Jasper

1. Instalar Jasper

Enlace para bajar la herramienta:
ftp://ftp.cs.stir.ac.uk/pub/staff/kjt/software/jasper-1.3.tar.gz

Para instalarlo basta con descomprimir en el directorio deseado. Aparecerá la siguiente estructura:

html
: directorio para páginas web y el JAR del simulador

protocol, simulator, support
: carpetas para el código

makefile
: fichero *make* para gestionar el código

build.bat, clean.bat, help.bat, run.bat, spotless.bat
: ficheros batch para gestionar el código en MS-DOS

2. Ejecutar Jasper

- como applet

  Simplemente hay que abrir un navegador hacia el fichero *index.html* o el del protocolo que queramos que hay en el directorio *html* de nuestra instalación de Jasper. Se cargará el applet donde se pueden especificar los parámetros que admite el protocolo y hacer la simulación (el navegador debe permitir applets y JavaScript).

  Nota: Al ejecutar Jasper como applet no se pueden utilizar las opciones de guardar, cargar o imprimir una simulación. Éstas sólo están disponibles en caso de ejecutarlo como aplicación.

- como aplicación

  Para empezar, tenemos 2 opciones:

  - Usando el *makefile*

    Editar el fichero *makefile* y descomentar la línea correspondiente al sistema operativo que se esté utilizando, según se indica en la línea 16. Para Unix, habrá que hacer el siguiente cambio:

```
16 # classpath option: uncomment 1 for Windows, 2 for Unix, 3
for OpenStep
17
18 # CPATH = -classpath ".;.."
19 CPATH = -classpath .:..
20 # CPATH =
-classpath .:..:/usr/local/share/kaffe/Klasses.jar
```

Si el S.O. es **Unix**, al escribir `make` o `make help` en la línea de comandos desde el directorio raíz de Jasper aparece un sencillo menú de ayuda. Nosotros vamos a utilizar la última opción, que es la que nos permite ejecutar los protocolos:

```
make protocolo
```

donde *protocolo* es cualquiera de los protocolos de la lista de ayuda. La primera vez se compilarán todas las clases antes de ejecutarlo.

Si estamos en **Windows**, hay que abrir una ventana de MS-DOS y ejecutar primero el fichero *build.bat*. Esto compila las clases y construye el simulador. Después, para lanzar la simulación del protocolo '*protocolo*', sólo hay que escribir:

```
run protocolo
```

- Usando *javac*

  Desde el directorio raíz de Jasper, compilar todas las clases de los directorios *protocol*, *simulator* y *support*. Por ejemplo, para Unix:

  ```
  javac [-classpath .:..] protocol/*.java
  simulator/*.java support/*.java
  ```

  Una vez hecho esto, basta con escribir:

  ```
  java [-classpath .:..] simulator/ProtocolSimulator
  protocolo [opciones]
  ```

  donde protocolo es cualquiera de los protocolos de la lista y opciones son las que admite cada protocolo (ver lista más abajo).

→ En caso de que querer especificar el valor de los parámetros del protocolo en vez de dejar los que lleva por defecto, **sólo** se puede utilizar el comando `java` desde el raíz de Jasper:

```
java [–classpath .:..] simulator/ProtocolSimulator
protocolo [opciones]
```

Las opciones son las siguientes:

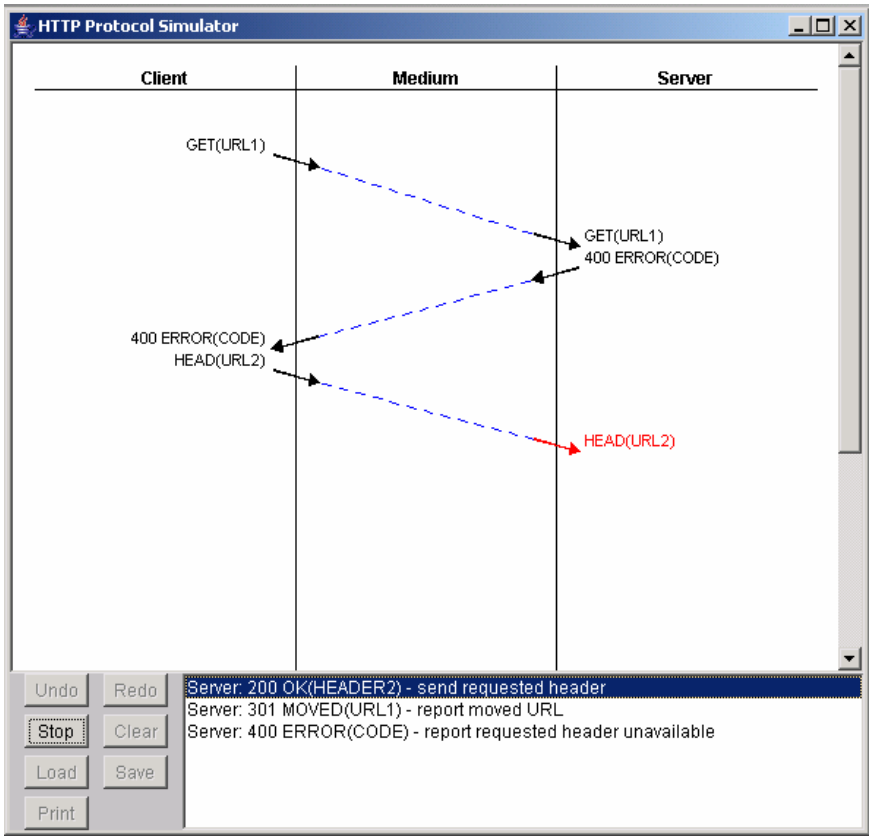
Protocolo	Parámetro	Tipo	Significado
ABP	(sin parámetros)		
ABRA	(sin parámetros)		
BOOTP	(sin parámetros)		
HTTP	(sin parámetros)		
IP	misordering	boolean	hace que el medio desordene los paquetes
	losarte	float	tasa de pérdida (0.0 = sin pérdidas – 1.0 = todo pérdidas)
	userMessageSize	int	tamaño de mensaje del usuario
	maxProtocolMessageSize	int	tamaño máximo de mensaje del protocolo
	maxMediumMessageSize	int	tamaño máximo de mensaje que transmite el medio
SMTP	(sin parámetros)		
SWP3 y SWP5 (Sliding Window Protocol, 3 o 5 columnas)	maxSeq	int	número máximo de secuencia
	winSize	int	tamaño de ventana
TCP	pushA	boolean	flag push de A
	pushB	boolean	flag push de B
	serviceAMessageSize	int	tamaño de paquete de A
	serviceBMessageSize	int	tamaño de paquete de B
	windowSizeA	int	tamaño de la ventana de recepción de A
	windowSizeB	int	tamaño de la ventana de recepción de B
	maxSendPacket	int	tamaño máximo de paquete que transmite el medio

TFTP	(sin parámetros)		
UDP	sourcePortA	int	puerto de salida de A
	sourcePortB	int	puerto de salida de B
	destPortA	int	puerto de destino para A (entrada en B)
	destPortB	int	puerto de destino para B (entrada en A)

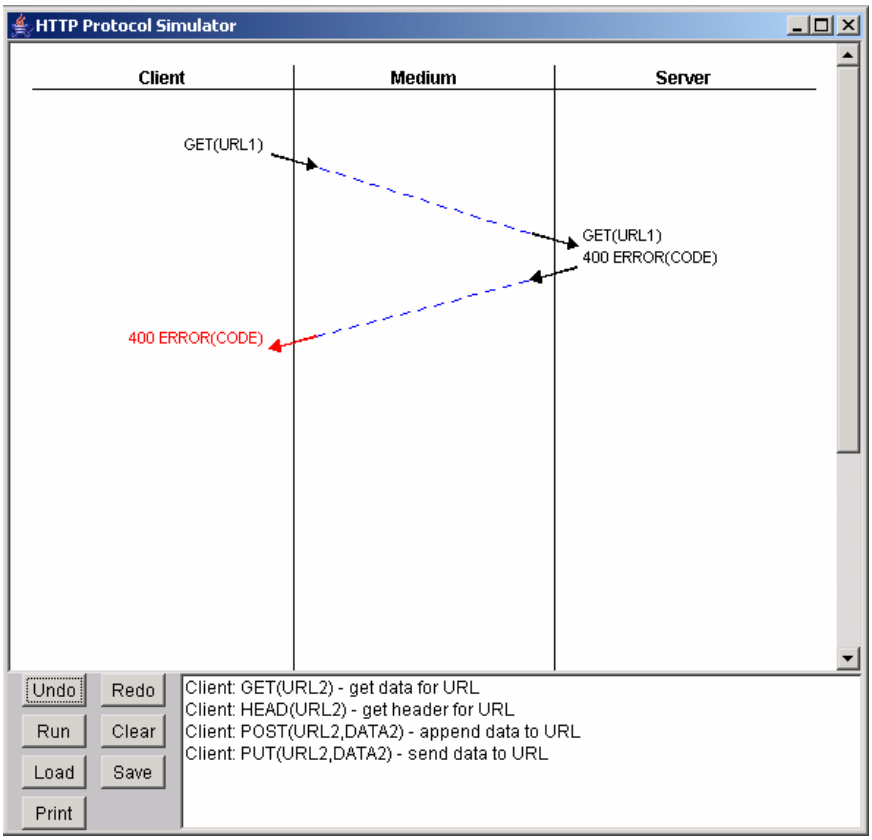
3. Usando Jasper

Vamos a ver las posibilidades con un ejemplo sencillo: HTTP

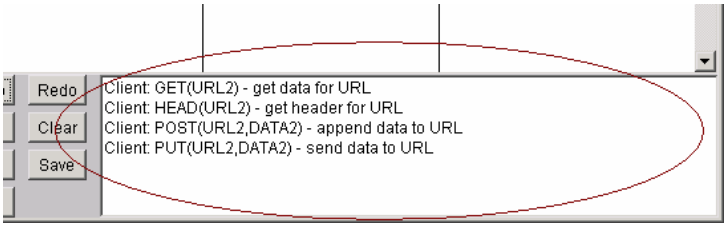
Al abrir la aplicación, podemos empezar pulsando el botón *Run*. Comienza una simulación con una serie de casos aleatorios hasta que pulsemos el botón *Stop*.



En cualquier momento en el que la simulación esté detenida, y siempre que haya trazas en ella, se puede utilizar *Clear* para borrarlo todo, o *Undo / Redo* para deshacer o volver a hacer el último paso.

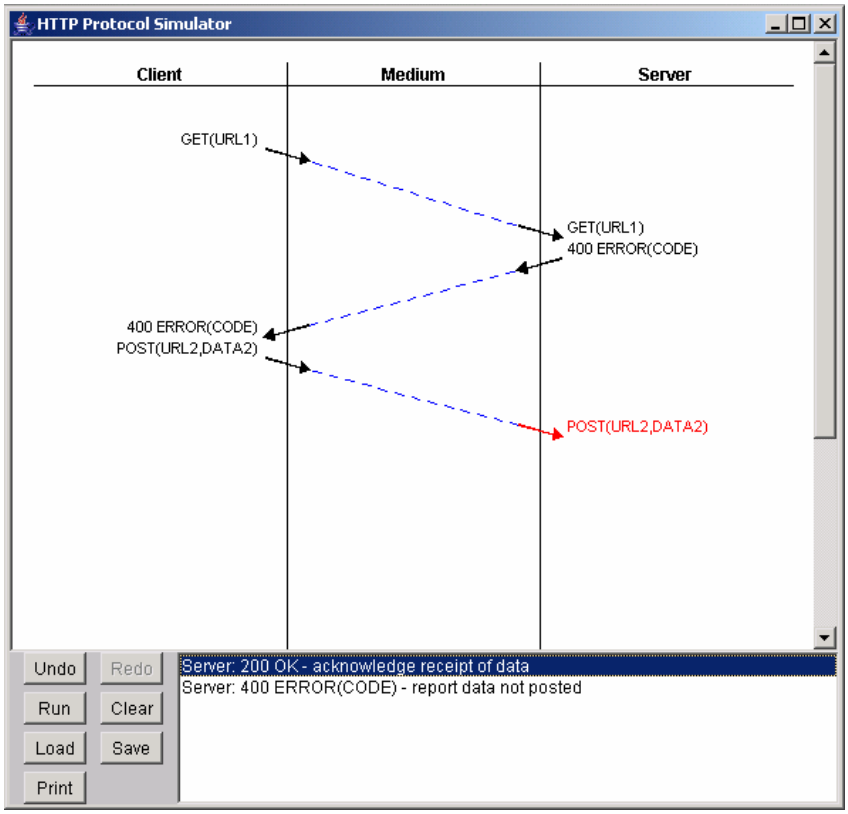


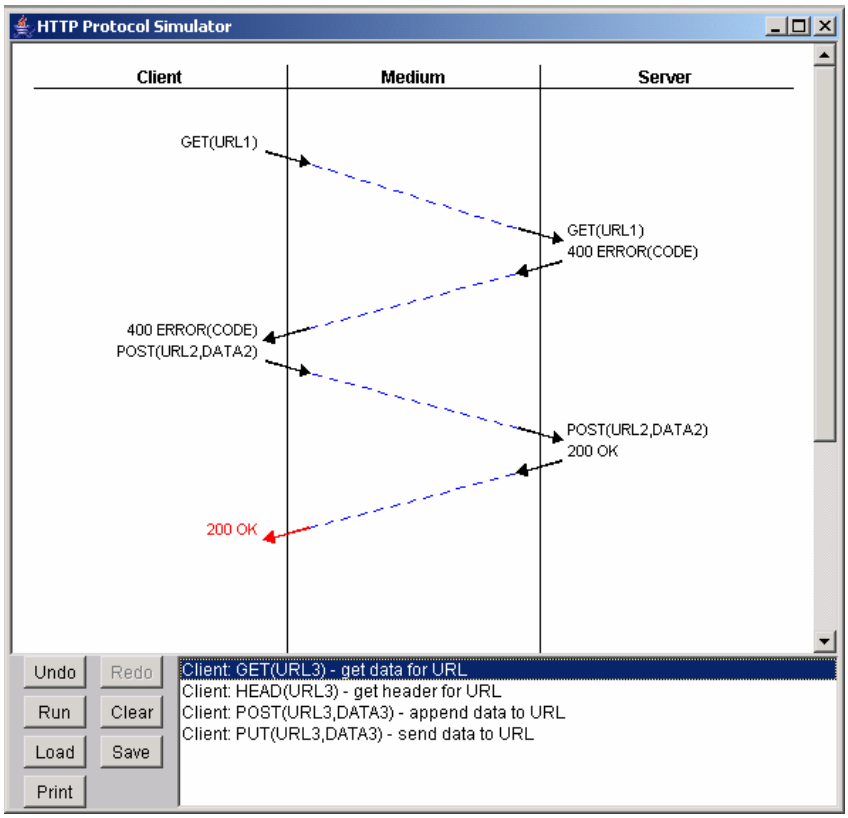
Por otro lado, podemos ir creando la simulación que deseemos paso a paso en vez de dejar que corra aleatoriamente, utilizando la parte de abajo a la derecha:



En cada momento, nos ofrece una lista con las posibilidades en ese punto de la simulación. Al pinchar sobre una de ellas, se pinta la traza correspondiente y la lista muestra los nuevos posibles mensajes, que dependen del que acabamos de elegir.

Por ejemplo, se acaba de enviar una petición POST(URL2, DATA2). En el cuadro de abajo aparecen las opciones 200 OK y 400 ERROR(CODE). Si pinchamos sobre 200 OK, se dibuja la traza y se abre una nueva lista de opciones, que corresponden a los mensajes que se pueden enviar en ese momento: cualquier petición nueva (GET, HEAD, POST o PUT).

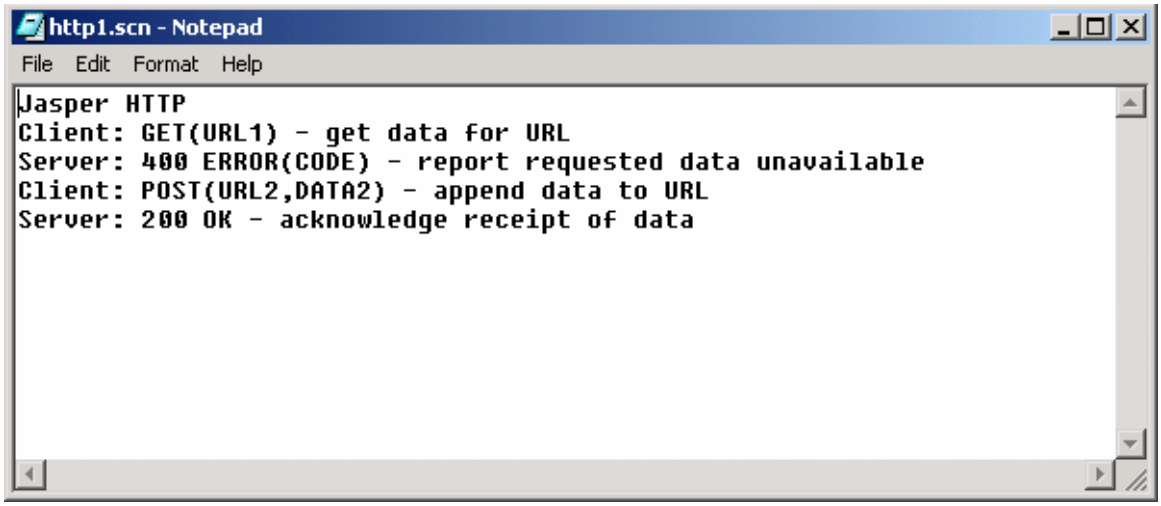




Una vez hecha la simulación, podemos guardarla (con extensión .scn) con el botón *Save* o imprimirla (por la impresora o en un fichero de impresión) usando *Print*. Cualquier fichero .scn guardado (o editado por nosotros) se puede cargar más tarde pulsando el botón *Load* y continuar simulando a partir de la última traza.

4. Ficheros .scn

Ejemplo de fichero .scn de Jasper:



Para crear un fichero .scn (*scenario*) mediante un editor de texto, los pasos a seguir son los siguientes:

1 La primera línea debe ser siempre:
`Jasper protocolo`

2 El resto de líneas tienen el formato:
`Emisor: mensaje [- comentarios]`
En cuanto a los mensajes, la flexibilidad de contenidos varía mucho entre la implementación de un protocolo y otro. Por ejemplo, en HTTP basta con escribir `Server: 200` para que aparezca el mensaje `200 OK` en la simulación, mientras que en SMTP hace falta escribir `Server: 250 Recipient OK` para que se pinte la traza, en vez de simplemente `Server: 250`.
A continuación se detalla la lista de comandos de HTTP y SMTP donde se indican los campos obligatorios y los variables.

HTTP	
Cliente	Servidor
`GET(URL#)`	`200 [OK] [(datos)]`
`PUT(URL#, datos)`	`301 [MOVED] (URL#)`
`POST(URL#, datos)`	`400 [ERROR] (texto)`
`HEAD(URL#)`	

SMTP	
Cliente	Servidor
TCP connect	220 Server ready
TCP disconnect	250 Server hello to client
HELO client	250 Sender OK
QUIT	550 Sender invalid
MAIL FROM sender[#]	250 Recipient OK
RCPT TO recipient[#]	550 Recipient invalid
DATA	354 Send mail
Mail message	250 Message accepted
	221 Server closing

NOTA: Jasper es sensible a mayúsculas y minúsculas.