



Escuela Politécnica Superior de Elche

SISTEMAS INFORMÁTICOS EN TIEMPO REAL

2º Ingeniería Industrial

PRÁCTICAS DE TCP-IP

Protocolo TCP

Luis Miguel Jiménez

Departamento de Ingeniería de Sistemas Industriales
Área de Ingeniería de Sistemas y Automática

© ISA-UMH

1. OBJETIVO

Esta práctica tiene como objetivo el estudio del protocolo TCP, analizando los mecanismos de conexión y desconexión, opciones de la cabecera, etc. Se experimentará así mismo con varios servicios (puertos) disponibles sobre este protocolo.

2. MATERIAL EMPLEADO

Se utilizarán los mismos equipos de la sesión práctica anterior, disponiendo el software **WireShark** como monitor de red.

Utilizaremos adicionalmente el programa **Socket Workbench** que nos permite establecer conexiones TCP, tanto en modo cliente como en modo servidor, utilizando la librería de sockets. (en la página web de la asignatura está disponible una versión de evaluación de este programa).

Las prácticas se realizan en grupos de dos personas disponiendo de 2 PCs (cliente y servidor). Los equipos del aula disponen de S.O. Windows XP.

3. PROGRAMA SOCKET WORKBENCH

El programa **Socket Workbench** permite establecer de forma sencilla una conexión TCP pudiendo actuar tanto en modo cliente como servidor. Nos permite transmitir cualquier tipo de datos y monitorizar las respuestas. Adicionalmente dispone de un modo de captura (*Pass Through*) que permite capturar los mensajes transmitidos por otros programas que utilicen conexiones TCP.

Adicionalmente dispone de varios de mensajes preconfigurados utilizados en diferentes protocolos (E-mail, HTTP).

Realizaremos en primer lugar la instalación del programa descargándolo de la página web indicada anteriormente. A continuación se describen los pasos básicos de su utilización.

- En el menú **Inicio/Programas** buscaremos la opción **Socket Workbench** y ejecutaremos el programa. En la figura 1 se muestra la pantalla inicial donde se nos pide el registro del programa. Pulsaremos la tecla **OK** sin indicar número de registro (en este modo el programa estará activo en modo evaluación por 30 días).
- Tras la confirmación aparecerá una ventana que nos permite seleccionar configuraciones predeterminadas (figura 2). Estas configuraciones se encargan de transmitir mensajes específicos de diferentes protocolos durante la conexión (http cliente, http-servidor, E-mail,...). Nosotros utilizaremos el programa en modo directo, sin ninguna opción preconfigurada, por lo que pulsaremos el botón **Close**.
- Aparecerá la ventana principal del programa. Antes de empezar a utilizar el programa vamos a cambiar la configuración de algunos parámetros para la realización de las prácticas posteriores. Desplegaremos el menú '**View**' (figura 3) y seleccionaremos la opción '**Options**'. Aparecerá una ventana de configuración (figura 4). En esta ventana seleccionaremos la etiqueta '**Socket**

Handling para acceder a la configuración del socket. Y desmarcaremos la opción **'Automatically attempt reconnection'** tal como se muestra en la figura. De este modo el programa no reintentará establecer una conexión cuando se produzca una desconexión desde el servidor.

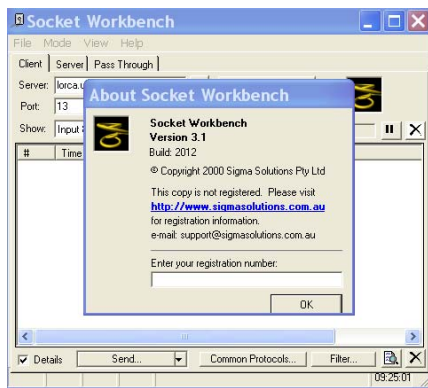


Figura 1 Arranque del programa

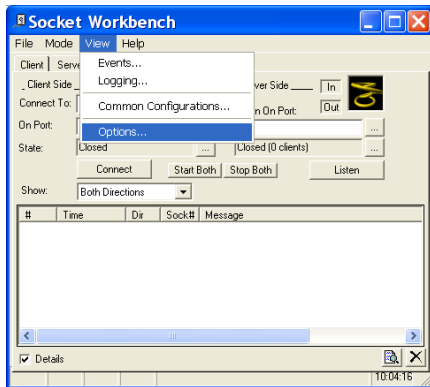


Figura 3 Configuración de opciones

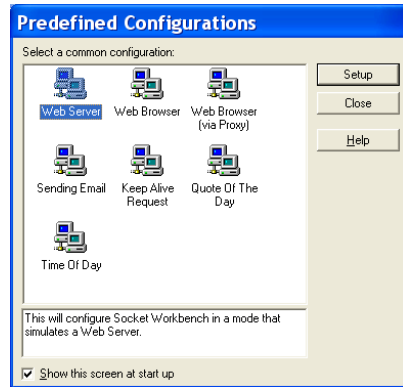


Figura 2 Configuraciones

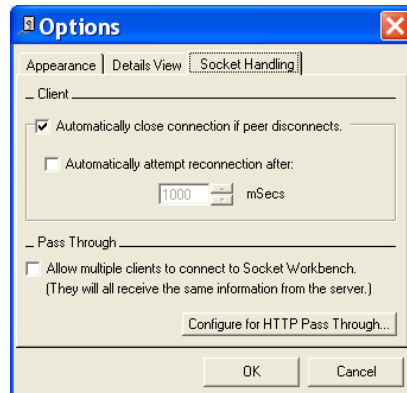


Figura 4 Opciones del socket

Uso básico del programa:

La figura 5 muestra la pantalla principal del programa

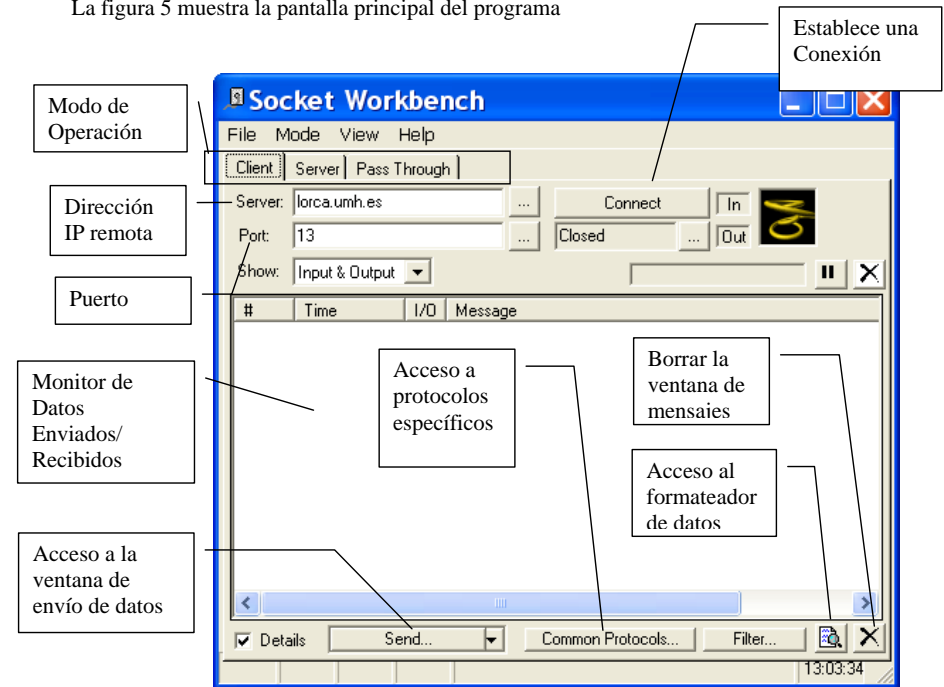


Figura 5 Pantalla principal del programa Socket Workbench (modo cliente)

Justo debajo del menú disponemos de tres etiquetas que dan acceso a los diferentes modos de operación:

- **Client:** el programa actúa como un cliente estableciendo conexiones y transmitiendo/ recibiendo datos.
- **Server:** el programa actúa como un servidor escuchando la red a la espera de una conexión desde un cliente.
- **Pass Through:** el programa actúa como un monitor de red permitiendo monitorizar los datos transmitidos por otros programas (Navegador Web, ...)

Durante el desarrollo de las prácticas lo usaremos en los dos primeros modos. En el modo cliente (figura 5) disponemos la opción de seleccionar la dirección IP (o nombre DNS) del servidor remoto y el puerto. El botón **Connect** nos permite iniciar una conexión mostrando en la ventana inferior todos los mensajes (segmentos) transmitidos y recibidos (no muestra los segmentos de conexión y desconexión, por lo que para su visualización utilizaremos el programa **Wireshark**)

Cuando el programa actúa como servidor (figura 6), solo nos permite seleccionar el puerto de escucha, activándose el proceso al pulsarse el botón **Listen**.

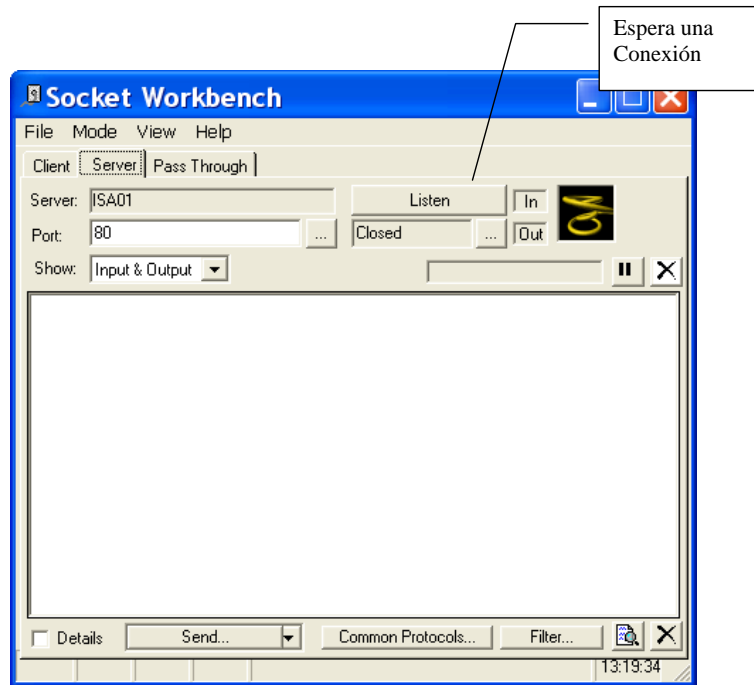


Figura 6 Pantalla principal del programa Socket Workbench (modo servidor)

4. SERVICIOS (PUERTOS)

El extremo servidor está formado por procesos que se mantienen continuamente escuchando puertos conocidos. A continuación describiremos los servicios soportados generalmente por cualquier implementación TCP/IP cuyos puertos utilizaremos en la práctica. Se indica para cada puerto su nombre simbólico (soportado por casi todos los clientes) y el número:

echo (7): el servidor devuelve lo que envía el cliente

discard (9): el servidor descarta todos los datos enviados por el cliente

daytime (13): el servidor devuelve la fecha y hora en formato legible y desconecta

time (37): el servidor devuelve la fecha y hora como un número de 32 bits que indica los segundos transcurridos desde (1/1/1900)

chargen (19): si el servicio es con conexión (TCP) envía un flujo continuo de datos aleatorios hasta que el cliente desconecta. Si el servicio es datagrama (UDP) envía un solo datagrama con números aleatorios cada vez que el cliente envía un datagrama.

quote (17): devuelve una cita como una cada de texto y termina la conexión.

En cada ejercicio deberá guardarse las tramas capturadas con el programa WireShark en un fichero para realizar posteriormente los informes de prácticas.

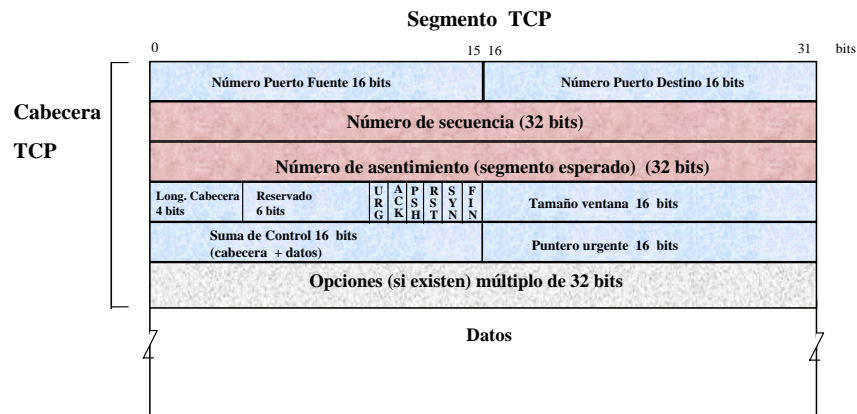
5. TAREAS

Se conectarán los equipos utilizando los concentradores suministrados para la práctica. Es importante no utilizar conmutadores ya que actúan como filtros impidiendo observar las tramas que circulan por la red.

Cuestión 1. Conexión/desconexión en el protocolo TCP

Descripción del Protocolo TCP:

El protocolo TCP (RFC 793) implementa un **servicio de flujo** (*stream*) de bytes orientado a la conexión asegurando la fiabilidad de los datos transmitidos y su secuenciamiento. Precisa de una etapa previa de conexión y una posterior de desconexión a la transmisión de datos. (*Ver apuntes*)



Formato de un segmento TCP

Vamos a estudiar el mecanismo de conexión y desconexión TCP:

CONEXIÓN:

1. El cliente (proceso que solicita la conexión) emite un datagrama de sincronismo (campo **Flag SYN** activado). Este segmento incluye la dirección IP y puerto del servidor con el cual desea establecer la conexión. Incorpora así mismo el número de secuencia inicial (**ISN**) del cliente. En el campo de opciones se puede transmitir el tamaño máximo de segmento (**MSS**). El segmento de conexión consume 1 byte aunque no incluya datos por lo que el siguiente número de secuencia será **ISN+1**.
2. Si el servidor (proceso que acepta conexiones) acepta dicha conexión emite un datagrama de sincronismo (campo **Flag SYN** activado) y asentimiento (campo **Flag ACK** activado). Este segmento incorpora el número de secuencia inicial (**ISN**) del servidor. El campo con el número de asentimiento coincidirá con el número de secuencia inicial del cliente más

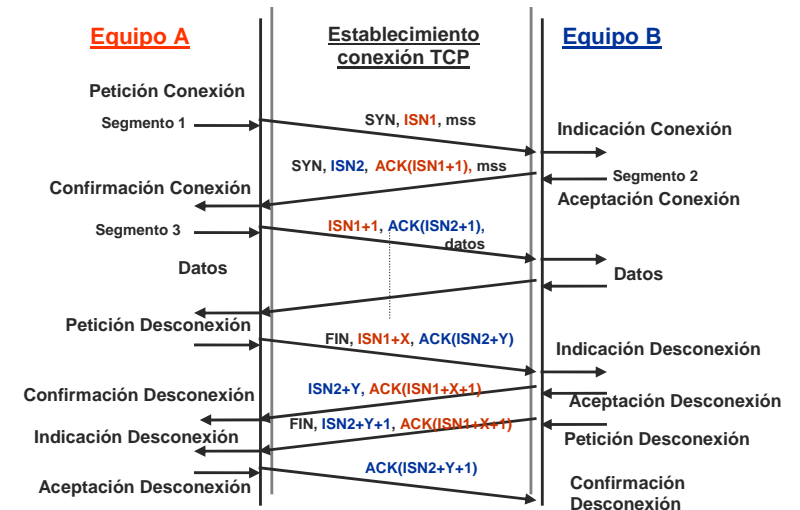
uno (ya que un segmento de conexión consume 1 byte). En el campo de opciones se transmite el tamaño máximo de segmento (**MSS**).

3. El cliente debe reconocer este asentimiento enviando un segmento normal (datos) con asentimiento (campo **Flag ACK** activado) al servidor indicando en el número de asentimiento el ISN del servidor más uno. Este segmento suele ser el primer segmento de datos.

DESCONEXIÓN:

1. El cliente o servidor emite un datagrama de finalización (campo **Flag FIN** activado). Como en el segmento de conexión, consume un byte aunque no contenga datos.
2. El otro extremo (servidor o cliente) acepta dicha desconexión en el sentido de transmisión indicado emitiendo un datagrama de asentimiento (campo **Flag ACK** activado). El número de secuencia de asentimiento corresponderá al número de secuencia de la petición de desconexión más 1. Este segmento puede incorporar datos dirigidos en el sentido contrario

El proceso anterior se debe realizar también en el otro sentido de la comunicación. Ambos sentidos son independientes y el cierre de una parte de la conexión no afecta a la otra, ya que solo indica que no se aceptarán más segmentos en ese sentido.



Tareas a realizar:

Al iniciar la práctica deberemos consultar los siguientes datos de configuración de nuestra conexión de red. Para ello ejecutaremos el siguiente comando en una consola:

Windows NT/2000/XP/Vista: **ipconfig /all**

A partir de los datos mostrados apuntaremos lo siguiente:

- Dirección IP/inet <IP_HOST_CLIENTE>:
- Dirección IP/inet <IP_HOST_SERVIDOR>:

En algunas cuestiones se utilizarán dos PCs por lo que deberemos apuntar las IPs de ambos equipos. Uno actuará en modo cliente y el otro en modo servidor.

Cuestión 1: Mecanismo de Conexión/Desconexión TCP

1. Arrancar **WireShark** en la máquina monitor activando el siguiente filtro de captura:

Filtro Captura=> **tcp and ip host 193.147.145.166**

2. Activar el la captura de paquetes.
3. Ejecutar el programa **Socket Workbench** en modo **cliente** (figura 5):

Opciones SocketWorkbench: modo cliente

- Servidor: **193.147.145.166**
- Puerto: **9 (discard)**

Pulsaremos el botón **Connect**. Se establece una conexión con la máquina remota.

A continuación pulsaremos el botón **Disconnect** que habrá aparecido sustituyendo al anterior.

De este modo hemos provocado una conexión y una desconexión TCP sin transferencia de datos.

4. En el programa **WireShark** parar la captura (**guardar**) y estudiar las tramas involucradas.

Discusión:

- a) ¿Qué tipos de segmentos TCP aparecen ?
- b) Justificar los segmentos aparecidos en función de la descripción previa del protocolo, analizando el mecanismo de conexión y la doble desconexión.
- c) Justificar el contenido de los diferentes campos de la cabecera TCP en función del tipo de segmento. Hacer hincapié especialmente en los campos *Número de secuencia* y *flags*
- d) Discutir el contenido del campo opciones: tipos de opciones, rellenos,...
- e) Utilizando la información de la cabecera de la primera trama de conexión averiguar el puerto '*efimero*' del cliente.

Cuestión 2. Servicios TCP

Vamos a comprobar a continuación la funcionalidad de otros puertos: **echo** y **daytime**,

Tareas a realizar:

1. Mantener los mismos filtros de la cuestión anterior.
2. Activar el modo de captura en el program **Wireshark**.
3. Ejecutar el programa **Socket Workbench** en modo cliente (figura 5):

Socket Workbench : modo cliente

Opciones:

- Servidor: **193.147.145.166**
- Puerto: **13 (daytime)**

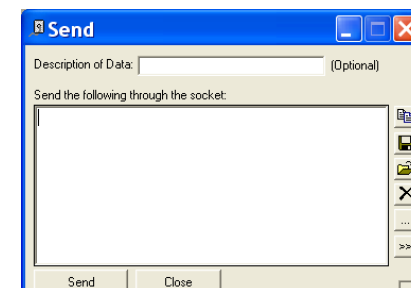
Pulsaremos el botón **Connect**. Se establece una conexión con la máquina remota.

En la ventana de monitorización habrá aparecido una trama de datos y se produce una desconexión automática

4. En el programa **WireShark** parar la captura (**guardar**) y estudiar las tramas involucradas.

Discusión:

- a) Analizar el mensaje enviado.
- b) Estudiar las diferencias en el mecanismo de desconexión. ¿que proceso inicia la primera desconexión? Estudiar el mecanismo de numeración de tramas y asentimientos.
- c) Repetir el proceso anterior con el puerto de **echo (7)**. En este caso la conexión quedará abierta hasta que realicemos la desconexión. Utilizaremos el botón **Send** para transmitir una trama al otro extremo de la conexión. Aparecerá una ventana donde podemos escribir el texto para el campo de datos y realizar su envío.



Cuestión 3. Timeout en el establecimiento de una conexión

Para esta práctica utilizaremos dos PCs de ensayos conectados al mismo concentrador:

- 1 PC con el programa **Socket WorkBench** en modo cliente
- 1 PC con el programa **Socket WorkBench** en modo servidor

El PC que actúe en modo cliente ejecutará adicionalmente el monitor de red **Wireshark**

Filtro Captura=> tcp and ip host <IP_HOST_SERVIDOR>

Tareas a realizar:

1. Consultaremos la dirección IP del PC que actúe como servidor. En este equipo ejecutaremos la aplicación **Socket Workbench** en modo servidor (figura 6).

Socket Workbench: modo servidor (PC servidor)

Opciones:

- Puerto: **6666**

Pulsaremos el botón **Listen**. Se queda a la espera de una conexión

2. Desde el PC de ensayos del cliente ejecutar el programa **Socket Workbench** en modo cliente (figura 5):

Socket Workbench: modo cliente (PC cliente)

Opciones:

- Servidor: **<IP_HOST_SERVIDOR>**
- Puerto: **6666**

Pulsaremos el botón **Connect**. Se establece una conexión con la máquina remota.

De este modo comprobaremos que la conexión es correcta. Desconectaremos la conexión con el botón **Disconnect**

3. Activar el modo captura en el equipo monitor (cliente) con **Wireshark**.
4. **Desconectar** el cable de red del PC que actúa como servidor para simular un fallo en la red.
5. Realizaremos de nuevo la conexión desde el cliente como en el punto 2.
6. En el programa **Wireshark** parar la captura (**guardar**) y estudiar las tramas involucradas.

Discusión:

- a) Estudiar las tramas de conexión que envía el cliente y no son respondidas desde el servidor.
- b) Analizar los tiempos de espera entre intentos de conexión (*timeout*).
- c) Reconectar el cable del servidor y comprobar la conexión.

Cuestión 4. Timeout en la retransmisión de tramas

Vamos a estudiar el mecanismo de recuperación de tramas perdidas en la transmisión de datos TC. Para esta práctica utilizaremos dos PCs de ensayos conectados al mismo concentrador como en la cuestión anterior:

Tareas a realizar:

1. Activar el modo captura en el programa **Wireshark** en el equipo cliente.
2. Consultaremos la dirección IP del PC que actúe como servidor. En este equipo ejecutaremos la aplicación **Socket Workbench** en modo servidor (figura 6).

Socket Workbench: modo servidor (PC servidor)

Opciones:

- Puerto: **6666**

Pulsaremos el botón **Listen**. Se queda a la espera de una conexión

3. Desde el PC de ensayos del cliente ejecutar el programa **Socket Workbench** en modo cliente (figura 5):

Socket Workbench: modo cliente (PC cliente)

Opciones:

- Servidor: **<IP_HOST_SERVIDOR>**
- Puerto: **6666**

Pulsaremos el botón **Connect**. Se establece una conexión con la máquina remota.

4. Enviaremos desde el cliente una trama de datos (texto) a través de la conexión (ver cuestión 2 para los detalles)
5. Desconectaremos el cable de red del PC que actúa como servidor para simular un fallo en la red.
6. Enviaremos desde el cliente otra trama de datos (texto) a través de la conexión.
7. Esperaremos 5 segundos y reconectaremos el cable para que la transmisión se recupere.
8. En el programa **Wireshark** parar la captura (**guardar**) y estudiar las tramas involucradas.

Discusión:

- a) Analizar los tiempos de espera entre intentos de retransmisión. (*timeout*).

Cuestión 5. Protocolo HTTP

Para terminar este estudio y como introducción a los protocolos de nivel aplicación, vamos a utilizar el programa socket WorkBench para analizar el funcionamiento de un protocolo de nivel aplicación HTTP. En este caso no será necesario el uso del PC monitor ya que utilizaremos la ventana de mensajes disponible en el propio programa.

Tareas a realizar:

1. Desde el PC de ensayos (Windows) ejecutar el programa **Socket Workbench** en modo cliente (figura 5):

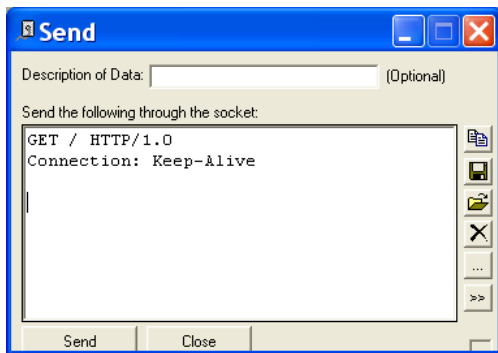
Socket Workbench: modo cliente (PC cliente)

Opciones:

- Servidor: 193.147.145.166
- Puerto: 80 (http)

Pulsaremos el botón **Connect** Se establece una conexión con la máquina remota.

2. Utilizaremos el botón **Send** para transmitir una trama al otro extremo de la conexión. Escribiremos la siguiente petición del protocolo HTTP (**nota debemos incluir dos retornos de línea al final para señalar el final de petición**). El servidor nos devolverá la página web principal en formato HTML.



Discusión:

- a) Analizar el mensaje enviado. Resaltar las marcas de fin de mensaje
- b) Analizar las tramas recibidas