Práctica 3.2.1: Implementación (parcial) de un cliente TFTP

Descripción:

En esta práctica el/la estudiante tendrá que implementar en Python un cliente (parcial) de la aplicación TFTP. En concreto tendrá que implementar la opción de bajar un fichero del servidor (opción *Read Request* o *RRQ*).

Material:

Para hacer esta práctica se dispone de los siguientes recursos:

1. Fichero con el código fuente del cliente parcialmente desarrollado (*tftp_cli_rrq.py*). Éste es el fichero que habrá que modificar para desarrollar la práctica.
2. Además se dispone de un servidor TFTP ya en ejecución en la máquina *dif-linuxserver.ehu.es* y puerto 50069 que el/la estudiante podrá usar para probar su cliente.

Protocolo de aplicación

El protocolo de la aplicación TFTP es bastante sencillo y ya ha sido explicado en clase. Todos los detalles los podemos encontrar en el RFC 1350. De todas formas, a continuación se resume el procedimiento RRQ:

1. El cliente envía el comando RRQ, el nombre del fichero a descargar e indica que lo quiere hacer en modo binario. Formato: El valor 1 en binario en 2 bytes para codificar el comando, el nombre del fichero, el valor 0 en binario en un byte para indicar el fin del nombre del fichero, la palabra "*octet*" (modo de descarga en binario) segida de un 0 en binario en un byte.
2. Si todo va bien, el servidor debería responder con un paquete DATA: Código 3 en binario en 2 bytes, el número de bloque en binario en 2 bytes (empezando por 1) y el siguiente bloque de datos del fichero de 512 bytes.
3. Para confirmar la recepción de datos, el cliente responde con un ACK: Código 4 en binario en 2 bytes y el número de bloque recibido en binario y en 2 bytes.

Los pasos 2 y 3 se repetirán hasta que el servidor envíe el último bloque de datos, que siempre será de menos de 512 bytes (incluído el tamaño 0).

Se ha de tener en cuenta que el puerto del que responde el servidor en el paso 2 podría ser distinto al que el cliente ha enviado el comando RRQ en el primer paso. Debido a esto es necesario que el cliente responda en el paso 3 al puerto usado por el servidor en el paso 2 (y no al puerto usado en el paso 1).

En esta práctica, puesto que tanto cliente como servidor estarán en la misma red local, la probabilidad de pérdida de paquetes es muy baja. Debido a esto, y para simplificar el trabajo, supondremos que no se perderán paquetes y, por tanto, no tomaremos ninguna medida para afrontar este caso.

Servidor

Habrá un servidor TFTP en ejecución en la máquina y puerto indicadas anteriormente. Tendrás que intentar que tu cliente se comunique con este servidor para comprobar su correcto funcionamiento. Este servidor tendrá disponibles los siguientes ficheros y sería interesante que

comprobaras el correcto funcionamiento de tu cliente con todos ellos:

- file511: Un fichero de 511 bytes.
- file512: Un fichero de 512 bytes .
- file513: Un fichero de 513 bytes.
- file: Un fichero de 11.754 bytes.

Cliente

El programa del cliente a implementar recibirá 2 parámetros en la línea de comandos: 1) Nombre DNS o dirección IP del servidor del que queremos bajar el fichero y 2) Nombre del fichero a bajar.

El código del cliente parcialmente ya implementado contiene:

- Los *import* necesarios
- Unas constantes para definir los comandos
- Tratamiento del número de parámetros de la línea de comandos
- Creación del socket UDP
- Gestión del fichero a escribir (apertura y cierre)
- Bucle principal para bajar un fichero
- Aviso final con tamaño del fichero bajado y velocidad de descarga conseguida

Además hay una serie de comentarios, allí donde corresponde, para indicar lo que falta por hacer para completar el cliente. Estos te serán de ayuda, siguiéndolos uno por uno.

Dinámica de trabajo

1. Terminar de implementar el cliente y probarlo con el servidor disponible.

Fichero *“tftp_cli_rrq.py”*:

```
#!/usr/bin/env python3

import sys
import socket
import time

NULL  = b'\x00'
RRQ   = b'\x00\x01'
WRQ   = b'\x00\x02'
DATA  = b'\x00\x03'
ACK   = b'\x00\x04'
ERROR = b'\x00\x05'

PORT = 50069
BLOCK_SIZE = 512

def get_file(s, serv_addr, filename):
      start = time.time()

      f = open(filename, 'wb')
      """A COMPLETAR POR EL/LA ESTUDIANTE:
      Enviar al servidor la petición de descarga de fichero (RRQ)
      """
      expected_block = 1
      while True:
```

```
            """A COMPLETAR POR EL/LA ESTUDIANTE:
            Recibir respuesta del servidor y comprobar que tiene el código
correcto (DATA), si no, terminar.
            Comprobar que el número de bloque es el correcto (expected_block),
si no, volver al comienzo del bucle.
            Escribir en el fichero (f) el bloque de datos recibido.
            Responder al servidor con un ACK y el número de bloque
correspondiente.
            Si el tamaño del bloque de datos es menor que BLOCK_SIZE es el
último, por tanto, salir del bucle.
            Si no, incrementar en uno el número de bloque esperado
(expected_block)
            """

      f.close()
      elapsed = time.time() - start
      bytes_received = (expected_block - 1) * BLOCK_SIZE + len(data)
      print('{} bytes received in {:.2e} seconds ({:.2e}
b/s).'.format(bytes_received, elapsed, bytes_received * 8 / elapsed))

if __name__ == '__main__':
      if len(sys.argv) != 3:
            print('Usage: {} server filename'.format(sys.argv[0]))
            exit(1)

      s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
      serv_addr = (sys.argv[1], PORT)

      get_file(s, serv_addr, sys.argv[2])
```

Posibles mejoras

1. Si en algún momento, el cliente recibe un mensaje de ERROR del servidor, mostrar en pantalla el código y el mensaje de error.
2. Que el programa acepte un tercer parámetro opcional que sea el nombre del fichero con el que el usuario quiere guardar en disco el fichero a bajar. Si no se indica este parámetro, el nombre a usar será el original del fichero.
3. Implementar la opción de subir un fichero al servidor (opción *Write Request* o *WRQ*). Ten en cuenta que ahora de alguna forma tendrá que poder indicar el usuario si lo que quiere es subir o bajar el fichero indicado. Para ello puedes añadir un parámetro más a la entrada del programa indicando la operación a realizar.