’Algorithmes Des Réseaux.” - Université de Strasbourg, 2020.