

STI 3^{ème} année – Réseau

TD 1: Socket et client NTP

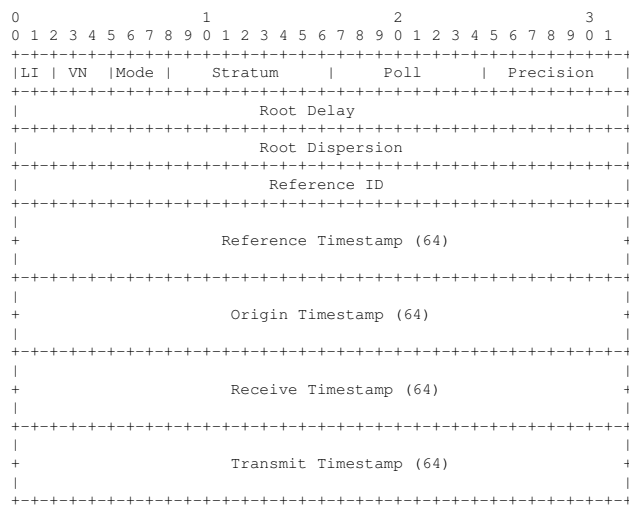
J. Briffaut

<https://lettier.github.io/posts/2016-04-26-lets-make-a-ntp-client-in-c.html>

1 NTP

Network Time Protocol (« protocole de temps réseau ») ou *NTP* est un protocole qui permet de synchroniser, via un réseau informatique, l'horloge locale d'ordinateurs sur une référence d'heure.

D'après la RFC1305¹ Network Time Protocol (Version 3), la structure d'un message NTP est composée de 384 bits (soit 48 octets) correspondants à 17 champs :



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

- **LI** indicateur d'insertion/retrait d'une seconde intercalaire la dernière minute du jour courant, on utilisera toujours **0**;
- **VN** est la version du protocole employée, et vaudra toujours **3**;
- **Mode** est le type de message, **3** pour une requête, **4** pour une réponse;
- **Stratum** : stratum de l'horloge locale, on utilisera **0**;
- **Poll** : intervalle minimum entre deux messages successifs, on utilisera **0**;
- **Precision** : précision de l'horloge locale;
- **Reference Timestamp**, on ignore, on utilisera **0**;
- **Originate Timestamp** **0** pour une requête. Pour une réponse, la copie du Transmit Timestamp de la requête correspondante;

1. <https://tools.ietf.org/html/rfc1305>

- **Receive Timestamp** 0 pour une requête. Pour une réponse, la date de réception de la requête correspondante;
- **Transmit Timestamp** est la date de transmission de ce paquet.

En regroupant les champs LI (2 bits), VN (3 bits) et mode (3 bits) sur 1 octet, afin de simplifier l'implémentation, et en découpant les champs 64 bits en 2 entiers de 32 bits, on obtient la structure suivante :

```
typedef struct
{
    uint8_t li_vn_mode;    // Eight bits. li, vn, and mode.
    // li. Two bits. Leap indicator.
    // vn. Three bits. Version number of the protocol.
    // mode. Three bits. Client will pick mode 3 for client.

    uint8_t stratum;       // Eight bits. Stratum level of the local clock.
    uint8_t poll;          // Eight bits. Maximum interval between successive messages.
    uint8_t precision;     // Eight bits. Precision of the local clock.

    uint32_t rootDelay;    // 32 bits. Total round trip delay time.
    uint32_t rootDispersion; // 32 bits. Max error aloud from primary clock source.
    uint32_t refId;        // 32 bits. Reference clock identifier.

    uint32_t refTm_s;      // 32 bits. Reference time-stamp seconds.
    uint32_t refTm_f;      // 32 bits. Reference time-stamp fraction of a second.

    uint32_t origTm_s;     // 32 bits. Originate time-stamp seconds.
    uint32_t origTm_f;     // 32 bits. Originate time-stamp fraction of a second.

    uint32_t rxTm_s;       // 32 bits. Received time-stamp seconds.
    uint32_t rxTm_f;       // 32 bits. Received time-stamp fraction of a second.

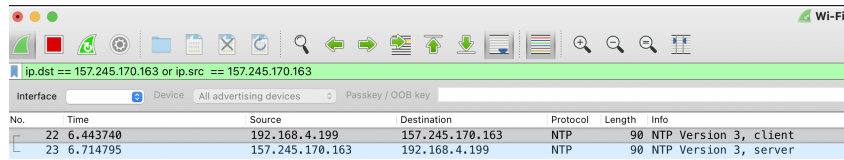
    uint32_t txTm_s;       // 32 bits and the most important field the client cares about. Transmit time-stamp
                           // seconds.
    uint32_t txTm_f;       // 32 bits. Transmit time-stamp fraction of a second.

} ntp_packet;            // Total: 384 bits or 48 bytes.
```

2 NTP Client UDP

Afin de récupérer l'heure auprès d'un serveur NTP, par exemple `us.pool.ntp.org`, en UDP. Votre client devra envoyer un packet UDP avec une partie data de 48 octets à 0 (vide). Le serveur enverra en retour un packet UDP avec les informations remplies.

Requête :



ip.dst == 157.245.170.163 or ip.src == 157.245.170.163

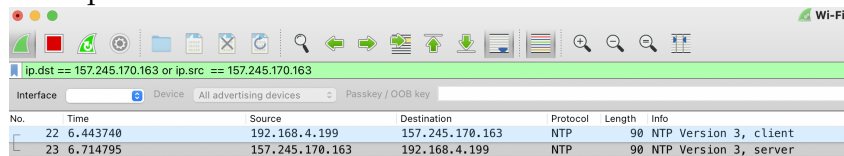
No.	Time	Source	Destination	Protocol	Length	Info
22	6.443740	192.168.4.199	157.245.170.163	NTP	90	NTP Version 3, client
23	6.714795	157.245.170.163	192.168.4.199	NTP	90	NTP Version 3, server

```

> Frame 22: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface 0
> Ethernet II, Src: 8c:85:90:54:78:c6, Dst: b8:27:eb:a4:99:df
> Internet Protocol Version 4, Src: 192.168.4.199, Dst: 157.245.170.163
> User Datagram Protocol, Src Port: 62859, Dst Port: 123
  Source Port: 62859
  Destination Port: 123
  Length: 56
  Checksum: 0xe06e [unverified]
  [Checksum Status: Unverified]
  [Stream index: 7]
  Network Time Protocol (NTP Version 3, client)
    Flags: 0x1b, Leap Indicator: no warning, Version number: NTP Version 3, Mode: client
    Peer Clock Stratum: unspecified or invalid (0)
    Peer Polling Interval: invalid (0)
    Peer Clock Precision: 1.000000 sec
    Root Delay: 0 seconds
    Root Dispersion: 0 seconds
    Reference ID: NULL
    Reference Timestamp: Jan  1, 1970 00:00:00.000000000 UTC
    Origin Timestamp: Jan  1, 1970 00:00:00.000000000 UTC
    Receive Timestamp: Jan  1, 1970 00:00:00.000000000 UTC
    Transmit Timestamp: Jan  1, 1970 00:00:00.000000000 UTC

```

Réponse :



ip.dst == 157.245.170.163 or ip.src == 157.245.170.163

No.	Time	Source	Destination	Protocol	Length	Info
22	6.443740	192.168.4.199	157.245.170.163	NTP	90	NTP Version 3, client
23	6.714795	157.245.170.163	192.168.4.199	NTP	90	NTP Version 3, server

```

> Frame 23: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface 0
> Ethernet II, Src: c8:25:e9:7e:85:48, Dst: 8c:85:90:54:78:c6
> Internet Protocol Version 4, Src: 157.245.170.163, Dst: 192.168.4.199
> User Datagram Protocol, Src Port: 123, Dst Port: 62859
  Source Port: 123
  Destination Port: 62859
  Length: 56
  Checksum: 0xa8f2 [unverified]
  [Checksum Status: Unverified]
  [Stream index: 7]
  Network Time Protocol (NTP Version 3, server)
    Flags: 0x1c, Leap Indicator: no warning, Version number: NTP Version 3, Mode: server
    Peer Clock Stratum: secondary reference (3)
    Peer Polling Interval: invalid (3)
    Peer Clock Precision: 0.000000 sec
    Root Delay: 0.101837158203125 seconds
    Root Dispersion: 0.0540313720703125 seconds
    Reference ID: 206.55.64.77
    Reference Timestamp: Nov 23, 2020 13:52:53.044355899 UTC
    Origin Timestamp: Jan  1, 1970 00:00:00.000000000 UTC
    Receive Timestamp: Nov 23, 2020 13:57:55.615823773 UTC
    Transmit Timestamp: Nov 23, 2020 13:57:55.615856664 UTC

```

2.1 Préparatifs

- Exercice 1** Créer un fichier `ntpclient-udp.c`, avec une fonction `main` (correct!)
- Exercice 2** Définir une fonction `void error(char* msg)` qui affiche le message à l'aide de `perror`, puis quitte le programme avec le code `EXIT_FAILURE`
- Exercice 3** Ajouter la définition de la structure `ntp_packet`
- Exercice 4** Vérifier que votre code compile correctement avec :
- ```
gcc -Wall -o ntpclient-udp ntpclient-udp.c
```

## 2.2 Création du message

Nous allons construire le message qui sera envoyé au serveur NTP.

- Exercice 5** Créer une variable `ntp_packet packet` avec tous les champs à zero
- Exercice 6** Ajouter 3 macros qui permettront de fixer les champs LI, VN et MODE. Comment fonctionnent ces macros ?

```
#define SET_LI(packet, li) (uint8_t) (packet.li_vn_mode |= (li << 6))
#define SET_VN(packet, vn) (uint8_t) (packet.li_vn_mode |= (vn << 3))
#define SET_MODE(packet, mode) (uint8_t) (packet.li_vn_mode |= (mode << 0))
```

- Exercice 7** Utiliser ces 3 macros pour fixer la valeur LI=0 (ignoré), VN=3 (NTPv3), MODE=3 (client) Le message est maintenant prêt à être envoyé au serveur NTP

## 2.3 Création de la socket

- Exercice 8** A l'aide de la fonction `socket`, créer une socket UDP (*man udp*). Vous nommerez le descripteur de fichier `sockfd`.
- Exercice 9** Gérer correctement les erreurs en appelant votre fonction `error`

Nous allons maintenant (dans les exercices 10 à 15) associer une IP et un port à cette socket pour envoyer notre message au serveur "fr.pool.ntp.org" sur le port 123 (port udp utilisé pour NTP).

- Exercice 10** Définir une variable de type chaîne de caractère `host_name` correspondant à "fr.pool.ntp.org"
- Exercice 11** A l'aide de la fonction `gethostbyname`, obtenez une structure `struct hostent *server` qui contiendra l'adresse IP du serveur NTP que l'on souhaite interroger. Pensez à gérer correctement les erreurs possibles.
- Exercice 12** Créer une structure `sockaddr_in` nommée `serv_addr`. Puis l'initialiser à zero à l'aide de la fonction `bzero`.
- Exercice 13** Fixer la famille de votre structure d'adresse en tant que *Internet version 4 protocols*
- Exercice 14** En utilisant `bcopy`, recopier l'adresse de votre `server` (champ `h_addr`) dans votre structure d'adresse (champ `sin_addr.s_addr`) afin d'initialiser l'adresse cible.
- Exercice 15** Initialiser le port de destination (champ `sin_port`) de votre structure d'adresse avec la valeur 123

## 2.4 Fixer les paramètres de connexion au serveur

Nous avons créé une socket de type UDP, et une structure d'adresse avec l'adresse et le port du serveur NTP que l'on souhaite interroger.

**Exercice 16** En utilisant la fonction `connect` fixer les paramètres de "connexion" qui seront utilisés ensuite par les fonctions `read/write` pour envoyer/recevoir des messages. (comme d'habitude, gérer correctement les erreurs).

## 2.5 Envoi de la requête NTP

Vous pouvez maintenant envoyer votre `packet` au serveur afin qu'il vous envoie en retour sa réponse (contenant "l'heure actuelle").

**Exercice 17** Utiliser la fonction `write` afin d'envoyer votre `packet`. (comme d'habitude, gérer correctement les erreurs).

**Exercice 18** Vérifier à l'aide de **wireshark** que votre requête est bien envoyée (et que vous recevez bien une réponse même si on ne la traite pas encore).

## 2.6 Réception de la réponse

**Exercice 19** Utiliser la fonction `read` afin de récupérer la réponse du serveur dans votre `packet`. (comme d'habitude, gérer correctement les erreurs).

## 2.7 Traitement de la réponse

Le serveur vous renvoie la date sous la forme d'un timestamp (un entier incrémenté toutes les secondes) depuis le NTP epoch (01/01/1900).

**Exercice 20** Afficher le `timestamp` que vous avez reçu (le champ `txTm_s`). Attention, pensez à convertir les données qui sont au format réseau, vers la représentation utilisée par votre hôte.

**Exercice 21** Comparez le timestamp que vous avez obtenu avec celui de `http://www.timestamp.fr`. Est-ce le même ? Pourquoi ?

**Exercice 22** Le timestamp Unix ayant une epoch le 01/01/1970, vous devez déduire la valeur 2208988800 afin d'obtenir un timestamp "compatible-Unix". Définissez la constante suivante :

### Listing 1 – Solution de l'exercice 22

```
#define NTP_TIMESTAMP_DELTA 2208988800ull
```

**Exercice 23** Afficher à nouveau le timestamp en déduisant cette constante, et comparer avec `http://www.timestamp.fr`

**Exercice 24** Finalement, en utilisant la fonction `ctime`, afficher la date actuelle.

## 2.8 Exercice supplémentaire

**Exercice 25** Modifier votre code pour prendre le nom du serveur de temps en paramètre de votre programme.

**Exercice 26** Utiliser `sendto` et `recvfrom` à la place de `write/read`