OPTIMIZACIÓN DE TIEMPOS EN EL ACCESO REMOTO VÍA INTERNET A PLANTA PILOTO PARA PRÁCTICAS

Manuel Domínguez, Ángel Alonso, Perfecto Reguera, José Javier Glez. Pacho, Juan José Fuertes
Instituto de Automática y Fabricación.
Universidad de León, 24071, León, España
Email:{diemdg, dieaaa, diepra, diejgp, iafjfm}@unileon.es

Resumen

En este trabajo se estudia la problemática asociada a los tiempos de acceso de los usuarios que se conectan vía Internet a una planta industrial, y qué influencia ejercen en su operación y manejo remoto.
Con el fin de tener una serie de estimaciones fiables , se realiza una medición de tiempos que permite extraer conclusiones y facilitar así, mediante un rediseño de todo el sistema, una mejora sustancial y una metodología en el desarrollo de este tipo de arquitecturas.

Palabras Clave: Teleoperación, internet, control, retardo, multihilo, JAVA.

1. INTRODUCCION

En las anteriores jornadas de trabajo sobre enseñanza vía Internet de la Ingeniería de Sistemas y Automática, se abordaron las diferentes utilidades que puede aportar Internet como herramienta de uso docente para el estudio de disciplinas de tipo tecnológico relacionadas con la Ingeniería Industrial, en las cuales, además de la dificultad propia debida a los contenidos, se une la necesidad de poder manejar equipos industriales complejos, no disponibles en muchas ocasiones por su elevado coste.

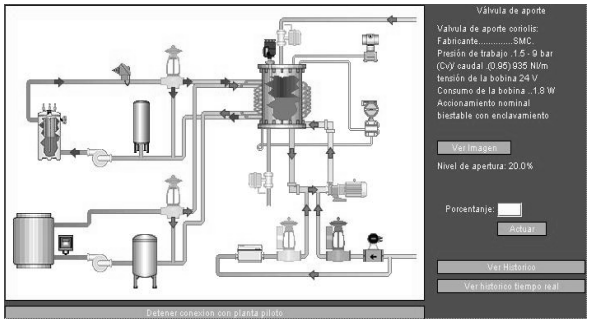

Figura 1: sinóptico de Planta Piloto.

El trabajo que aquí se presenta parte de la utilización como equipo complejo de una planta piloto, con un reactor de 50 litros, circuitos de calentamiento, enfriamiento y recirculación e instrumentación industrial, que es accesible vía Internet utilizando una estructura cliente-servidor de tres capas, ver Figura 2, en la que tanto el cliente como el servidor se construyen con JAVA y la capa intermedia se haya constituida por una base de datos [1]. Se dispone de un interface de usuario como el representado en la Figura 1, con un sinóptico para realizar sobre él la monitorización de 24 variables además de la operación remota sobre la misma.

La idea que se pretende transmitir en el presente trabajo es la problemática asociada con la representación y actuación remota en tiempo real de variables físicas a través de Internet y la estructura que se ha utilizado para conseguir una operatividad razonable, los problemas que se han observado, sus soluciones, conclusiones obtenidas y líneas futuras que deben ser acometidas.

2. PROBLEMAS DETECTADOS

Al utilizar la forma de proceder indicada en el apartado anterior, aparecieron una serie de problemas que afectaban fundamentalmente a la interactividad del usuario:

- Lentitud de carga del applet en el ordenador del usuario debido a la estructura gráfica del sinóptico.

- Retardo importante desde que el usuario solicita una acción sobre un elemento físico, por ejemplo activar una bomba, hasta que ésta se visualiza en pantalla, por ejemplo que la bomba está realmente activada. Esta situación propicia la descoordinación entre las señales de mando y las de visualización pudiendo crear problemas al usuario en determinadas secuencias de operación que implican a variables con dinámicas rápidas.

- Retardo importante en la representación gráfica de los datos en tiempo real y, lo que es más importante, también pérdida de algunos de ellos.

- Los retardos observados eran tanto más importantes cuanto menor era la velocidad del canal de comunicaciones.

3. SOLUCIONES ADOPTADAS

Dado que el principal problema es el retardo, se hace necesario utilizar una métrica que cuantifique los retardos existentes tanto de representación como de actuación en comunicaciones a fin de poder evaluar las mejoras realizadas. Realizando medidas de tiempo del sistema: cuando el usuario realiza una acción desde el applet, cuando el elemento físico ha actuado, cuando esa actuación vuelve de nuevo al applet del usuario y cuando se ha representado el estado de un elemento y la evolución en el tiempo de un nuevo dato, se llega a los siguientes resultados:

Tabla 1: Valores de tiempos (ms)

Conexión	Actuación	Recepción	Representación
Unileon	3712.24	2158.14	289.65
Mód-LE	6023.12	3502.14	287.12
Mód-Wan	7158.14	3723.18	295.89

La adquisición de esos valores exige que tanto clientes como servidor se encuentren sincronizados, para lo cual se ha utilizado el protocolo SNTP. A la vista de esos resultados, se adoptaron una serie de acciones sobre el sistema para garantizar un funcionamiento razonable:

- Reducir el periodo de muestreo sobre el sistema físico.
- Optimizar el tiempo de envío y recepción de datos del sistema desde el punto de vista del applet del usuario.
- Optimizar la visualización de los estados de los datos en el applet del usuario.
- Optimizar la representación gráfica en tiempo real de los datos que el usuario desee.

3.1. REDUCCIÓN PERIODO MUESTREO

Inicialmente, ver Figura 2, se partía de un tiempo de muestreo sobre el sistema de 5 segundos que se eligió con ese valor porque la evolución de los procesos del sistema no iba a ser muy rápida: controles de nivel y temperatura, se suponía que elegir un periodo de muestreo menor aumentaría, de forma innecesaria, la carga del controlador y realizar conexiones ODBC (OptoConnect las realiza) a una base de datos SQL cada menos tiempo aumentaría la carga en el PC que soporta la base de datos y disminuiría el ancho de banda del mismo (la base de datos, el IIS y el motor de servlets se encuentran en el mismo PC).

A raíz de la necesidad de controlar un variador de frecuencia que controla la bomba del circuito de recirculación de que dispone el sistema físico y al solventar problemas con las conexiones ODBC realizadas por el OptoConnect, surgió la necesidad de muestrear datos a un ritmo mayor y de ahí que el tiempo de muestreo fuera reducido a 1 segundo (cota mínima que permite el programa OptoConnect para tomar los datos del controlador y salvaguardarlos en una base SQL). Con esta acción, se ha conseguido reducir el tiempo de actuación sobre el sistema físico y el de recepción de datos, pero no tanto como se había estimado.

El tiempo de actuación se ha reducido hasta valores que rondan el segundo dentro de la red local universitaria, ver Tabla 2; es decir, si un usuario desea abrir una válvula, desde que solicita la acción hasta que la válvula se abre realmente no se superan los 2 segundos. Sin embargo, el tiempo de confirmación, desde que una válvula se ha abierto hasta que el usuario recibe una confirmación del hecho, no se redujo tanto como se hubiera esperado y ronda valores aleatorios de entre 1 y 5 segundos en red local universitaria.

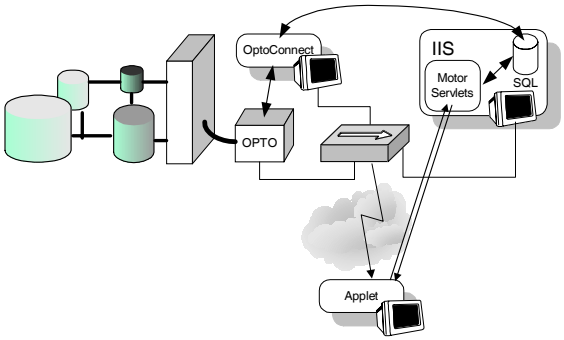

Figura 2: estructura general.

La razón de este hecho viene dada porque el programa utilizado para capturar los datos del controlador (por tanto del sistema físico) y almacenarlos en una base de datos SQL (OptoConnect) no realiza esta acción hasta, aproximadamente, entre 1 y 3 periodos de muestreo después de que una acción física haya tenido efecto; es decir, si una válvula se ha abierto, este hecho no se verá reflejado en la base de datos hasta transcurridos unos 2-3 segundos después, lo que sumado al tiempo de muestreo definido, que es de un segundo, consigue que un cambio físico en una variable dada no se vea reflejado en la base de datos hasta unos 3 segundos después de que realmente se haya producido (para determinar el tiempo en el que el usuario recibirá la notificación del cambio, hay que sumar a este tiempo de 3 segundos el retardo que se produce en el envío de los datos de la base de datos hasta el applet del cliente, que debería coincidir, en media, con el tiempo que se tarda en enviar los datos desde el applet del cliente a la base)

Evidentemente, depende de la naturaleza del proceso que se esté controlando que esos tiempos sean suficientes o no. En este caso concreto son más que suficientes, pero aún así se cree que se puede mejorar

el tiempo de escritura de datos físicos en la base SQL si se indica al controlador que dedique más tiempo de su CPU a comunicaciones (enviar datos al OptoConnect) y menos a control (para el control que se debe realizar, el controlador no necesita dedicar mucho tiempo de su CPU), porque el cuello de botella existente actualmente se encuentra en el programa OptoConnect, que no decide de forma determinista cuándo escribir datos en la base de datos.

3.2 OPTIMIZAR COMUNICACIONES

Cuando se desea operar de forma remota a través de una red, un problema fundamental lo constituye el retardo existente en las comunicaciones. Este problema se agrava en entornos "no deterministas" como lo constituyen las redes Internet/Intranet.

Desde el principio del desarrollo de este proyecto, se consideró que este sería el principal escollo y por ello todas las decisiones de diseño de software de la aplicación tenían como objetivo fundamental minimizar en lo posible el retardo en las transmisiones(se observará que el efecto del retardo no es tan determinante).

Dado que la arquitectura por la que se optó es la de una arquitectura tricapa que consta de un cliente, un servidor y una capa intermedia y que tanto el cliente como el servidor se han desarrollado en JAVA, para comunicar los clientes con el servidor, se barajaron dos posibles soluciones:

- Construir un servidor en JAVA que se comunique mediante sockets con los clientes (applets en este caso).
- Utilizar HTTP Tunneling como medio de comunicación y configurar un servlet en servidor que atienda a un cliente concreto.

Aunque la primera opción es más eficiente que la segunda en las transmisiones, presenta el inconveniente de que hay que asignar un puerto por cada cliente que se conecte, con lo que puede haber problemas con firewalls a través de Internet en el sentido de que no permitan el tráfico sobre determinados puertos. Además los costes de mantenimiento del servidor son elevados.

La segunda opción es menos eficiente que la primera, pero presenta las ventajas de que no presentará problemas con ningún firewall porque utiliza el puerto 80 (puerto permitido por todos los firewalls) para establecer las comunicaciones y además toda la gestión del servicio corre a cargo de un servidor de servlets, con lo que el mantenimiento es mínimo.

La opción tomada es la segunda, utilización de applets-servlets y la forma de establecer la comunicación es siempre la misma tanto para lectura como escritura:

- Desde un applet se llama a un servlet como se llama a cualquier pagina web.
- El servidor de servlets detecta que la llamada corresponde a un servlet y éste define un canal de entrada (lectura) y uno de salida (escritura).
- El applet, una vez que el servlet ha sido llamado exitosamente abre un canal de entrada y otro de salida, se envía una petición al servlet y dependiendo de si la petición corresponde a una escritura (actuar sobre el sistema físico) o a una lectura (tomar datos del proceso físico) se espera en el canal de entrada la recepción de datos o no.
- Se cierran los canales abiertos en el applet y cuando se desee realizar otra nueva petición, se vuelven a abrir y se realiza.

Cuando se utilizó por primera vez esta forma de proceder, dado que se necesitaba continuamente estar muestreando datos del sistema para poder obtener el estado de los elementos físicos, se definió una arquitectura de un solo hilo en la que un hilo se encargaba de realizar las peticiones de datos, las escrituras de acciones y la representación de los estados del sistema físico. Esta solución presentaba una ineficiencia bastante alta porque no sólo era lenta sino que, si el canal de comunicaciones era lento (en WAN lo es siempre), se perdían, desde el punto de vista del usuario, algunos datos. Esta pérdida no era importante porque los datos reales son salvaguardados perfectamente, sin pérdida, en la base SQL (esto es cierto siempre que el PC que corre OptoConnect y el PC que corre el sistema gestor de base de datos presenten entre ellos un canal de comunicaciones rápido).

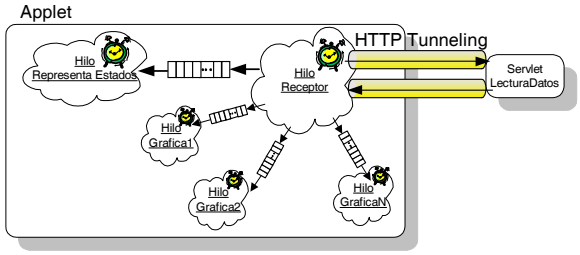

Figura 3: Arquitectura de Applet de Usuario.

El siguiente paso que se propuso fue construir una arquitectura multihilo, ver Figura 3, en la que existe un hilo dedicado a la recepción de datos y distribuye los datos que le llegan a hilos encargados de representar los estados y dibujar las gráficas de datos reales provenientes del sistema físico.

Para comunicar el hilo principal con el resto de los hilos se hace necesario el uso de buffers circulares sincronizados en la lectura que permitan ecualizar las distintas velocidades que pueden presentar los hilos entre sí y que actúen como colchones de seguridad

que eviten la pérdida o repetición de datos porque un hilo corra más rápido que otro.

Con esta arquitectura, la eficiencia en el canal aumentó de forma considerable obteniendo tiempos de recepción del cambio de todos los datos del sistema que rondan el segundo (considerando lectura de la base de datos + transmisión por el canal + recepción en el applet cliente dentro de red local universitaria). En la siguiente tabla, se presentan una serie de valores según la velocidad del canal de comunicaciones.

Tabla 2: Valores de tiempos (ms)

Conexión	Actuación	Recepción	Representación
Red-LE	1833.57	1177.13	110.12
CyberGijón	3500.00	1236.11	103.14
Alehop	4015.23	755.05	101.10
Mód-LE	2022.36	1268.52	99.38
Wanadoo	3844.26	1319.67	107.18
Genentech	3943.82	912.29	101.25

3.3. VISUALIZACIÓN ESTADOS

A la hora de representar los estados de los elementos físicos, inicialmente se partió de un hilo perteneciente a la clase SwingWorker (todavía se mantiene esta estructura) que se dedicaba a solicitar datos del proceso físico (base de datos SQL), escribir datos en el proceso físico (base de datos SQL) y también se encargaba de la representación de los estados de los mismos y de la representación gráfica de las variables.

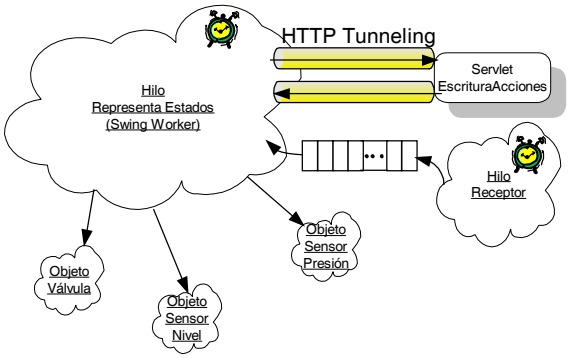

Figura 4: representación estados

Debido a la ineficiencia de esta forma de proceder, ahora sólo se dispone de un hilo SwingWorker, ver Figura 4, que se dedica a representar los estados, atender a las acciones del usuario y a solicitar acciones sobre el sistema (la acción de escribir en la base de datos no es excesivamente exigente en recursos). La representación de los estados la realiza en función de los datos reales suministrados por el hilo principal del applet y de la definición de niveles de alarma que cada uno de los elementos del sinóptico representado en la Figura 1 presenta (cada objeto recibe un dato físico y es el propio objeto el que decide el estado en que se encuentra y cambia sus propiedades, por ejemplo el color, en función del mismo)

Se ha conseguido una mejora evidente respecto a la primera situación, pero aún así se ha observado que la carga introducida por este hilo (% de uso de CPU) y por el uso de objetos swing para representar el sinóptico del sistema es bastante elevada, carga que se podría reducir si se utilizaran objetos AWT, técnica esta última probada en la representación gráfica de las variables en tiempo real y que pasa a exponerse en el siguiente punto.

3.4. VISUALIZACIÓN GRÁFICAS

Para representar de forma gráfica la evolución de las variables una vez observados los problemas de consumo de CPU por utilizar objetos swing, se optó por comprobar la funcionalidad utilizando objetos AWT y objetos Thread (en vez de SwingWorker) para definir una arquitectura multihilo.

La solución propuesta se basa en el hilo principal del applet del cliente que escribe en un array de buffers circulares de forma continua (está sólo sincronizada la lectura, por lo que el hilo productor sobrescribe datos si un hilo consumidor no ha sido capaz de extraerlos) y un hilo por cada gráfica que el usuario desee visualizar. El número de hilos de representación coincide con el número de gráficas ejecutándose en el sistema y la ventaja de esta forma de proceder es que la carga que introduce en el sistema la representación es del orden del 5-10% de la CPU (PC Pentium II 200 MMX) o el 2-5% (PC Pentium III 500 Mhz) cuando utilizando swing la carga rondaba del orden del 37% de CPU en el mismo PC y para la misma funcionalidad.

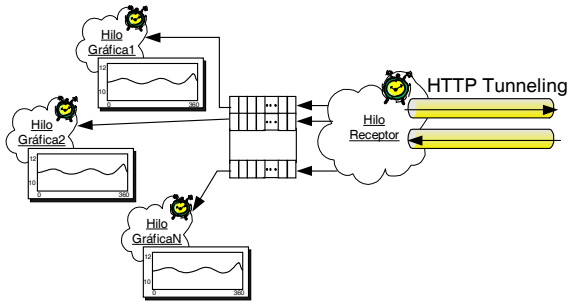

Figura 5: representación gráficas.

4. CONCLUSIONES

Para utilizar Internet como un medio de comunicación que permita la teleoperación de procesos industriales o la realización de prácticas

remotas, es necesario que la velocidad del servicio sea adecuada.

La arquitectura por la que se ha optado es altamente abierta, flexible y expandible aunque si se deseara implantar en un proceso industrial real, dependiendo del número de variables que deben ser monitorizadas, la arquitectura debería ser modificada en el sentido de que si el número de variables es elevado, sería mucho más eficiente sólo enviar al applet cliente aquellas que realmente se han cambiado (se necesita algo de inteligencia en la parte del servidor) y también sería muy interesante sólo actualizar la representación del applet por zonas y no todo él (la versión 1.4 de swing lo permite)

Es necesario mejorar la eficiencia de la representación de los estados de los elementos físicos utilizando AWT, minimizar la descarga del applet (empaquetándolo en un fichero JAR por ejemplo).

Una mejora sustancial sería la descentralización de tareas, por ejemplo disponer de un solo PC para la base de datos, otro PC para el servicio web y otro PC para servir video que permita ver las imágenes capturadas por una cámara de vídeo o una webcam.

Tanto en el caso de visualización de estados como en el de representación gráfica de variables se puede producir una pérdida de datos desde el punto de vista del usuario, este es un problema cuyo origen viene determinado porque la petición de datos desde el applet-cliente al sistema se realiza mediante HTTP realmente y aunque se intentara realizar una nueva petición un segundo después, no sería posible hasta que la petición anterior haya sido satisfecha (los datos hayan sido enviados por el canal que se abre bajo http). Para evitar este problema, la solución más factible y cómoda es enviar no sólo los datos del instante actual, sino datos de instantes anteriores (en el periodo de tiempo: valor de actuación menos valor de recepción) y que los hilos encargados de la representación de la evolución de las variables y de los estados de los dispositivos tengan en cue4nta este hecho o también se puede conseguir que el servlet al que se han solicitado datos envíe todos aquellos que no se han enviado desde la última solicitud que se ha realizado. Esto obliga de nuevo a disponer de una cierta inteligencia en la parte del servidor.

La forma de proceder descrita en el punto anterior también posibilitaría no sólo representar variables con una dinámica del orden del segundo, sino también variables con dinámicas mucho más rápidas, puesto que el controlador utilizado permite obtener muestras de datos varias veces por segundo y no sólo una por segundo para poder luego, mediante OptoConnect, almacenar todas esas muestras en la base de datos SQL. Con esto se consigue, en cada segundo, una precisión de centenas o decenas de milisegundos.

La arquitectura propuesta ha conseguido tiempos, en media, inferiores a los 5 segundos independientemente de la ubicación geográfica desde la que se actúe sobre el sistema siempre que el canal de comunicaciones presente un ancho de banda suficiente (mayor de 56 Kb/s).

Referencias

[1] Dominguez, M., Marcos, D., Reguera, P., Gonzalez, J.J., Blazquez, L.F., (2001) "Connection Pilot Plant to the internet", *IFAC Internet Based Control Education, IBCE01*, Madrid. España.

[2] Meléndez, J., Colomer, J., Rosa De La, J. Fabregat, R., Macaya, D."Experiencias en teleoperación de procesos y telenseñanza en la Universitat de Girona". EIWISA'01

[3] Opto22 (2000) OptoConnect User´s Guide. http://www.opto22.com/.

[4] Ramos, C., Herrero, J., Martínez, M., Blasco, X. "Internet en el desarrollo de prácticas no presenciales con procesos industriales". EIWISA'01

[5] The Source for java tecnology. http://java.sun.com/.