PROGRAMACIÓN EN RED (SOCKETS)

Contenido

- Presentación
  - Objetivos y entorno de desarrollo
  - Servicio web simplificado
    - Especificaciones del protocolo
      - Mensajes
      - Ejemplos de dialogo
    - Gestión de datos en el servidor

Objetivos y entorno de desarrollo

- El objetivo de esta práctica es implementar un aplicación en red como
  - Usuario del nivel de transporte y
  - Según el modelo cliente-servidor
- Se implementará un servicio web simplificado
- Entorno de desarrollo
  - Estación de trabajo con S.O. Debian GNU/Linux 9 (*stretch*) (nogal.usal.es)
  - Sockets de Berkeley
  - Lenguaje de programación C

Especificaciones del protocolo

- El servicio que vamos a implementar se corresponde con la versión 1.1 del protocolo HTTP definido en la RFC 2616 (http://www.ietf.org/rfc/rfc7235.txt)
- No obstante, nuestro servidor no implementará todo el protocolo, sino únicamente un subconjunto muy reducido de éste
- Además, HTTP se proporciona sobre TCP, pero nosotros realizaremos también una versión para UDP

Mensajes HTTP (I)

- HTTP emplea dos tipos de mensajes:
  - Peticiones de los clientes a los servidores
  - Respuestas de los servidores a los clientes
  - Ambos poseen una estructura similar formada por tres campos:
    - Línea inicial que se incluye siempre
    - una o más cabeceras (en la versión 1.1 solo una es obligatoria)
    - Cuerpo que no siempre estará presente
- La línea inicial y la cabecera de los mensajes HTTP son siempre líneas de caracteres terminadas con un par CR-LF (retorno de carro "\r" (ASCII 13 (0x0D)), - avance de línea "\n" (ASCII 10 (0x0A)))
- La cabecera finaliza con una línea que solo contenga CR-LF

Mensajes HTTP (II)

- Para definir el tamaño del cuerpo del mensaje se utiliza la cabecera
  - Content-Length: 3495 (en bytes)
  - El cuerpo comienza después de la línea con solo CR-LF que da fin a la cabecera.

Mensajes HTTP (III)

- **Órdenes del cliente al servidor**

  GET /index.html HTTP/1.1<CR><LF>

  Host: www.usal.es<CR><LF>

  Connection: keep-alive<CR><LF> (la conexión permanece abierta)

  <CR><LF> (Línea en blanco enviada también por el navegador)

- **Respuestas del servidor**
  - **Correcta**
    - HTTP/1.1 200 OK<CR><LF>
    - Server: Servidor de Nombre_alumno<CR><LF>
    - Connection: keep-alive<CR><LF>
    - Content-Length: 54
    - <CR><LF>
    - <html><body><h1>Universidad de Salamanca</h1></body></html>

Mensajes HTTP (III)

- **Errores**
  - No existe el objeto
    - HTTP/1.1 404 Not Found<CR><LF>
    - Server: Servidor de Nombre_alumno<CR><LF>
    - Connection: keep-alive<CR><LF>
    - Content-Length: 38
    - <CR><LF>
    - <html><body><h1>404 Not found</h1></body></html>
  - Orden errónea (algo distinto a GET …)
    - HTTP/1.1 501 Not Implemented<CR><LF>
    - Server: Servidor de Nombre_alumno<CR><LF>
    - Connection: close<CR><LF>
    - Content-Length: 38
    - <CR><LF>
    - <html><body><h1> 501 Not Implemented </h1></body></html>

Mensajes HTTP (IV)

- Ejemplo de dialogo (lo que ve el cliente):
  - Cliente> GET /index.html k
    - <html><body><h1>Web de la Universidad de Salamanca</h1></body></html>
  - Cliente> GET /ejemplo.html k
    - <html><body><h1>404 Not Found</h1></body></html>
  - Cliente> DAME /index.htm
    - <html><body><h1>501 Not Implemented</h1></body></html>
  - (Se cierra la comunicación)

- Nuestros clientes podrán solicitar varios objetos sin que se cierre la conexión para cada petición especificando la cabecera Connection: keep-alive
  - Esta es una característica opcional de la versión 1.1 del protocolo
  - Si no se indica nada su valor es Close; es decir, para cada petición se establece, se atiende y se cierra la comunicación con el cliente
  - Cuando no se deseen más objetos se omitirá esta cabecera o se seleccionará el valor Connection: close. Cerrando de esta forma la comunicación con el servidor

Requisitos (III)

- Lo que realmente viaja por la red (los mensajes del protocolo)
  - Cliente> GET /index.html k
    - Petición del cliente al servidor:
      - GET /index.html HTTP/1.1<CR><LF>
      - Host: url_servidor<CR><LF>
      - Connection: keep-alive<CR><LF>
      - <CR><LF>
    - Respuesta del servidor:
      - HTTP/1.1 200 OK<CR><LF>
      - Server: Servidor de Nombre_alumno<CR><LF>
      - Connection: keep-alive<CR><LF>
      - Content-Length: 54
      - <CR><LF>
      - <html><body><h1>Universidad de Salamanca</h1></body></html>
  - Cliente> GET /ejemplo.html k
    - Petición del cliente al servidor:
      - GET /ejemplo.html HTTP/1.1<CR><LF>
      - Host: url_servidor<CR><LF>
      - Connection: keep-alive<CR><LF>
      - <CR><LF>
    - Respuesta del servidor:
      - HTTP/1.1 404 Not Found<CR><LF>
      - Server: Servidor de Nombre_alumno<CR><LF>
      - Connection: keep-alive<CR><LF>
      - Content-Length: 38
      - <CR><LF>
      - <html><body><h1>404 Not Found</h1></body></html>

Requisitos (III)

- Lo que realmente viaja por la red (los mensajes del protocolo)
  - Cliente> DAME /index.htm
    - Petición del cliente al servidor:
      - DAME /index.htm HTTP/1.1<CR><LF>
      - Host: url_servidor<CR><LF>
      - Connection: close<CR><LF> (se puede omitir)
      - <CR><LF>
    - Respuesta del servidor:
      - HTTP/1.1 501 Not Implemented<CR><LF>
      - Server: Servidor de Nombre_alumno<CR><LF>
      - Connection: close<CR><LF> (se puede omitir)
      - Content-Length: 38
      - <CR><LF>
      - <html><body><h1>501 Not Implemented</h1></body></html>
  - (Se cierra la comunicación)

Gestión de los datos en el servidor

- El servidor buscará las páginas web solicitadas en un directorio relativo a la carpeta donde se encuentren los ejecutables llamado **www**
- Alojad allí varios ejemplos de archivos http de diferentes longitudes que servirán para validar el funcionamiento de nuestro servicio

Requisitos (IV)

- Programa Servidor
  - Aceptará peticiones de sus clientes tanto en TCP como en UDP
  - Registrará todas las peticiones en un fichero de "log" llamado peticiones.log en el que anotará:
    - Fecha y hora del evento
    - Nombre del ejecutable
    - Descripción del evento:
      - Comunicación realizada: nombre del host, dirección IP, protocolo de transporte, nº de puerto efímero del cliente
      - Una línea por cada objeto solicitado indicando si se ha atendido correctamente o en caso contrario especificar la causa del error.
      - Comunicación finalizada: nombre del host, dirección IP, protocolo de transporte, nº de puerto efímero del cliente
  - Se ejecutará como un “daemon”.
- Programa Cliente
  - Se conectará con el servidor bien con TCP o UDP
  - Leerá por parámetros el nombre del servidor, el protocolo de transporte TCP o UDP y en el caso de TCP una k si se desea que la conexión permanezca abierta o una c en caso contrario. En UDP siempre será una c puesto que no hay conexión. Ejemplos:
    - cliente nombre_o_IP_del_servidor TCP
    - cliente nombre_o_IP_del_servidor TCP
    - cliente nombre_o_IP_del_servidor UDP
  - Realizará peticiones al servidor como se ha indicado anteriormente
  - Realizará las acciones oportunas para su correcta finalización

Requisitos (II): pruebas

- Durante la fase de pruebas el cliente podrá ejecutarse como se muestra en el ejemplo de diálogo anterior, pero en la versión para entregar el cliente
  - Leerá de un fichero las órdenes que ha de ejecutar. El nombre del fichero lo recibirá como parámetro
  - Escribirá las respuestas obtenidas del servidor y los mensajes de error y/o depuración en un fichero con nombre el número de puerto efímero del cliente y extensión .txt

Requisitos (III): versión entregable

- Para verificar que esta práctica funciona correctamente y permite operar con varios clientes, se utilizará el *script* lanzaServidor.sh que ha de adjuntarse obligatoriamente en el fichero de entrega de esta práctica
- El contenido de lanzaServidor.sh es el siguiente:

```
# lanzaServidor.sh
# Lanza el servidor que es un daemon y varios clientes
# las ordenes están en un fichero que se pasa como tercer
  parámetro
servidor
cliente nogal TCP ordenes.txt &
cliente nogal TCP ordenes1.txt &
cliente nogal TCP ordenes2.txt &
cliente nogal UDP ordenes.txt &
cliente nogal UDP ordenes1.txt &
cliente nogal UDP ordenes2.txt &
```

Requisitos (IV): documentación

- Entregar un informe en formato PDF que contenga:
  - Detalles relevantes del desarrollo de la práctica
  - Documentación de las pruebas de funcionamiento realizadas