Programmation réseaux

TP 3 – Protocole SNTP

Février 2007

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.

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 :

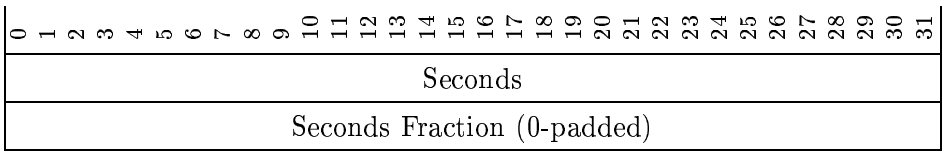
- la date à laquelle la requête a été transmise (selon le client) ;
- la data à laquelle la requête a été reçue (selon le serveur) ;
- la date à laquelle la réponse a été transmise (selon le serveur) ;
- la date à laquelle l'horloge du serveur a été réglée pour la dernière fois.

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.

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 :

0 1	2 3 4	5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31
LI	VN	Mode	Stratum	Poll	Precision
Root Delay					
Root Dispersion					
Reference Identifier					
Reference Timestamp (64)					
Originate Timestamp (64)					
Receive Timestamp (64)					
Transmit Timestamp (64)					

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 Classes fournies

Vous trouverez les fichiers `Timestamp.java` et `NTPMessage.java` sur la page `http://www.pps.jussieu.fr/~jch/enseignement/reseaux/`.

Timestamp La classe `Timestamp` représente une date au format NTP. Elle contient un certain nombre de constructeurs et méthodes. Dans un premier temps, la méthode suivante vous sera utile :

- `currentTime()`, qui contruit un `Timestamp` représentant l'heure courante (à 1 ms près environ).

La classe `Timestamp` définit une méthode `toString` ; il est donc possible d'en passer les instances à `println`.

NTPMessage La classe `NTPMessage` représente un message SNTP (requête ou réponse). Elle aussi contient un grand nombre de constructeurs et méthodes ; dans un premier temps, les suivantes vous seront utiles :

- `NTPMessage(Timestamp)`, qui construit une requête contenant l'argument comme *Transmit Timestamp* ;
- `format()`, qui construit un `byte[]` contenant la représentation « sur le fil » du message ;
- `toString()`, qui produit une forme vaguement lisible d'un message NTP.

Exercice 1 – *Premier client NTP*

Écrivez un programme Java 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.

Vous aurez besoin pour cela des méthodes `send()` et `receive()` de la classe `DatagramSocket`.

Dans ce TP, vous pouvez utiliser les serveurs NTP qui tournent sur les machines `nivose` et `lenteja` ; il n'est malheureusement pas possible d'utiliser des serveurs NTP distants du fait des mesures de sécurité[1] dont nous bénéficions.

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

Exercice 2 – *Un client robuste*

Modifiez le client précédent pour qu'il consiste de deux threads. Le thread émetteur envoie, à intervalles *irréguliers*[2] d'une minute environ des requêtes NTP ; le thread récepteur reçoit les réponses et les affiche. (Vous pourrez utiliser la méthode `nextInt()` de la classe `java.util.Random` pour calculer le délai, et la méthode `sleep()` de la classe `Thread` pour attendre.)

Modifiez ce client pour qu'il décode les paquets reçus du serveur, et maintienne une estimation de la différence entre les horloges locale et distante. (Il faudra pour cela rajouter des accesseurs à la classe `NTPMessage`.)

Exercice 3 – *Un client bavard*

Modifiez maintenant le client précédent pour qu'il puisse converser simultanément avec plusieurs serveurs, et maintienne des statistiques séparées pour chaque serveur connu.

De nombreuses extensions sont possibles. On peut par exemple 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.

[1] Peut-être quelque peu exagérées.

[2] Pourquoi ?