Trabajo no presencial: Implementación de un servicio de gestión de imágenes publicitarias en base a un sistema de subasta

Programación de Sistemas Concurrentes y Distribuidos
Grado de Ingeniería Informática
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

13 de diciembre de 2017

1. Objetivos

Los objetivos de este trabajo no presencial son los siguientes:

- Desarrollar una aplicación distribuida cliente/servidor.
- Profundizar en el modelo de comunicación síncrona para procesos distribuidos.
- Asentar los conocimientos adquiridos para la programación de aplicaciones distribuidas en C++.
- Trabajar en equipo.

2. Descripción del sistema

Consideremos una empresa que se dedica a la publicidad dinámica, y que gestiona vallas electrónicas, de manera que en una valla puede mostrar imágenes y cobrar por el servicio. El trabajo consiste en el desarrollo de un prototipo en C++ del sistema de gestión de la vallas, que se esquematiza en la figura 1.

La empresa dispone de dos vallas publicitarias idénticas, y su negocio consiste en cobrar por mostrar en ellas imágenes que le suministran los clientes. El sistema se compone de tres módulos: el módulo de subastas, el módulo de control de vallas y el módulo de administración:

Módulo de subastas: Este módulo oferta, mediante subasta, el uso de las vallas. Propone una subasta para alquilar una valla por un periodo de tiempo. Los clientes interesados en mostrar alguna imagen acuden a la subasta. Aquél que la gane, suministrará al servicio la URL con la imagen que desea mostrar, que será descargada por el servicio, y mostrada en una de las ventanas por el tiempo con el que se sacó en subasta. Las peticiones

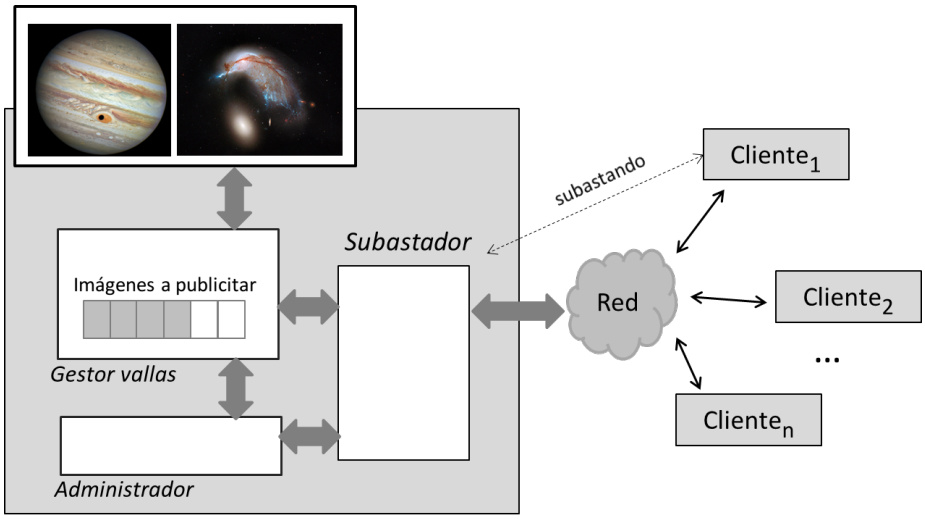

Figura 1: Esquema del sistema

de servicio que han ganado se van encolando, con toda la información necesaria, en una cola que este módulo comparte con el de gestión de vallas.

Módulo de gestión de vallas: Va tomando las peticiones encoladas y atendiéndolas, mostrando cada imagen el tiempo adecuado según salió en la subasta.

Módulo Administrador: Atiende a determinadas órdenes del administrador. Como mínimo, las siguientes:

1. Mostrar información histórica del sistema (número de imágenes mostradas, tiempo total y medio que las imágenes han estado en pantalla, etc.)
2. Mostrar información del estado del sistema (número de peticiones encoladas, tiempo contratado estimado, etc.)
3. Iniciar la terminación ordenada del servicio. Para ello, ya no se abrirán nuevas subastas, y el sistema esperará hasta la terminación de la que se pueda encontrar en marcha, así como a atender todas las subastas ganadas que se encuentren en la cola, para finalizar su ejecución.

3. Sistema de subasta inglesa

El módulo de subastas va a lanzar sucesivas subastas (hasta que el administrador del sistema le avise de que debe terminar), proponiendo el tiempo que se mostrará la imagen, y fijando un precio mínimo (que no hace público). La subasta se regirá por la modalidad de *subasta inglesa*, de acuerdo con el protocolo estándar definido por la Foundation for Intelligent Phisical Agents, que se describe detalladamente en `http://www.fipa.org/specs/fipa00031/`.

4. Algunas consideraciones a tener en cuenta

- Para descargar imágenes desde una URL, y mostrarlas, se puede adaptar el ejemplo desarrollado en *descargarYMostrarImagen.zip*. Por un lado, utiliza "The CImg Library" (`http://cimg.eu/`) para el procesamiento de imágenes (en realidad, solo para mostrarlas) y, por otro lado, la librería "curl" (`https://curl.haxx.se//`) para el acceso a recursos via internet.

5. Instrucciones para el desarrollo y la entrega del trabajo

5.1. Organización inicial del trabajo

El trabajo tiene una naturaleza no presencial. Inicialmente, los alumnos se organizarán en grupos de tres estudiantes, obligatoriamente (los propios alumnos serán responsable de formar estos tríos). Una vez formado, se deberá enviar un correo a *alvaper@unizar.es* con el asunto *[PSCD] Equipo de trabajo TP6* en el que se comunique quiénes son los componentes del trío. Para cada componente concreto se deberá especificar su NIP y nombre completo con los dos apellidos. Esta información es indispensable y debe ser completa. Cada trío deberá programar una aplicación distribuida conforme las pautas de implementación descritas en las secciones previas. La fecha límite para enviar la composición de cada equipo de trabajo es el 22 de Diciembre del 2017 a las 23:59 horas. Aquellos tríos que no comuniquen su composición antes de esta fecha no serán evaluados.

No se admitirán trabajos realizados individualmente o en pareja. Si algún alumno no encuentra compañeros de trabajo después de un tiempo razonable, deberá ponerse en contacto con el profesor (antes de la fecha límite de composición de equipos de trabajo) y este le asignará a algún equipo. No obstante, esta solución será de carácter excepcional.

5.2. Ficheros a entregar como resultado

Cuando se finalice el trabajo los alumnos deberán entregar un fichero comprimido *trabajoNP_NIP1_NIP2_NIP3.zip* (donde *NIP1*, *NIP2* y *NIP3* sean el NIP de los tres autores del trabajo) con el siguiente contenido:

1. Un fichero de texto denominado *autores.txt* que contendrá el NIP, los apellidos y el nombre de los tres autores del trabajo en las primeras líneas del fichero. Por ejemplo:

   ```
   NIP      Apellidos              Nombre
   ----------------------------------------
   345689   Rodríguez Quintela     Sabela
   787654   Hernández Castillo     Luis
   925674   Bribón Palomares       Genaro
   ```

   Además, el fichero deberá contener obligatoriamente un *informe de dedicación*. Este informe debe recoger las horas que cada alumno del trío ha dedicado a la realización del trabajo. En concreto, es importante

que aparezca el número de horas totales por alumno y un desglose más detallado de días trabajados y número de horas dedicadas en esos días concretos, así como de las tareas a las que se ha dedicado el tiempo: análisis, diseño, implementación, test, etc. Esta información no será utilizada para evaluar el trabajo de los alumnos (es decir, no tiene ninguna influencia directa o indirecta en la nota de los alumnos). Su objetivo es medir el esfuerzo medio que ha costado realizar el trabajo a los alumnos.

2. Un documento pdf, llamado `disenio.pdf`, donde se explique claramente el diseño de la aplicación. Este documento debe contener como mínimo una descripción detallada de: el diseño interno de los procesos del sistema (una descripción de alto nivel de la lógica de control de los procesos, decisiones relativas a la gestión de errores, estructuras de datos utilizadas, o algoritmos concretos de interés), los protocolos de interacción concretos que han sido implementados (es decir, cuál es el formato de los mensajes, en qué orden se intercambian los mensajes, y cómo se gestionan los posibles errores de interacción), los mecanismos programados para gestionar los aspectos de concurrencia donde fuera necesaria, etc. En general, deberán estar debidamente justificadas todas aquellas decisiones de diseño que hayan sido adoptadas durante el desarrollo del sistema.

3. Un documento pdf, llamado `pruebas.pdf`, donde se expliquen claramente las pruebas que han sido realizadas para comprobar el correcto funcionamiento de la aplicación. Este documento es un compromiso de garantía de funcionamiento del sistema, y debe contener una descripción detallada de: el método seguido para la fase de pruebas, las pruebas concretas realizadas, los resultados obtenidos y, si procede, las acciones correctivas programadas para su resolución.

4. Un directorio denominado `src`, que contendrá los ficheros fuente C++ utilizados en la implementación del trabajo.

5. Un directorio `bin`, que contendrá todos los ficheros resultantes de la compilación de los fuentes en `src`.

6. Un fichero "Makefile" para la compilación y limpieza.

7. Los siguientes scripts:

   - `lanza_servicio.sh`: Lanza tanto el "Subastador" como el "Gestor de vallas", así como el proceso "Administrador", encargado de todo lo que se considere necesario.
   - `lanza_robots.sh`: Abre tres terminales distintas, cada una con un cliente con un identificador distinto (el desarrollador determina cómo es un identificador). Cada cliente participará, de manera automática, en tres subastas sucesivas, e informará por la salida estándar del cómo evoluciona el proceso. El objetivo es mostrar, de manera simulada, el comportamiento del sistema. Es interesante que se introduzcan ciertas pausas en la ejecución con el objetivo de "ir viendo" la evolución. Es de ayuda para esta parte mirar la documentación de la instrucción (unix/linux) `xterm -e`.

- `lanza_cliente_manual.sh`: Toma como parámetro la IP y el puerto de un proceso subastador, así como la URL de una imagen, y lanza un cliente manual. Este cliente interacciona con el usuario para llevar a cabo transacciones con el sistema (inicia la participación en una subasta, va fijando el precio que considere oportuno, en caso de ganar la subasta pasa la URL de la imagen, etc.).

Es importante seguir las convenciones de nombrado y la estructura de ficheros y directorios (src, bin, etc.) descrita.

5.3. Procedimiento y fechas de entrega de la práctica

Para la entrega del fichero `.zip` descrito anteriormente, se utilizará el comando *someter* en la máquina *hendrix.cps.unizar.es*. Los tres alumnos que forman un trío deberán someter individualmente el mismo fichero `.zip` generado como resultado de su trabajo conjunto. La fecha límite para someter el trabajo es el 14 de Enero del 2018 a las 22:00h.

5.4. Procedimiento y recomendaciones para su evaluación

El procedimiento de entrega anterior constituye el primer paso para la evaluación de esta actividad. El segundo paso tiene un carácter presencial y para ello se habilitará una sesión especial en el laboratorio de prácticas (Laboratorio 0.01, Edificio Ada Byron). Una vez entregados los trabajos, se publicarán con antelación suficiente las fechas y horas concretas en las que se realizará esta evaluación presencial. Es obligatorio la asistencia de todos los miembros del equipo a la misma. El trabajo debe entregarse en los términos indicados anteriormente, debe funcionar correctamente y no haber sido copiado. En particular, hay que asegurarse de que la práctica funciona correctamente en los ordenadores del laboratorio (hay que vigilar aspectos como los permisos de ejecución, juego de caracteres utilizado en los ficheros, etc.). También es importante someter código limpio. El tratamiento de errores debe ser adecuado, de forma que si se producen debería informarse al usuario del tipo de error producido. Además se considerarán otros aspectos importantes como calidad del diseño del programa, adecuada documentación de los fuentes, correcto formateado de los fuentes, etc.

Para el adecuado formateado de los fuentes, es conveniente seguir unas pautas. Hay varias, y es posible que podáis configurar el entorno de desarrollo para cualquiera de ellas. Una posible, sencilla de seguir, es la "Google C++ Style Guide", que se puede encontrar en:

`https://google-styleguide.googlecode.com/svn/trunk/cppguide.html`