PR6 – Programmation réseaux
TP n° 10 : Un client SNTP

Introduction

Le protocole *Network Time Protocol* (NTP) permet de régler l'horloge d'un hôte à partir d'une ou plusieurs sources de temps distantes. NTP est défini dans les RFC 1305 (version 3) et 5905 (version 4), des documents de 120 pages chacun qui incluent 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. Les RFC 2030 et 4330 définissent le *Simple Network Time Protocol* (SNTP), 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 forcément vrai.
Techniquement, le protocole (S)NTP est un protocole pair-à-pair, mais on peut voir SNTP comme un protocole strictement client-serveur. Le but de ce TP est d'implémenter un client SNTP.

Protocole SNTP

Lorsqu'il décide que c'est nécessaire, le client SNTP envoie au serveur une requête sous forme d'un paquet UDP qui contient le temps auquel ce message est transmis. Le serveur répond avec une réponse qui contient quatre temps :

- le temps t_o auquel la requête a été transmise selon l'horloge du client ;
- le temps t_r auquel la requête a été reçue selon l'horloge du serveur ;
- le temps t_t auquel la réponse a été transmise selon l'horloge du serveur ;
- le temps auquel l'horloge du serveur a été réglée pour la dernière fois.

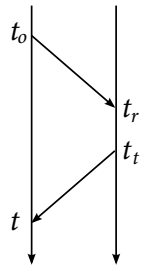
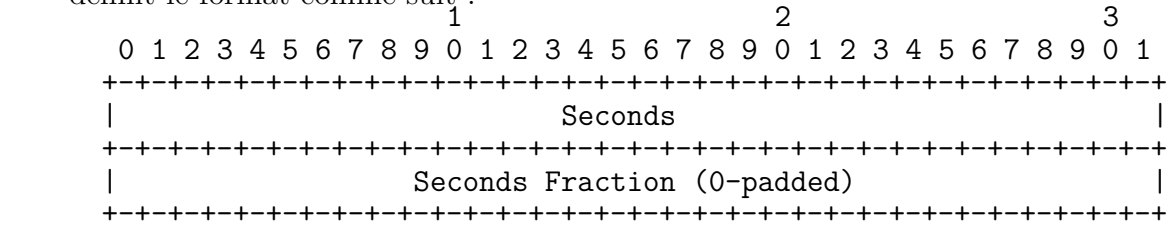


FIGURE 1 – Les temps associés à une réponse NTP

En outre, la réponse contient un certain nombre de données que nous ignorons dans ce TP, et qui permettent à NTP (mais pas nécessairement SNTP) d'avoir une idée de la précision des résultats.
Remarquez que la réponse contient une copie des données que le client avait incluses dans la requête. C'est une technique habituelle en UDP, qui permet au client de déterminer la requête qui va de pair avec une réponse donnée même en présence de paquets perdus.

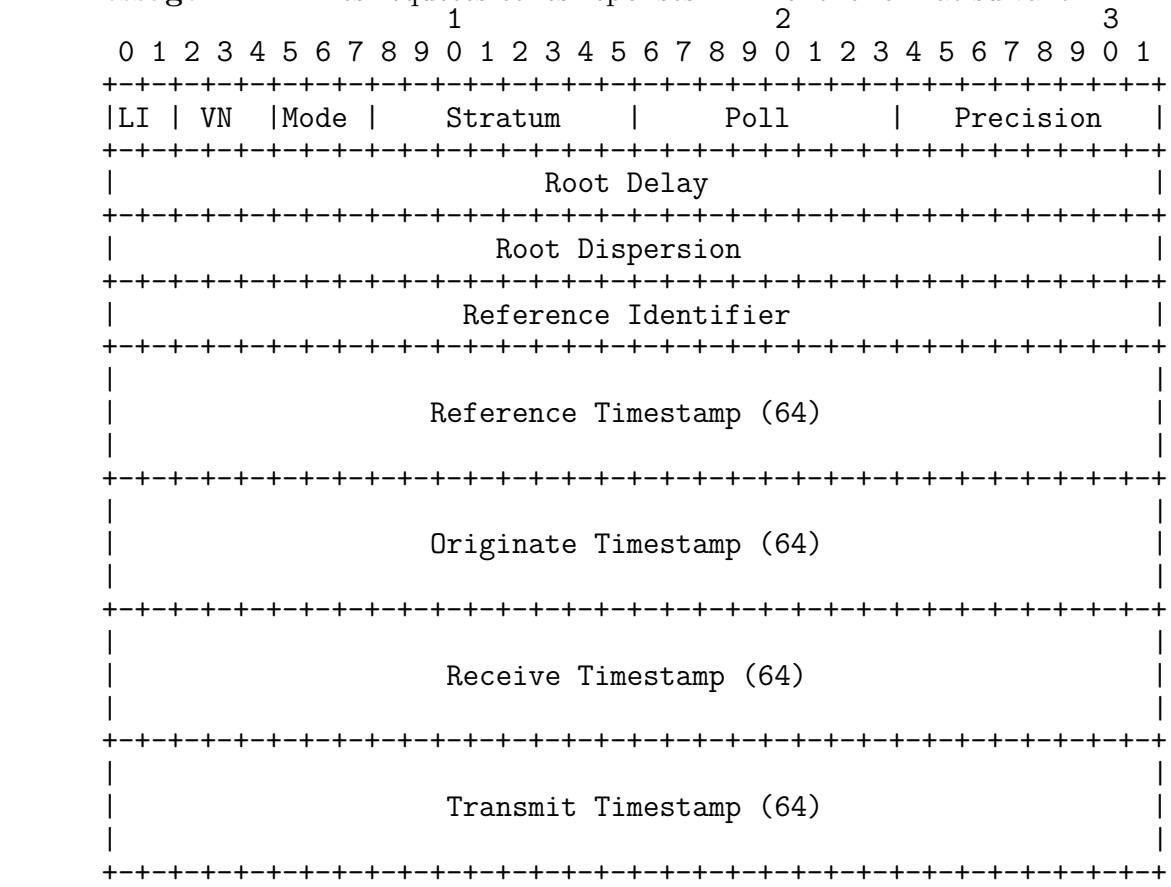
a) Format des données

Temps NTP Un temps NTP est le temps écoulé depuis 0 h le 1er janvier 1900. Il est représenté sur 64 bits, sous forme de deux entiers de 32 bits, le premier exprimant les secondes, et le second, des fractions de 2^{-32} secondes. On peut aussi le voir comme un nombre décimal à virgule fixe de 64 bits avec la virgule entre les bits 31 et 32. La RFC 4330 définit le format comme suit :



Attention, la représentation est gros-boutiste (*big endian*), alors que votre processeur est probablement *little-endian*.

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



Les champs d'un tel message sont définis comme suit :

- *VN* est la version du protocole employée, et pour une requête vaudra 3 en IPv4 (4 en IPv6) ;
- *Mode* est le type de message, et vaudra 3 pour une requête, 4 pour une réponse ;
- *Reference Timestamp* est le temps auquel l'horloge de l'émetteur a été mise à jour ; il vaudra 0 dans les requêtes que nous ferons ;

- *Originate Timestamp* (le t_o de la figure 1) est la copie du *Transmit Timestamp* de la requête correspondant à une réponse ; il vaudra 0 dans les requêtes que nous transmettrons ;
- *Receive Timestamp* (le t_r de la figure 1) est le temps auquel a été reçue la requête correspondant à une réponse ; il vaudra 0 dans les requêtes que nous transmettrons ;
- *Transmit Timestamp* (le t_t de la figure 1) est le temps auquel a été transmis le paquet.

Les autres champs vaudront 0 dans une requête, et seront ignorés dans les réponses.
Le paquet peut être suivi de 160 octets de données supplémentaires, que nous ne transmettrons jamais, et qui seront ignorés lors de la réception.

b) Conversion d'heures

Le format des temps utilisé sous Unix, et retourné par des fonctions comme `clock_gettime` et `timespec_get`, est différent de celui utilisé par NTP : sous Unix, un temps est stocké comme un nombre de secondes et nanosecondes (microsecondes dans les anciennes API) depuis le 1er Janvier 1970.
On peut convertir un temps Unix (`sec`, `nsec`) en temps NTP (`seconds`, `fraction`) à l'aide des équations suivantes :

```
seconds = sec + 0x83aa7e80;
fraction = nsec * 4295 / 1000;
```

La conversion inverse peut se faire à l'aide des équations suivantes :

```
sec = seconds - 0x83aa7e80;
nsec = fraction * 1000 / 4295;
```

c) Conversion de boutisme

Par ailleurs, `sec` et `nsec` étant représentées par des types entier [1], et les *timestamps* NTP aussi, il faudra faire attention au « boutisme ». Pour la portabilité de votre code, il faudra les convertir. Vous pouvez utiliser `htonl`/`ntohs` ou leur équivalent 64 bits `htobe64`/`be64toh` si votre système le supporte : voyez `man endian`.

I) Un client NTP

Écrivez un programme C qui exécute les actions suivantes :

1. formate une requête NTP dans un tampon ;
2. envoie cette requête dans un paquet UDP à partir d'un port p vers le port 123 d'une machine munie d'un serveur NTP ;
3. attend une réponse sur le port UDP p ;
4. affiche le contenu de la réponse sous une forme lisible par un être humain normal.

Quelques remarques :

1. 32 bits sur les anciennes distributions, 64 sur les nouvelles, pour éviter le bug de l'an 2038

– Vous pourrez obtenir l'heure courante fournie dans une `struct timespec ts` en exécutant

```
clock_gettime(CLOCK_REALTIME,&ts);
```

ou

```
timespec_get(&ts,TIME_UTC);
```

– Pour des raisons de portabilité du code, le paquet SNTP ne peut pas être un `struct` envoyé tel quel sur la socket. Il faut donc utiliser un tableau de `char`. Faites bien attention au « boutisme » des valeurs codées sur plus d'1 octet.

Les champs LI, VN et Mode doivent être accolées ensemble dans un même octet (`char`) à l'aide des opérations de manipulation de bits (« ou » `|`, « et » `&`, décalages de bit `<<` et `>>`) ; pour un résumé lisez cet article de wikipedia).

– Votre programme ne doit pas utiliser de nombres à virgule flottante.

– Si vous testez votre client depuis le réseau de l'université, utilisez le serveur NTP suivant : `ntp` depuis les machines de l'ufr. Sinon, vous pouvez utiliser, par exemple, `2.fr.pool.ntp.org` depuis chez vous.

Qu'arrive-t-il à votre client si la requête ou la réponse sont perdues ?

II) Un client utile

Soit t la temps à laquelle une réponse a été reçue, et t_o, t_r et t_t les trois temps contenus dans celle-ci (voir figure 1). Alors :

– le délai total est donné par

$$d_t = t - t_o;$$

– le délai dû au serveur est donné par

$$d_s = t_t - t_r;$$

– le délai dû au réseau est donné par

$$d_r = d_t - d_s = (t - t_o) - (t_t - t_r);$$

– si on suppose que la route entre le client et le serveur est symétrique, une bonne estimation de la différence entre l'horloge locale et l'horloge distante est donnée par :

$$\delta = \frac{t_o + t}{2} - \frac{t_r + t_t}{2} = \frac{(t - t_t) + (t_o - t_r)}{2}.$$

Modifiez votre client pour qu'il affiche une estimation de la différence ainsi que les trois délais (sans utiliser de nombres à virgule flottante, sauf éventuellement au moment de l'affichage).

III) Un démon

Modifiez votre client pour qu'il envoie des requêtes indéfiniment, à intervalles irréguliers [2] de 30 secondes environ, et qu'il maintienne une estimation de l'écart entre l'horloge locale et l'horloge distante.
De nombreuses extensions sont possibles. On peut par exemple converser simultanément avec plusieurs serveurs et maintenir des statistiques séparées pour chaque serveur connu, éliminer les échantillons qui ont subi un délai trop important, comparer les heures données par plusieurs serveurs et éliminer les échantillons trop fantaisistes, ajouter une interface graphique qui affiche en temps réel les délais vis-à-vis des différents serveurs *etc.*

2. Pourquoi irréguliers ?