Universidad Nacional Autónoma de México

Facultad de Ciencias

Redes de computadoras

Programación de un cliente de HTTP v1.1

Profesor: Paulo Contreras Flores
Ayudante: Virgilio Castro Rendón
Ayudante Lab: Daniel Campuzano Barajas

1. Programar un cliente de la versión 1.1 del protocolo HTTP con el lenguaje de programación Python, se hará sin el uso de módulos complementarios como *httplib* o alguno parecido, aunque si será necesario usar otros módulos como *socket* o *sys*. Se conectará a cualquier servidor Web usando el protocolo HTTP.

2. Al final de este documento en la figura 2, se muestra el código esqueleto como propuesta del programa.

3. Deberá investigar cómo se conforma la solicitud HTTP de un cliente a un servidor, al menos la solicitud enviada al servidor por el programa cliente deberá incluir los siguientes parámetros:

   - Método, URL y versión del protocolo HTTP, por ejemplo *GET /imagen.jpg HTTP/1.1*, los dos primeros valores los proporcionará el usuario del programa.
   - Host, proporcionado por el usuario.
   - User-Agent, proporcionado por el usuario a partir de una lista de opciones.
   - Accept, este valor puede ser fijo
   - Accept-Charset, este valor puede ser fijo
   - Accept-Encoding, proporcionado por el usuario.
   - Accept-Language, este valor puede ser fijo
   - Connection, proporcionado por el usuario.

4. El programa final recibirá al menos seis argumentos cuando se ejecute, de la forma
   `$ python clientHTTP.py host http_method url user_agent encoding connection`
   en donde

   - `host`, corresponde a la dirección IP del servidor HTTP o a su nombre de dominio, por ejemplo *www.fciencias.unam.mx*.
   - `http_method`, corresponde al método de HTTP que se usará para enviar la solicitud al servidor, para este cliente solamente se usará *HEAD* o *GET*.

- `url`, corresponde al archivo o recurso solicitado al servidor web, por ejemplo / para directorio raiz, o *imagen.jpg*.
- `user_agent`, tendrá tres opciones numeradas (del 1 al 3), cada opción corresponderá a un *user agent* diferente definido por el programador.
- `encoding`, corresponde al parámetro de la solicitud de la codificación de la respuesta, por ejemplo *gzip*, *deflate* o *identity*.
- `connection`, se refiere a la forma del establecimiento de la conexión, por ejemplo *keep-alive* o *close*.

por ejemplo

```
$ python clientHTTP.py www.fciencias.unam.mx GET / 1 identity close
```

5. Es importante aclarar que los parámetros deben de separarse por un retorno de carro (*carriage return*, CR) y por un salto de línea (*linefeed*, LF), y que la solicitud de HTTP debe de terminar de la misma forma con un retorno de carro y con un salto de línea. En la siguiente figura se muestra el tráfico de Wireshark en la que se observa estos delimitadores con su valor en ASCII, del lado izquierdo de la salida de los datos de Wireshark se encuentra los valores en hexadecimal de estos datos, y del lado derecho se encuentra su interpretación en código ASCII, si la interpretación no corresponde a un caracter imprimible, Wireshark colocará un punto. En los recuadros de color verde se señalan el retorno de carro y el salto de línea, en color azul los simbolos que conforman los parámetros de HTTP, figura 1.

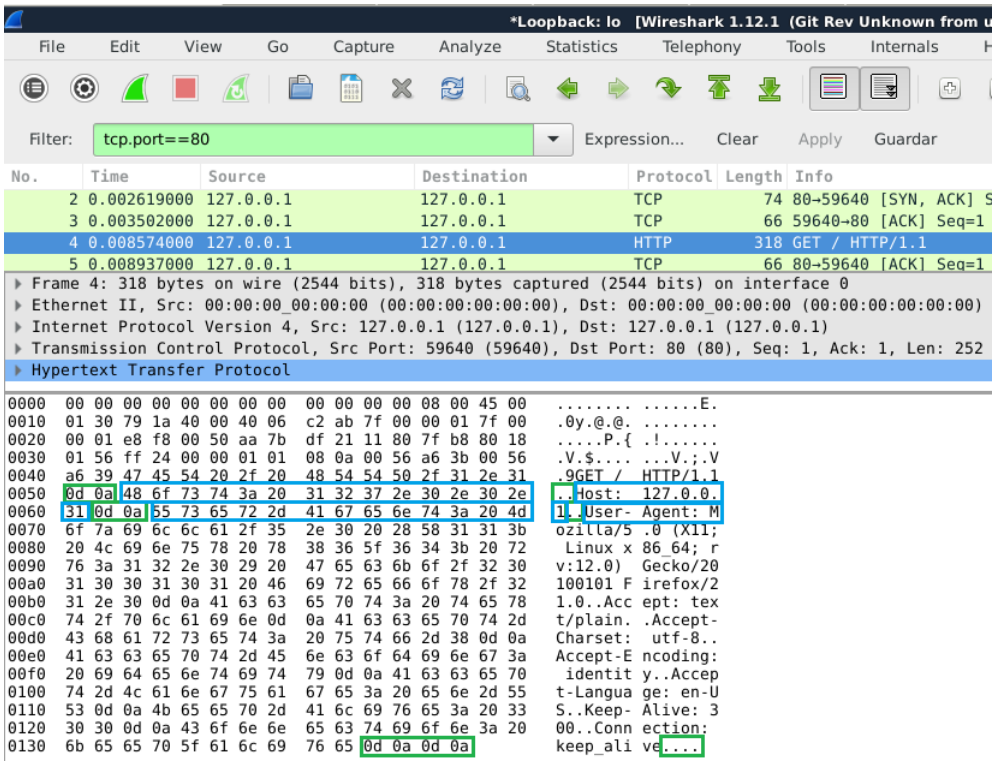

Figura 1: Interpretación ASCII - hexadecimal de la solicitud de HTTP.

6. La respuesta HTTP enviada por el servidor deberá ser mostrada por el cliente en pantalla (en la terminal en donde se ejecutó el cliente). Puede ser sólo la respuesta HTTP en algunos casos, y en otros la respuesta HTTP junto con el archivo solicitado, ambos en pantalla. Pruebe solicitar varios recursos.

7. Incluir una función sencilla de ayuda que muestre cómo se debe de ejecutar el programa y las opciones de *User-Agent* con las que se cuenta.

8. Durante la programación del cliente se recomienda el uso de Wireshark para visualizar el tráfico de la comunicación con el servidor y detectar posibles fallas en las solicitudes de HTTP, o en otros aspectos. Se recomienda colocar un filtro para que sólo se muestren las conexiones dirigidas al puerto 80 y a la dirección IP del servidor Web.

9. Entregue el código fuente, junto con un breve reporte en donde muestre con varios ejemplos diferentes el correcto funcionamiento de su programa. Además, conteste en el reporte lo siguiente:

   - Investigue para que se usan los métodos HEAD, GET, POST, PUT, DELETE.
   - Investigue los diferentes códigos de estado que usa HTTP, enliste las categorías.
   - Investigue el uso del campo *encoding*.
   - Investigue el uso del campo *connection*.

```python
# Modulos de Python
import socket
import sys


def proccessArguments():

    # Recibe de la linea de comandos un argumento,
    # la direccion IP del servidor o nombre de dominio,


    return arguments



def constructHTTPRequest(host_server):


    return HTTP_request



def TCPconnection(host_server, HTTP_request):

    # Crea un socket de TCP
    s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)


    # Conexion del cliente al servidor dado,
    # en el puerto 80 para HTTP
    s.connect((host_server,80))


    # Envia la peticion HTTP al servidor
    s.send(HTTP_request)


    # Mientras reciba informacion del servidor, la guardara
    # en HTTP_response, e imprimira en pantalla
    while True:
            HTTP_response = s.recv(1024)
            if HTTP_response == "": break
            print HTTP_response,


    # Una vez que la recepcion de informacion ha terminado
    # se cierra la conexion con el servidor
    s.close()


    print "\n\nConexion con el servidor finalizada\n"


arguments    = proccessArguments()
HTTP_request = constructHTTPRequest(*arguments)
TCPconnection(arguments[0],HTTP_request)
```

Figura 2: Esqueleto de código propuesto.