La interfaz Hombre-Máquina en el Mundo de Internet

Roberto Cárdenas C

RESUMEN

En este artículo se estudia las implicaciones que tiene el poder tener sistemas de control conectados por una interred usando protocolos estandarizados y se analiza en especial su repercusión en la interfaz hombre-má quina.

Palabras Clave: JavaScript, Servidor Web, navegador, ANSI/EIA-709.1, formulario, HTML.

ABSTRACT

This paper is focused to investigate the implications of control systems networked by an internet using stantdard protocols and its influence in human-machine interface.

Key Words: JavaScript, Web, navegador, ANSI/EIA-709.1, HTML.

INTRODUCCIÓN

Los sistemas de control en interred combinan las redes de control (recientemente especificadas en ANSI/EIA-709.1) y redes IP para crear soluciones poderosas a un amplio número de aplicaciones de supervisión y control.

Algunas de las actuales tendencias y tecnologías tienen un impacto sustancial sobre los sistemas de control en interred del futuro, tales como Calidad del Servicio (QoS), Redes Virtuales Privadas (Vpns), medios físicos (ATM, CATV, Gigabit Ethernet), protocolos de seguridad y tecnologías de objetos distribuidos.

Estas tecnologías nos permitirán tener arquitecturas de red como la que se observa en la figura 1

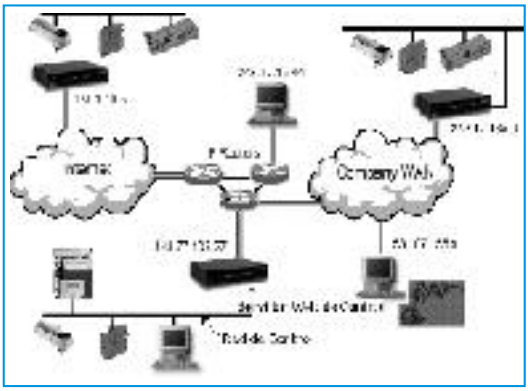

Figura 1. Red de Control en Interred

Esta arquitectura da nacimiento a un nuevo tipo de dispositivo que he llamado Servidor Web de Control el cual debe ofrecer el "throughput" alto que demandan las aplicaciones de control de procesos, automatización de edificios, generación de energía, transporte o telecomunicaciones.

Estas máquinas deben hacer un "tunneling" de paquetes ANSI/EIA-709.1 sobre la red IP para obtener sistemas abiertos e interoperables.

Al analizar un sistema de control tradicional (sin interred), vemos que una parte esencial es la interfaz hombre-máquina, como se muestra en la Figura 2:

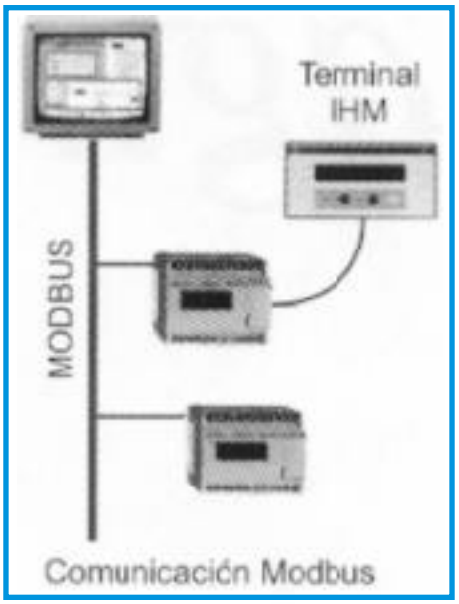

Figura 2. Sistema de control sin Interred

Estas interfaces funcionan sobre terminales dedicadas y despliegan gráficas, textos, alarmas y cuentan con ayudas sensibles al contexto, matemáticas, análisis de tendencias, diagnósticos, control de procesos y software de aplicación como el PL7 [1].

Pero en el nuevo contexto (con interred) ¿Qué pasa con la interfaz hombre-máquina, la cual o las cuales pueden estar situadas en cualquier parte del mundo? ¿En qué cambian las comunicaciones y en especial los protocolos ante este nuevo escenario?

En este artículo intentaremos encontrar algunas respuestas.

Servidores Web de Control

Como ya se mencionó, este nuevo escenario da lugar a la aparición de nuevas máquinas, las cuales están conectando las redes de control con el mundo IP. La pregunta es ¿cómo enviarán estas máquinas la información hacia las interfaces hombre-máquina?.

Visto desde el punto de vista de la pila TCP/IP la respuesta casi obvia es que estas máquinas se deberán portar como un servidor Web y así permitir el acceso a variables que representan la temperatura, población, velocidad, etc, a través de un navegador Web.

Esto permite el acceso a datos de supervisión y control desde cualquier sitio sin necesidad de herramientas especiales de software sobre LANs, WANs o Internet.

Si se requiere diagnóstico remoto, calibración de equipo, supervisión de alarmas, el servidor Web integral simplificaría con estos métodos el acceso a cualquier parte del sistema de control.

Desde la perspectiva de la red IT este tipo de servidor sería visto como un host IP típico y como tal deberá soportar protocolos estándar como TCP/IP, UDP, DHCP, SNMP (MIB II), ICMP, SNTP, TOS, MD5, HTTP y FTP. Adicionalmente, los parámetros de direccionamiento, utilización de ancho de banda IP y seguridad podrán ser ajustados vía la red IP.

La pregunta es ¿cómo enviarán estas máquinas la información hacia las interfaces hombre-máquina?.

Procesamiento Web de los datos

La existencia de un lenguaje estándar HTML soportado por los navegadores, la sencillez del mismo, el uso de editores HTML y la virtud de que sus documentos o páginas permitan interactuar con los usuarios, hacen que este lenguaje sea implementado en los servidores.

Los documentos de Web pueden agruparse en tres categorías, dependiendo del momento de cambio de la información del documento. La información de un documento estático permanece sin cambio hasta que el autor la modifica.

La información de un documento dinámico puede cambiar en el momento en que es solicitado al servidor. La información presentada en un documento activo puede cambiar una vez que se ha cargado el documento en un visualizador.

Habitualmente se han empleado programas CGI (Common Gateway Interface) para realizar el procesamiento en los equipos servidores y, de esta manera, mostrar al usuario información procesada que se puede considerar dinámica.

Un ejemplo de esta situación es la existencia de páginas Web con formularios que los usuarios rellenan; posteriormente se hacen búsquedas en bases de datos alojadas en el servidor y los resultados se transmiten al cliente en formato HTML [2].

Las páginas ASP (Active Server Pages) de Microsoft proporcionan un medio sencillo y eficiente de programar páginas Web activas en el servidor, ofreciendo un modelo alternativo a las aplicaciones CGI.

Con el fin de ofrecer un soporte tecnológico que permita implementar páginas Web dinámicas en cliente se han desarrollado los lenguajes de guiones (Scripts), los applets de Java y los ActiveX de Microsoft.

Por diversas razones históricas, económicas, técnicas y de seguridad, la solución más utilizada son los Scripts con sintaxis Java.

Los applets de Java y las aplicaciones Actives funcionan como programas que se transmiten por la red desde los servidores a los clientes y se ejecutan en estos últimos. Sus mayores problemas son la seguridad (en el caso de los Actives), la necesidad de recursos en el cliente y la dificultad de desarrollo.

JavaScript [3] está diseñado para poder representar y manipular la información mediante un navegador, pero no es capaz de leer un archivo, ni de enviar datos al servidor o al computador del usuario. Esto significa que no se puede escribir un programa en JavaScript, que lea un directorio de un computador o que lo borre. En cambio sí es posible crear un archivo de comandos que supervise y grabe la sesión del uso del navegador, que acumule o guarde en un archivo lógico las páginas visitadas y lo que se ha introducido (por ejemplo contraseñas).

Además de ser basado en objetos (tiene una colección de objetos), JavaScript está controlado por eventos. Siempre que sucede algo en una página Web, se produce un evento. Evento que puede ser todo: la pulsación sobre un botón, el movimiento del puntero del ratón cuando se carga una página o cuando se transmite un formulario, etc.

La ventaja de usar JavaScript es que la demanda de cómputo en el cliente es mucho más baja y el tiempo de descarga para la página es también mucho menor ya que solo es necesario descargar una pequeña cantidad de texto en vez de una aplicación entera.

Interactividad con el usuario

Una de las características fundamentales de una interfaz hombre-máquina es su interactividad con el usuario. En un ambiente Web que usa HTML esto lo podemos obtener haciendo uso de formularios, los cuales están destinados a recoger datos para que estos sean tratados en un servidor destinado para tal efecto.

El concepto de incrustar información de estado en los URL (Universal Resource Locator) se ha generalizado y extendido [4]. El HTML permite que el servidor declare que un documento es una forma con uno o más elementos que debe suministrar el usuario.

Para cada elemento, la forma especifica un nombre. Al seleccionar un hipervínculo en esta forma, el visualizador agrega una cadena de argumentos al URL con cada uno de los elementos nombrados. Así un programa dinámico puede obtener valores suministrados por el usuario.

Por ejemplo, suponga que una forma tiene tres elementos llamados AA, BB y CC , y que el usuario ha suministrado los datos sí, no, tal vez.

La ventaja de usar JavaScript es que la demanda de cómputo en el cliente es mucho más baja y el tiempo de descarga para la página es también mucho menor

Al reunirlos en un URL, el visualizador crea la cadena:

?AA=sí,BB=no,CC=talvez

que agrega el visualizador al URL indicado.

Al igual que con el resto de los elementos de HTML[2], se dispone de dos etiquetas para indicar el comienzo y el final de un formulario: <FORM> y </FORM>. Todas las definiciones de los elementos que componen un formulario deben ir dentro de dichas etiquetas.

La etiqueta <FORM> dispone de una serie de atributos, y en esta ocasión comentaremos los mas relevantes para nuestro estudio:

- Name: (Nombre) indica el nombre del formulario que sirve para referenciarlo con lenguajes destinados a tal efecto (JavaScript), VbasicScript),

- Method: (Valor) hace referencia al método utilizado para enviar el contenido del formulario al servidor. Puede tomar los valores get (se envían los datos recogidos junto con el nombre del programa que los va a tratar en forma de par nombre=valor) o post (se producen dos conexiones con el servidor y se envían los datos como si se tratara de un archivo, teniendo éste que recuperarlos de la entrada estándar),

- Action: (URL) hace referencia al programa que se ejecutará en el servidor y recogerá los datos.

Los formularios disponen, de forma genérica, de tres clases de entrada de datos que son: Input, Textarea y Select. A su vez Input se divide en otros muchos tipos de entrada de datos. Veamos específicamente el tipo Submit.

El tipo Submit tiene el siguiente aspecto:

```
<INPUT Type="Submit"  Name="Nombre" Value="Valor">
```

Este tipo tiene el aspecto de un botón que cuando es pulsado provoca el envío de los datos del formulario al servidor y programa indicado en el atributo Action de la etiqueta <FORM>. En este caso, Valor representa el texto que aparece en el interior del botón, como indicación al usuario de la acción que realiza dicho botón.

Para desplegar información en el cliente podemos usar el método document.write del objeto document. Este método escribe el contenido de una cadena de caracteres directamente en la página.

Por ejemplo el siguiente Script escribe la palabra Encendido en la página:

```
<SCRIPT>
var x = "Encendido";
 document.write (x)
</SCRIPT>
```

En este ejemplo la variable x se ha declarado con la palabra clave var. Una de las diferencias más importantes entre JavaScript y la mayoría de los lenguajes de programación consiste en el hecho de que JavaScript no conoce tipos de datos explícitos. Por esta razón, no hay posibilidad alguna de definir una variable de manera que acepte o bien un valor entero (Integer), una cadena o un valor de coma flotante.

Cualquier variable de JavaScript puede contener cualquiera de estos tipos de datos y esto llega tan lejos que una variable puede ser interpretada de diferentes modos en contextos con dependencias funcionales distintas.

Ahora veamos un ejemplo de una página HTML que lee valores de entrada de variables de red de control y permite fijar un valor para una variable de salida de la red de control [4]:

Como se esperaba se usa FORM y se observa el funcionamiento del objeto ILONWEB. Con la función ShowValue se despliegan los valores de las variables enviadas por el servidor y con TextField se capturan los datos que van a ser transmitidos con Submit. Cuando se produce el contacto con el servidor, esta codificación produce el siguiente resultado en la pantalla del usuario:

La anterior codificación pretende mostrar el código HTML necesario para transferir información. En la Figura 3 podemos observar una interfaz hombre-máquina elaborada en JavaScript, la cual es un perfeccionamiento del ejemplo anterior:

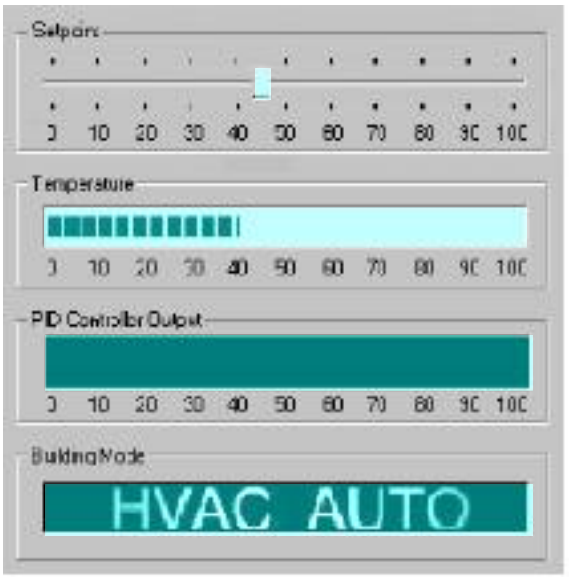

Figura 3. Interfaz Hombre-Máquina elaborada en JavaScript

Los cuatro "controles" de esta página no son controles del todo realmente; son imágenes gráficas que son manipuladas en el navegador usando JavaScript.

Experimentando con Internet

Examinemos un experimento realizado con la participación activa de los estudiantes de la Maestría en Teleinformática,[6].

Como se observa en la figura 4, existe en teoría un equipo de control conectado a un Servidor Web, los cuales en conjunto simulan el Servidor Web de Control comentado anteriormente.

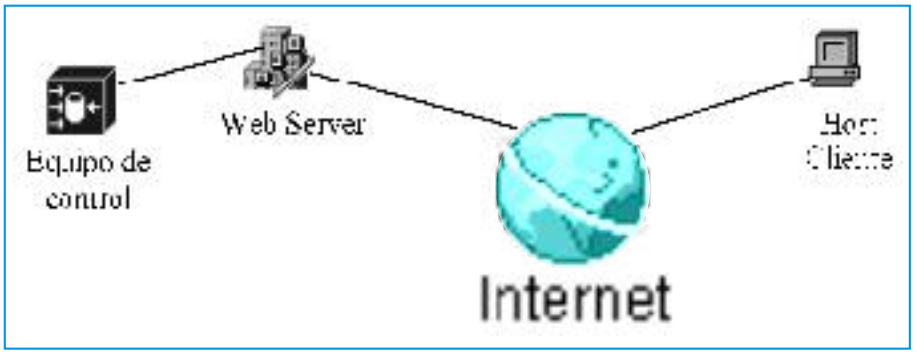

Figura 4. Simulación de Red Control en Interred

El equipo de control está actualizando una página HTML que contiene información acerca de un estado binario "Encendido" o "Apagado".

En un ejemplo anterior vimos cómo despliega el estado "Encendido" en la pantalla del usuario. Igualmente el estado "Apagado" se desplegará con el siguiente código:

```
<SCRIPT>
var x = "Apagado";
document.write (x)
</SCRIPT>
```

Estos códigos están incluidos en la codificación de la página HTML que se despliega.

Para actualizar la página en el Servidor Web se siguió el siguiente procedimiento:

- Para almacenar la página HTML control.htm que contiene el estado "Encendido" se creó en el computador que simula el equipo de control el directorio c:/control_on

- Para el estado "Apagado" se crea el directorio el directorio c:/control_off, el cual almacena la página control.htm que tiene el código de "Apagado".

La actualización la hacemos en el servidor ftp.geocites.com, el cual es un servidor FTP al que se tiene acceso con el USER y PASSWORD que se eligieron en el momento de cargar por primera vez páginas Web. Este es un servidor de acceso público.

El contacto con este servidor se hace desde la consola de comando de DOS, la cual se obtiene ejecutando command. En la línea de comando digitamos ftp ftp.geocities.com y se establece la conexión.

Una vez hemos hecho contacto con ftp.geocities.com actualizamos la página para los estados "Encendido" o "Apagado" con procedimiento de la figura 5:

Figura 5. Actualización de página HTML

El computador cliente o en el cual se despliega la interfaz hombre-máquina hace contacto por medio del navegador con la página www.geocities.com/carlosomarramos/control.htm la cual será desplegada al usuario.

Sin embargo nada se habría logrado hasta este momento si la página desplegada en el cliente no se estuviera refrescando automáticamente para desplegar cualquier cambio en el equipo de control.

Para hacer esto usamos el objeto window con la propiedad location y la función reload( ).

El objeto window es en todo documento el objeto de jerarquía superior y por tanto está en la cima de la jerarquía de objetos de JavaScript. Este objeto maneja todo lo relacionado a la ventana de despliegue.

La propiedad location contiene información sobre la dirección URL del documento que aparece.

Reload() hace una solicitud al servidor para que envíe la página ubicada en la dirección URL. El código de esta solicitud es:

window.location.reload().

El anterior código servirá para crear la función reloadMe() así:

<SCRIPT>
function reloadMe()
{
window.location.reload()
}
</SCRIPT>.

Pero ¿Cuándo realiza el cliente la petición de recarga?

Para ello se crea un temporizador con el método de window, timerID.

Este método retrasa en milisegundos la ejecución de una instrucción. Si escribimos:

timerID=setTimeout("reloadMe()",4000),

se estará ejecutando reloadMe() cada 4000 milisegundos o 4 segundos.

La codificación de la página HTML es:

```
<html>

 <head>
  <meta http-equiv="Content-Type"content="text/html;
charset=iso-8859-1">
  <meta name="GENERATOR" content="Microsoft
FrontPage Express 2.0">
   <title>REDESII</title>
 </head>

 <body bgcolor="#FFFFFF">

  <script>
    functionreloadMe()
      {
          window.location.reload()
      }
  </script>

<p align="center"><font color="#FF0000"
size="7"><strong>REDES II</strong></font></p>

<p align="center"><font color="#800000"
size="6"><strong>PROYECTO DE
CONTROL</strong>
</font></p>

<p></p>

<P><font color="#000000" size="5">El equipo
conectado a geocities.com está:</font>
<font color="#008040" size="5">

<SCRIPT>
      var x="Encendido";

      //este es el ON
      document.write(x)
</SCRIPT>

</font><font color="#FF0080" size="5">

<SCRIPT>
      //var x=" Apagado";

      //este es el OFF
      //document.write(x)
  </SCRIPT>

  </font></p>
  <script>
       TimerID=window.setTimeout("reloadMe()",
        4000)
  </script>

  <form>
    <p align="center">
  <input type="button" name="stop_reloading_button"
   value="Parar Recarga"
   onclick="clearTimeout(timerID)"> </p>
  </form>
 </body>
</html>
```

Este código despliega lo siguiente en el cliente (Figuras 6a y 6b), según el caso (encendido o apagado):

Figura 6a. Despliegue en el cliente del estado Encendido.

Figura 6b. Despliegue en el cliente del estado Apagado.

El tiempo transcurrido entre el reporte de los estados en 6a y 6b fué de 7 segundos.

Para parar la recarga usamos el controlador de eventos onClick, el cual activa clearTimeout(timerID) cuando se pulsa el botón ve Parar Recarga. ClearTimeout(timerID) borra timerID colocado con la instrucción setTimeout.

El tiempo medido entre la actualización de la página y el despliegue en el cliente fluctuó entre 7 y 45 segundos. Las sesiones FTP y WWW se corrieron en el mismo computador y el acceso a Internet se hizo con Multinet a través de una lí nea telefónica a una velocidad de 16.800 bps.

CONCLUSIONES

El nuevo enfoque (con interred) redefine el funcionamiento interno de la interfaz hombre-máquina ya que la vuelve ubicua y se debe aplicar a aquellas situaciones donde los retardos de decenas de segundos sean tolerables. Estas interfaces funcionarán con codificación HTML (o similar), aprovechando su bidireccionalidad, su posibilidad de programación en lenguajes como JavaScript, y su despliegue gráfico, mientras que el protocolo universal de comunicaciones sería el HTTP (HyperText Transfer Protocol), el cual es el encargado del transporte de las páginas WWW (World Wide Web).

Como se demuestra aquí, una posible opción es que las máquinas de control actualicen servidores FTP (File Transfer Protocol), para que finalmente la información sea consultada en servidores WWW. Se espera que con el progreso de la interconexión masiva de medios físicos como fibra óptica o métodos de transmisión avanzados (DSL), la implantación de ATM en los backbones y el aumento de la velocidad en los servidores, los tiempos de respuesta se reduzcan sensiblemente.

BIBLIOGRAFÍA

[1] TELEMECANIQUE, MM7 Integrated Man-Machine dialogue, Rueil-Malmaison-France,1992

[2] BOBADILLA J. y ALONSO S., HTML Dinámico a través de ejemplos, Santa Fé de Bogotá: Alfaomega S.A., 2000

[3] KOLBECK R., El gran libro de JavaScript, Barcelona: Marcombo S.A., 1997

[4] COMER, D.E., Redes de Computadoras, Internet e Interredes, México: Prentice-Hall Hispanoamericana, S.A., 1997

[5] ECHELON, i.LON™ 1000 Internet Server Starter Kit, Palo Alto-CA: Echelon, 2000

[6] Ramos C. y Guillén E.
www.udistrital.edu.co/comunidad/estudiantes/posgrados/posgrado

Roberto Cárdenas C.
Ingeniero Electrónico, U. Distrital, Magíster en Teleinformática, U. Distrital, Profesor de la Facultad de Ingeniería en la Maestría en Teleinformática U. Distrital
e-mail: bitek@multi.net.co