PR6 – Programmation réseaux
TP n° 8 : Un client DNS minimal

Le protocole DNS (*Domain Name server*) permet de traduire les noms de domaine Internet en adresse IP. Le but de ce TP est de faire un client qui, comme les utilitaires `dig` ou `host` (mais en plus léger), permet d'afficher l'adresse IP d'une machine dont le nom est passé en paramètre [1]. Le protocole DNS utilise des datagramme UDP. Chaque message est formé de 5 régions :

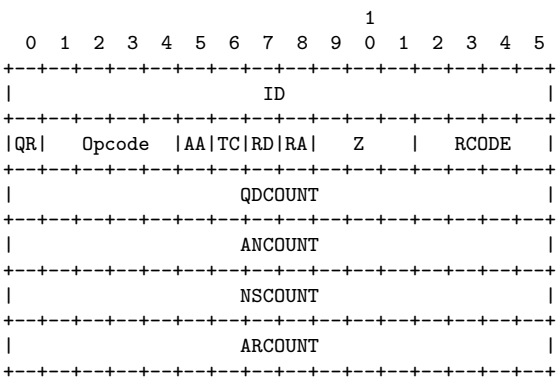
1. entête
2. question(s)
3. réponse(s)
4. données d'autorité
5. données additionnelles

Chacune des régions de 2 à 5 peut contenir plusieurs entrées. L'entête donne le nombre d'entrées de chaque région. Notre client doit

1. Construire une requete DNS contenant un nom de domaine (passé en ligne de commande)
2. l'envoyer au serveur DNS, en IPv4, via UDP
3. recevoir la réponse
4. en extraire l'adresse IP et l'afficher

Entête

L'entête (utilisée par les questions comme les réponses) a le format suivant (RFC 1035) :



Les champs sont définis comme suit :

- *ID* est une suite de 2 octets arbitraires qui permet d'associer requêtes et réponses. Il est tiré au sort pour la requête. Celui de la réponse doit correspondre, mais ce n'est pas à vérifier ;
- *QR* vaut 0 pour une requête, 1 pour une réponse ;
- *Opcode* vaut 0 pour une requête standard et sera ignoré dans la réponse ;
- *AA* doit être 0 pour la requête et vaut 1 dans la réponse si elle fait autorité, 0 sinon (on peut l'ignorer dans une réponse)
- *TC* vaut 0 dans la requête. S'il vaut 1 dans la réponse, il y a eu troncation, ce que notre client ne traitera pas (dans ce cas, afficher une erreur),

1. D'ailleurs, un petit client DNS est contenu dans à peu près toutes vos applications réseau, dont il est nécéssaire au fonctionnement : navigateur Web, client mail...

- *RD* vaut 1 dans la requête (récursivité désirée), et sera ignoré dans la réponse ;
- *RA* et *Z* vaudront 0 pour une requête, et seront ignorés dans la réponse ;
- *RCODE* est un code d'erreur, il vaudra 0 dans une requête et, dans une réponse, vaut :
  - 0 : No error (donc on peut traiter la suite)
  - 1 : Format error (vous avez mal fait le TP, débuggez)
  - 2 : Server failure
  - 3 : Name Error (le domaine n'existe pas. Avertir l'utilisateur dans ce cas !)
  - 4 : Not Implemented
  - 5 : Refused

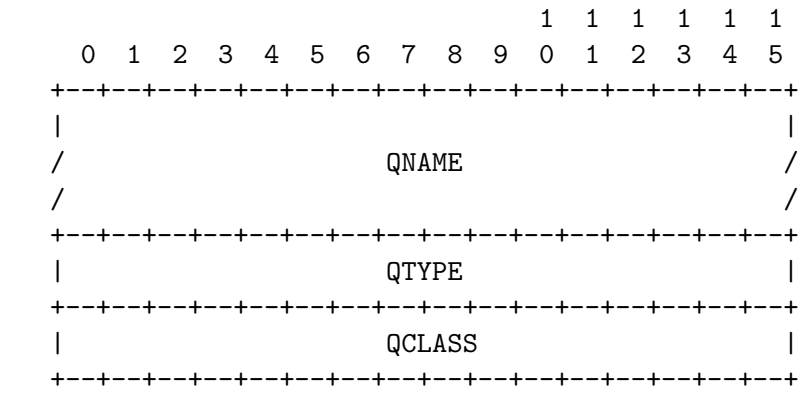
  les quatres autres champs indiquent respectivement le nombre de questions, réponses, données d'autorité et données supplémentaires qui suivent l'entête.

Exercice 1 : écriture d'une requête

Il faut composer un message consistant seulement en une entête et une entrée de requête (QDCOUNT = 1). Ce message pourra être composé soit avec une ou plusieurs structures remplie champ par champ, soit avec un `char *msg` que l'on remplira par des `memcpy` et autres opérations semblables [2].

Pour faire court, tous les champs de l'entête doivent être à 0 sauf ID, RD et QDCOUNT. Attention, comme pour presque tous les protocoles réseau, les champs de plus d'un octet doivent être en *network order* (gros boutistes).

L'entête sera suivie d'une requête unique. Elle suit ce format :



Le champ QNAME est une suite de chaînes, chacune préfixée par leur longueur (sur un octet), et se termine par un O représentant la chaîne nulle. Le nom de domaine est décomposé en mots selon les points et chaque mot est précédé de sa longueur codée sur 8 bits. Ainsi l'adresse `www.wikipedia.fr` donne quatre chaînes, y compris la chaîne nulle finale :

03	77	77	77	09	77	69	6b	69	70	65	64	69	61	02	66	72	00

Ici chaque case représente un nombre hexdécimal sur 8 bits, les cases bleues sont les longueurs des mots et les autres cases les lettres de chaque mot. Notez que contrairement à ce que le schéma laisse à penser, la longueur de QNAME peut être impaire, et les champs suivants commenceront donc à des offsets impairs.

`QTYPE` est l'enregistrement demandé. Il vaut 1 pour un enregistrement `A` (adresse IPv4) et 28 pour `AAAA` (adressse IPv6).

2. cette seconde option est plus facile mais moins élégante

`QCLASS` doit valoir 1 (pour le domaine Internet). Là encore, ces deux champs sont bien sûr en notation big-endian.

Vous observerez que la longueur du message n'est pas constante mais dépend de celle du nom de domaine recherché.

Exercice 2 : envoi de la requête

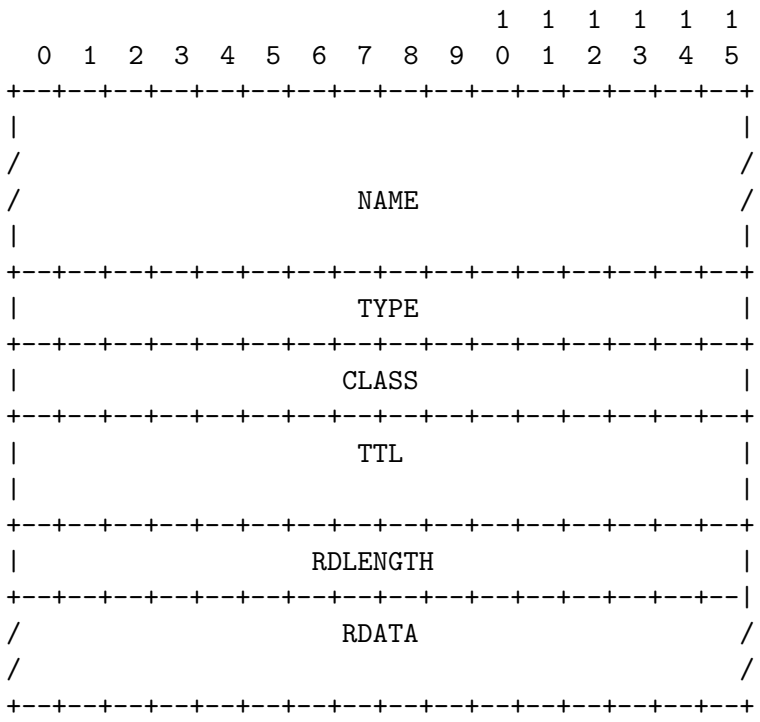
Envoyer par UDP, en IPv4, le message qu'on vient de composer au serveur DNS. L'adresse IP de celui-ci se trouve dans `/etc/resolv.conf` sur une machine Unix. Vous l'obtenez aussi avec `dig`. Elle peut être mise en dur dans le code ou prise en ligne de commande. Le port UDP du service DNS est 53.

Exercice 3 : réception et affichage de la réponse

Après l'envoi, attendre une réponse (un seul paquet UDP) puis l'analyser pour en extraire l'adresse IP demandée. Il faut pour cette analyse :

- d'abord analyser l'entête pour voir s'il y a eu une erreur et, le cas échéant, l'afficher et terminer. Attention on parle des erreurs du protocole DNS, mais toute erreur survenant en couche plus basse (`recvfrom`, etc.) doit aussi être affichée !
- puis lire combien d'entrées de requêtes et de réponse il y a. En effet, une réponse DNS contient usuellement sa question. On va l'ignorer mais le problème est qu'on ne sait pas où commence la réponse !
- il faut donc connaître la taille de la zone de requête. Pour cela, il faut lire séquentiellement le champ QNAME de chaque entrée pour savoir sa longueur et donc continuer le parsing ;
- enfin on arrive dans la zone de réponse elle-même. Celle-ci est donnée dans le format décrit ci-dessous. Pour chaque entrée, extraire l'adresse, la convertir avec `inet_ntop` et l'afficher ;
- on ignorera les autres champs (données d'autorité et additionnelles) de la réponse.

Le format d'une entrée de réponse est le suivant :



- NAME est le nom de domaine. Attention, il peut être compressé, voir ci-dessous !
- TYPE et CLASS sont comme QTYPE et QCLASS d'une requête ;
- TTL est le nombre de secondes de validité de la réponse ;
- RDLENGTH est la longueur de RDATA, normalement 4 ou 16 pour IPv4 et IPv6 ;
- enfin RDATA est, pour une réponse de type A ou AAAA, l'adresse demandée.

Exercice 4 : gestion (minimale) des noms de domaine compressés

Une difficulté de lecture de la réponse vient du fait qu'elles peuvent utiliser une compression de nom. Dans les champs QNAME et NAME du message de réponse, si une longueur a ses deux premiers bits à 0 (donc est <64) c'est bien une longueur de chaîne, comme décrit exercice 1. Mais si les deux premiers bits sont à 1, alors les 14 bits qui suivent sont un pointeur OFFSET vers une chaîne existante. La RFC 1035 spécifie trois formats possibles pour QNAME ou NAME :

- Une suite de longueur+chaîne terminant par un octet nul ;
- Un pointeur du type décrit ci-dessus ;
- Une suite de longueur+chaîne terminant par un pointeur.

Donc, si votre serveur DNS compresse les noms des réponses (par exemple en pointant dans la zone de question) il faut gérer de façon minimale ce mécanisme, non pas pour lire la réponse, mais juste pour calculer correctement la longueur des champs QNAME et NAME, afin de savoir où commencent les autres.

Exercice 5 : Requêtes AAAA et multiples

1. Vous avez fait une requête A. Gérer aussi (grâce à une option en ligne de commandes) les requêtes AAAA.

2. Votre programme pour l'instant ne pose qu'une seule question à la fois. Composer une seule requête comprenant deux entrées (deux questions) pour le même domaine : une A et une AAAA, pour avoir les IP des deux versions.

Attention, beaucoup de serveurs DNS n'implémentent pas bien le protocole DNS ! Ainsi le serveur DNS de l'UFR (IP 192.168.70.200) envoie un message d'erreur 1 quand il y a une requête de deux questions... correctement écrite ! Celui de Google (IP 8.8.8.8) ne répond qu'à la première question. Celui de free (IP 212.27.40.240) répond correctement. D'autres ne répondent même pas et `recv` attend en vain...