Université Paris Diderot – Paris 7 **Programmation réseau**
L3 Informatique **Anne 2010-2011, 2ème semestre**

TP n°4

Routage et Protocole SNTP

1 Tables de routage

Exercice 1 [Lire les tables de routage]

1. En utilisant la commande `ip route show` qui vous donne les tables de routage depuis votre machine déterminez l'adresse IP de la passerelle (*gateway*) qui est utilisé lorsque vous vous connectez à internet depuis les machines de la salle de tp.
2. Trouvez en faisant `ip route get` la route allant vers une destination donnée (par exemple `google.com`).
3. Reprenez la question 1. avec l'option `-6` après `ip`) pour obtenir la table de routage au format IPv6.

2 Protocole SNTP

Introduction

Le protocole NTP (Network Time Protocol) permet de régler avec une grande précision l'horloge d'un hôte à partir d'une ou plusieurs sources de temps distantes. NTP est défini dans le RFC 1305, un document de 120 pages qui inclut une discussion détaillée des algorithmes de filtrage que les clients NTP doivent employer.

Pour beaucoup d'applications, la complexité de NTP n'est pas nécessaire. RFC 2030 définit SNTP (Simple Network Time Protocol), une version simplifiée de NTP qui ne définit que le protocole et pas les algorithmes à employer. Tout client ou serveur NTP est *a fortiori* un client ou serveur SNTP, mais l'inverse n'est pas vrai.

Le but de ce TP est d'implémenter un client SNTP.

2.1 Le protocole

Lorsqu'il détermine que c'est nécessaire, le client SNTP envoie au serveur une requête sous forme d'un paquet UDP qui contient la date à laquelle ce message est transmis. Le serveur répond (aussi vite que possible) avec une réponse qui contient quatre dates :

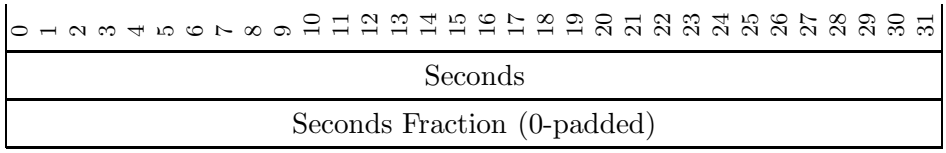
– la date à laquelle la requête a été transmise (selon le client) ;
– la data à laquelle la requête a été reçue (selon le serveur) ;
– la date à laquelle la réponse a été transmise (selon le serveur) ;
– la date à laquelle l'horloge du serveur a été réglée pour la dernière fois.

En outre, la réponse contient un certain nombre de données que nous ne considérerons pas dans ce TP, et qui permettent à NTP (mais pas nécessairement SNTP) d'avoir une idée de la précision des résultats.

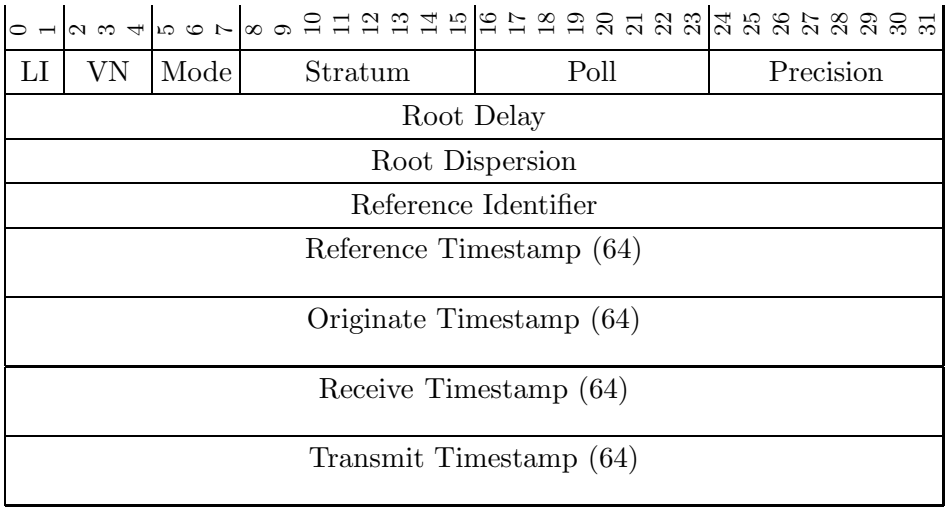
On remarquera que la réponse contient la date que le client avait incluse dans la requête ; ceci permet au client d'identifier une réponse comme allant de pair avec une requête donnée, et donc de contourner les problèmes dûs à la nature non-fiable du transport.

2.1.1 Format des données

Date NTP Une date NTP est représentée sous la forme d'un entier de 64 bits exprimant un temps, en unités de 2^{-32} secondes, écoulé depuis 0 h le 1er janvier 1900. RFC 2030 définit le format d'une telle date comme suit :



Message NTP Les requêtes et les réponses NTP ont le format suivant :



Les champs d'un tel message sont définis comme suit :
– *LI* sera ignoré dans ce TP, et vaudra toujours 0 ;
– *VN* est la version du protocole employée, et vaudra toujours 3 ;
– *Mode* est le type de message, et vaudra 3 pour une requête, 4 pour une réponse ;

– *Stratum*, *Poll* et *Precision* seront ignorés dans ce TP ;
– *Reference Timestamp* sera ignoré dans ce TP ;
– *Originate Timestamp* vaut 0 pour une requête, et, pour une réponse, est la copie du *Transmit Timestamp* de la requête correspondante ;
– *Receive Timestamp* vaut 0 pour une requête, et pour une réponse, est la date de réception de la requête correspondate ;
– *Transmit Timestamp* est la date de transmission de ce paquet.

Le paquet peut être suivi de 160 octets de données supplémentaires, que nous ne transmettrons jamais, et qui seront ignorés lors de la réception.

2.2 Code fourni

Vous trouverez sur la page web des TP quelques fichiers C pouvant vous aider :
Le fichier `timestamp.c` permet de manipuler un timestamp NTP :
– `int ts_current(struct timestamp *ts)`
Initialise la structure avec l'heure courante.
– `char* tstop(struct timestamp *ts)`
Renvoie une chaîne de caractère représentant le timestamp.

Le fichier `ntp.c` permet de manipuler le contenu d'un message NTP.
– `int init_req(struct ntp_msg* msg, struct timestamp *current)`
Initialise un message NTP avec le timestamp passé en argument.
– `fprint_ntp_msg(FILE* f, struct ntp_msg* msg)`
Affiche le contenu d'un message NTP.
– `void ntp_msg_read(struct ntp_msg *dest, unsigned char* buf, int offset)`
– `ntp_msg_write(struct ntp_msg* src, unsigned char* buf, int offset)`

2.3 Exercices

Exercice 2 [Premier client NTP]

Écrivez en C un programme qui exécute les actions suivantes :

1. crée une requête NTP ;
2. envoie cette requête dans un paquet UDP à partir d'un port p vers le port *ntp* d'une machine munie d'un serveur NTP ;
3. attend une réponse sur le port UDP p ;
4. affiche le contenu de la réponse.

Pour de mauvaises raisons techniques, vous ne pouvez pas tester votre programme sur Internet, mais uniquement dans l'UFR. Que se passe-t-il si vous essayez par exemple `pool.ntp.org` ?

Exercice 3 [Découverte de serveur]

Certains supports physiques, comme le réseau Ethernet utilisé dans les salles machines de l'UFR, permettent de diffuser une trame à tous les ordinateurs qui

y sont connectés. Cette possibilité est re-exportée par la couche IPv4 à l'aide d'une adresse IP spéciale, appelée adresse de diffusion, ou adresse de *broadcast*[1]. Un paquet émis à destination de cette adresse sera reçu par tous les ordinateurs connectés sur le même lien physique. Ce paquet ne sera jamais retransmis par un routeur.

En envoyant une requête NTP à l'adresse *broadcast* de l'UFR (192.168.70.255), tous les serveurs NTP accessibles directement vous répondront.

Modifiez votre programme pour qu'il soit capable d'attendre plusieurs réponses. Vous pourrez supposer qu'après 500ms de silence, il n'y aura plus de réponse. **Attention :** pour pouvoir envoyer un paquet *broadcast*, il faut activer l'option `SO_BROADCAST` sur la socket.

Déterminez le nombre de serveurs NTP de l'UFR.

1. dans le cas d'IPv6, il n'existe plus d'adresse dite de *broadcast*. Ce mécanisme a été remplacé par le mécanisme plus général d'adresse *multicast*.