Reloj basado en NTP con Java

Práctica de laboratorio 1

Aplicaciones Telemáticas
ETSIT-UPV

J. E. López Patiño
F.J. Martínez Zaldívar

Índice

1. Introducción y objetivos

En la presente práctica el alumno implementará en Java un sencillo programa con el que mostrará la hora obtenida de un servidor de tiempo, empleando los paquetes UDP que se transfieren en el protocolo NTP *(Network Time Protocol)*.

El alumno deberá ser capaz de realizar la edición y compilación de ficheros Java así como de entender mínimamente el formato del paquete UDP de NTP, interpretarlo correctamente y deducir la hora local con las correcciones oportunas. Tanto la compilación como la ejecución se llevará a cabo en línea de comandos, sin utilizar ningún IDE *(Integrated Development Environment)*.

1.1. Formación de grupos

Los grupos para realizar esta práctica serán de uno o dos alumnos. El nombre del grupo de GitHub Classroom estará formado por la concatenación del primer apellido de cada alumno; si el grupo consta de tan solo un alumno, el nombre del grupo será entonces, la contatenación de ambos apellidos. Ténganse en cuenta las consideraciones comentadas en el apartado "Resultados a entregar" para evitar que cualquier incumplimiento implique una merma en la calificación.

1.2. Software y hardware necesario

El software mínimo necesario será el JDK de Java. Cualquier máquina con cualquier sistema operativo, este software y conexión a internet será suficiente. Adicionalmente se requerirá un editor de texto plano y el sistema de control de versiones Git.

1.3. Aplicación a realizar

La ejecución de la aplicación debe simplemente mostrar la hora en el instante de ejecución en formato `hh:mm:ss` en pantalla. Se deberán hacer las correcciones oportunas en función del huso horario (UTC+01:00 u obsoletamente GMT+01:00) y del horario de verano (+0/1 h).

1.4. Repositorio compartido

Cada grupo de prácticas (formado por uno o por dos alumnos) comparte un repositorio con la organización de GitHub `AATT-ETSIT`.

Antes de comenzar la práctica, el alumno deberá primeramente verificar que `git` está instalado en el ordenador de prácticas. Si la cuenta de usuario del ordenador de prácticas no es nominal del alumno, se sugiere que la configuración (nombre y email del usuario: `git config user.name "nombre"` y `git config user.email "correo"`) de git no se realice de manera global, sino particular por directorio o repositorio y al final, borrar todo el directorio de trabajo al acabar la sesión de la práctica.

A continuación debe escoger un directorio en el cual realizar una clonación del repositorio de la práctica para trabajar en modo local. En el directorio de trabajo hay a su vez dos directorios: `doc` con la presente documentación y `codigo` con un fichero plantilla `JavaNTP.java` que hay que rellenar y hacerlo funcionar. Un instante antes de finalizar la sesión de prácticas hay que subir obligatoriamente el repositorio local (desde el ordenador utilizado en la sesión de prácticas) al repositorio remoto (GitHub). Este último commit debe tener una etiqueta o `tag` cuyo valor sea `"pr1.0"`. Se dispondrán de 24 horas adicionales para finalizar algún detalle que pudiera haber quedado pendiente, en cuyo caso deberá de nuevo subirse la versión definitiva con el `tag "pr1.1"`. (Recuérdese que las etiquetas hay que subirlas hacia el respositorio remoto explícitamente con el comando `git push origin --tags`, si `origin` es el alias del servidor donde está ubicado el repositorio remoto.)

A lo largo del desarrollo de la práctica, el grupo es completamente libre de actualizar el repositorio local o remoto como lo estime oportuno, con las ramas o los *commits* que estimen necesarios.

2. NTP

2.1. Protocolo

NTP o *Network Time Protocol*, versión 4, puede encontrarse en la RFC 5905, cuya última versión data de junio de 2010. En los documentos RFC 5906, 5907 y 5908 puede encontrarse información adicional. En otras RFC, como 958, 1305 o 4330 aparecen versiones anteriores de este protocolo.

NTP es un protocolo empleado ampliamente para sincronizar los relojes de computadores conectados a Internet.

2.2. Formatos de tiempos en NTP

Existen tres formatos de tiempo en NTP: *NTP Short Format*, *NTP Timestamp Format* y *NTP Date Format* (véase la figura 1). Nosotros utilizaremos el denominado *NTP Timestamp Format*, mostrado como segunda

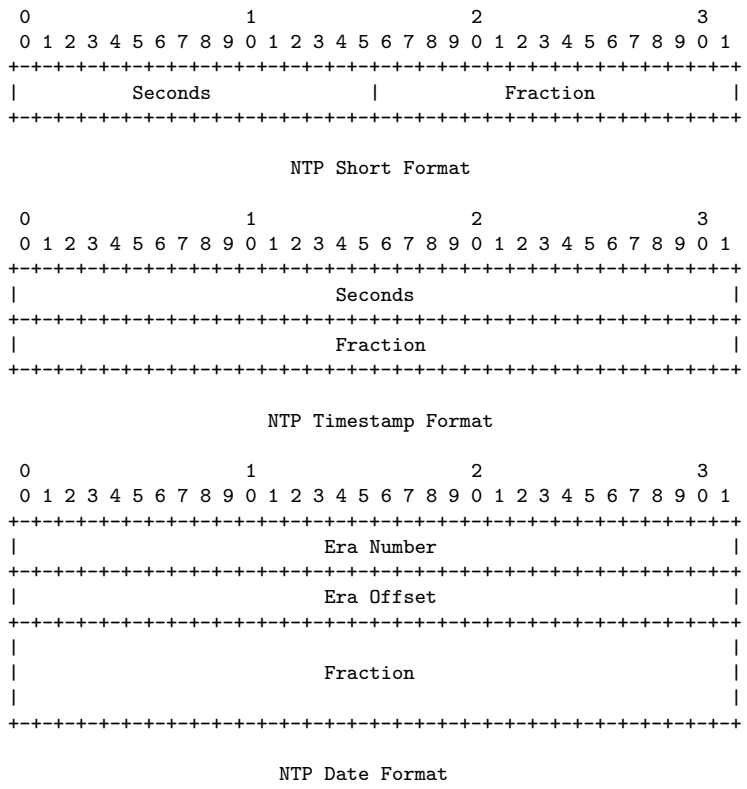

Figura 1: Formatos de tiempo en NTP

subfigura dentro de la figura 1, en el que el tiempo se codifica con formato de punto o coma fija. Con este formato se dedican 32 bits **sin signo** a la parte entera de la cantidad de segundos que han transcurrido desde el 1 de enero del año 1900, y otros 32 bits también sin signo para para la parte fraccionaria.

2.3. Paquetes UDP

El protocolo NTP emplea exclusivamente paquetes UDP. En la práctica se *rellenará* un paquete UDP para realizar una petición de tiempo a un servidor y este responderá con otro paquete UDP con la información requerida.

En la RFC 5905 aparece la estructura del paquete y también las distintas formas de representar el tiempo. La versión básica sin extensiones consta de **48 octetos** (numerados del 0 al 47). La figura 2 reproduce este formato, donde la representación de la información es *big endian* (los octetos se almacenan y transmiten de izquierda a derecha, de arriba a abajo, y dentro de un octeto, el primer bit que aparece o se transmite es el de mayor peso). En cada fila se muestran 32 bits numerados desde 00 hasta el 31, es decir, 4 octetos;

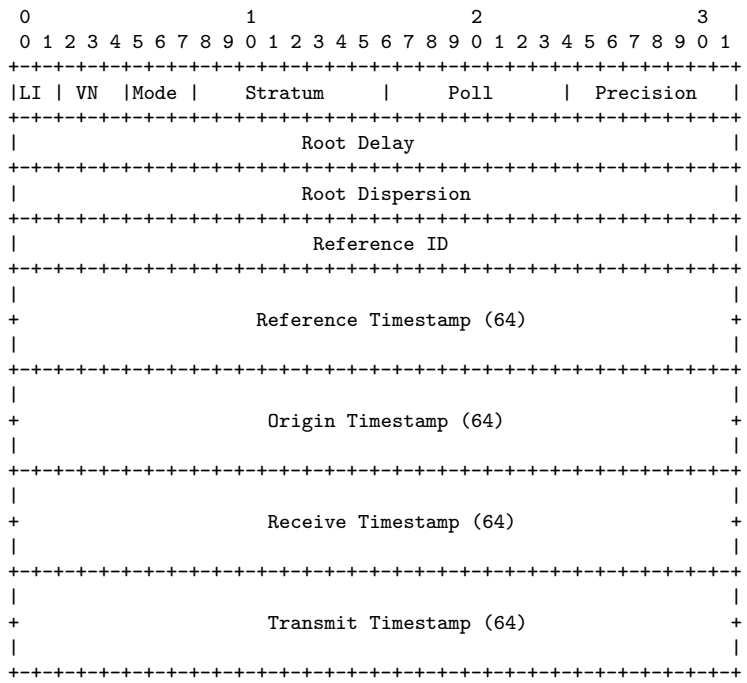

Figura 2: Estructura de paquete UDP de NTP

tenemos un total de 12 filas, por lo que el paquete constará efectivamente de 48 octetos, tal y como hemos comentado anteriormente. Existen campos adicionales que son opcionales y sin interés para la presente práctica.

El significado de los campos que aparecen en la figura 2 es el siguiente (aunque en la práctica se empleará un subconjunto mínimo de estos tal y como hemos comentado anteriormente):

- `LI` *(Leap Indicator)*: indicador de salto (2 bits)

Valor	*Significado*
00	Sin advertencia
01	El último minuto del día tendrá 61 segundos
10	El último minuto del día tendrá 59 segundos
11	Reloj no sincronizado (o valor desconocido)

- `VN` *(Version Number)*: número de versión (3 bits, actualmente la versión es la 4: 100)

- `Mode`: modo o rol asumido en el protocolo

Valor	*Significado*
000	Reservado
001	Activo simétrico
010	Pasivo simétrico
011	Cliente
100	Servidor
101	Broadcast
110	Mensaje de control NTP
111	Reservado

- `Stratum`: estrato o nivel (8 bits) al que pertenece el servidor

Valor	*Significado*
0	Sin especificar o inválido
1	Servidor primario (equipado con receptor GPS)
2–15	Servidor secundario (vía NTP)
16	Desincronizado
17–255	Reservado

- `Poll` (8 bits con signo): máximo intervalo entre mensajes sucesivos (dado en $\log_2$ segundos, por ejemplo, un valor de 6 significaría envío de mensajes cada $2^6 = 64$ segundos). Valores típicos: entre 6 y 10.

- `Precision` (8 bits con signo): precisión del sistema de reloj en $\log_2$ segundos. Por ejemplo, un valor de -18 implicaría una precisión de $2^{-18} \approx 3 \cdot 10^{-6}$, es decir del orden del microsegundo

- `Root Delay`: tiempo de retardo de ida y vuelta total hasta el reloj de referencia en formato *NTP shorf format* (32 bits)

- `Root Dispersion`: dispersión respecto al reloj de referencia en formato *NTP short format* (32 bits)

- `Reference ID`: identificación de la referencia (32 bits). Estos 32 bits se interpretan como cuatro caracteres ASCII *(kiss code)* para estratos 0 y 1; para estratos superiores es la dirección IPv4 (si se utiliza IPv6, entonces son los cuatro primeros octetos del *hash* MD5 de la dirección IPv6)

ID	Fuente
GOES	Geosynchronous Orbit Environment Satellite
GPS	Global Position System
GAL	Galileo Positioning System
PPS	Generic pulse-per-second
IRIG	Inter-Range Instrumentation Group
WWVB	LF Radio WWVB Ft. Collins, CO 60 kHz
DCF	LF Radio DCF77 Mainflingen, DE 77.5 kHz
HBG	LF Radio HBG Prangins, HB 75 kHz
MSF	LF Radio MSF Anthorn, UK 60 kHz
JJY	LF Radio JJY Fukushima, JP 40 kHz, Saga, JP 60 kHz
LORC	MF Radio LORAN C station, 100 kHz
TDF	MF Radio Allouis, FR 162 kHz
CHU	HF Radio CHU Ottawa, Ontario
WWV	HF Radio WWV Ft. Collins, CO
WWVH	HF Radio WWVH Kauai, HI
NIST	NIST telephone modem
ACTS	NIST telephone modem
USNO	USNO telephone modem
PTB	European telephone modem

- `Reference Timestamp`: tiempo cuando el reloj fue ajustado por última vez en formato *NTP timestamp* (64 bits)
- `Origin Timestamp`: tiempo en el cliente en el momento de hacer la petición hacia el servidor, en formato *NTP timestamp* (64 bits)
- `Receive Timestamp`: tiempo en el servidor, en el momento en el que llegó la petición del cliente, en formato *NTP timestamp* (64 bits)
- `Transmit Timestamp`: tiempo en el servidor, en el momento en que que envió la respuesta hacia el cliente, en formato *NTP timestamp* (64 bits)

3. Detalles de implementación del ejercicio

3.1. Envío de paquete UDP hacia el servidor

De todo el paquete UDP de 48 bytes, los únicos campos con información necesaria para efectuar la petición son la versión, `VN`=4 (100 en binario) y el modo, `Mode`=3 (011 en binario), ambos en el octeto número 0 del paquete; es decir que el primer octeto deberá contener un

$$00\ 100\ 011 = \ 0010\ 0011 = 0\text{x}23.$$

El resto de octetos pueden estar perfectamente a 0.

3.2. Recepción del paquete UDP del servidor

El formato del paqute UDP recibido es idéntico al transmitido. El campo que nos interesa es el de `Transmit Timestamp`, cuya parte entera (32 bits) se haya ubicada en los octetos 40–43 (obviaremos los 32 bits correspondientes a la parte fraccionaria ubicada en los octetos 44–47). A estos 32 bits de la parte entera correspondiente a la cantidad de segundos transcurridos desde el 1 de enero de 1900, la denominaremos t_{NTP} o simplemente `tiempo`.

3.2.1. Tipos de datos en Java

La parte entera del tiempo NTP está codificada con 32 bits, luego parece oportuno a priori emplear como tipo de datos `int` en Java ya que estos se plasman en 32 bits. Pero tenemos un problema potencial de rango en la representación, ya que este tiempo está codificado en 32 bits sin signo, pero los tipos numéricos simples de Java son con signo en complemento a 2, luego el rango está en el intervalo

$$[-2^{31} \dots 2^{31} - 1] = [-2\,147\,483\,648 \dots 2\,147\,483\,647],$$

y, por ejemplo, el 31 de diciembre de 1999 ya tenía el tiempo NTP $t_{\mathrm{NTP}} = 3\,155\,587\,200$ s, lo cual ya estaría fuera de rango, luego el tipo `int` producirá interpretaciones incorrectas de las magnitudes; es por ello que necesitaremos un tipo de datos de Java `long` con el que ya no tendremos este problema, ya que su rango es

$$[-2^{63} \dots 2^{63} - 1] = [-9\,223\,372\,036\,854\,775\,808 \dots 9\,223\,372\,036\,854\,775\,807]$$

Sin embargo aparece un problema de conversión de datos ya que tendremos que construir un `long` a partir de 4 bytes (con los cuatro bytes de mayor peso en el `long` a cero), no existiendo ninguna instrucción Java a tal efecto. Para ello se propone concebir una solución que gráficamente se puede observar en las figuras 3, 4, 5 y 6.

En la figura 3 se propone emplear dos variables, ambas de tipo `long` denominadas `aux` y `tiempo` (esta inicializada a cero) respectivamente. El primer byte ubicado en la posición 40 del paquete UDP recibido se carga directamente en la variable `aux` (instrucción `aux = vector[40];`). En la carga se produce una extensión de signo (denotada con una serie de bits con la letra `s`) que para nuestros propósitos hay que cancelar realizando la operación

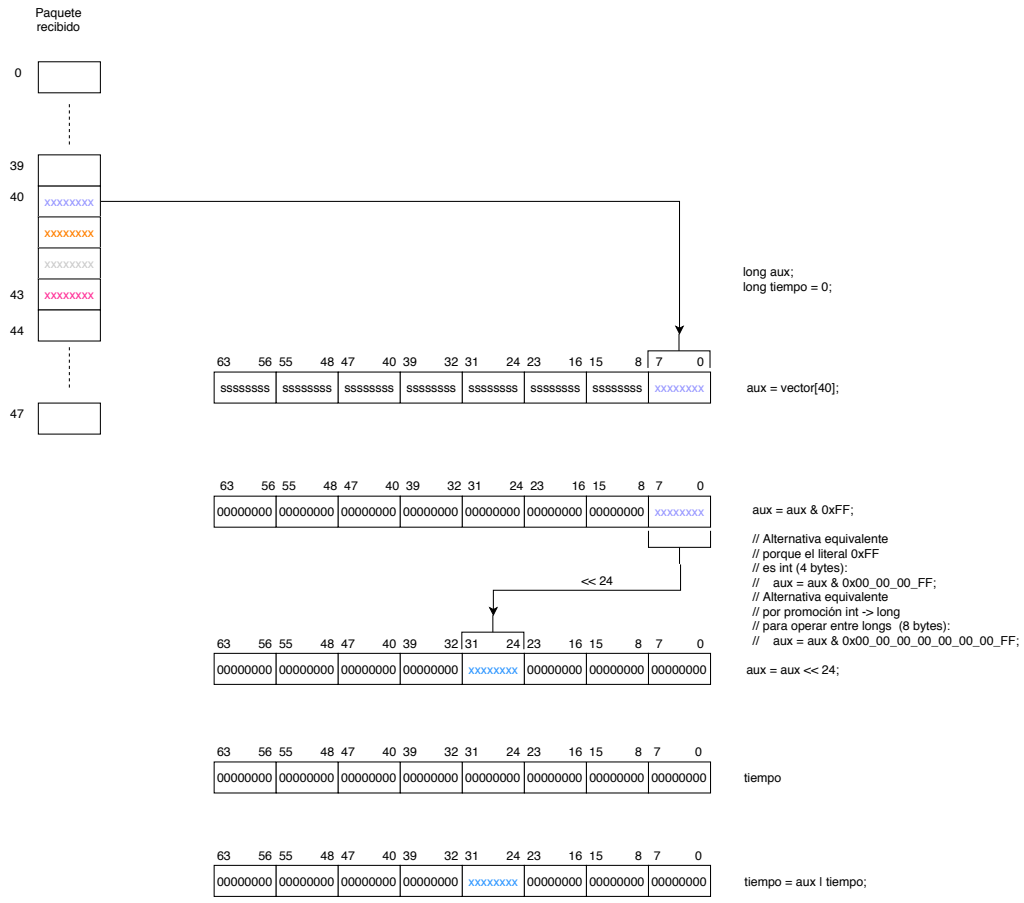

Figura 3: Primera iteración en conversión

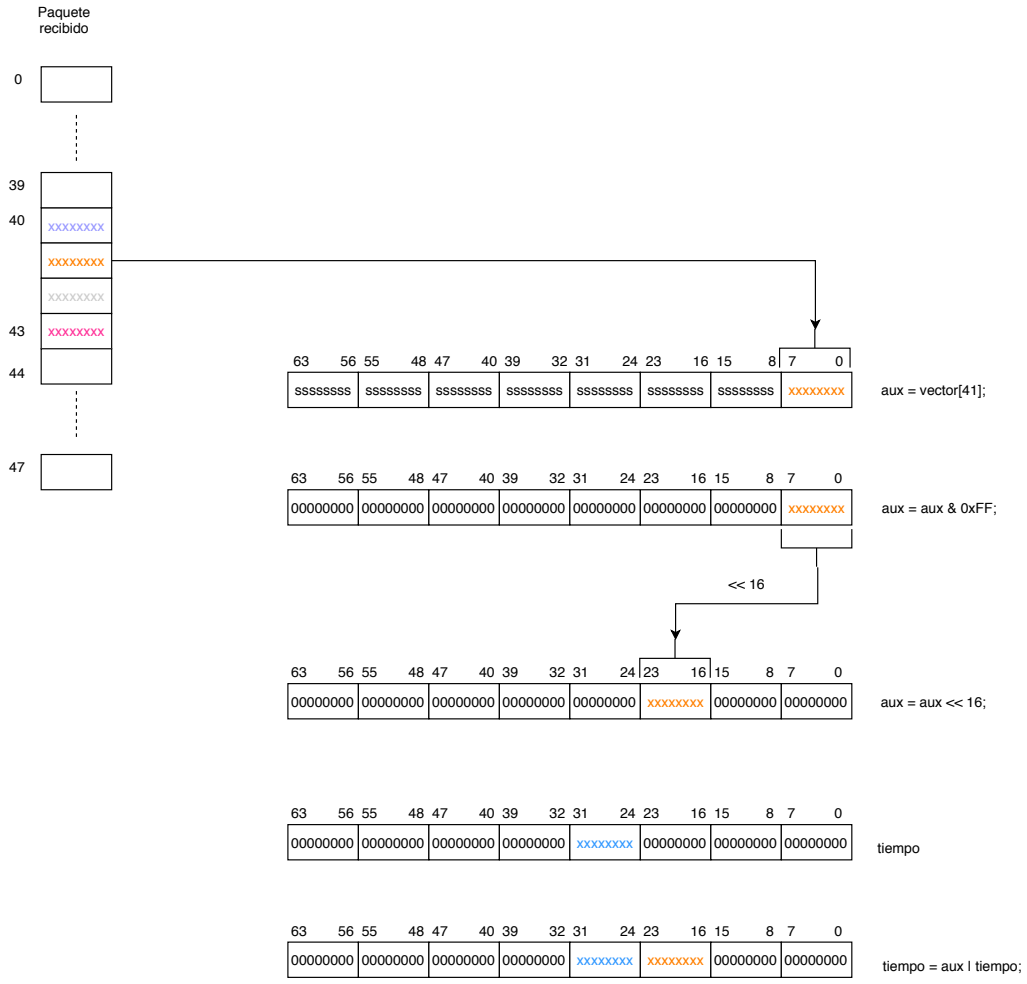

Figura 4: Segunda iteración en conversión

lógica AND con la máscara `0xFF` (instruccion `aux = aux & 0xFF;`). A continuación, a esta variable `aux` se le debe aplicar un desplazamiento lógico de 24 posiciones a la izquierda (instrucción `aux = aux << 24;`) para alinearla con la posición que deberá ocupar definivamente en la variable `tiempo`. Para homogeneizar el proceso con los siguientes pasos, esta *carga* de `aux` en `tiempo` se ha llevado a cabo con una operación lógica OR (instrucción `tiempo = aux | tiempo;`).

En las restantes figuras (figuras 4, 5 y 6) puede observarse un proceso equivalente para ubicar convenientemente los restantes 3 bytes en el sitio oportuno en la variable `tiempo` de tipo `long`.

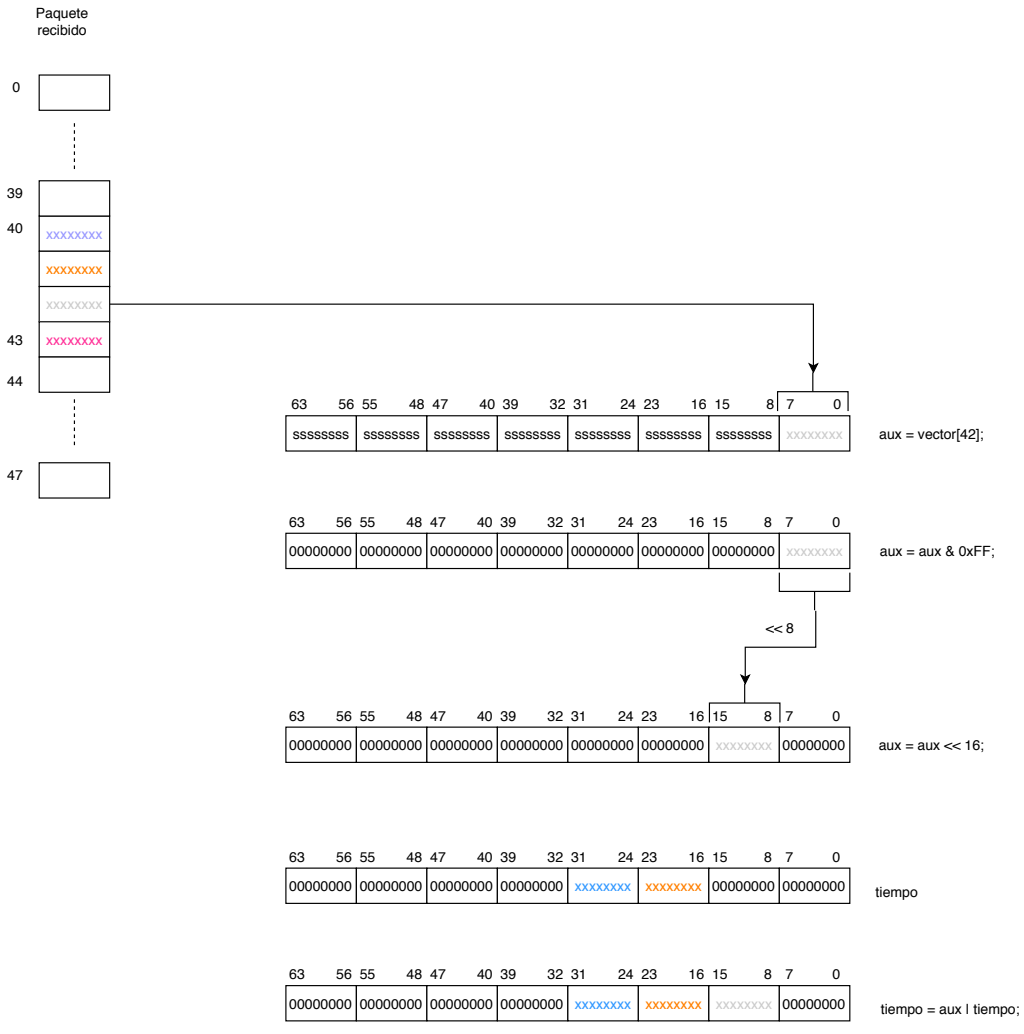

Figura 5: Tercera iteración en conversión

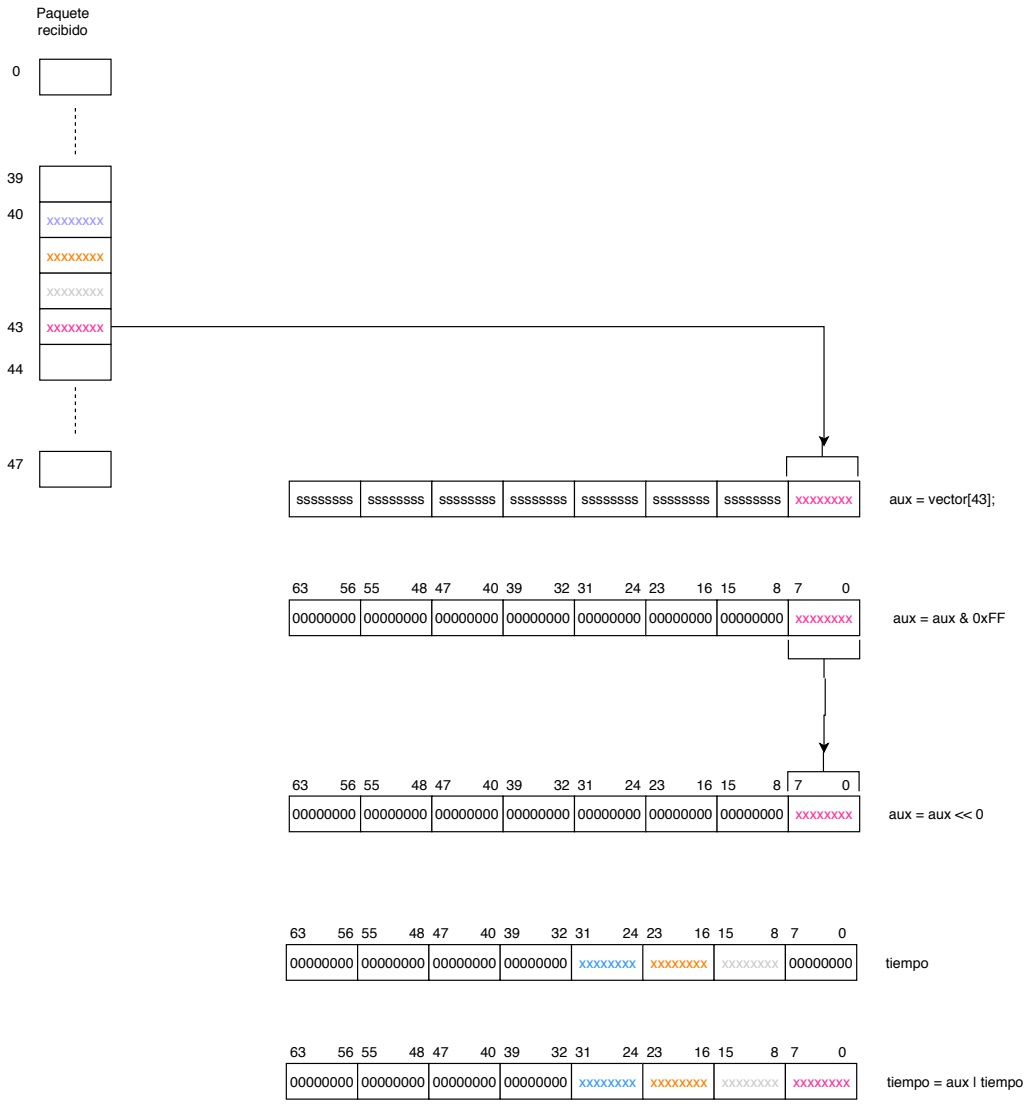

Figura 6: Cuarta iteración en conversión

Sugerencia:

Se propone que se diseñe un método que admita como parámetro un array de 4 bytes y devuelva un *long* correspondiente de alguna manera a la conversión descrita en este subapartado. Una sugerencia parcial puede ser la siguiente:

```
long deVector4BytesALong(byte[] vector) {
  long result;
  ...
  return result;
}
```

Adicionalmente se sugiere que se vaya rellenado ya el fichero `JavaNTP.java` con una clase que incluya al método `public static void main(String[] args)` en la que se desarrollará toda la práctica, incluyendo entre otros y oportunamente el método propuesto anteriormente. Se sugiere que simultáneamente se vayan realizando pruebas parciales intentando verificar todo lo que se vaya realizando, en la medida de las posibilidades.

3.2.2. Horas, minutos y segundos a partir del tiempo UTC

Nos ceñiremos exclusivamente a intentar obtener la hora dentro del día actual, el minuto dentro de la hora actual, y el segundo dentro del minuto actual. Para ello, ya que un minuto contiene 60 s, y una hora contiene 60 min, es decir, 3600 s y por tanto, un día tendrá 24 h, es decir, $3600 \cdot 60 = 86\,400$ s, podemos deducir fácilmente las tres cantidades:

$$\text{hora} = \frac{\texttt{tiempo}\,\%86\,400}{3600}, \quad \text{minutos} = \frac{\texttt{tiempo}\,\%3600}{60}, \quad \text{segundos} = \texttt{tiempo}\,\%60 \tag{1}$$

donde % denota la operación módulo y las divisiones son divisiones **enteras**, y `tiempo` es el tiempo UTC obtenido del protocolo NTP (es decir, la variable de tipo `long` obtenida de la conversión de los 4 bytes que comienzan en la posición 40 del datagrama, tal y como se ha descrito anteriormente).

Téngase en cuenta algunas correcciones necesarias para mostrar definitivamente los resultados: el tiempo recibido es tiempo UTC y en España, y concretamente la península Ibérica, la hora oficial es UTC+1. Adicionalmente, y en función de cuándo se realice la práctica, debería tenerse en cuenta la corrección de la hora local atendiendo a si se está en horario de invierno o de verano (1 hora adicional).

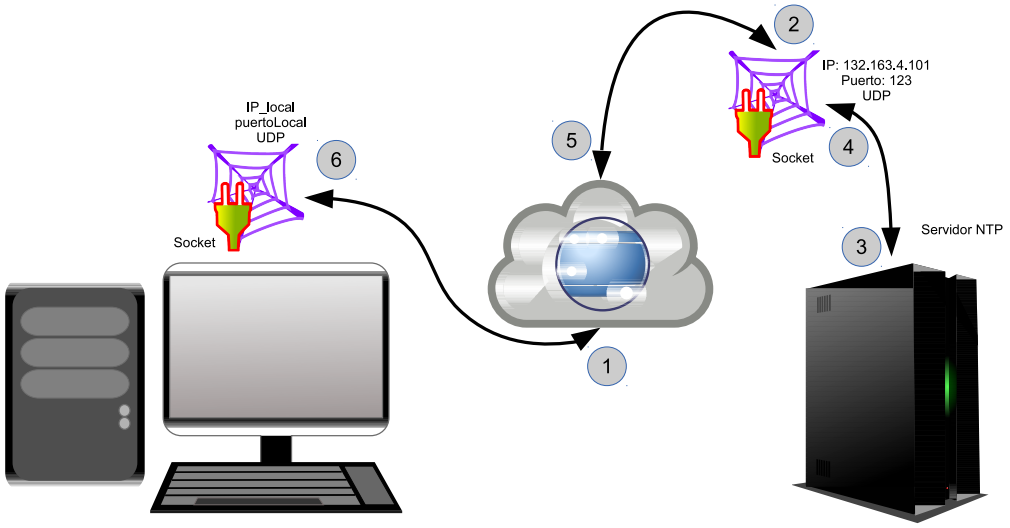

Figura 7: Modelo de funcionamiento

Sugerencia

Se propone que se implemente un método tal que a partir de un `long` que representa el tiempo UTC devuelva en un array de bytes las horas, los minutos y los segundos en las posiciones 0, 1 y 2 respectivamente del mismo, tal y como algebráicamente se ha descrito en los párrafos anteriores.

```
byte [] deUTC2hms(long t_UTC) {
  byte[] hms = new byte[3];
  ...
  return hms;
}
```

3.3. `DatagramPacket`, `DatagramSocket` e `InetAddress`

La figura 7 resume de manera simbólica las conexiones ordenadas a realizar entre nuestro cliente y el servidor NTP.

Las clases de Java para operar con datagramas o paquetes UDP serán `DatagramPacket`, `DatagramSocket` e `InetAddress`. A continuación se resume el esquema:

1. Crear un array de 48 bytes (por defecto, Java lo rellenará con ceros) que conformará el paquete UDP de NTP de petición de tiempo.

2. Inicializar el paquete UDP de NTP: simplemente escribiendo en el byte 0-ésimo del vector anterior lo mostrado en la subseción 3.1

3. Crear un *datagrama* instanciando la clase `DatagramPacket` con un constructor en el que se debe indicar tanto el array a enviar (creado anteriormente), su longitud, la dirección IP destino y el puerto destino. (Búsquese con la ayuda de la información de tipo `javadoc`, la sobrecarga oportuna del constructor del `DatagramPacket`.)

   La dirección IP destino habrá que especificarla empleando un objeto de la clase `InetAddress`. Para ello puede emplearse el método estático `InetAddress.getByName(String host)` tal que introduciéndole, por ejemplo, en un string la dirección IP con formato "xxx.xxx.xxx.xxx", devuelve el objeto necesario para incluirlo como parámetro en `DatagramPacket`.

   Búsquense en internet direcciones IP de servidores NTP (el puerto asignado a NTP es el 123). Si la búsqueda no es fortuita o no hay certeza de un correcto funcionamiento, pruébese con la dirección 130.206.3.166 correspondiente a la máquina `hora.rediris.es`. Se debería ser sumamente cuidadoso de no inundar al servidor con peticiones que pudieran provocar la denegación del servicio.

4. Créese adicionalmente un array de 48 bytes, que será el buffer o paquete UDP del protocolo NTP que contendrá la respuesta recibida desde el servidor NTP. Con este array se creará un nuevo `DatagramPacket`, esta vez, para la recepción. Se empleará un constructor con una sobrecarga en la que sólo sea necesario indicar dicho array y su longitud (véanse las acciones indicadas en la figura 10).

5. A continuación se abrirá un *socket*, es decir, se creará un objeto instanciando la clase `DatagramSocket`, donde será necesario especificar sólo un puerto local (indíquese un puerto superior a 1024).

6. Ahora se enviará el datagrama que contiene la petición NTP através del socket empleando el método `send` de dicho objeto, cuyo parámetro será el `DatagramPacket` de la petición (véase la figura 8)

7. Inmediatamente se llamará al método `receive` del mismo socket utilizando como parámetro el `DatagramPacket` creado para la recepción; se producirá una espera hasta que llegue la respuesta del servidor.

8. El paquete UDP transmitido por nosotros llegará eventualmente al servidor y éste responderá con un nuevo paquete UDP hacia el origen de la petición (es decir, nosotros, véase la figura 9).

9. La respuesta del servidor nos la encontraremos en el array de 48 bytes empleado para instanciar el `DatagramPacket` de recepción. Los bytes 40–43 contienen con formato *big endian* los 32 bits del tiempo NTP (véanse las acciones indicadas en la figura 10). Empléense las fórmulas (1) para determinar la hora, los minutos y los segundos e imprímase en pantalla con el formato indicado. Ténganse muy en cuenta las consideraciones necesarias comentadas en subapartados anteriores para transformar los 4 octetos consecutivos que empiezan en la posición 40 en formato *big endian* en una variable de tipo `long` que asumirá el rol de `tiempo` NTP o t_{NTP}.

3.3.1. Excepciones

La creación de algunos de los objetos o la ejecución de algunos métodos de los mismos requieren que ciertas excepciones sean *cazadas*. Las excepciones que es necesario considierar pueden obtenerse de la documentación, o simplemente de los errores de compilación. Recuérdese que el esquema general era:

```
try {
  // Operaciones susceptibles de provocar excepciones
} catch (ExcepcionAControlar e) {
  // Tratar la excepción. En nuestro entorno de prueba no
  // será necesario realizar ninguna acción en especial, pero
  // en una situación más realista sí que habría que hacerlo
} finally {
  // Se puede dejar también vacío
}
```

Sugerencia:

Se propone que los apartados enumerados anteriormente conformen ya prácticamente el contenido definitivo del método `main` de la clase en la que estamos trabajando. Para deducir el tiempo UTC y convertirlo en forma de horas, minutos y segundos emplearemos los metodos propuestos en los subapartados anteriores.

4. Resultados a entregar

Debe actualizarse el fichero denominado `JavaNTP.java` ubicado en el directorio `codigo` del repositorio GitHub compartido para la presente práctica. El profesorado evaluará la práctica teniendo en cuenta el contenido del fichero java y llevando a cabo una ejecución del mismo; para ello clonará localmente

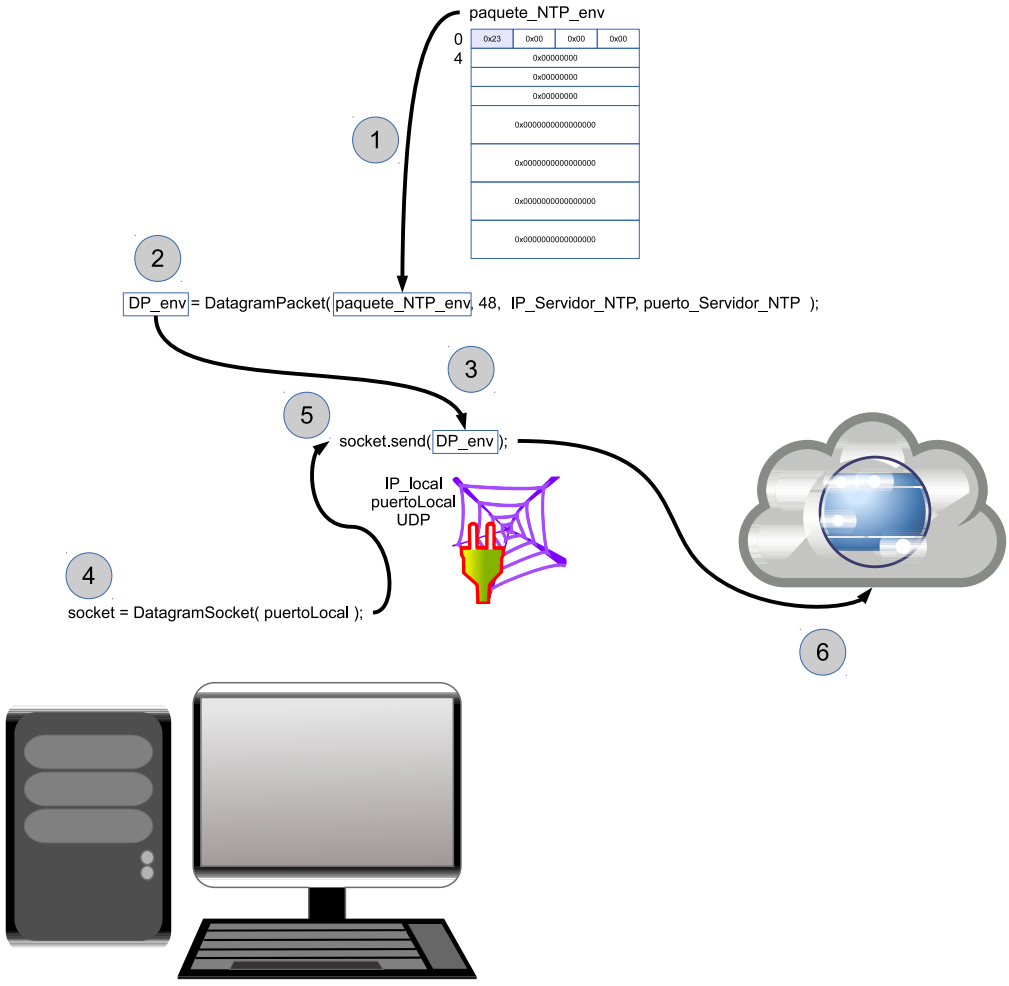

Figura 8: Envío de petición

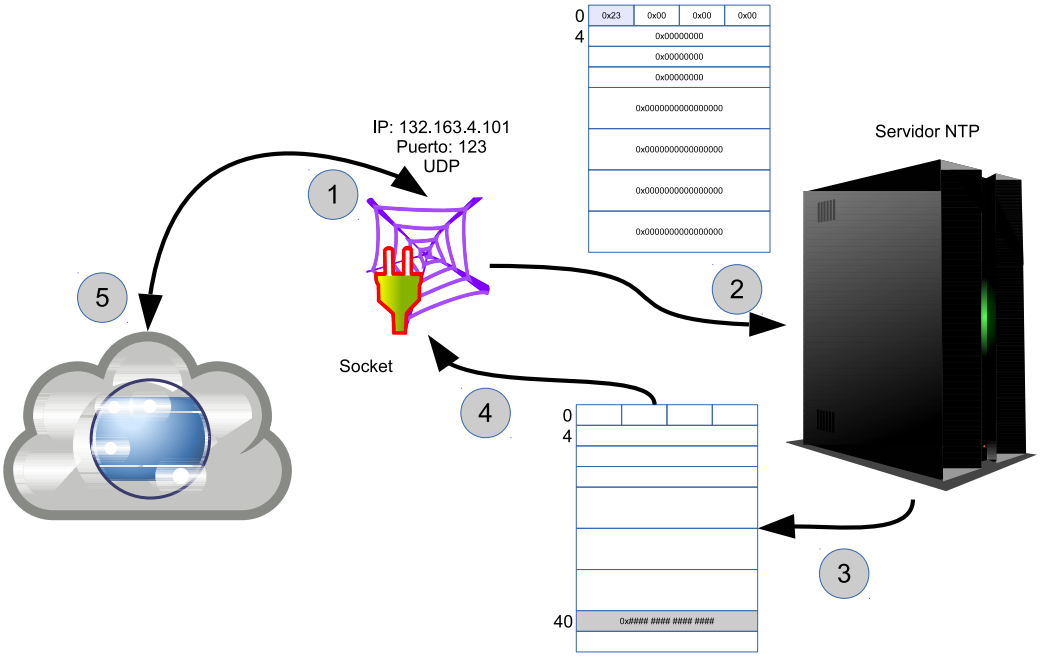

Figura 9: Recepción de petición y respuesta del servidor

el repositorio del grupo y en el directorio raíz de trabajo procederá de la siguiente manera:

```
> javac codigo\JavaNTP.java
> java codigo.JavaNTP
hh:mm:ss
```

Recuérdese que si el grupo está compuesto por dos personas, el nombre del grupo debe estar formado por la concatenación de los primeros apellidos de ambos; en caso contrario el nombre serán los dos apellidos del alumno.

Ténganse en cuenta todos los detalles relativos a las versiones a subir al repositorio GitHub (al acabar la sesión de prácticas y al finalizar los detalles pendientes 24 horas a lo sumo posterioremente), así como sus etiquetas o `tags` comentados al principio del presente documento. El incumplimiento de estas condiciones supondrá una merma en la calificación de la práctica.

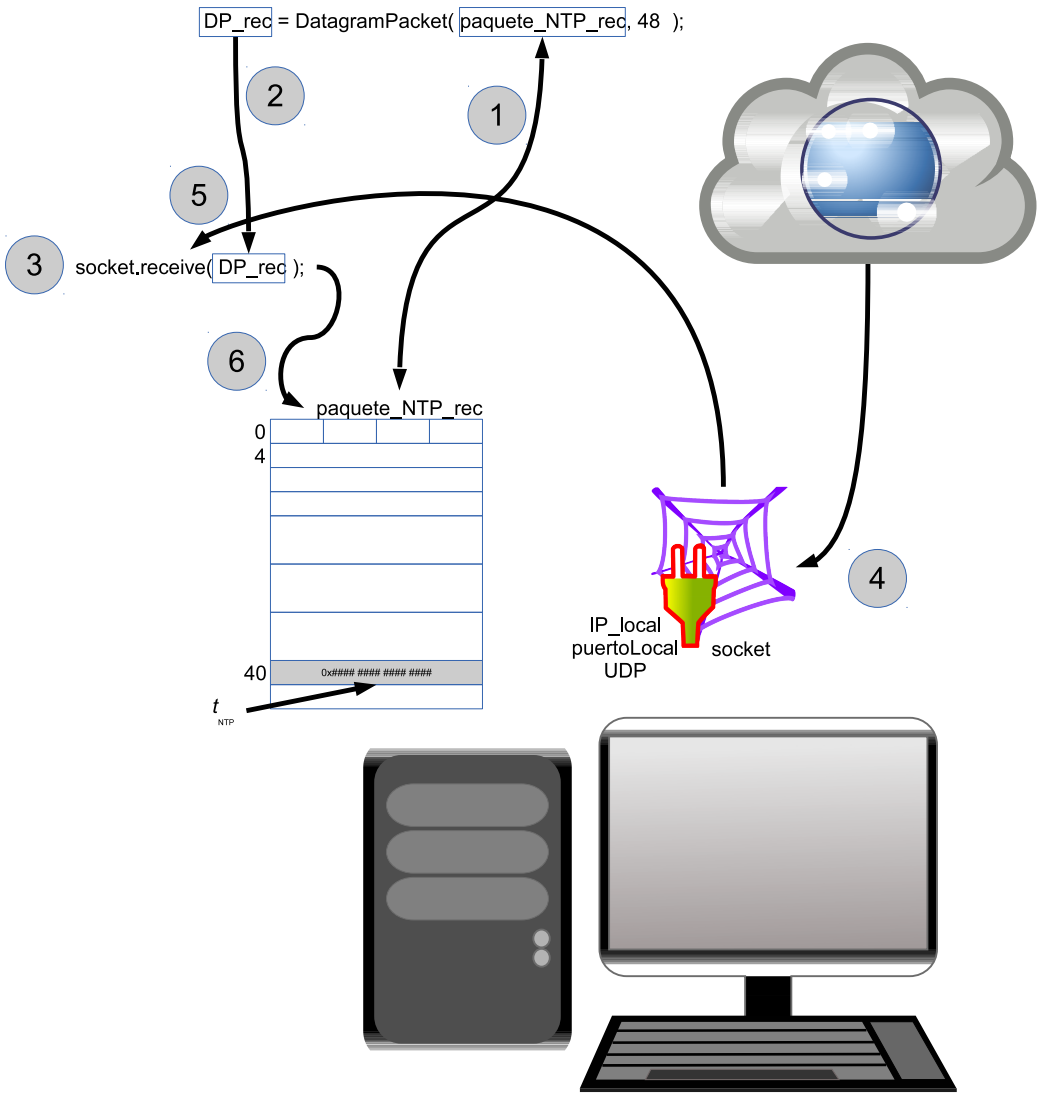

Figura 10: Preparación del `DatagramPacket` de recepción y recepción de la respuesta