TFC_101101_iperezga:

Simulación del comportamiento de aplicaciones colaborativas sobre un servidor Jakarta Tomcat.

Iñaki Pérez García (iperezga@uoc.edu)

Ingeniería Técnica en Telecomunicaciones (Esp. Telemática)

Consultor: Pau Fonseca i Casas

15 de Enero de 2011

Dedicatoria y agradecimientos

Este proyecto está dedicado a mi mujer Nuria y a mis dos hijos Janire y Unai porque son lo más importante de mi vida. No habría podido terminarlo sin la ayuda de Nuria. Gracias por tu esfuerzo, cariño y paciencia durante estos años.

Gracias a mis padres por haberme educado con valores como la constancia y el trabajo, y por no haber dudado ni un momento en ayudarme cuando lo he necesitado.

Gracias al Dr. Pau Fonseca por los ánimos que he recibido durante todo el proyecto. Sin ellos no habría seguido adelante en los momentos difíciles.

Por último, gracias a los demás familiares y amigos por aceptarme como soy y por todo vuestro apoyo y comprensión.

Índice

1 Presentación del proyecto

1. Introducción 4
2. Objetivos 5
3. Entorno de trabajo 7

2 Investigación básica

1. Introducción 8
2. Resumen 9
3. Lenguaje SDL básico
   - Introducción 10
   - Herramientas utilizadas 11
   - Ejemplos 13
4. Simulación básica
   - Introducción 16
   - Herramientas utilizadas 16
   - Ejemplos 20
     - Ej.1 Mixed Lan 20
     - Ej.2 Red con clientes wifi 22
     - Ej.3 Comunicación con servidor externo 23
5. Servidor Tomcat y servlets básicos
   - Introducción 25
   - Herramientas utilizadas 26
   - Ejemplos 29
6. Conclusiones 32

3 Investigación avanzada

1. Introducción 33
2. Resumen 34
3. Lenguaje SDL avanzado
   - Introducción 35
   - Modelo 1 35
   - Modelo 2 36
4. Simulación avanzada
   - Introducción 38
   - Simulación contra servidor real 39
   - Simulación parametrizada 43

5. Servidor Tomcat y servlets básicos

- Introducción..... 46
- Múltiples instancias de Tomcat..... 47
- Habanero. Aplicaciones colaborativas..... 50

6. Resultados de las pruebas 53
7. Conclusiones 62

4 Seguridad del servidor

1. Contraseñas shadow 64
2. Bit de inmutabilidad 64
3. Registro de eventos del sistema 65
4. Cortafuegos con iptables 66
5. Encriptación de las conexiones 68

Anexo A Planificación del proyecto 72
Anexo B Indice de figuras 73
Anexo C Bibliografía y referencias 75
Anexo D Defensa. Presentación virtual 77

Presentación del proyecto

Introducción

El presente proyecto se plantea dentro del área de simulación de redes y sistemas informáticos y de telecomunicaciones.

De una manera más específica se desarrollará en base al proyecto Castelldefels, que tiene como objetivo modelar el sistema informático que da soporte al Campus virtual de la Universidad Oberta de Cataluña. Con el presente proyecto se pretende complementar los servicios que se ofrecen en dicho Campus mediante aplicaciones colaborativas, como por ejemplo pizarras compartidas, bloc de notas compartido, o cualquier tipo de herramienta que permita colaborar de forma interactiva entre profesores y alumnos.

En el pasado el proyecto Castelldefels se centraban en el software de simulación OPNET, se trata de un simulador de redes con licencia privativa. Actualmente la herramienta principal utilizada en el desarrollo del proyecto es OMNET++, se trata igualmente de un simulador de redes pero con licencia gratuita y de código abierto.

Es en ésta línea de software libre en la que se intentará desarrollar el presente proyecto. Se utilizará una distribución Linux, en concreto Debian, como sistema operativo, en la que se creará un entorno de trabajo con aplicaciones libres como por ejemplo Openproj, OpenOffice, Eclipse, etc. Pero sobre todo la aplicación protagonista será el servidor web Tomcat, que forma parte del proyecto Jakarta de la Apache Software Foundation.

Éste servidor web tiene soporte para servlets, que son pequeños programas que se ejecutan en el servidor y que son utilizados por los usuarios. Están programados en lenguaje Java y dichos programas son propiamente las aplicaciones colaborativas mencionadas anteriormente. En concreto lo que se pretende es comprobar el rendimiento y la capacidad del sistema mediante la simulación de múltiples conexiones al servidor por parte de lo que serían los estudiantes del Campus virtual.

Por último se pretende dar un paso más en el mundo de la seguridad informática, aplicando algunas medidas de seguridad a nuestro sistema. En este caso también se trata de aplicaciones libres, iptables y openssl nos permitirán crear un entorno encriptado en el cual, a pesar de que ningún sistema es invulnerable por completo, estaremos más tranquilos.

Objetivos

Para poder entender el uso que se le quiere dar al servidor que se va a implementar voy a hacer una comparación del sistema que voy a simular con un supermercado. De ésta manera podremos conocer la capacidad y su posible aplicación en un entorno de aprendizaje virtual. Por ejemplo varias personas pueden ir a la cafetería y utilizar la máquina de café, esto podría ser una aplicación colaborativa, otra distinta podría ser la frutería, otra la charcutería, pero en general el supermercado sería el servidor Tomcat. Por ejemplo el parking podría representar la capacidad total del servidor.

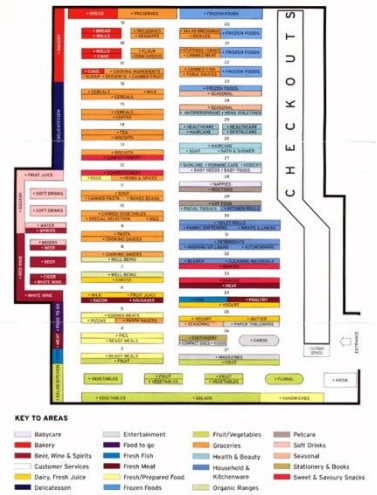


Fig.1 Organización de un supermercado

El supermercado tiene un aforo definido, es decir una capacidad máxima que se reparte entre las diferentes zonas. Pero, ¿que ocurriría si todo el mundo que llega al supermercado decide ir a la cafetería? Seguro que no habría mesas para todos. Este comportamiento es precisamente el que se quiere modelar. En el entorno del campus virtual de la Uoc, se conectan estudiantes de diferentes carreras, por ejemplo empresariales, ingenierías, etc. Cada carrera sería comparativamente una zona del supermercado, con una capacidad definida.

Fig.2 Cafetería del supermercado

El objetivo principal es definir el comportamiento y la capacidad del servidor cuando se están ejecutando simultaneamente varias aplicaciones dirigidas cada una de ellas a un grupo de usuarios específico. También existirán algunas aplicaciones críticas que serán utilizadas por la mayoría de los usuarios y que se serán las que reciban una mayor carga de conexiones y tráfico.

Objetivos específicos:

- ✓ Investigación sobre los sistemas de simulación existentes para modelar las conexiones de los usuarios a las diferentes aplicaciones disponibles en el servidor, tales como OMNET++ y el lenguaje SDL.
- ✓ Investigación sobre el servidor Jakarta Tomcat, así como sobre los servlets que se utilizarán como aplicaciones colaborativas, utilizando el lenguaje de programación Java. También se pretende dotar de algunas medidas de seguridad a éste servidor.

Para conseguir alcanzar éstos objetivos se tendrá en cuenta el trabajo relizado por compañeros de semestres anteriores como Jose Miguel Vera Roa, que investigó sobre los perfiles de usuario que se conectan al campus virtual y Alberto Manuel Cabeza, que modeló el comportamiento de diferentes perfiles de usuario usando SDL.

Entorno de trabajo

Se preparará un entorno de trabajo basado en software libre para su posterior simulación y análisis de rendimiento, comportamiento y capacidad. La base para éste entorno de trabajo será el sistema operativo Debian, más concretamente en su versión 5.0.6. Sobre ésta distribución de linux se instalarán diferentes herramientas para documentar el proyecto, como Openproj y OpenOffice, también se utilizará packet tracer como apoyo para representar gráficamente las redes simuladas.

Por otro lado, las aplicaciones principales sobre las que se harán las simulaciones serán el servidor Tomcat, cuyo motor de servlets de se presentará en combinación con el servidor web Apache. Para gestionar los servlets contaremos con el IDE eclipse, estos servlets, programados en lenguaje Java serán los que hagan de aplicaciones colaborativas. Por último y si el tiempo lo permite, se intentarán realizar simulaciones sobre el servidor de aplicaciones JBoss.

Todas las aplicaciones del entorno de trabajo descritas hasta el momento estarán instaladas en una máquina dentro de una Lan. En otra máquina distinta de la Lan se instalarán las aplicaciones para realizar las simulaciones.

A pesar de que se intentará realizar pruebas con todo tipo de herramientas de simulación, centraremos la atención en el lenguaje SDL para modelar la estructura del sistema, los procesos, los usuarios, la comunicación de las diferentes partes de éste sistema mediante señales, etc.

También se utilizará OMNeT++ para simular pricipalmente el comportamiento de Tomcat, aunque para llegar a ese punto habrá que hacer otra serie de simulaciones para aprender y comprender el funcionamiento del programa.

OMNeT++ es un simulador de eventos, usado habitualmente para modelar el tráfico de redes de telecomunicaciones, protocolos, evaluación del rendimiento de sistemas software y, en general, modelar cualquier sistema que pueda simularse con eventos discretos, pero en éste proyecto se le intentará dar uso sobre un sistema real.

Por último, para obtener algo más de información sobre el tráfico de red generado durante las simulaciones, también contaremos con WireShark, un analizador de paquetes de red.

Investigación Básica

Introducción

Al inicio del presente proyecto se estableció un plan de trabajo en el que se contemplaba la implementación de un entorno de trabajo en el que se incluían una serie de herramientas o aplicaciones que nos permitirán alcanzar los objetivos propuestos.

Durante cuatro semanas se ha estado realizando la instalación de dichas herramientas, en la medida de lo posible se ha utilizado software libre, y además se ha venido realizando un trabajo de investigación básica que comprende la lectura de documentación relativa al software y las tecnologías utilizadas así como la ejecución de distintos ejemplos para familiarizarse con su utilización.

El entorno de trabajo se divide en dos partes. Una de las partes es la que comprende las herramientas de simulación y la otra abarca el software necesario para el funcionamiento del servidor que vamos a denominar "real". Sobre estas dos partes se ha estado trabajando de manera paralela.

Como en cualquier proyecto han surgido contratiempos por temas como licencias, incompatibilidades de software, etc., pero al final se han solventado cambiando de versiones, o incluso sustituyendo unas aplicaciones por otras similares.

Durante las pruebas con ejemplos, que se han encontrado a través de internet o que incluyen las propias aplicaciones, se ha hecho más hincapié en aquellos que nos dan mayores posibilidades de alcanzar los resultados deseados, e incluso viendo el potencial de alguno de ellos se ha cambiado alguno de los objetivos que se tenían en caso de que sobrara tiempo.

En definitiva en el presente documento se dará una visión más bien generalista de las tecnologías empleadas y de los conocimientos adquiridos sobre éstas. Más adelante se aplicarán estos conocimientos para resolver el problema particular planteado en el proyecto, manteniendo el objetivo principal de complementar los servicios ofrecidos por el campus de la Uoc e intentando contribuir de esta manera con el proyecto Castelldefels.

Resumen

El objetivo principal del trabajo de investigación básica realizado es la comprensión del software y las tecnologías utilizadas en el entorno de trabajo.

En primer lugar se utilizará el lenguaje de descripción y especificación, SDL (Specification and Description Language). Se hace una descripción del estándar y unos conceptos básicos, para continuar explicando los símbolos utilizados, una serie de ejemplos y por último nos fijaremos en uno de estos ejemplos que se aproxima al sistema a simular. Para la visualización de los diagramas se ha utilizado el programa Sandrilla.

Después nos centraremos en las herramientas de simulación. Existe gran cantidad de software de simulación, sin embargo nos hemos centrado en el programa OMNeT++. Se hará una pequeña introducción utilizando el programa OPNET, que es la versión comercial, pero el desarrollo posterior se realizará con la versión de software libre.

Cabe destacar que además de estos programas, existen módulos, o software complementario dirigidos a distintos tipos de simulaciones. En este caso se utilizará ReaSE junto con. Se hará una descripción de éstas herramientas, de sus características principales y se ejecutarán algunos ejemplos.

Por último nos quedará la parte "real" del proyecto. Se dará una explicación de los pasos que se han seguido para instalar y configurar el servidor Tomcat. También ha sido necesario instalar software adicional y por último se han incluido los programas servlets que son el objetivo final de éste servidor. Al igual que con otras aplicaciones se explicarán algunos ejemplos muy útiles para ésta parte del proyecto. En este caso se ha utilizado Eclipse para observar el código de estos pequeños programas escritos en lenguaje Java.

Lenguaje SDL (Specifications and Description Language)

Introducción

El lenguaje SDL se utiliza en la descripción y especificación de sistemas de telecomunicación. Es una recomendación descrita por la ITU-T, unión internacional de telecomunicaciones, concretamente la recomendación Z.100. No llega a ser una norma o un estándar, pero ésta recomendación es de gran ayuda como manual de referencia en la utilización del lenguaje.

En la última revisión de la recomendación se extendió su uso a lenguajes de programación orientados a objetos, se hicieron algunos cambios para hacer su utilización más sencilla y para que se pudiera utilizar conjuntamente con otros lenguajes como por ejemplo UML (Unified Modeling Language).

Este lenguaje es muy utilizado en el ámbito de la simulación, ya que permite definir una serie de canales a través de los cuales se envían señales y es precisamente en éstas señales en las que se basan las simulaciones.

Se puede utilizar de dos formas diferentes, gráficamente o de forma gramatical. Utilizaremos la primera de ellas ya que es más sencilla de interpretar. SDL/GR (Graphical Representation) tiene una gran cantidad de símbolos para definir un sistema, se detallarán los que se utilicen, aunque existan muchos más.

SDL puede definir y especificar un entorno que contiene un sistema. Este sistema es la entidad principal en éste lenguaje y está formado por bloques. Estos bloques se pueden comunicar a través de canales entre ellos mismos y permitiendo a su vez que el sistema se comunique con el entorno. Los bloques contienen especificaciones de procesos que se comunican también entre ellos a través de canales y que permiten que los bloques se comuniquen entre ellos. Por último los procesos contienen una máquina de estados finita, en la que se pueden definir variables, parámetros, acciones, etc.

Los canales de los distintos niveles permiten enviar señales para que sistema, bloques, proceso y entorno puedan interactuar. Pero como hemos comentado anteriormente, la representación gráfica nos permite interpretar todo de forma más sencilla, por lo tanto a continuación se muestra un esquema en SDL que detalla la explicación anterior.

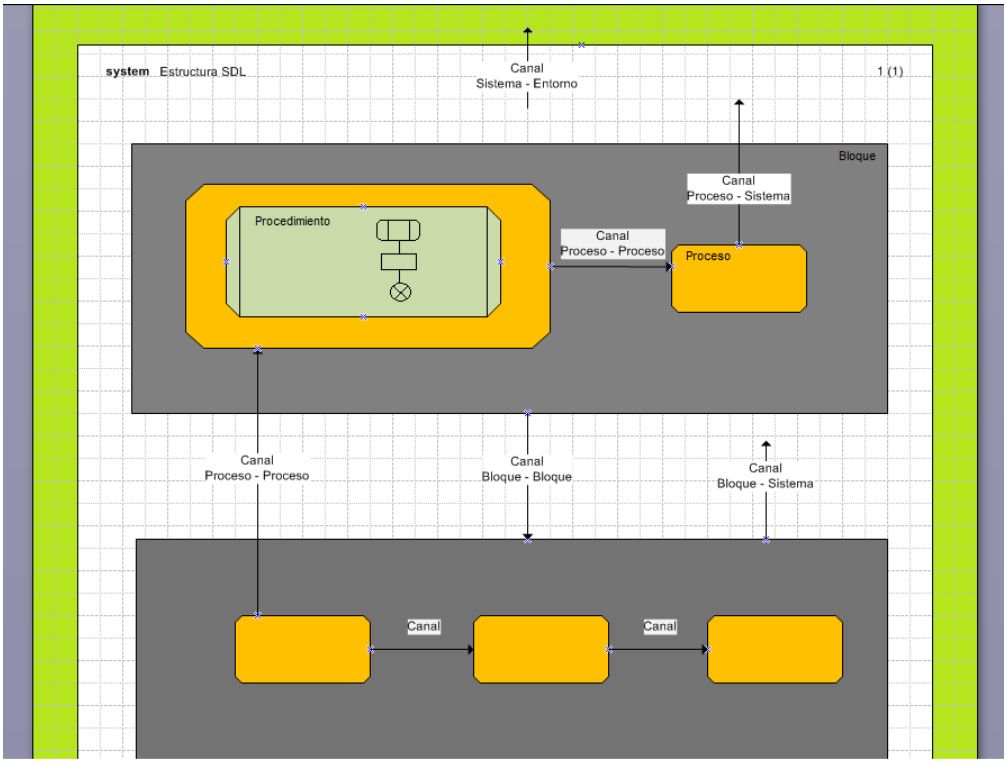


Fig.3. Estructura del lenguaje SDL.

Es un lenguaje en el que la simbología nos permite estandarizar los conceptos, lo que nos permite por ejemplo la reutilización de los proyectos y también permite la existencia de programas que generan código a partir de él. Se puede llegar a distintos niveles de detalle en el modelado de los sistemas, aunque por lo general se hace el mayor hincapié en la definición de los procesos.

Herramientas utilizadas

Esta es la única parte del proyecto en la que se ha utilizado software privativo. En concreto se ha utilizado una Microsoft Office Visio 2007 Profesional que ha sido obtenido de MSDNAA (Microsoft Developer Network Academic Alliance).

Microsoft Office Visio es una herramienta para crear diagrámas de flujo. Permite crear diagramas relacionados con la gestión de proyectos, circuitos eléctricos y lógicos, redes de ordenadores, UML, etc. Sin embargo no dispone de una categoría para crear diagramas SDL.

Para tener un entorno en el que se utilice la simbología descrita en el estándar Z.100, es decir SDL, necesitamos combinar la interfaz gráfica de Visio con un software llamado Sandrila.

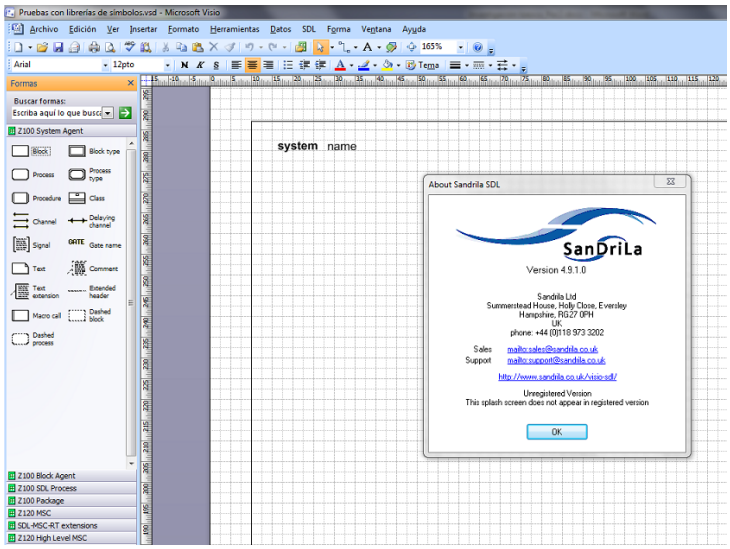


Fig.4 Sandrila y Visio

Dispondremos de este software en una carpeta dentro del sistema, y debemos indicar a Office Visio la ruta hasta esa carpeta para que reconozca los símbolos, plantillas y complementos de Sandrila.

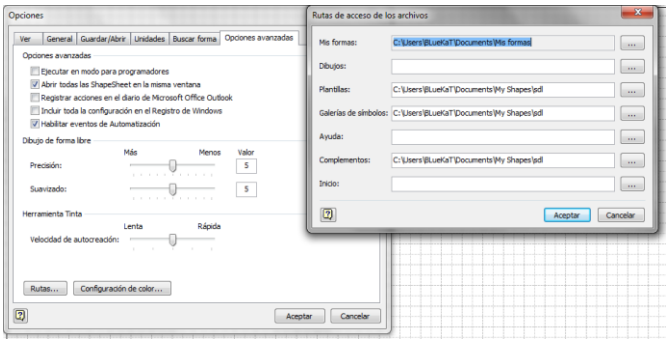


Fig.5 Configuración de Visio para incluir Sandrila

Por último cuando creemos un nuevo dibujo deberemos importar las librerías de símbolos que necesitemos. Las más utilizadas son Z100 system, Z100 block, Z100 process y Z100 package.

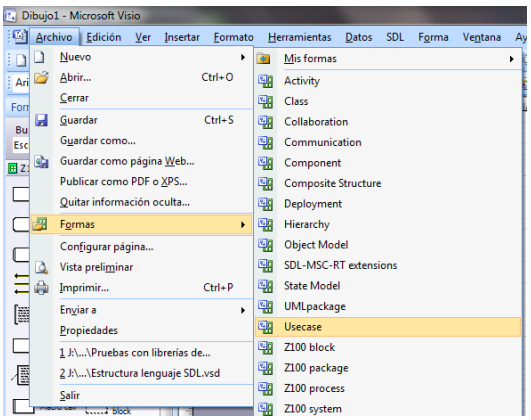


Fig.6 Librerías de símbolos

Ejemplos

A continuación se muestran unos ejemplos que nos permitirán profundizar algo más en la comprensión del lenguaje SDL para poder aplicarlo posteriormente a nuestro caso particular.

En primer lugar podemos observar el modelado de un sistema en el que se aprecian las definiciones de señales, los bloques y los canales de comunicación.

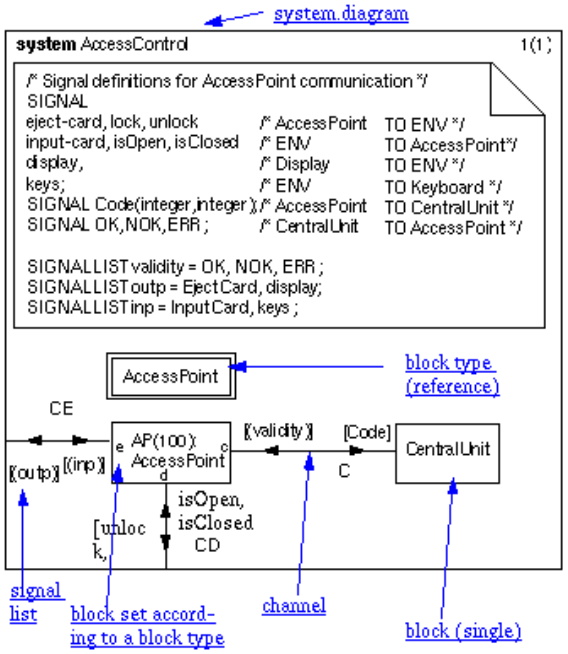


Fig.7 Diseño de sistema en SDL

En este ejemplo los canales son asíncronos, pero podrían ser también síncronos. También se puede diferenciar entre canales internos, entre bloques, y externos, que nos permitirán comunicarnos con el entorno. Otro de los aspectos es que pueden ser unidireccionales o bidireccionales.

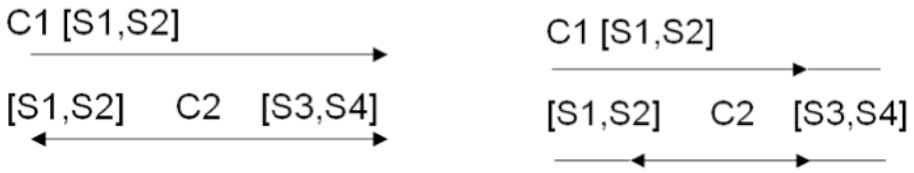


Fig.8 Canales, síncronos y asíncronos

En el diagrama del sistema aparece un bloque tipo, en este caso se llama AccessPoint. Aprovecharemos éste bloque tipo para ver que en su interior tenemos los procesos, y que también existen canales para comunicar los procesos y sus correspondientes señales. También pueden existir bloques simples o individuales que se emplean una única vez.

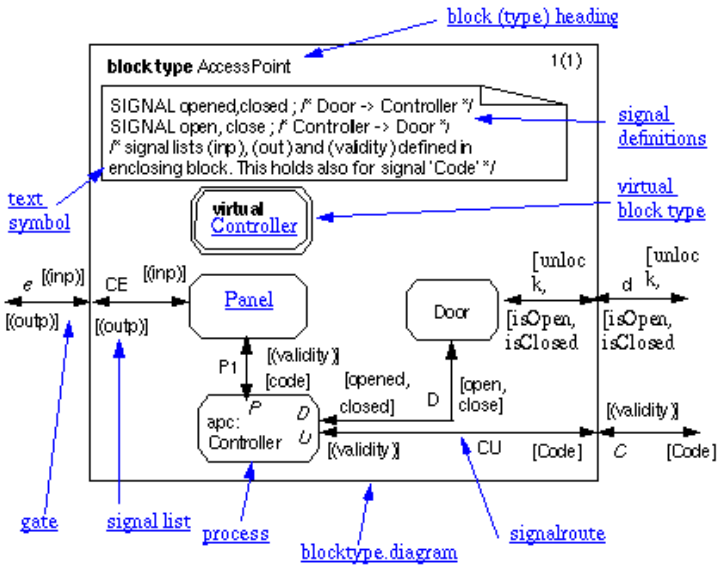


Fig.9 Especificación de un bloque en SDL

En este diagrama se puede observar la comunicación entre procesos. Nos fijaremos en el proceso denominado Controler. Es un proceso tipo que también está bien definido, lo que nos permitirá adentrarnos en el último nivel del sistema, los procedimientos de un proceso.

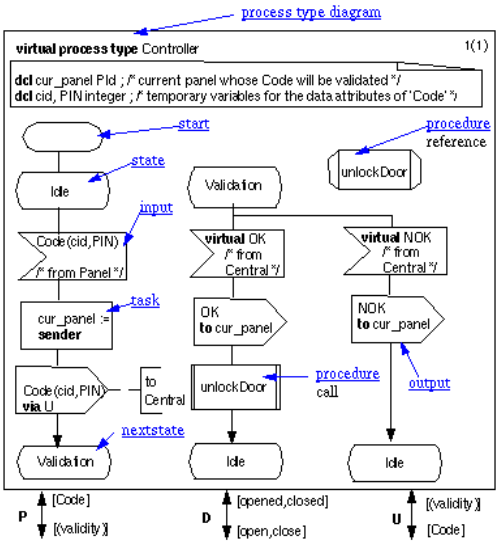


Fig.10 Especificación de un proceso en SDL

A este nivel se pueden observar variables, estados, entradas y salidas, acciones y se pueden incluir referencias a otros procedimientos para poder crear llamadas a éstos. También a éste nivel definimos canales y señales.

Por último, llegados a este punto únicamente nos queda hacer un resumen de los símbolos utilizados y mostrar un ejemplo de proceso que puede ser interesante en nuestro proyecto.

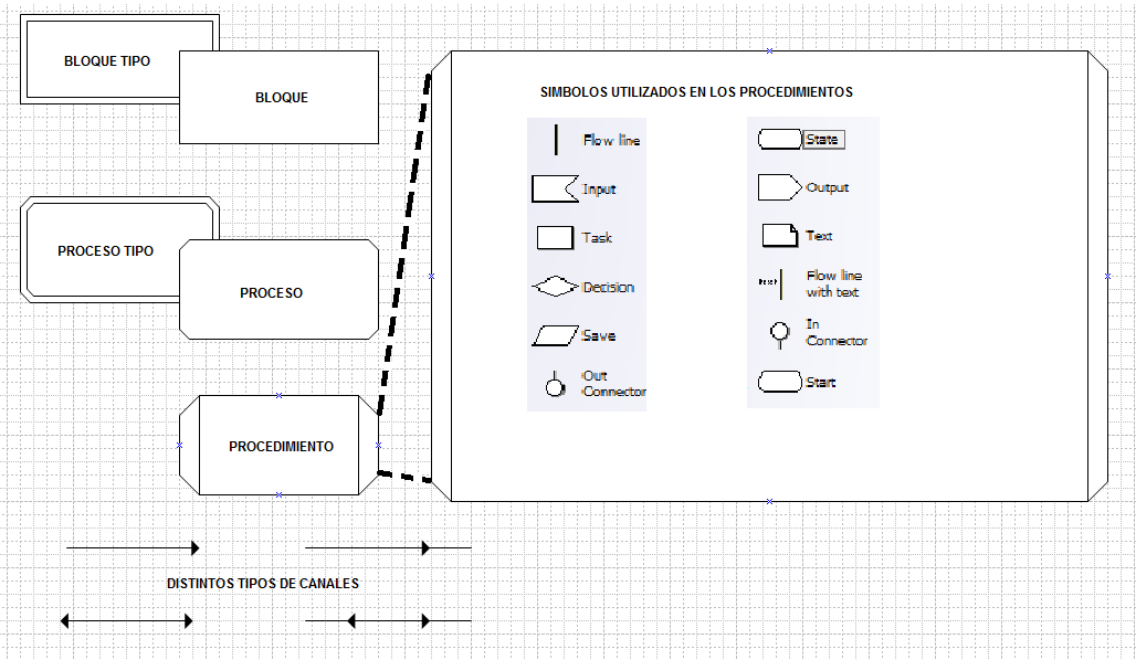


Fig.11 Simbología comunmente utilizada en SDL

En el siguiente ejemplo se puede observar un proceso en el que existen dos estados. Uno en el que se emiten datos y en el otro estado se está esperando el Ack que confirma de que se han recibido correctamente.

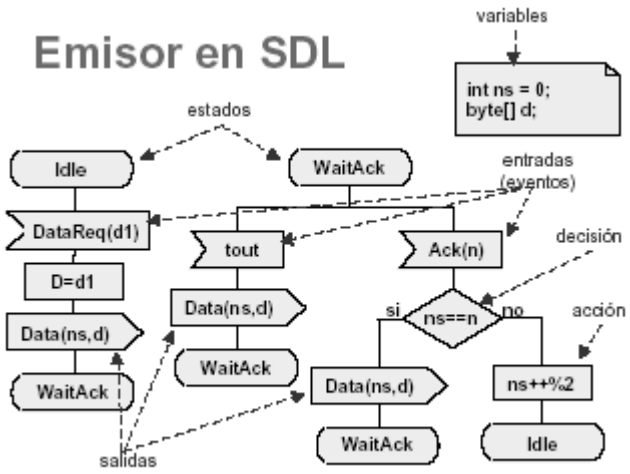


Fig.12 Ejemplo de emisor en SDL

Simulación básica

Introducción

En este apartado se describirán los pasos que se han dado para implementar el entorno de trabajo en el cual realizaremos las simulaciones.

En primer lugar, a modo de comentario, hemos utilizado una distribución de Linux como sistema operativo en el que desarrollar el trabajo de simulación. En concreto la distribución es una Ubuntu en su versión 9.10.

Fig.13 Logo Ubuntu

En cuanto a OMNeT++, hemos instalado la versión 4.0. Existe una versión posterior, la 4.1, pero se ha optado por la versión 4.0 por cuestiones de compatibilidad con los paquetes de software INET framework y la extensión de éste, el proyecto ReaSE.

Herramientas utilizadas

A continuación se detallan algunos de los pasos seguidos durante la instalación.

En primer lugar tenemos que instalar una serie de paquetes, sobre todo de programación, compiladores, librerías, etc. para tener la funcionalidad esperada en el programa principal, estos paquetes se denominan dependencias. Esta instalación se puede realizar mediante el siguiente comando:

```
$ sudo apt-get install build-essential gcc g++ bison flex perl \
tcl-dev tk-dev blt libxml2-dev zlib1g-dev openjdk-6-jre \
doxygen graphviz openmpi-bin libopenmpi-dev libpcap-dev
```

A continuación, una vez obtenido el paquete comprimido con la versión de OMNeT++, lo descomprimimos mediante el siguiente comando:

```
tar xvfz omnetpp-4.1-src.tgz
```

Una vez descomprimido se habrá creado una carpeta con todos los archivos necesarios y nos situamos en ella. Tenemos que configurar el código, después compilarlo y después instalarlo. Estas acciones se realizan de la siguiente forma:

```
cd omnetpp-4.1
./configure
make
make install
```

Es importante indicar en las variables de sistema, la ruta en la que se encuentran los ejecutables del programa y la librería tcl8.4, debemos incluir las siguientes 2 líneas en el archivo bashrc de nuestro usuario:

```
export PATH=$PATH:/home/inaki/omnetpp-4.p1/bin
export TCL_LIBRARY=/usr/share/tcltk/tcl8.4
```

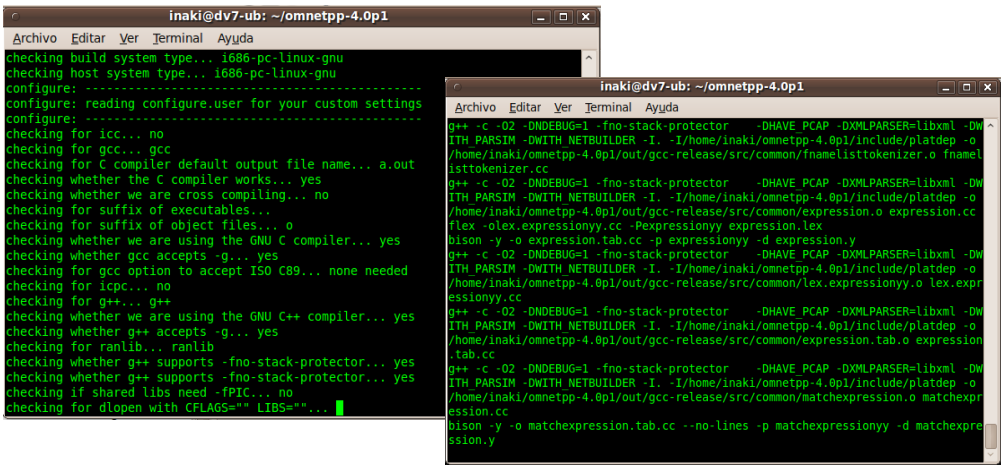


Fig.14 Configurando y compilando OMNeT++

Ya tenemos instalado OMNeT++ 4.0. A continuación podemos ejecutar el programa con el comando ./bin/omnetpp. Encontraremos una interfaz de usuario basada en eclipse en la que incluiremos los componentes de la simulación.

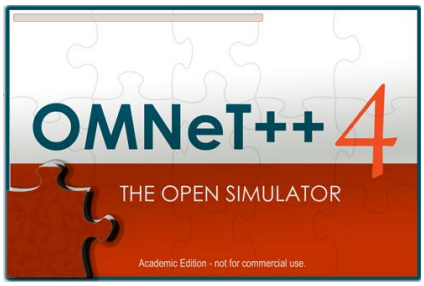


Fig.15 Logo OMNeT++ 4.0

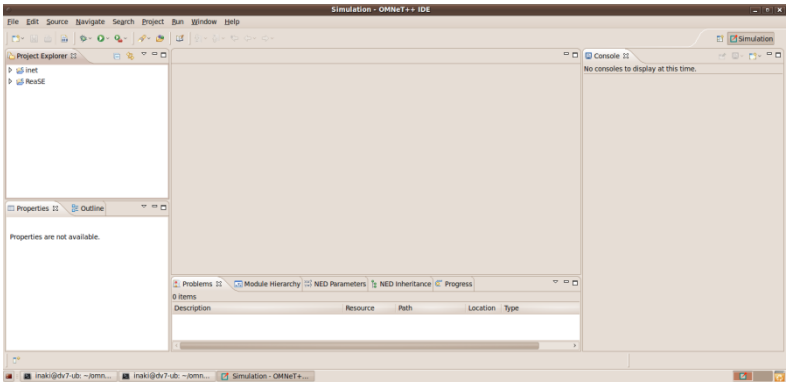


Fig.16 Workbench de OMNeT++

El siguiente paso para crear el entorno de simulación es importar los proyectos Inet y ReaSE. Tendremos que especificar el directorio raíz de estos proyectos, que serán los que se creen después de descomprimir los archivos obtenidos en las webs correspondientes. Las rutas son las siguientes:

/home/inaki/inet-framework-inet-a258c19
/home/inaki/ReaSE

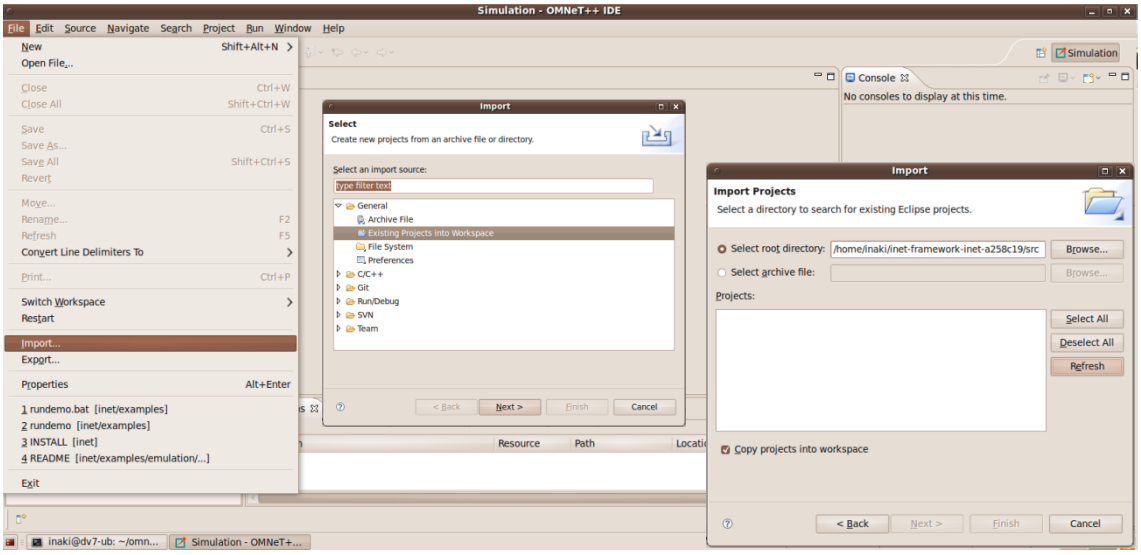


Fig.17 Importando los proyectos Inet y ReaSE

Llegados a este punto tenemos que configurar el código que será compilado, referenciar Inet a la extensión ReaSE y por último compilarlo.

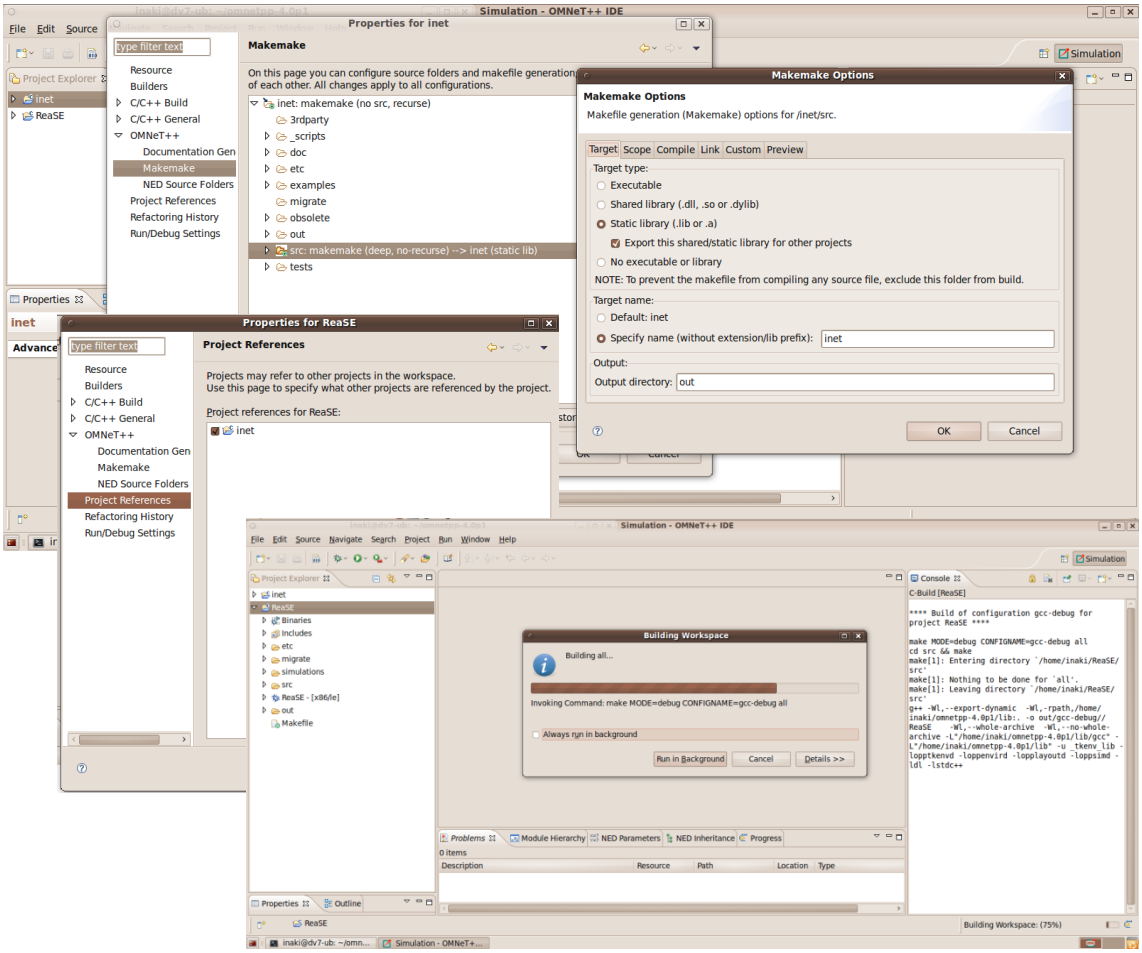


Fig.18 Configurando y compilando los proyectos Inet y ReaSE

Una vez compilado el proyecto podemos configurar la ejecución. Es en este punto en el que elegimos la simulación que vamos a realizar. De momento seleccionaremos ejemplos y veremos la forma de configurarlos, ejecutarlos y obtener los resultados.

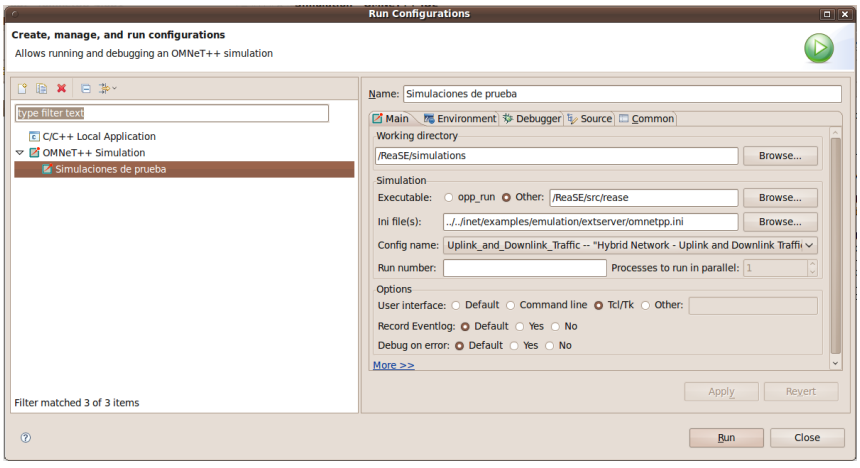


Fig.19 Configurando la ejecución de una simulación

Ejemplos

Ej.1. Mixed Lan (Red mixta)

Hemos seleccionado un ejemplo en el que se puede observar una red mixta. Esta red tiene un parámetro de inicialización que se pide al iniciar la ejecución. Por otro lado tenemos una ventana en la que se aprecia la topología de la red a simular y en la que podremos configurar valores pinchando sobre los objetos de la red.

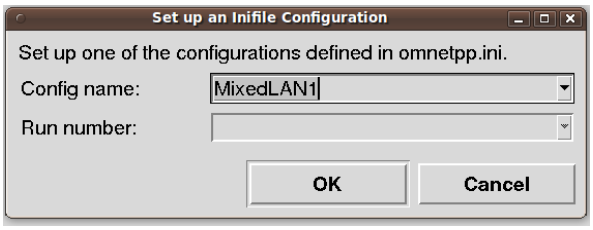


Fig.20 Parámetros de inicialización de una simulación

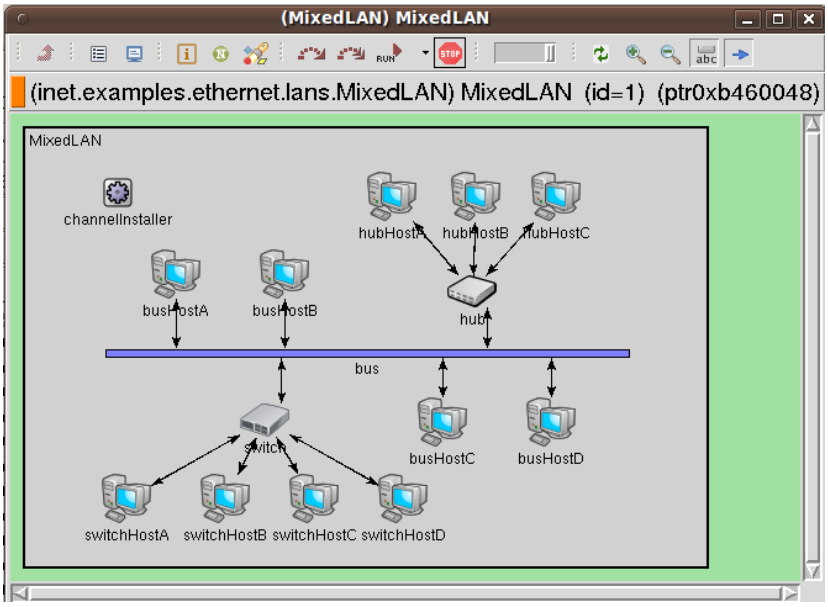


Fig.21 Topología de la simulación MixedLan

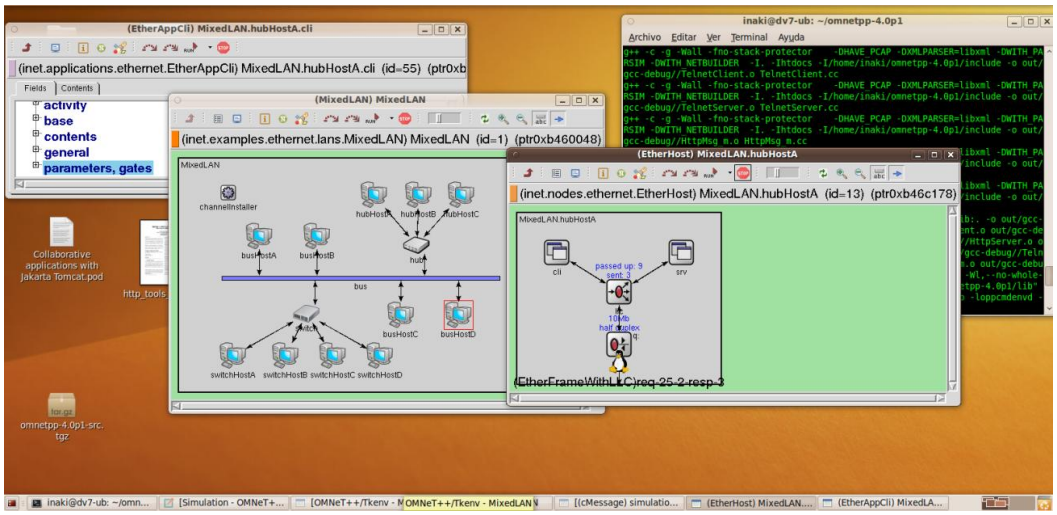


Fig.22 Configuración de los parámetros de los componentes de la simulación

También podemos ver una animación de la ejecución en la que las figuras de unos pingüinos simulan ser paquetes que viajan por ésta red.

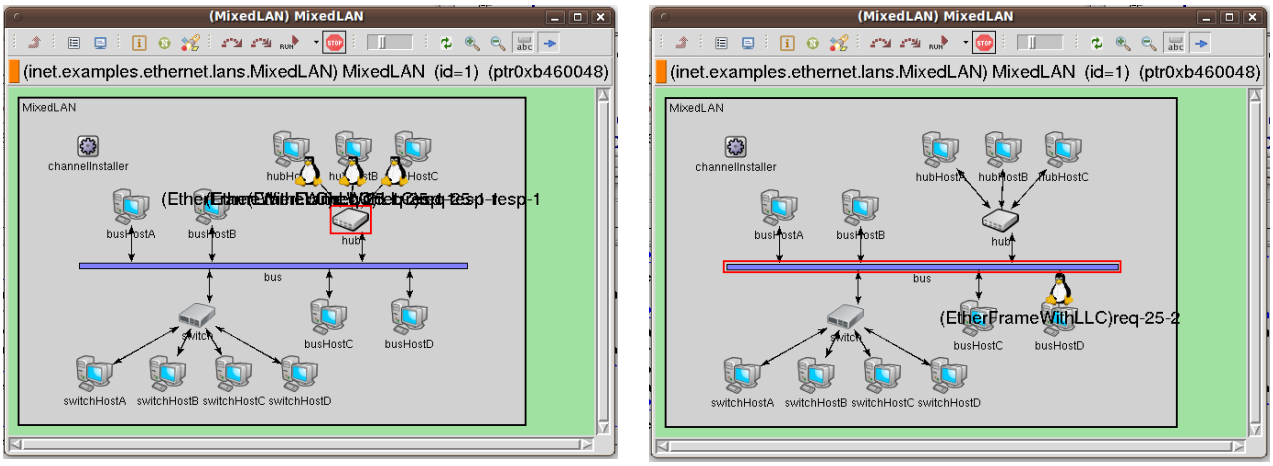


Fig.23 Ejecución de la simulación. Se puede observar una animación

Existe también durante la ejecución un listado de los eventos que van sucediendo y el tiempo en el que producen.

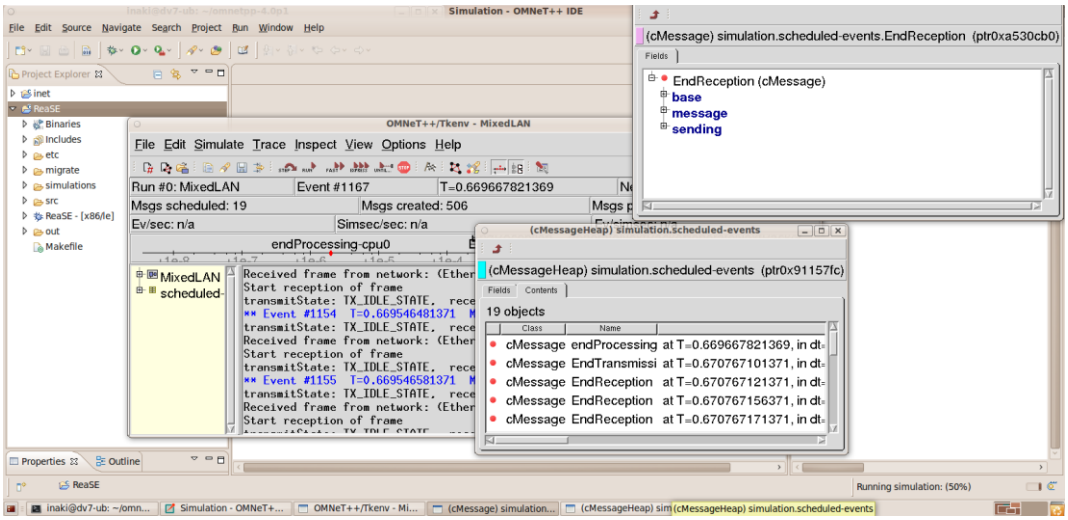


Fig.24 Listado de eventos

Antes de iniciar la ejecución podemos configurar algunos parámetros como el tiempo de duración de la ejecución, elegir un archivo en el que se guardará la salida generada, etc.

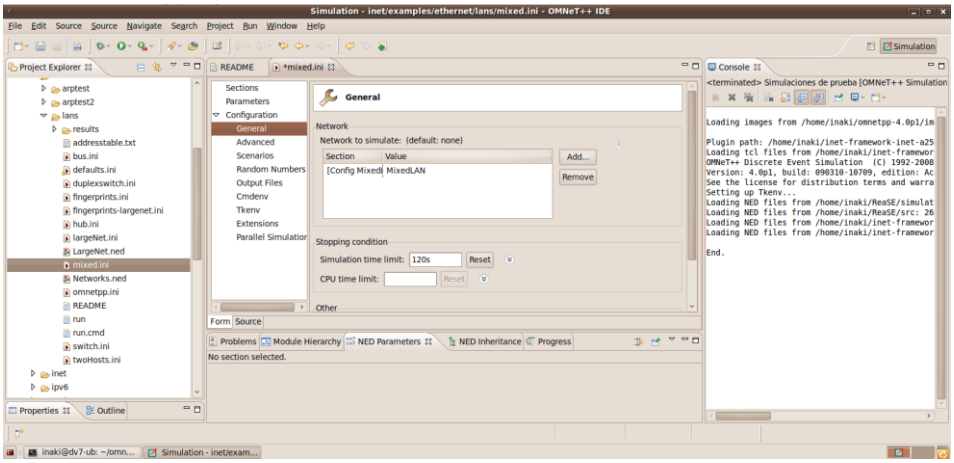


Fig.25 Configuración mediante archivos .ini

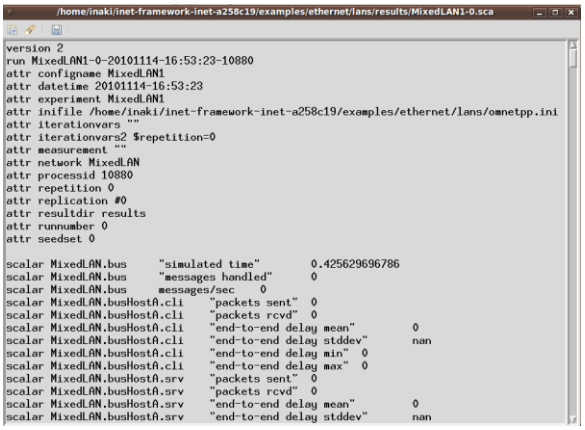


Fig.26 Archivo de salida, resumen de los resultados obtenidos

A continuación vamos a continuar con otros dos ejemplos en los que se aprecian unas características que nos vendrán bien en la siguiente etapa del proyecto cuando creemos la simulación sobre el sistema "real".

Ej.2 Red con clientes wifi

En el primero de los ejemplos podemos observar cómo se puede configurar un número determinado de clientes, en este caso inalámbricos, mediante el parámetro de inicialización.

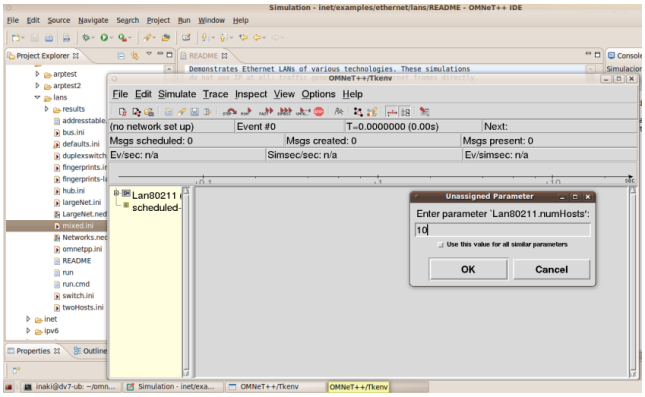


Fig.27 Inicialización de parámetros. Número de hosts

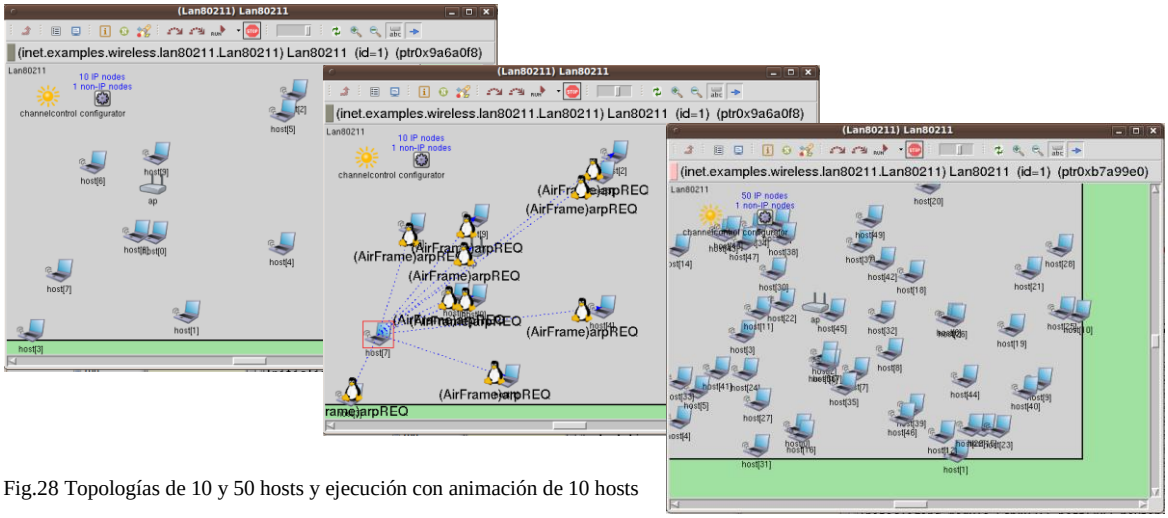


Fig.28 Topologías de 10 y 50 hosts y ejecución con animación de 10 hosts

Ej.3. Comunicación con servidor externo

Por último tenemos un ejemplo en el que se puede configurar la dirección de los dispositivos de una red para que apunten a máquinas externas reales.

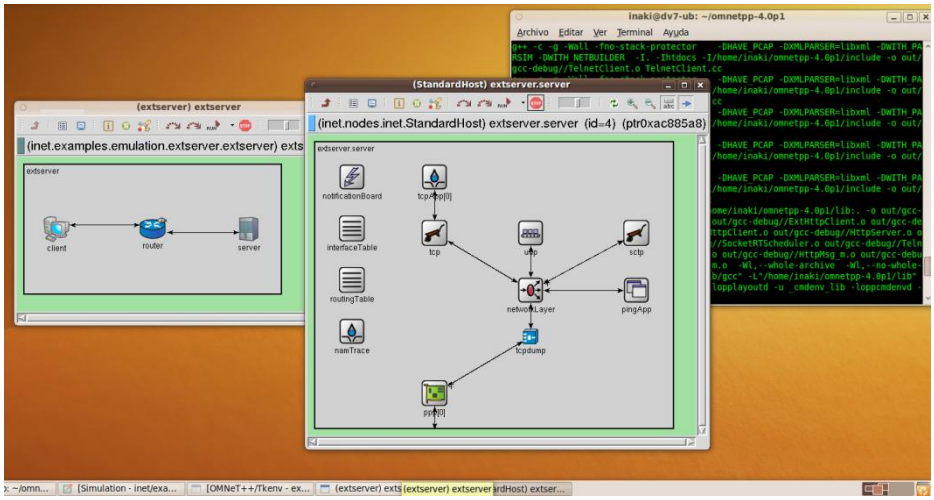


Fig.29 Topología de la simulación del servidor externo

Tendremos a nuestra disposición en el fichero de inicialización la posibilidad de definir la dirección ip externa, así como una puerta de enlace e incluso la cantidad de tráfico que puede manejar.

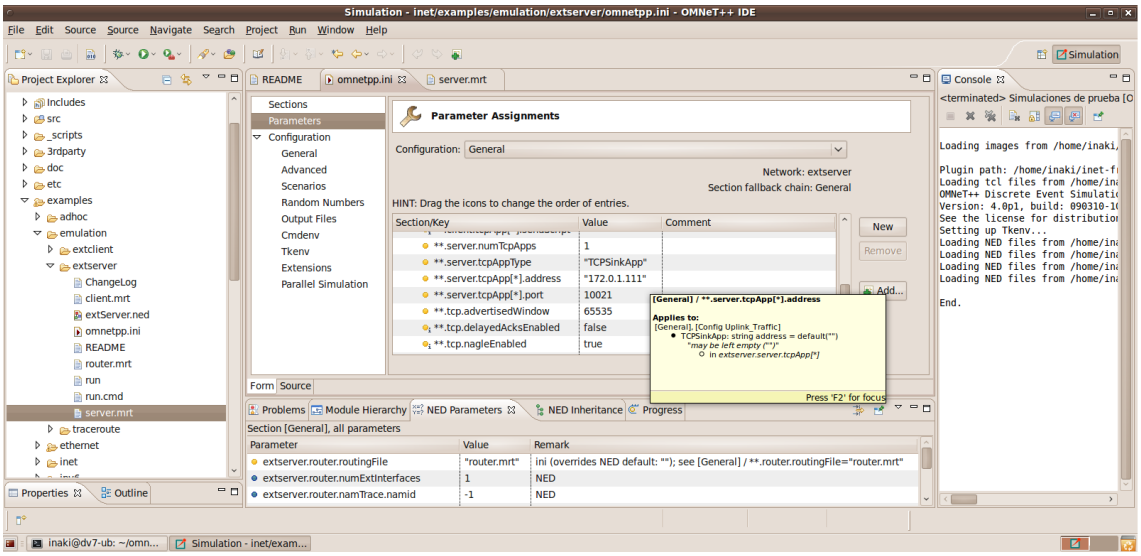


Fig.30 Configuración de la simulación. Detalle de configuración de dirección ip del servidor.

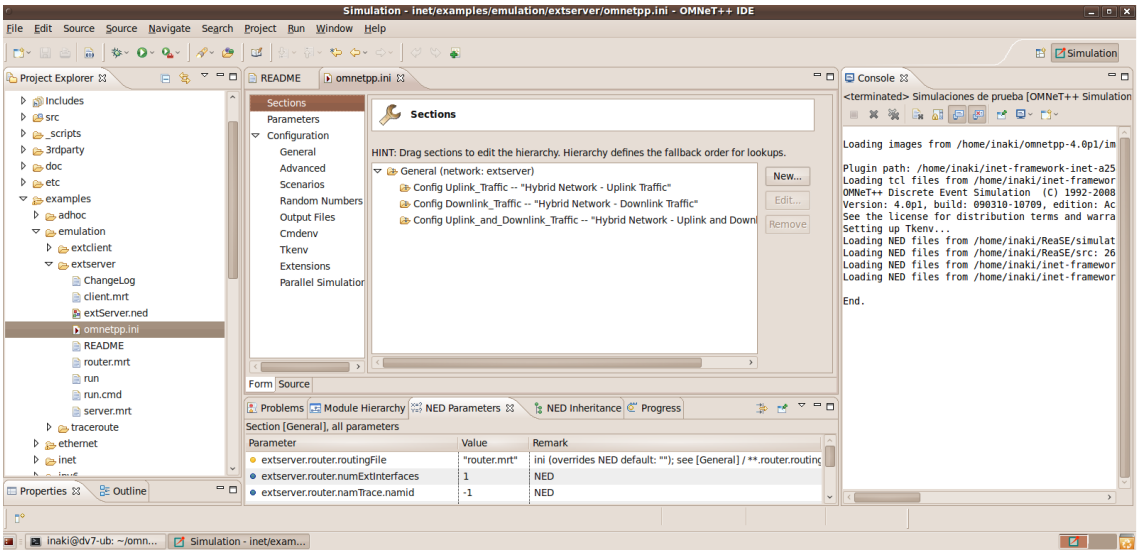


Fig.31 Distintos modos de ejecución de la simulación

Lamentablemente tenemos un problema llegados a este punto al ejecutar la simulación. Falta la librería pcap, por lo que se genera un error y se para la simulación. Ésta librería es necesaria para generar el tráfico de red que va al exterior, es decir a la otra máquina. Se corregirá este error más adelante incluyendo la librería necesaria y recompilando OMNeT++.

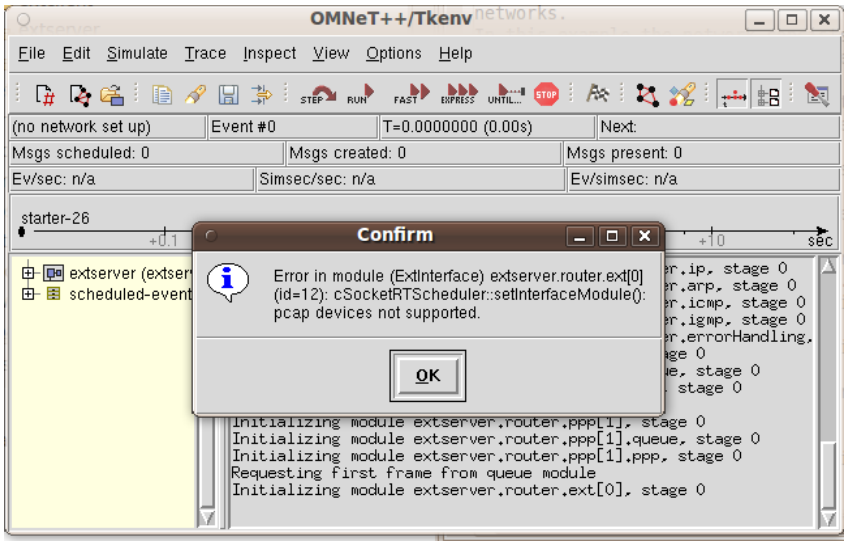


Fig.32 Error con librerías pcap

En la próxima etapa del proyecto combinaremos estas tres simulaciones para crear el entorno o escenario que necesitamos para nuestro caso particular.

Servidor Tomcat Básico

Introducción

Tomcat es un servidor web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones, como JBoss o JOnAS. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache.

Tomcat puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

Los servlets son pequeñas aplicaciones programadas en java que nos permiten ejecutar programas en el lado del servidor. El servidor realiza determinadas acciones con los datos que ha proporcionado el cliente y devuelve una respuesta. Por ejemplo mediante el módulo de java JDBC, el servidor puede acceder a una base de datos y devolver la respuesta a la consulta realizada por un cliente.

Lo que buscamos es una serie de servlets que sean útiles para la interacción entre estudiantes, profesores o miembros de la universidad en general.

El sistema operativo en el que se aloja en este caso el servidor tomcat es debian. En concreto la versión 5.05. También es necesario instalar una serie de dependencias de java, librerías como por ejemplo jdbc, etc.

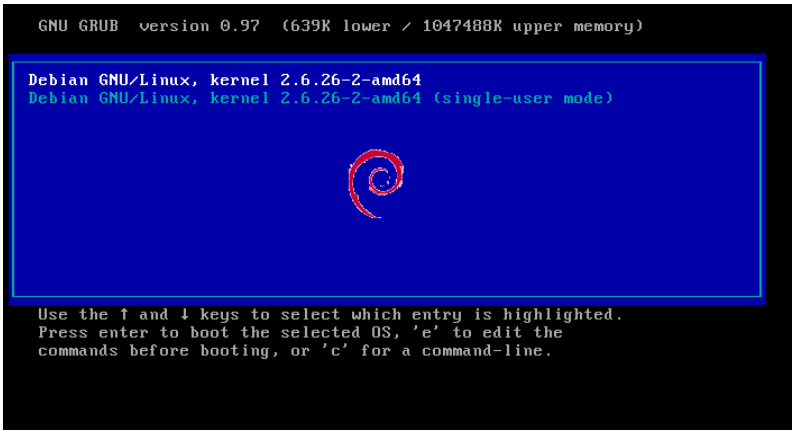


Fig.33 Arranque de Debian 5.05 y logo

Herramientas

Se mostrará el proceso que se ha seguido para conseguir que finalmente se ejecute el servicio con diversos servlets.

Como en el caso anterior con OMNeT++, Tomcat también necesita instalar una serie de paquetes adicionales, como por ejemplo el kit de programación en java JDK, el plugin java, etc. Todo lo necesario sobre java se instala con el siguiente comando.

dpkg --get-selections | grep sun-java
Se instalan los siguientes paquetes
sun-java6-bin
sun-java6-jre
sun-java6-plugin

La versión en concreto de java es jre6.0_22, además se ha instalado también el java development kit, jdk6.0_11.

Una vez instalado java procedemos a descomprimir el archivo obtenido de la web de apache software foundation en el que se encuentran todos los paquetes relacionados con tomcat. Se crea una carpeta con el nombre y la versión, /home/inaki/apache-tomcat-7.0.4

Al igual que con OMNeT++, tenemos que incluir una ruta en los archivos .bashrc de los usuarios. Es la que se muestra a continuación que corresponde con la raíz de java.

export JAVA_HOME=/usr/lib/jvm/java-6-sun

Siguiendo estos pasos hemos terminado con la instalación del servidor Tomcat. Existe la opción de configurar el arranque del servicio como un demonio, para que arranque automáticamente con el sistema, pero para arrancarlo más fácilmente tenemos que ejecutar el siguiente comando.

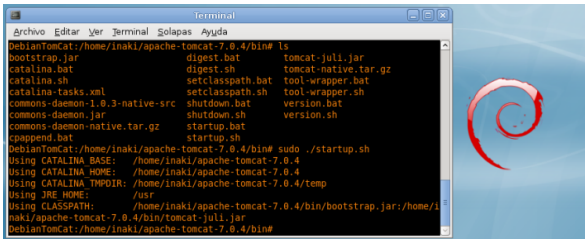


Fig.34 Arrancando Tomcat

A continuación ya podemos comprobar el correcto funcionamiento del servidor abriendo el navegador y escribiendo: *http://127.0.0.1:8080*

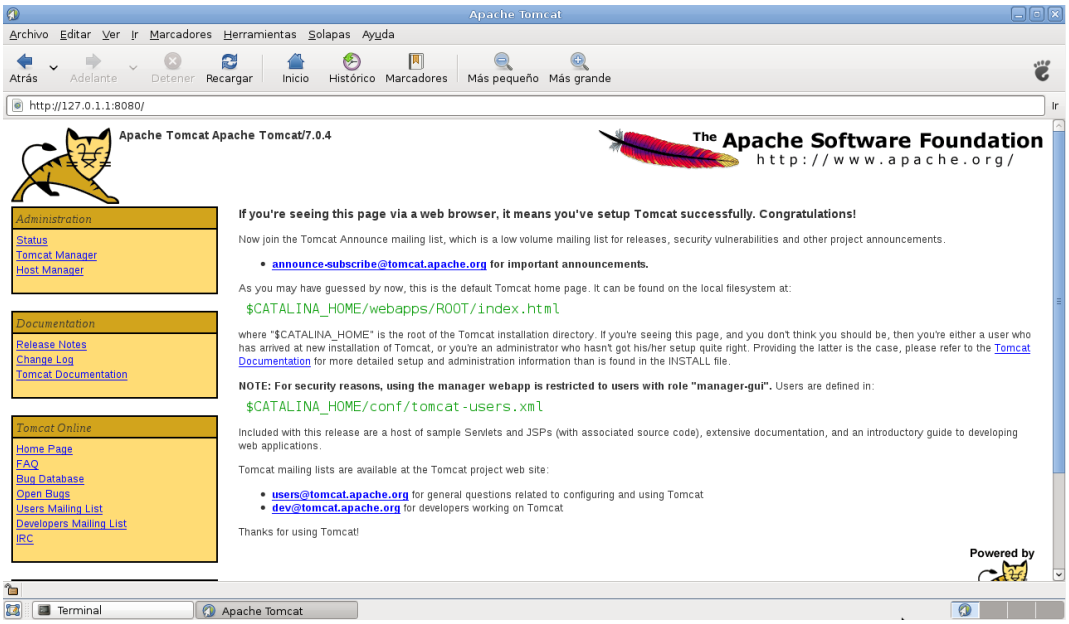


Fig.35 Página principal del servidor Tomcat 7.0.4

Ahora debemos hacer algún cambio en el archivo $CATALINA_HOME/conf/tomcat-users.xml. Tenemos que añadir las siguientes dos líneas para poder acceder como usuario a través del navegador para administrar el servidor.

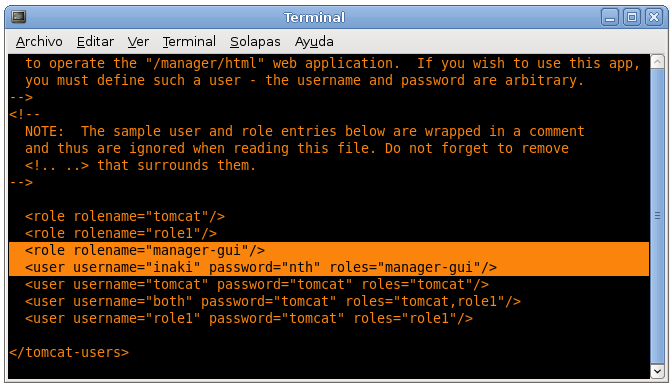


Fig.36 Configuración de usuarios

De esta manera al acceder al Tomcat Manager nos pedirá un nombre de usuario y una contraseña.

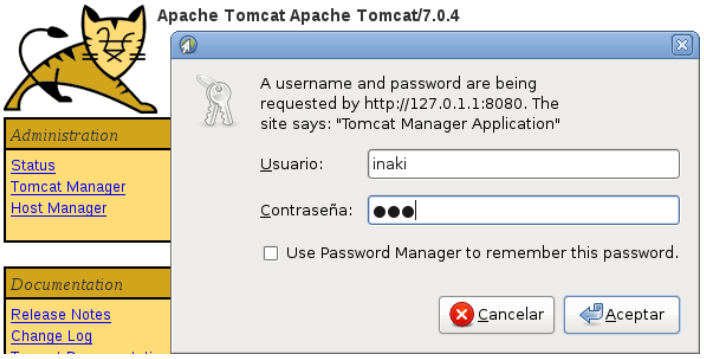


Fig.37 Login en Tomcat Manager Application

Accedemos de esta manera al gestor de aplicaciones web, desde donde podemos manejar la estructura de directorios, por ejemplo para establecer la página inicial, acceder a los ejemplos o consultar el estado del servidor.

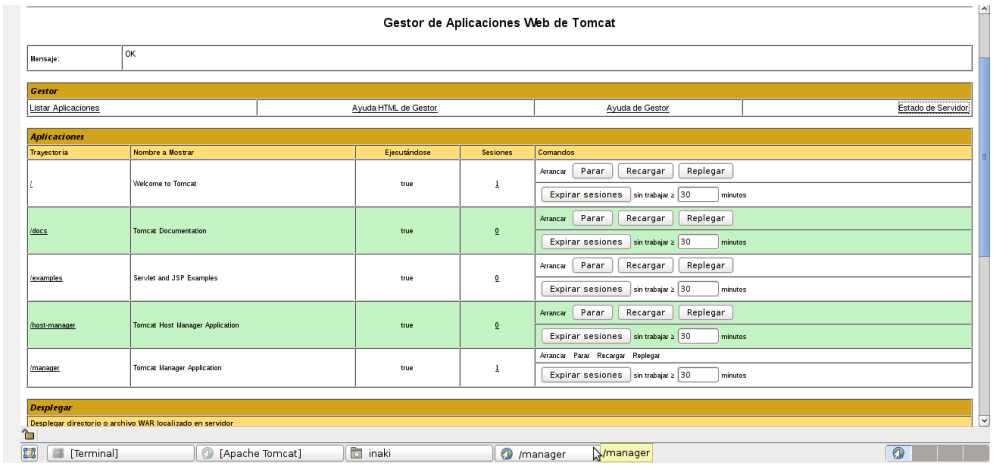


Fig.38 Gestor de aplicaciones

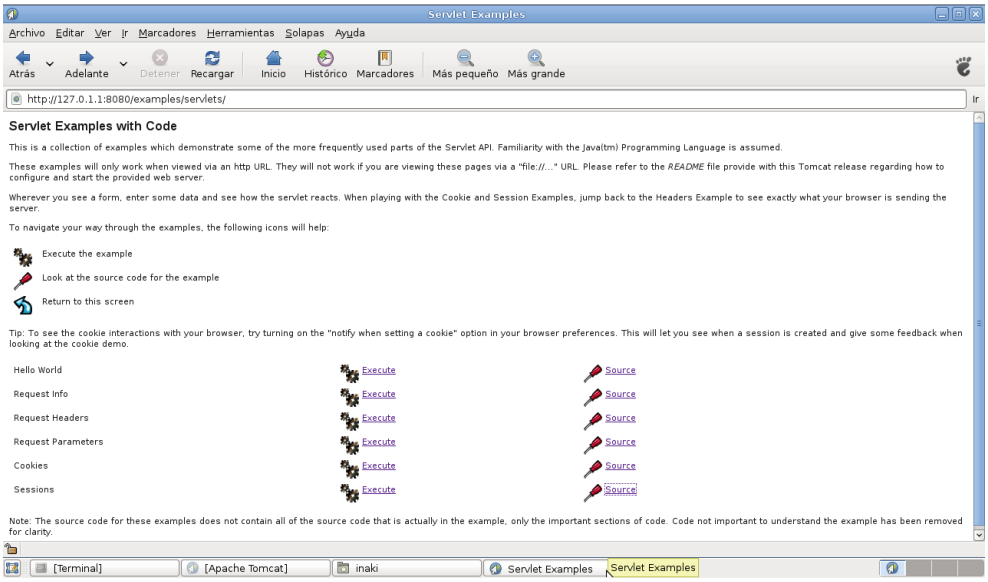


Fig.39 Directorio de ejemplos

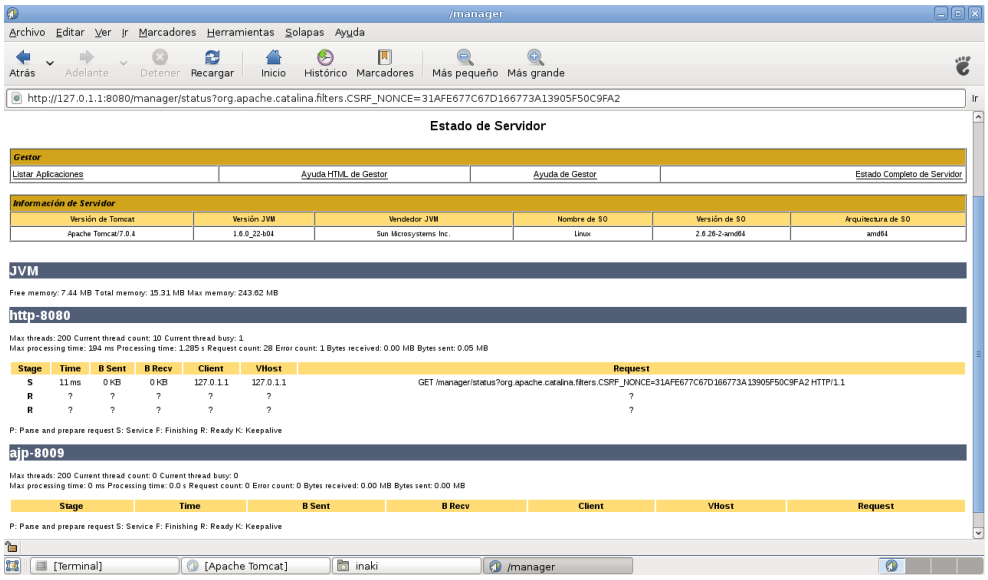


Fig.40 Estado del servidor

Otro de los archivos de configuración comúnmente utilizados en Tomcat es el archivo server.xml. Al igual que el archivo que contiene los usuarios está en la ruta $CATALINA_HOME/conf/. Editando éste archivo se puede configurar el puerto en el que escucha el servidor web, así como otros puertos, por ejemplo podríamos configurar la conexión segura utilizando secure sockets layer (SSL) descomentando las líneas oportunas y activando este servicio.

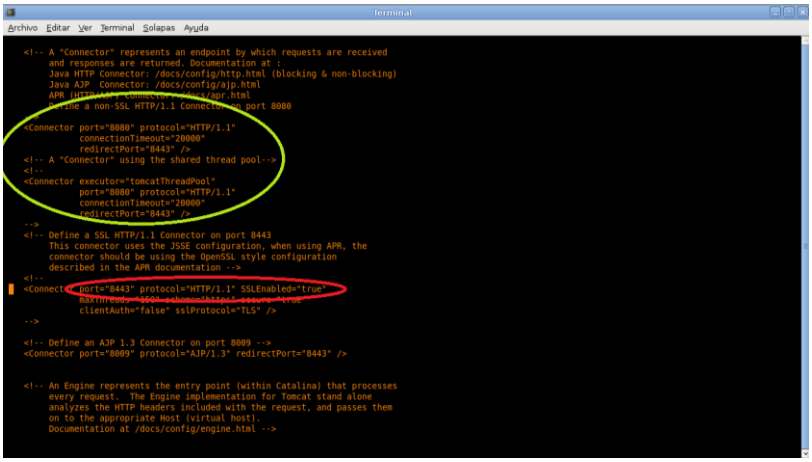


Fig.41 Configuración de puertos

Ejemplos

En este apartado se mostrarán algunos ejemplos aunque se espera avanzar más en el tema de las aplicaciones colaborativas durante la siguiente etapa del proyecto.

En primer lugar se muestra un ejemplo simple en el que mediante el paso de parámetros por parte del cliente, el servidor responde modificando la página que se está visualizando. Es uno de los ejemplos que vienen por defecto junto con la aplicación.

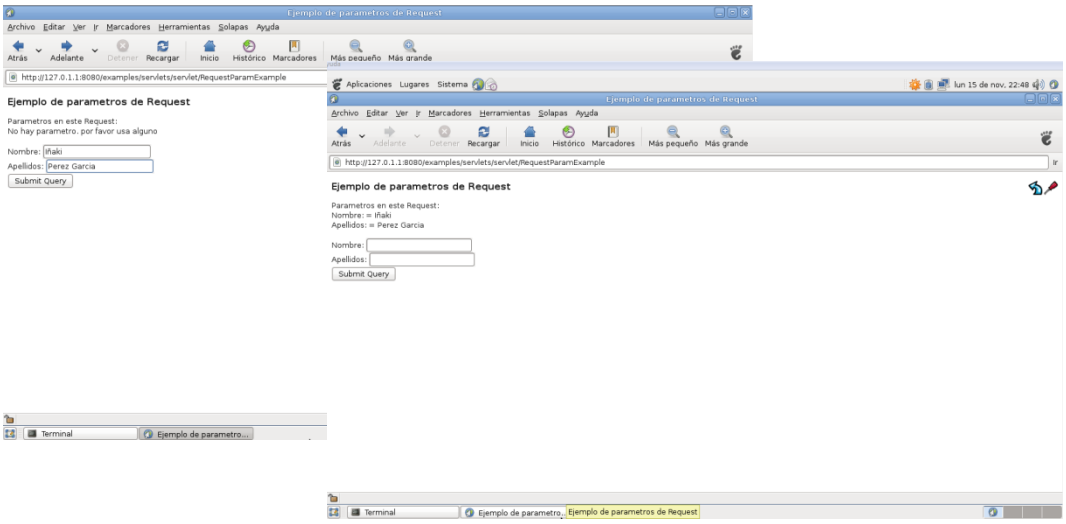


Fig.42 Ejemplo sencillo de servlet con parámetros

El siguiente ejemplo es algo más complejo. Corresponde a una serie de servlets que recogen y muestran información relativa a una universidad, datos de matrícula, alumno, formularios, etc.

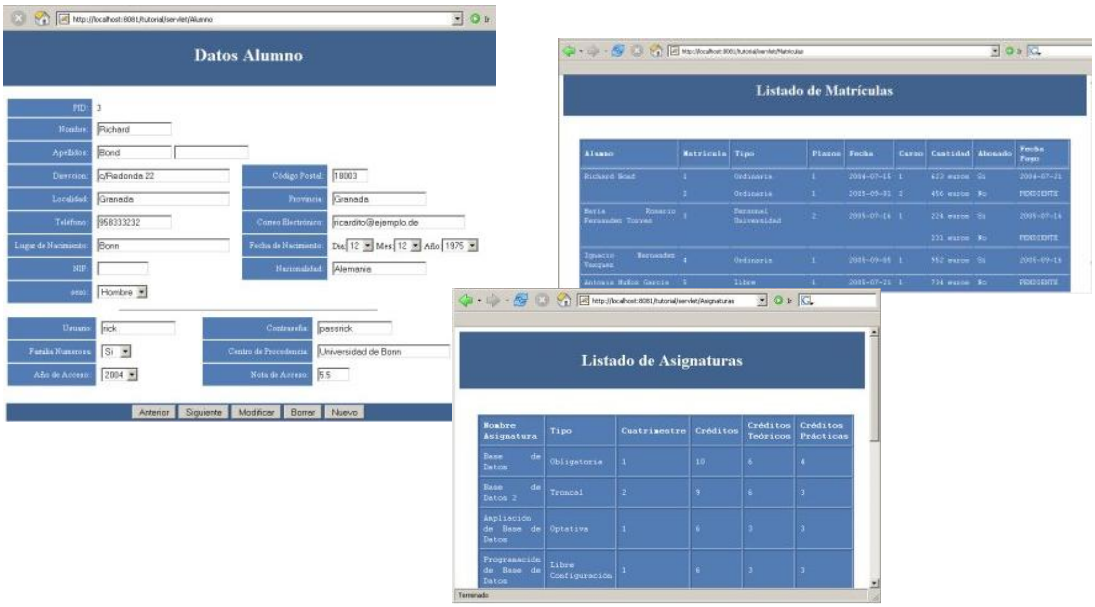


Fig.43 Ejemplo de servlet con conexión base de datos JDBC

Es posible modificar el código java mediante la aplicación eclipse para ajustar los servlets a nuestras necesidades.

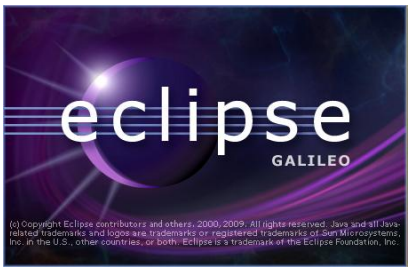


Fig.44 Logo Eclipse

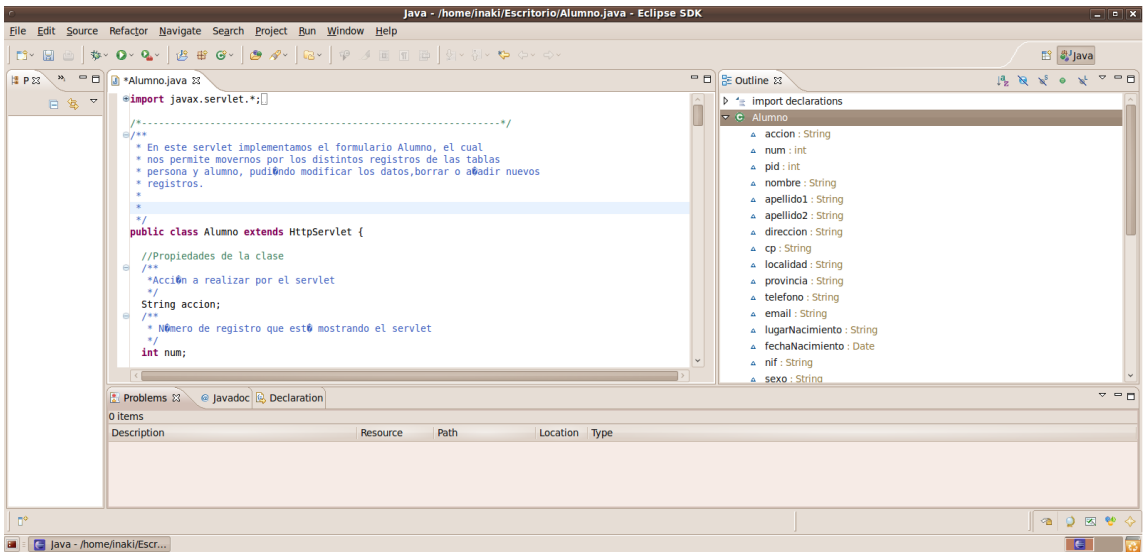


Fig.45 Edición del código de un servlet con Eclipse

Por último dentro del apartado de ejemplos de servlets para el servidor Tomcat se muestran una serie de aplicaciones colaborativas programadas en lenguaje java cuya investigación más profunda se realizará en la siguiente etapa del proyecto.

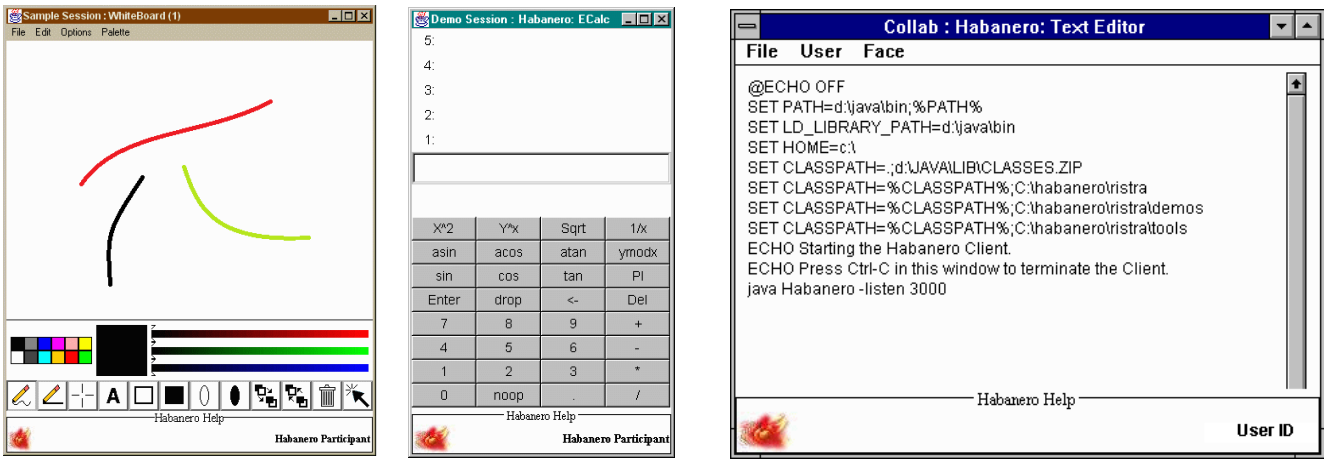


Fig.46 Ejemplos de aplicaciones colaborativas del proyecto Habanero

WhiteBoard permite compartir imágenes y dibujos en una sesión. Es posible dibujar libremente o mostrar imágenes gif y jpeg.

Ecalc presenta un visor a cada usuario en el que se pueden realizar operaciones matemáticas básicas y algunas funciones algo más complejas.

Text Editor permite abrir archivos de texto para que sean modificados por distintos usuarios, es posible dar formato al texto y distinguir a los usuarios por colores.

Conclusiones

Durante esta segunda etapa del proyecto se ha realizado una investigación básica en tres áreas, el lenguaje SDL, la simulación con OMNeT++ y el servidor Tomcat con sus servlets.

Se ha llegado a un nivel de comprensión del software utilizado en el proyecto bastante aceptable, aunque se han tenido algunos problemas de configuración y se ve una dificultad bastante elevada para la siguiente etapa en la que se deberán modificar y combinar los ejemplos mostrados en el presente documento, así como dotar de mayor funcionalidad a las aplicaciones principales.

Existe una gran cantidad de información gráfica de las capturas de pantalla que será necesaria para recordar ciertas partes de los programas utilizados, aunque se espera que el nivel de detalle en las explicaciones de la instalación y configuración del software del entorno de trabajo sea mucho menor en la memoria final. Llegados a tal punto se dará más importancia al caso particular del presente proyecto.

Investigación avanzada

Introducción

Al inicio del presente proyecto se estableció un plan de trabajo en el que se contemplaba la implementación de un entorno de trabajo en el que se incluían una serie de herramientas o aplicaciones que nos permitirán alcanzar los objetivos propuestos.

Durante cuatro semanas se realizó la instalación de dichas herramientas, así como un trabajo de investigación básica para familiarizarse con el software y las tecnologías utilizadas.

Se trabajo con ejemplos de cada una de las partes que comprenden el proyecto, es decir, el modelado con SDL, la simulación con omnetpp y el propio servidor Tomcat.

Durante ésta tercera etapa del proyecto se ha intentado profundizar en la utilización de todas estas herramientas planteando dos modelos a partir de los cuales se pretenden alcanzar los objetivos particulares planteados.

Se ha intentado hacer una simulación que conectara con el servidor real de forma física, pero ante ciertos problemas detectados se ha optado por una simulación contra un servidor virtual, utilizando parámetros de las pruebas de comportamiento realizadas contra esa parte real.

En definitiva en el presente documento se profundiza en la utilización de las herramientas y tecnologías software para adaptarlas a nuestros objetivos. También se explican con detalle los problemas que han ido apareciendo y como se plantean soluciones alternativas ya que todos estos errores y su solución forman parte de cualquier proyecto y sirven para adquirir conocimiento añadido.

Por último se hará una exposición de los resultados obtenidos en las diferentes pruebas, la comparación de éstos y las correspondientes conclusiones. Uno de los objetivos de ésta etapa es la comparación de los dos modelos definidos con SDL. Se mantiene el objetivo principal de complementar los servicios ofrecidos por el campus de la Uoc e intentando contribuir de esta manera con el proyecto Castelldefels.

Resumen

El objetivo principal del trabajo de investigación avanzada realizado es la adaptación del software y las tecnologías utilizadas en el entorno de trabajo a los dos modelos que serán comparados.

En primer lugar se utilizará el lenguaje de descripción y especificación, SDL. Mediante el programa Sandrilla se modelan dos escenarios distintos. En el primero de ellos existe un único servidor al que se conectan todos los estudiantes de la Uoc. En el segundo modelo, los estudiantes se dividen dependiendo de la carrera en la que estén matriculados y cada grupo se conecta a uno de los servidores, que en realidad son múltiples instancias del mismo servidor Tomcat que escuchan en puertos diferentes.

Después nos centraremos en las herramientas de simulación. En primer lugar se ha intentado adaptar una simulación de ejemplo llamada extServer para que los clientes simulados se conecten de forma física al servidor o servidores reales. Se han encontrado varios problemas que han impedido su correcto funcionamiento, se explicarán estos errores con detalle y también cuales eran las intenciones y objetivos buscados con esta simulación.

De forma alternativa se ha realizado otra simulación en la que se utilizan parámetros obtenidos de ciertas pruebas de comportamiento realizadas sobre la parte real.

Por otra parte se dará una explicación de los pasos que se han seguido para configurar el servidor Tomcat para que se ejecute de forma múltiple. También se ha hecho una suposición de cuál sería la manera de integrar las aplicaciones colaborativas de Habanero sin llegar a profundizar en el caso ya que no dispondríamos del tiempo suficiente para adaptar estos servlets.

También se da una explicación generalista de todas las aplicaciones disponibles y una explicación algo más detallada de varias de éstas aplicaciones que son interesantes para el proyecto. Se realizan pruebas de comportamiento sobre éstas aplicaciones.

Por último nos quedará el análisis de los resultados obtenidos a partir de las simulaciones y las pruebas de comportamiento. Se obtendrán gráficos a partir de archivos output de las aplicaciones.

Lenguaje SDL (Specifications and Description Language)

Introducción

En esta etapa del proyecto ya se conocen los principales elementos que componen un diagrama SDL y por lo tanto se aplicarán para realizar los modelos que nos interesan.

Como se puede observar se han modelado dos sistemas distintos. Uno algo más sencillo en el que varios clientes se conectan a un único servidor Tomcat y el otro en el que distintos tipos de estudiantes se conectan a distintos servidores mediante múltiples instancias de Tomcat.

Servidor único (Modelo 1)

En este caso se puede observar que todos los clientes son del mismo tipo. Lo único que harán los clientes es abrir cada uno de ellos un navegador que realice la petición de conexión al servidor tomcat. Ésta simulación nos servirá para saber qué cantidad de clientes es capaz de soportar Tomcat. La respuesta del servidor se devuelve a cada cliente en particular. Se recogerán capturas del tráfico de red con wireshark.

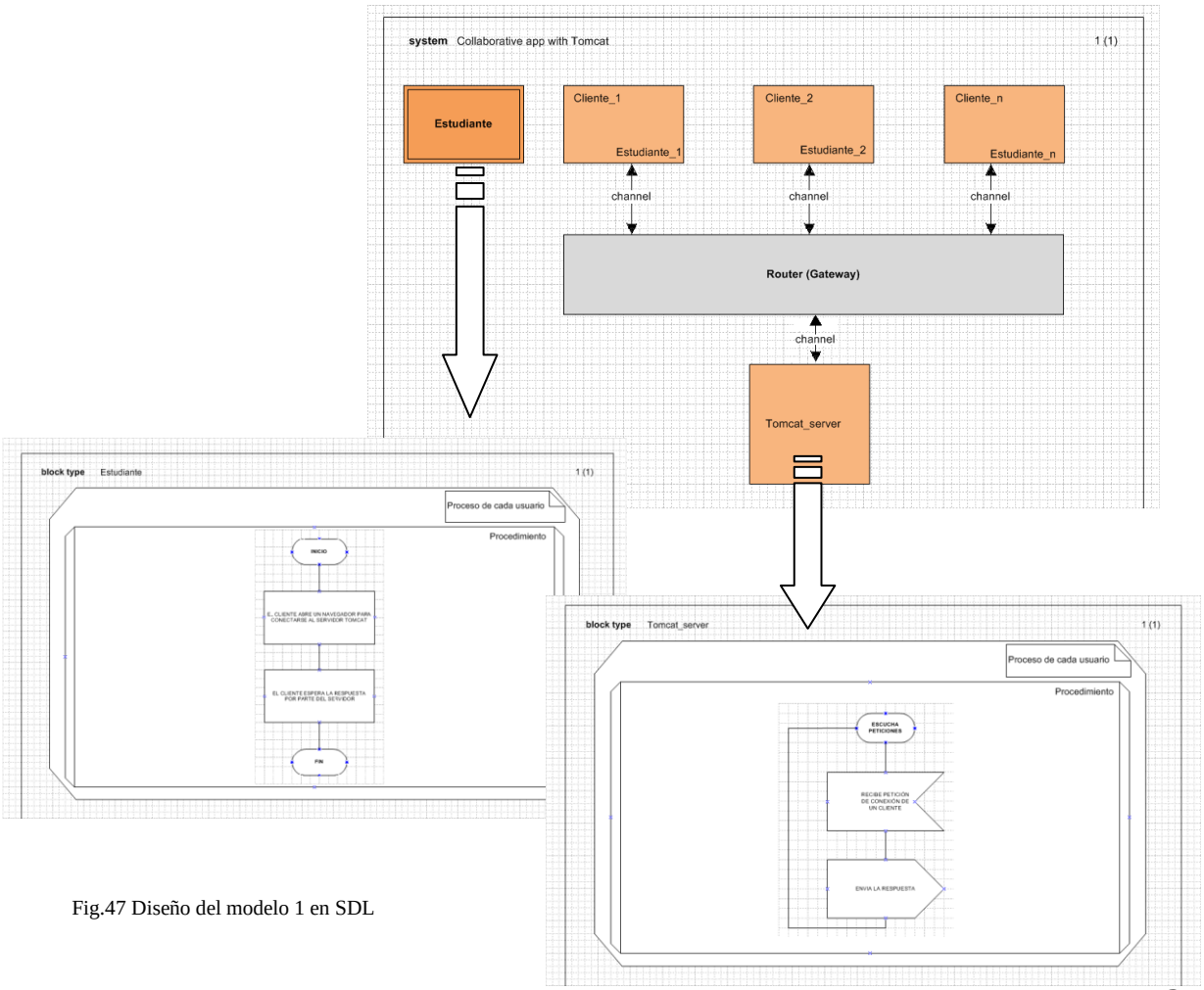


Fig.47 Diseño del modelo 1 en SDL

Múltiples instancias del servidor Tomcat (Modelo 2)

En este caso existen distintos tipos de clientes, que representan a distintos tipos de estudiantes que se distinguen por la carrera que estudian. El objetivo de la simulación es comprobar la respuesta del servidor ante distintas cantidades de tráfico. En realidad cada tipo de estudiante se conectará a una aplicación colaborativa diferente. También existen distintos servidores, cada uno de los cuales escuchará en un puerto diferente a su respectivo tipo de cliente. En este caso se dispone de un servidor interno en Omnet, que recogerá la respuesta de todos los servidores Tomcat ofreciéndonos la información de salida a analizar.

A continuación se observa el SDL del sistema completo.

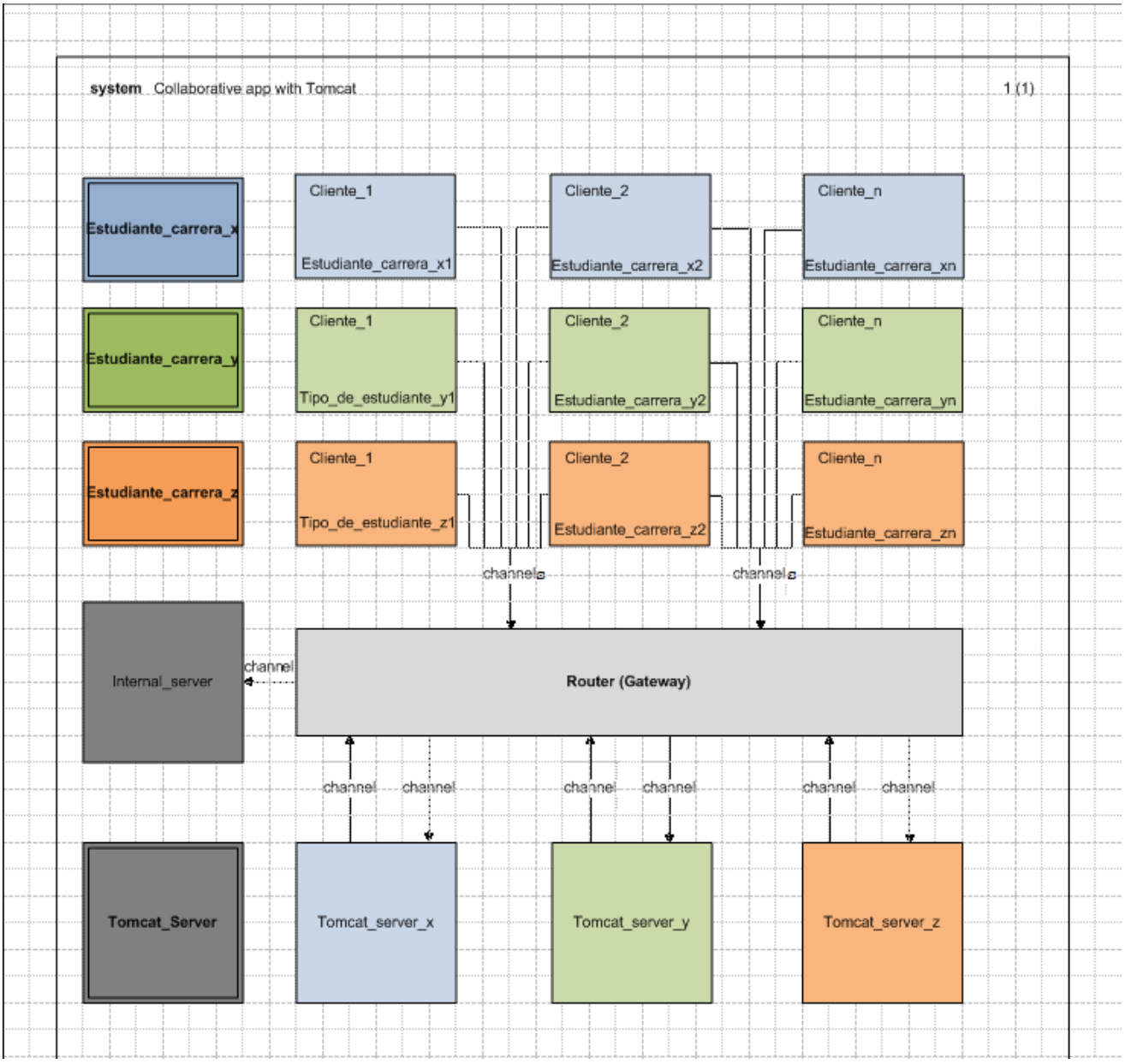


Fig.48 Diseño del modelo 2 en SDL

A continuación se muestran los procedimientos de los distintos tipos de estudiantes.

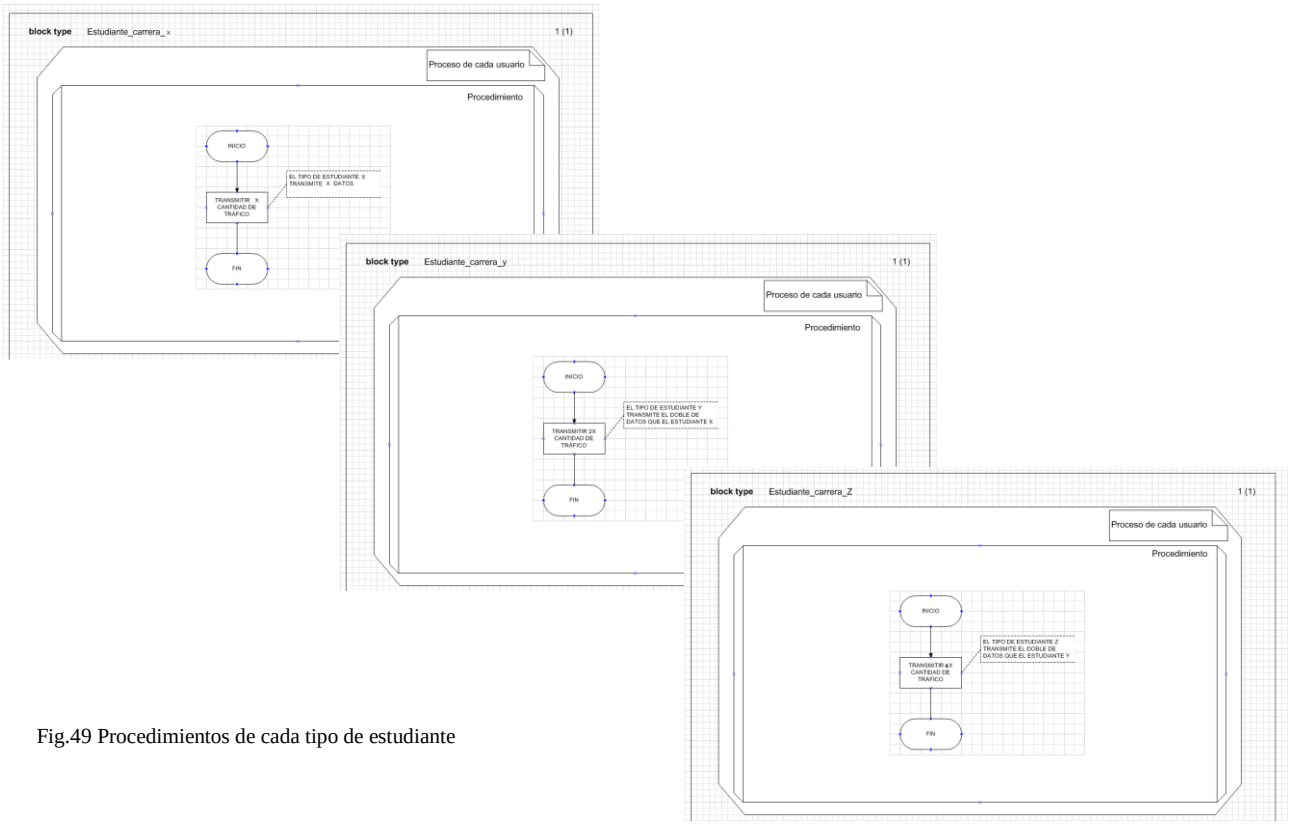


Fig.49 Procedimientos de cada tipo de estudiante

También se podemos ver los procedimientos de los distintos tipos de servidores Tomcat.

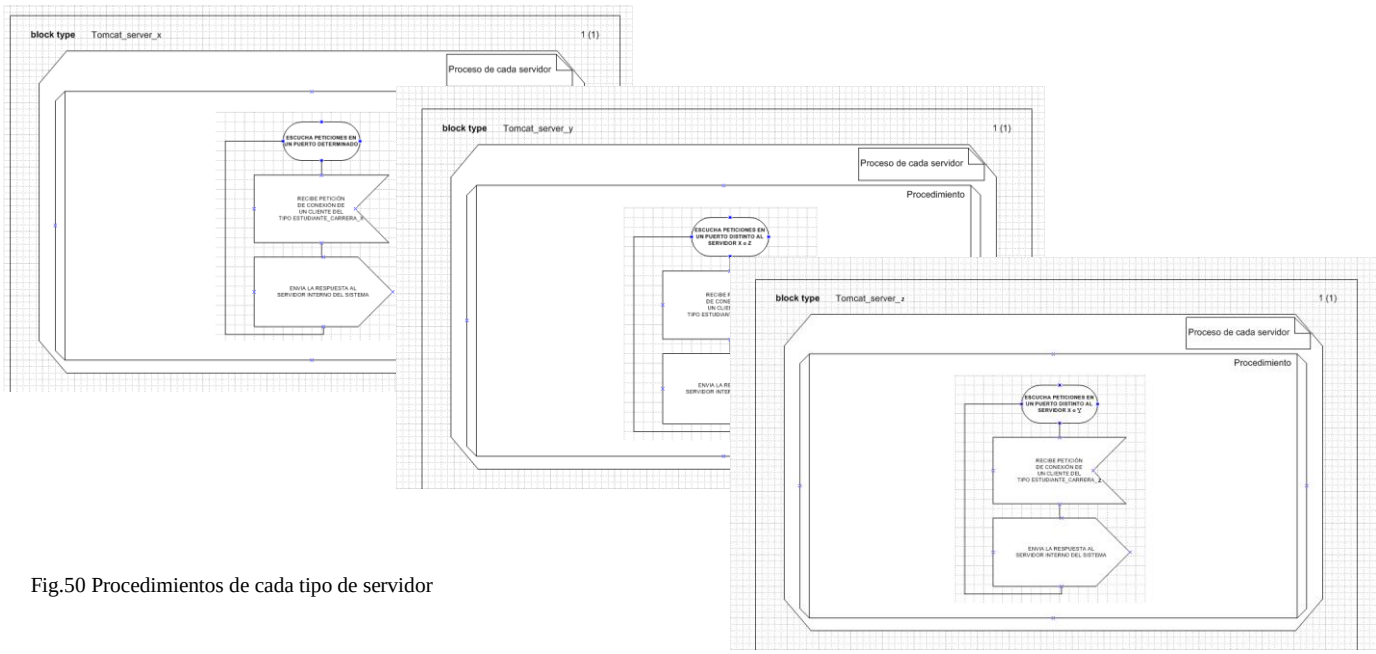


Fig.50 Procedimientos de cada tipo de servidor

Y por último los procesos del servidor interno

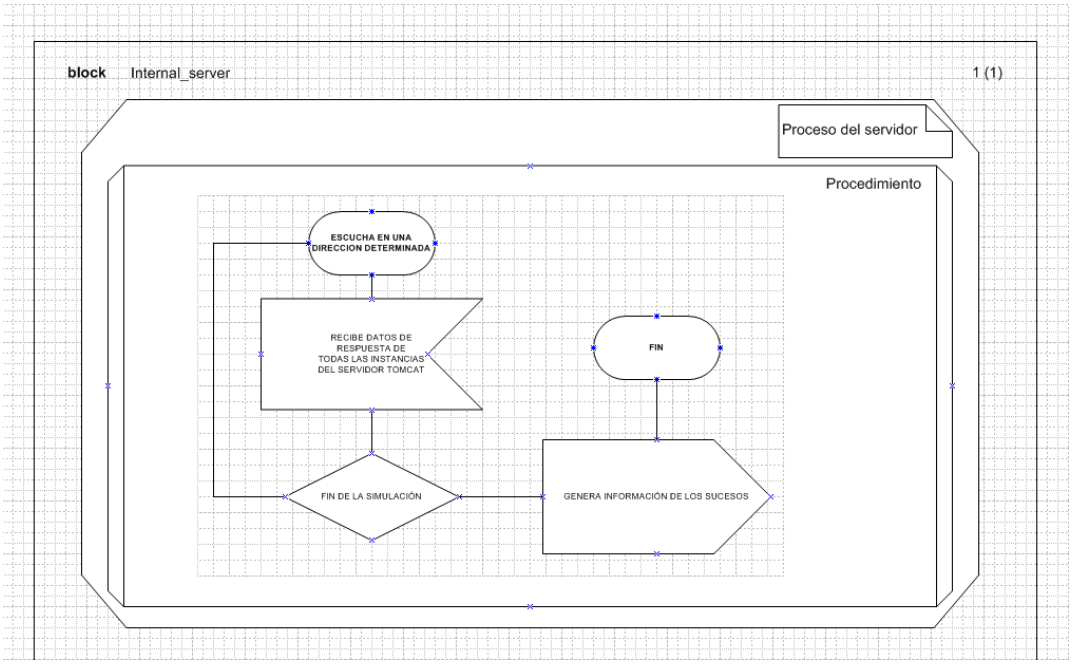


Fig.51 Procesos del servidor que recoge los resultados

Simulación avanzada

Introducción

En esta etapa nos centraremos en la simulación de los dos modelos descritos anteriormente con el objetivo de obtener unos resultados de cada sistema y compararlos.

En un primer momento se ha llevado a cabo la modificación de uno de los ejemplos incluidos con Inet Framework. En concreto se trata de la simulación extServer que nos permite conectar físicamente con la parte real, en la que se encuentran el servidor Tomcat y las aplicaciones colaborativas.

Se han encontrado problemas durante la configuración de las ejecuciones de la simulación. Alguno de estos errores han sido solventados y serán documentados, pero al final, como se explicará más adelante se ha optado por una alternativa en la que se realiza otro tipo de simulación parametrizada con valores obtenidos de las pruebas de comportamiento de la parte real.

De todas formas se realiza una exposición de ambas simulaciones.

Simulación contra servidor real

En primer lugar se intenta ejecutar la simulación con un único cliente para probar la conexión con el servidor. Para ello se mantiene el escenario por defecto y se configuran algunos parámetros que incluyen la dirección del servidor real y el puerto de conexión.

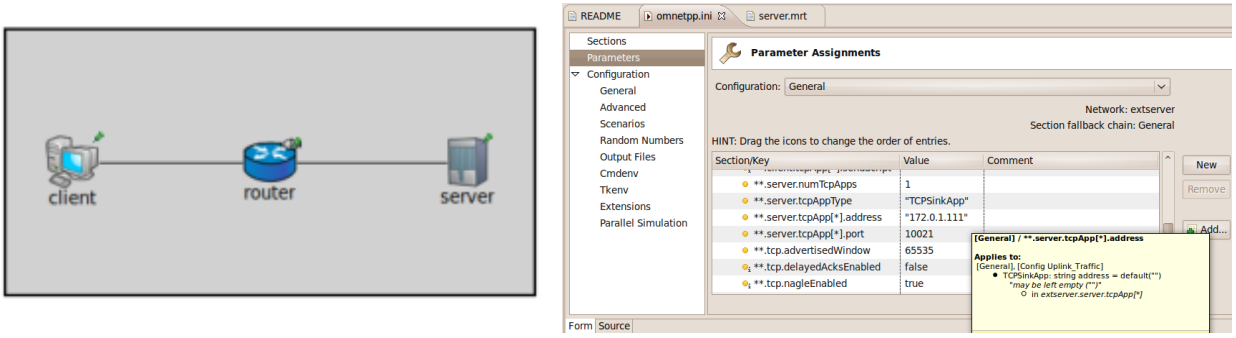


Fig.52 Escenario simple y configuración de la simulación extServer

Una vez construido el código se procede a ejecutar la simulación, encontrando el siguiente error.

Fig.53 Error con librerías pcap.

Se refiere a los dispositivos de captura de tráfico. En primer lugar para solventar el fallo se ha procedido a reinstalar un paquete llamado libpcap-dev que es el encargado de la captura de tráfico. Después se compila el programa por completo, pero no se resuelve el problema. En segundo lugar se piensa en la posibilidad de que como el sistema está instalado en una máquina virtual, no se puede establecer la conexión con la tarjeta de red de forma correcta, por lo que se reinstala todo en una máquina real, pero tampoco se solventa el error. Por último se comprueban las configuraciones y se consigue seguir adelante cambiando el siguiente parámetro.

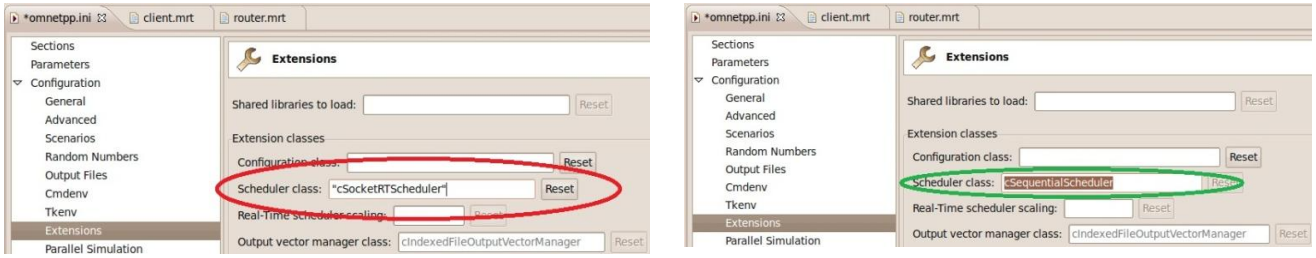


Fig.54 Solución del error de las librerías pcap

Se puede comprobar en la consola, como antes del error llegábamos hasta un punto de la ejecución de la simulación y una vez arreglado conseguimos avanzar.

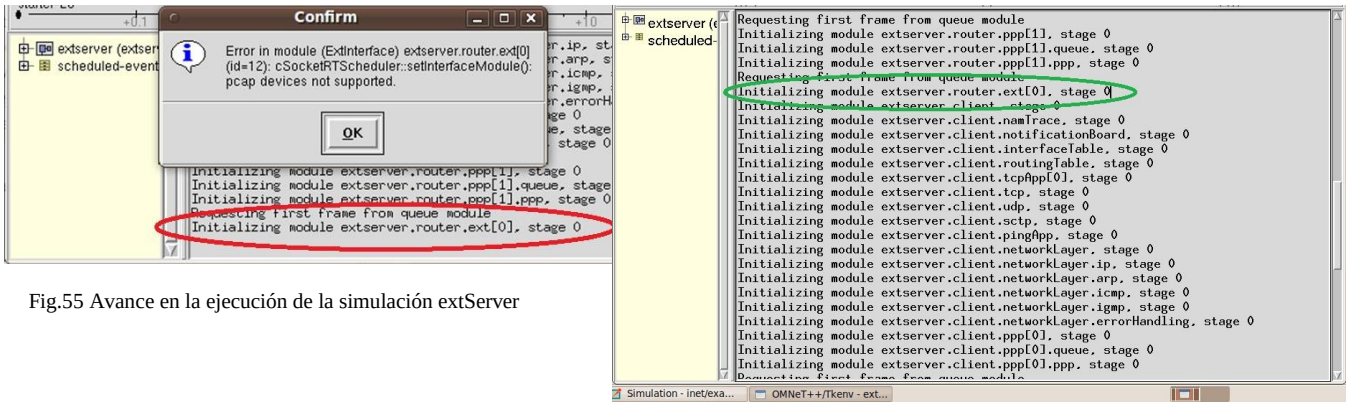


Fig.55 Avance en la ejecución de la simulación extServer

De todas formas llegamos hasta otro punto de la ejecución en la que obtenemos otro error de otro tipo.

Fig.56 Nuevo error con tabla de rutas

Esta vez se trata de un error en la tabla de rutas del router. Es un fichero incluido en la simulación que tiene el siguiente contenido.

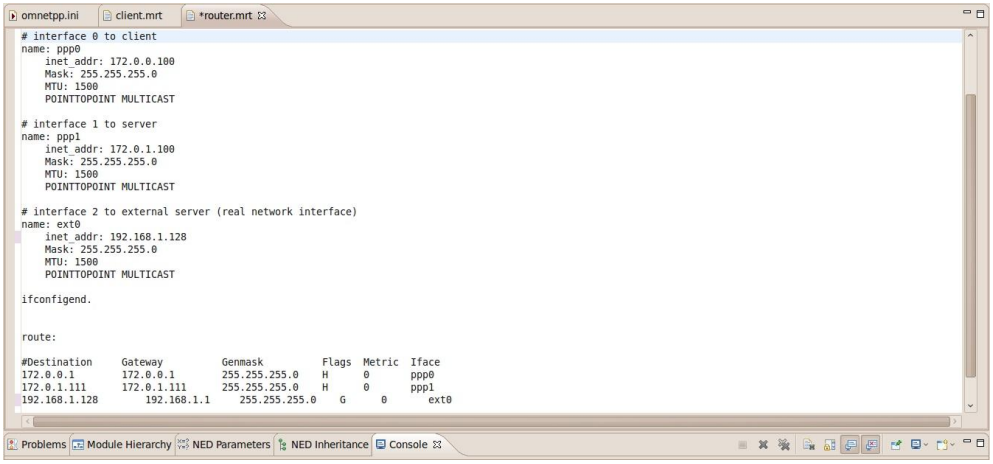


Fig.57 Fichero de configuración de la tabla de rutas del router

También se intenta solucionar cambiando el rango de direcciones ip, conectando la máquina de la simulación y la del servidor a través de un router, también sin directamente sin utilizar router, etc. sin llegar a obtener ningún resultado positivo, por lo que se opta por continuar de momento con la parte real, descubriendo en uno de los momentos, que aunque se consiguiera solventar este problema con las rutas no serviría de nada.

Se llega a esta conclusión ya que las tramas que se enviarían desde la simulación serían interpretadas de forma errónea por el servidor de aplicaciones colaborativas, ya que se prueba la conexión con este desde un navegador obteniendo el siguiente resultado.

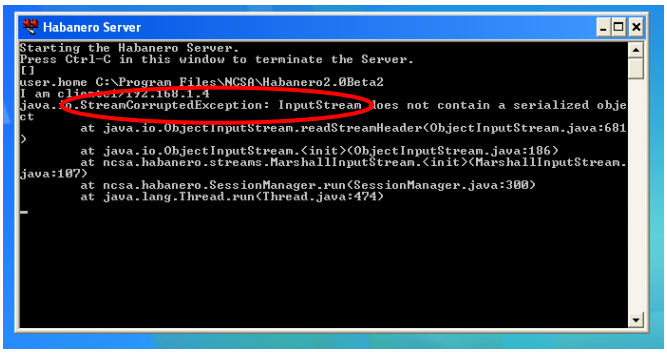


Fig.58 Error en la interpretación de las tramas que se envían desde un navegador

Se tendría que haber solucionado el siguiente esquema de conexión para que la simulación funcionara correctamente.

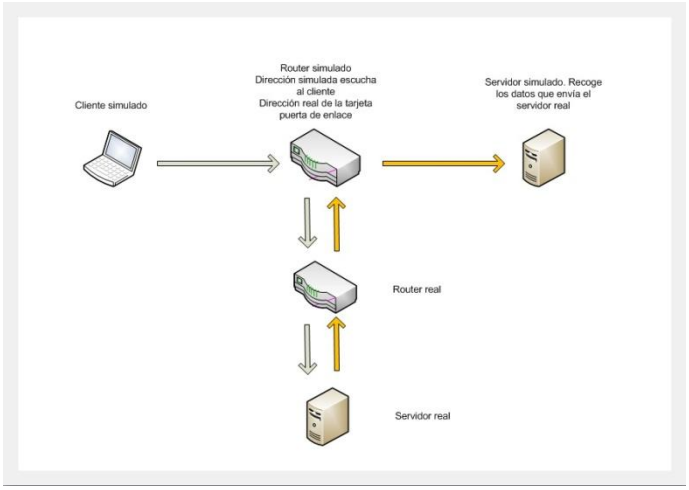


Fig.59 Interconexión entre la simulación y la parte real

Se había configurado incluso los puertos de conexión, que corresponden con los mismos en los que escucha el servidor Tomcat.

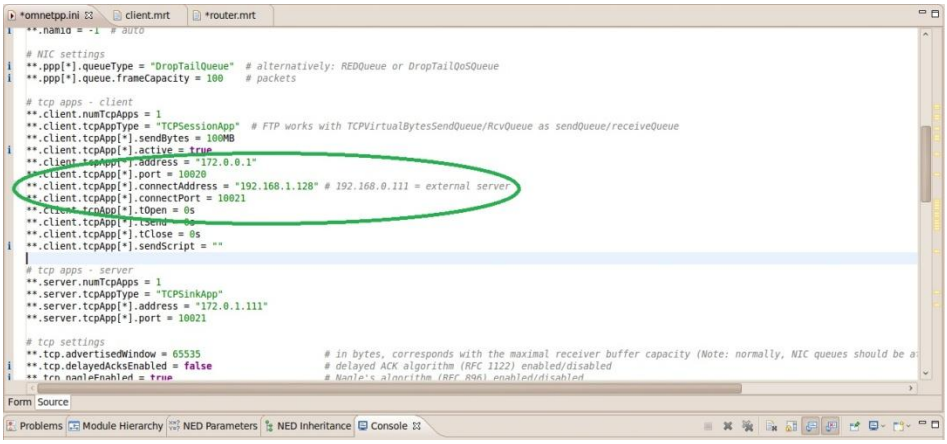


Fig.60 Configuración de los puertos de conexión y dirección del servidor externo

Y también se había contemplado la posibilidad de crear tres interfaces de red para la situación del modelo 2 en la que intentaría conectar con tres servidores. Este parámetro también era configurable.

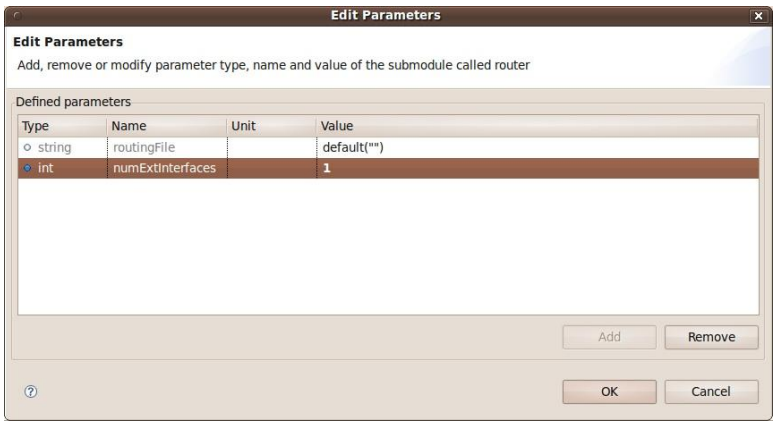


Fig.61 Configuración del número de interfaces del router

Por lo tanto se detallan a continuación los escenarios con los que habríamos trabajado con estas simulaciones para pasar posteriormente a otro tipo de simulación que nos permita alcanzar los objetivos esperados.

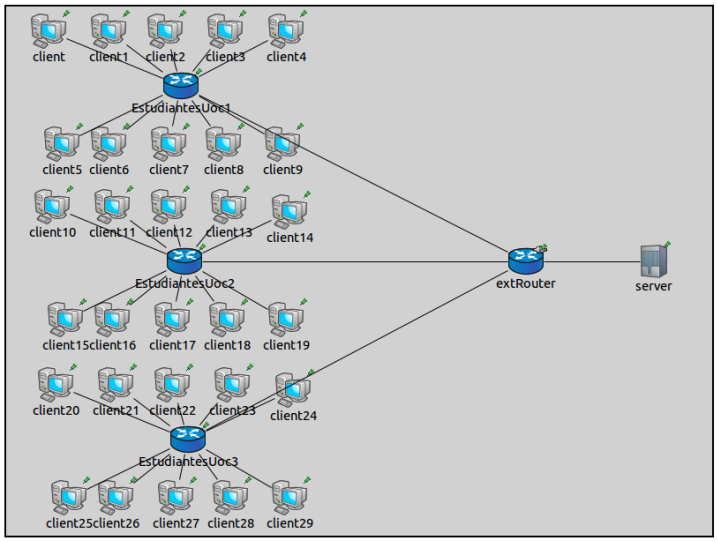


Fig.62 Escenario del modelo 1 (Servidor único)

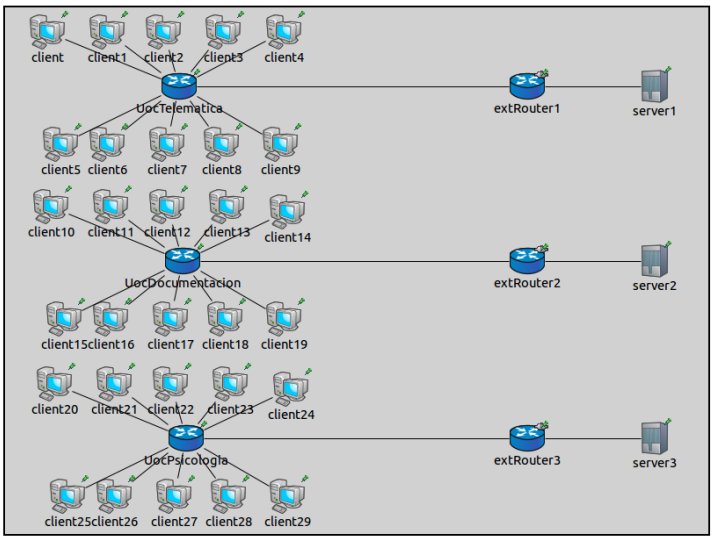


Fig.63 Escenario del modelo 2 (Múltiples instancias del servidor)

Simulación contra servidor parametrizado

Para realizar una simulación que proporcione unos resultados aproximados de la respuesta del servidor sin llegar a conectar con él, es necesario conocer algunos datos sobre el comportamiento de éste. Se han realizado varias pruebas con las aplicaciones colaborativas que se detallarán más adelante, pero necesitamos conocer al menos la cantidad de datos que tenemos que manejar en la simulación. Los archivos que se han utilizado y su tamaño se pueden observar a continuación.

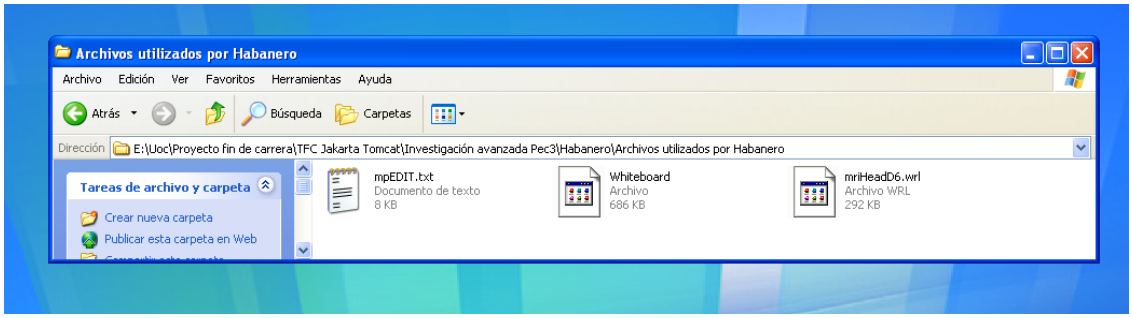


Fig.64 Tamaño de los archivos a manejar por las aplicaciones

Se aprecia un archivo de 8 KB, otro de 686 KB y un tercero de 292 KB. Durante las siguientes simulaciones también se han tenido en cuenta los dos modelos, pero con ciertas limitaciones. Tenemos una cantidad límite de clientes a partir de la cual la simulación no se ejecuta correctamente. Durante las pruebas sobre la parte real se han hecho conexiones con 30 clientes en el modelo 1 y también 30 clientes en el modelo 2, en este caso 10 clientes por servidor. Debido a la limitación de omnetpp, se realizará la simulación con 15 clientes para el modelo 1 y 5 clientes por servidor en el modelo 2, manteniendo de esa manera la misma proporción que en el caso real. Para el modelo 1 se ha hecho la media del tamaño de los archivos obteniendo un tamaño de archivo de 330 KB. Se procede a configurar las simulaciones con estos datos.

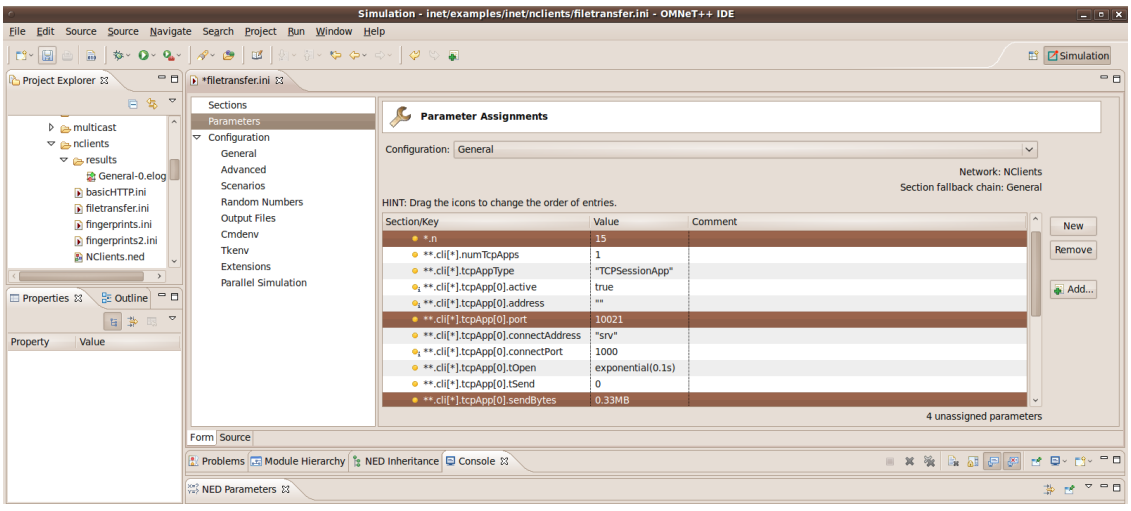


Fig.65 Configuración del archivo de inicialización para el modelo 1

A continuación se realizan las simulaciones, esta vez de forma correcta. Cabe comentar que esta simulación utiliza el protocolo ftp para transferir los archivos comentados anteriormente entre los clientes virtuales y el servidor virtual.

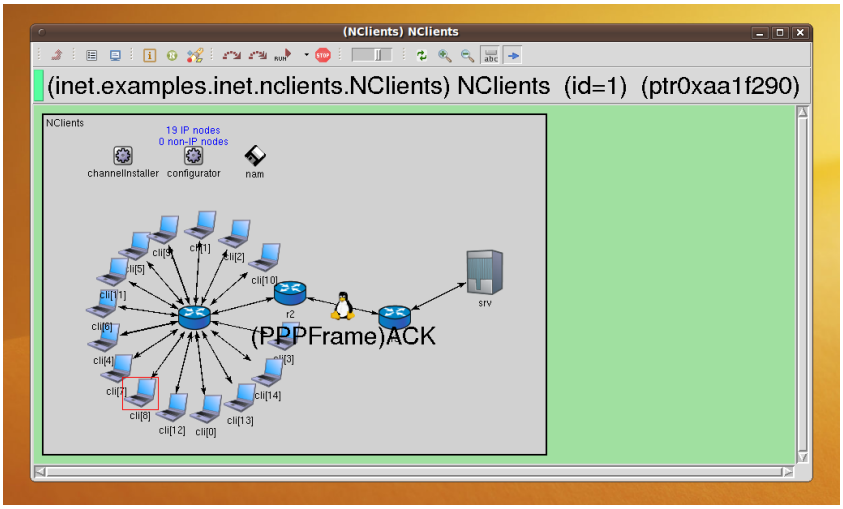


Fig.66 Escenario ejecutándose del modelo 1

Podemos observar también como se graban todos los eventos de la ejecución.

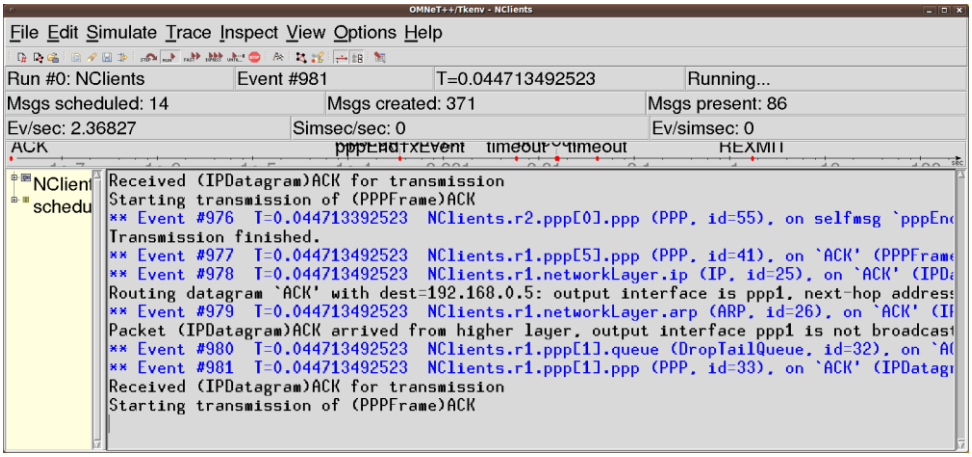


Fig.67 Registro de los eventos de la simulación

Así como los escenarios cambian en cada modelo a simular, la anterior pantalla en la que se registran los eventos es bastante parecida en todos los casos. Más adelante, en el análisis de los resultados obtenidos veremos cómo se tienen en cuenta estos eventos. A todas las simulaciones se les ha puesto un límite de 10 segundos. Es importante mantener el mismo criterio para que al analizar los resultados tengamos respuestas coherentes.

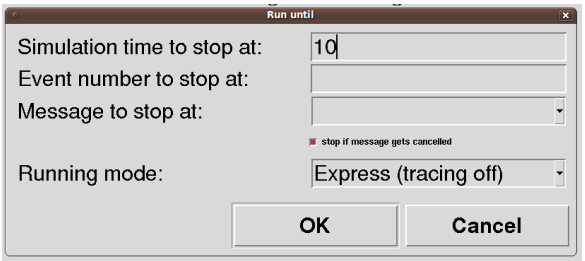


Fig.68 Límite de tiempo de las simulaciones

Se procede por lo tanto con las simulaciones del modelo 2. Se realizará una simulación por servidor, ya que en el caso real el comportamiento de cada servidor no influye en los otros.

Se muestra únicamente el escenario de uno de los servidores ya que en los tres casos la topología de red es la misma.

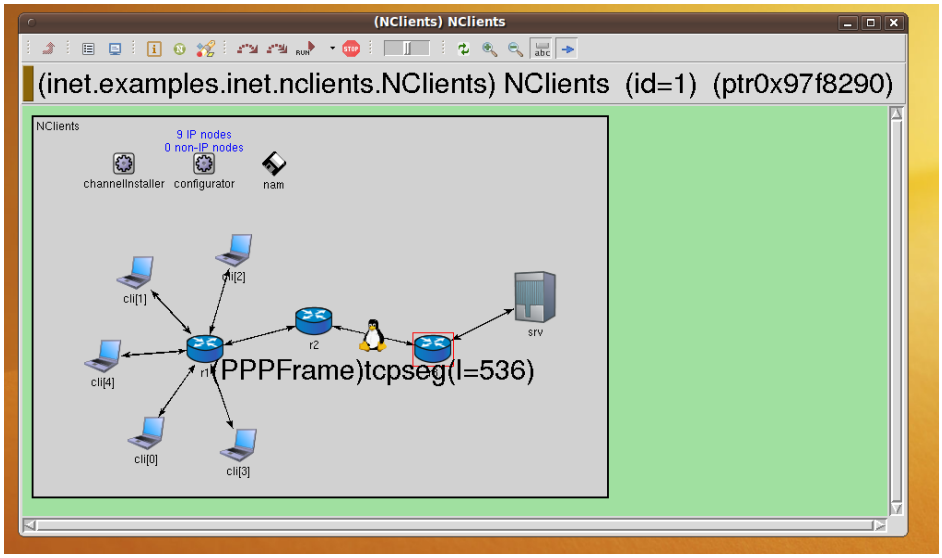


Fig.69 Escenario ejecutándose del modelo 2

Lo que si se tiene en cuenta es que la ejecución de la simulación de cada servidor tendrá el archivo de inicialización configurado de distinta forma.

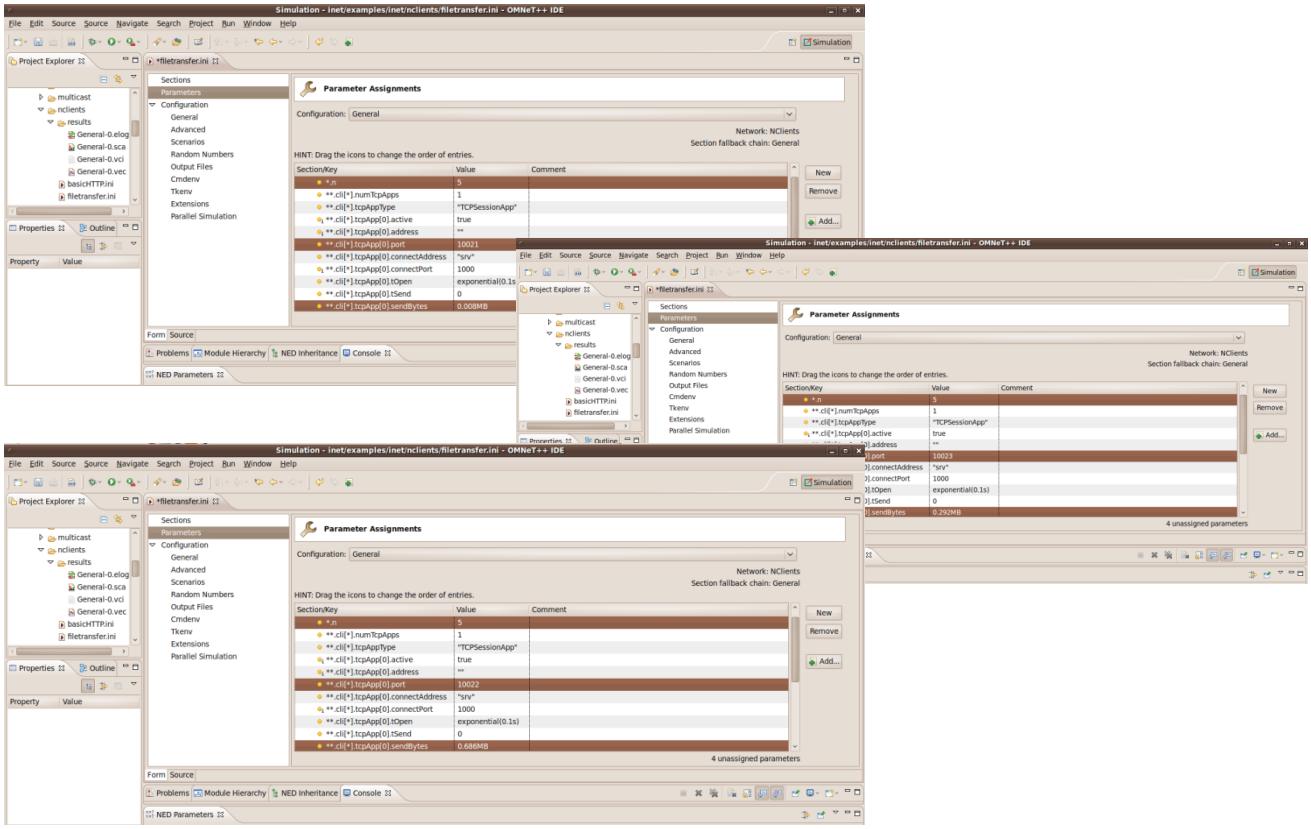


Fig.70 Configuraciones para las simulaciones del modelo 2

Los valores como número de clientes, puerto de conexión y tamaño del archivo a trasferir se aprecian resaltados en las capturas de pantalla anteriores.

Por último para concluir con ésta parte de simulaciones, cabe comentar, que cada vez que finalizamos una simulación debe aparecer la siguiente pantalla, de esa manera nos aseguramos que el programa llama a la finalización de todos los módulos y los archivos de output, o resultados, quedan registrados de forma correcta.

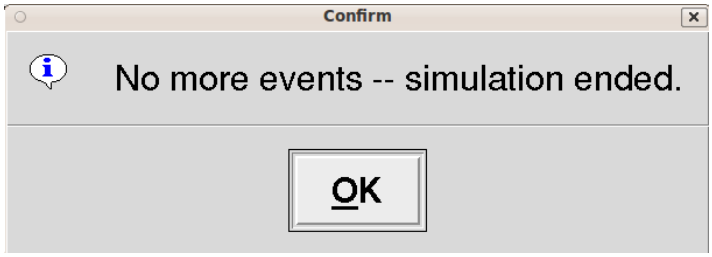


Fig.71 Fin de las simulaciones

Servidor Tomcat y servlets

Introducción

Este apartado está dedicado a configurar el servidor Tomcat para que se ejecute de forma múltiple. Cada instancia del servidor escuchará en un puerto diferente y cada una de estas contendrá una de las aplicaciones colaborativas que atenderá al tipo de estudiante que le corresponda.

Es importante comentar, que aunque se dan las pautas iniciales para instalar los clientes de Habanero, es decir, los clientes de las aplicaciones colaborativas a modo de servlets en sus contenedores correspondientes, no se ha dotado al sistema de ésta funcionalidad por varios motivos. En primer lugar, estamos ejecutando el servidor Tomcat en una máquina con Debian 5, en la que nos es imposible instalar la versión de java development kit jdk1.1.5. Por otro lado, deberíamos adaptar el código de Habanero para que funcionara correctamente junto con Tomcat y además todas las aplicaciones están dentro de una estructura de directorios muy compleja.

Debido al tiempo disponible y a la complejidad de la tarea dejaremos ese reto para otra ocasión más propicia y únicamente se dan unas pinceladas sobre los servlets en Tomcat.

Múltiples instancias de Tomcat

Existen dos maneras de crear múltiples servidores Tomcat en el mismo servidor. Una de ellas es copiar de la estructura de directorios del servidor las partes necesarias en otros directorios, configurar las variables globales para que apunten a éstos directorios y ejecutar una vez el servidor. La otra forma, más sencilla, es copiar toda la estructura de directorios del servidor tantas veces como necesitemos y arrancar múltiples servidores, cada uno con la configuración deseada. Ésta segunda manera de trabajar es la que se ha elegido y se detalla a continuación.

En primer lugar cabe comentar que ya tenemos uno de los servidores configurado, por lo tanto procedemos a copiar la estructura de directorios otras dos veces. En nuestro caso se van a tener tres instancias del servidor Tomcat.

Fig.72 Estructura de carpetas para múltiples instancias del servidor Tomcat

Después se configura cada servidor adicional para que escuche en otro puerto diferente.

server.xml

```
<Service name="Catalina">

  <!--The connectors can use a shared executor, you can define one or more named thread pools-->
  <!--
  <Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
      maxThreads="150" minSpareThreads="4"/>
  -->


  <!-- A "Connector" represents an endpoint by which requests are received
       and responses are returned. Documentation at :
       Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
       Java AJP  Connector: /docs/config/ajp.html
       APR (HTTP/AJP) Connector: /docs/apr.html
       Define a non-SSL HTTP/1.1 Connector on port 8080
  -->
  <Connector port="10022" protocol="HTTP/1.1"
             connectionTimeout="20000"
             redirectPort="8444" />
  <!-- A "Connector" using the shared thread pool-->
  <!--
```

*server.xml

```
<!-- A "Service" is a collection of one or more "Connectors" that share
     a single "Container" Note:  A "Service" is not itself a "Container",
     so you may not define subcomponents such as "Valves" at this level.
     Documentation at /docs/config/service.html
 -->
<Service name="Catalina">

  <!--The connectors can use a shared executor, you can define one or more named thread pools-->
  <!--
  <Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
      maxThreads="150" minSpareThreads="4"/>
  -->


  <!-- A "Connector" represents an endpoint by which requests are received
       and responses are returned. Documentation at :
       Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
       Java AJP  Connector: /docs/config/ajp.html
       APR (HTTP/AJP) Connector: /docs/apr.html
       Define a non-SSL HTTP/1.1 Connector on port 8080
  -->
  <Connector port="10023" protocol="HTTP/1.1"
             connectionTimeout="20000"
             redirectPort="8445" />
  <!-- A "Connector" using the shared thread pool-->
  <!--
```

Fig.73 Configuración de los puertos de cada servidor

Se puede observar que debemos cambiar también el parámetro redirectPort, para que cada servidor redirija a su propio puerto y no existan conflictos entre servidores. Al principio de este fichero de configuración también existe un parámetro llamado serverPort, con el valor 8000, que también debemos cambiar para que cada instancia tenga el suyo propio.

Una vez configurados los puertos tendremos que arrancar cada servidor de la manera que hemos explicado al principio, cada instancia se arranca de forma independiente.

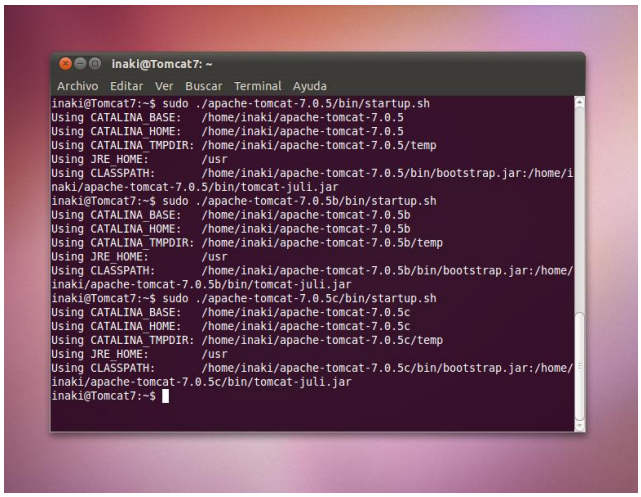


Fig.74 Arranque de múltiples servidores

Por último se puede observar el resultado con los tres servidores funcionando simultáneamente.

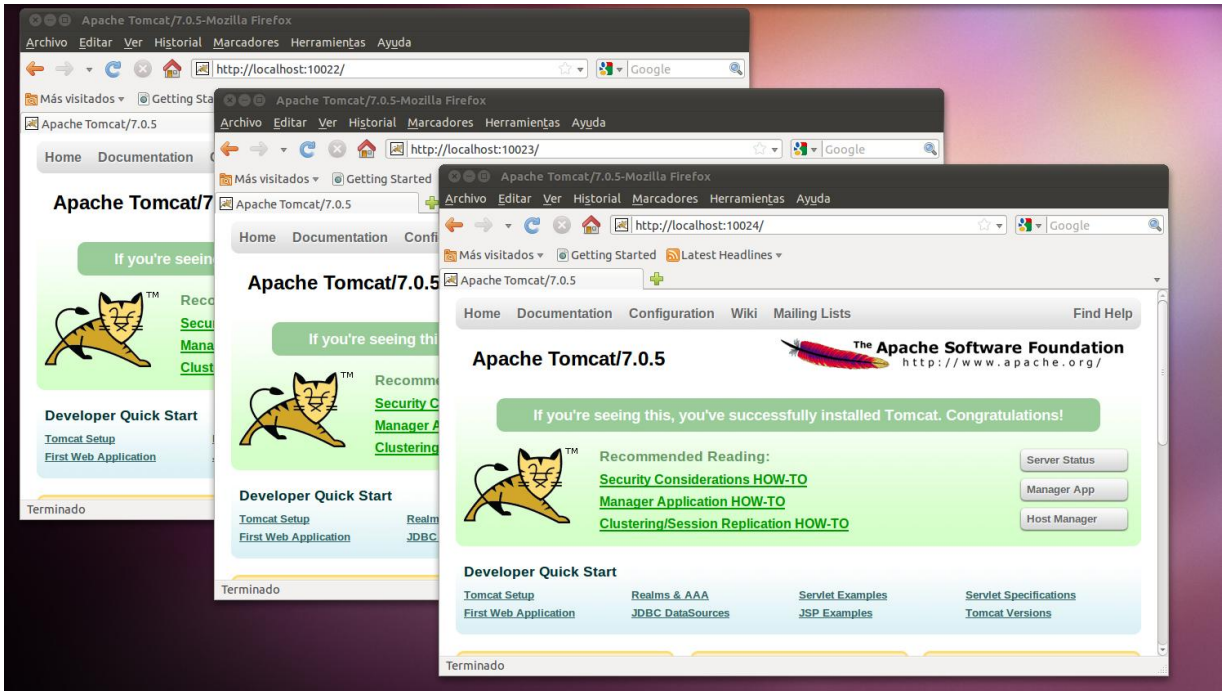


Fig.75 Varios servidores funcionando simultáneamente

Se va a dar una breve explicación de cómo se incluye un programa en Java a modo de servlet dentro de un contenedor del servidor Tomcat.

En primer lugar, para cada programa o servlet tendremos la siguiente estructura de directorios dentro de la carpeta apache-tomcat2.0.5. Tendremos que crear una carpeta con el nombre del servlet, que en este caso se denomina Habanero, dentro de ésta otra de nombre WEB-INF, y dentro de ésta las siguientes carpetas:

- clases -> Contendrá los ficheros .class con el código de los programas ya compilados.
- lib -> Contendrá las librerías necesarias para el correcto funcionamiento de estos programas.
- src -> Contendrá ficheros .java con el código sin compilar de los programas, aunque no son necesarios.

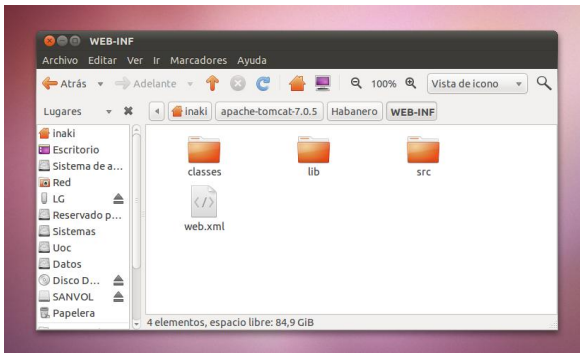


Fig.76 Estructura de directorios del contenedor de servlets

Por último es necesario disponer del fichero web.xml, que será el que se ejecute cuando un cliente se conecta al servidor y lanza el servlet. En su contenido deben aparecer al menos las siguientes líneas.

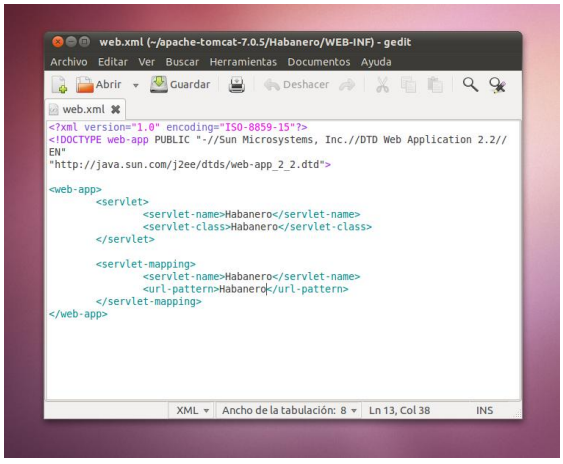


Fig.77 Archivo que ejecutarán los clientes cuando conecten al servlet

Como hemos comentado anteriormente, para este caso particular tendríamos un fichero web.xml mucho más complejo y deberíamos incluso modificar el propio código de las aplicaciones.

Habanero. Aplicaciones Colaborativas

En este apartado se va a describir el proyecto Habanero. Es un entorno colaborativo que contiene un set de aplicaciones y herramientas desarrollado en la Universidad de Illinois. Como no podía ser de otra manera también se han dado problemas a la hora de ejecutarlo. Para poder ejecutar la suite es necesario tener instalado el java development Kit, jdk en su versión 1.1.5. Se trata de una versión antigua y ese es el motivo por el que se han dado errores. En principio se ha intentado instalar en Debian, junto a Tomcat, pero esa versión de jdk solo está disponible para Windows y solaris. Por lo tanto después del fallido intento de instalación en Debian, se procede a instalar Habanero en Windows 7, con el mismo resultado. Al ser una versión antigua no es posible ejecutarlo en un sistema operativo de 64 bits. Al final se consigue hacer funcionar el entorno en un sistema operativo Windows xp con sp2.

Comenzaremos haciendo una breve descripción de cada una de las aplicaciones que forman parte de Habanero y después elegiremos tres aplicaciones que serían las más adecuadas para los tres supuestos tipos de estudiantes con los que se harán las pruebas.

- **Savina** -> un navegador web colaborativo.
- **Telnet** -> una versión colaborativa de Telnet.
- **The Lattice** -> una herramienta para ver el formato .pdb.
- **mpEdit** -> un editor multiplataforma colaborativo.
- **3DXYZ** -> una herramienta para visualizar el formato .xyz.
- **Whiteboard** -> utilizado para leer .gif, .jpg y .ppm incluye un modo para presentaciones Power Point, además de herramientas de dibujo para remarcar zonas de las imágenes y permite guardarlas.
- **Chat** -> un entorno de chat basado en texto con capacidad de logging.
- **Voting Tool** -> permite crear, ordenar y mostrar datos de votaciones.
- **Tic Tac Toe** -> un código de ejemplo sobre un juego.
- **Checkers** -> otro código de ejemplo del juego de las damas.
- **VisibleHuman** -> permite visualizar imágenes de rayos x.
- **BigCalculator** -> una calculadora colaborativa.
- **cVRML** -> permite visualizar archivos .vrm de realidad virtual.
- **CollabCristal** -> permite visualizar archivos tridimensionales.

Las tres aplicaciones seleccionadas son Whiteboard para alumnos de la carrera de Ing. tec. Telecomunicación, que nos permitirá visualizar imágenes de redes. Por otro lado utilizaremos mpEdit, que nos permite abrir textos y editarlos desde los distintos clientes, podría ser útil para los alumnos del grado de documentación. Y por último la aplicación cVRML nos permite abrir archivos de realidad virtual, se han buscado una imagen de un cerebro y podría ser útil para alumnos de la disciplina de psicología. Podemos observar un detalle de cada una de estas aplicaciones.

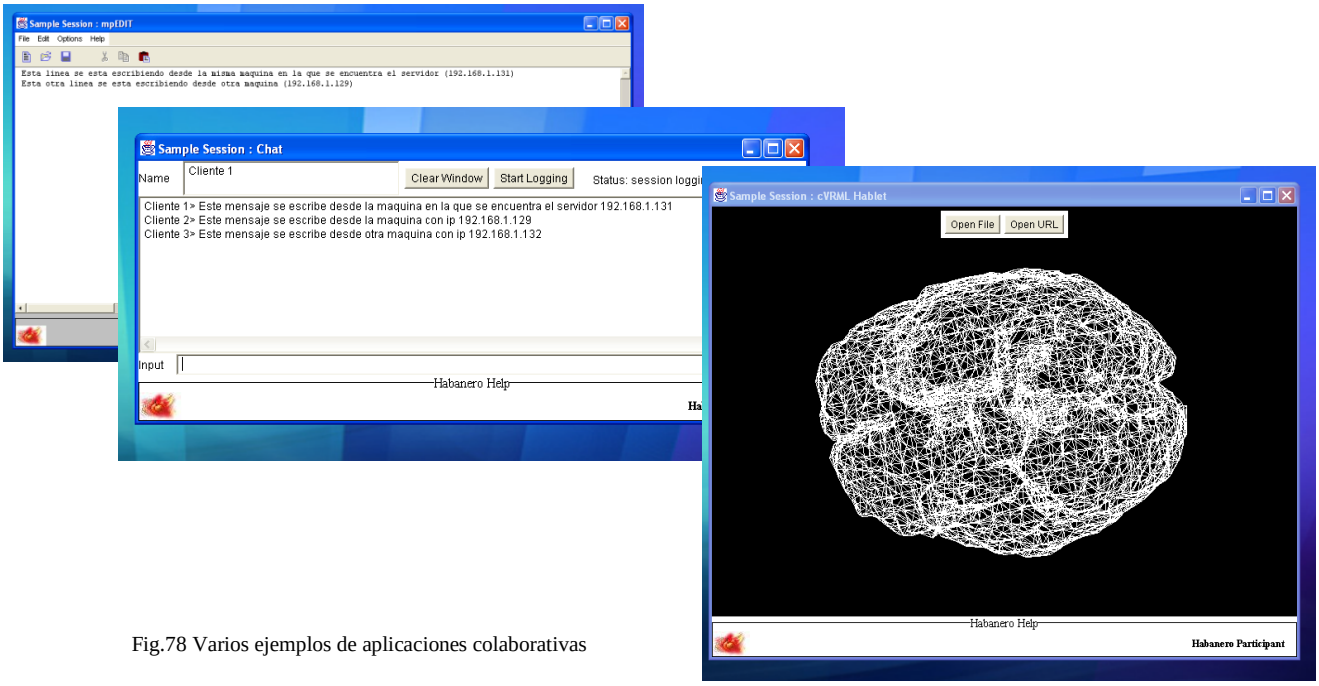


Fig.78 Varios ejemplos de aplicaciones colaborativas

En la siguiente captura se observa también el cliente de Habanero en el que configuramos los parámetros de la sesión. En el primer modelo descrito en sdl teníamos un único servidor y todos los clientes se conectarán a éste.

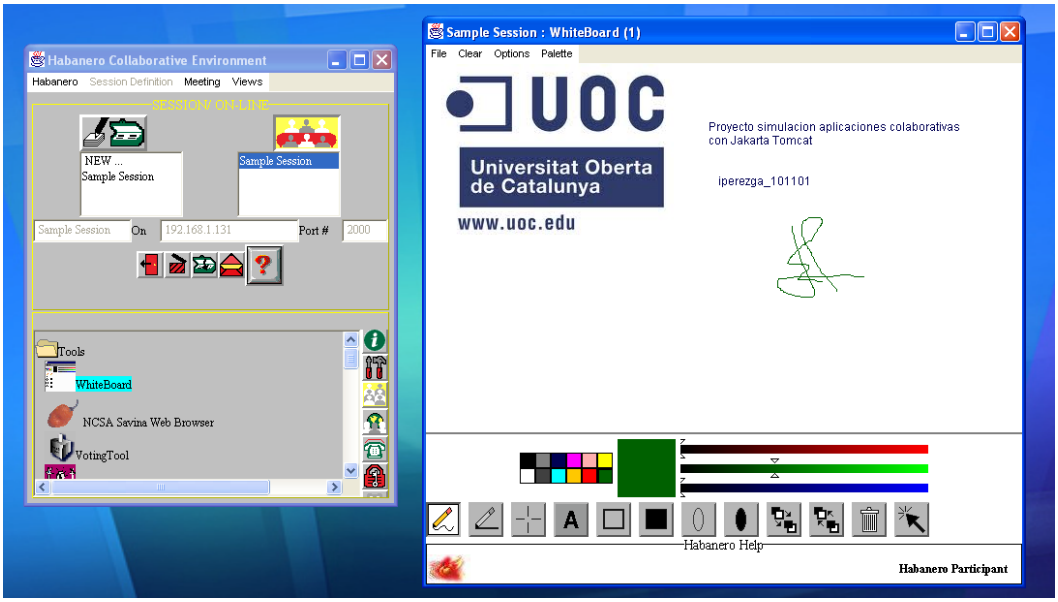


Fig.79 Ejecución del cliente de Habanero junto a WhiteBoard. Modelo 1 (Servidor único)

Es posible configurar el puerto entrante a la hora de ejecutar el script que lanzará el cliente de Habanero.

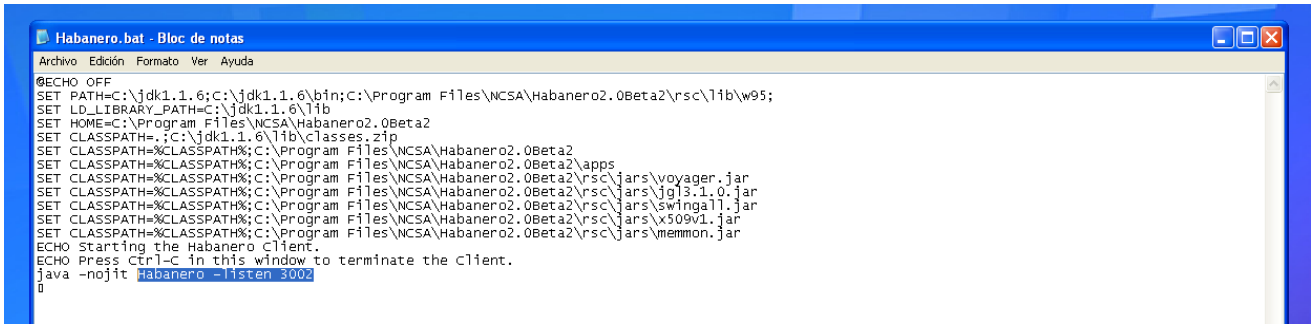


Fig.80 Configuración de puertos en el script de inicio del cliente

Para crear un entorno como el descrito en el modelo 2 de sdl tendremos que arrancar tres servidores. Después cada grupo de clientes se conectará a uno de estos servidores configurando un nombre para la sesión, la dirección ip en la que se encuentra en servidor y el puerto de conexión.

Fig.81 Varios clientes conectados a varios servidores. Modelo 2

En la imagen anterior se observa una de las máquinas en la que se estaban ejecutando tres clientes y un servidor. Para la realización de las pruebas de comportamiento se pidió ayuda a otras dos personas para que crearan actividad en los clientes de las otras dos máquinas. Estas máquinas estaban conectadas a través de un router. En una primera prueba del modelo 1 se ejecutaron 10 clientes en cada máquina que se conectaban a un único servidor, es decir 30 clientes. En la segunda prueba se ejecutaban también 10 clientes, pero esta vez teníamos 3 servidores, para los supuestos grupos de alumnos de telemática, documentación y psicología. En el siguiente apartado se detallan estas pruebas, así como las pruebas de las simulaciones y los resultados.

Resultados de las pruebas

En primer lugar se describirán las pruebas de comportamiento realizadas sobre los servidores de aplicaciones colaborativas. Uno de los datos más relevantes es que cuando se ejecutan más de 30 clientes contra un único servidor en el modelo 1, éste deja de responder y todo el sistema se bloquea. Se han realizado capturas de tráfico con WireShark durante 1 minuto, a partir de las cuales se puede comprobar la cantidad de paquetes transmitidos.

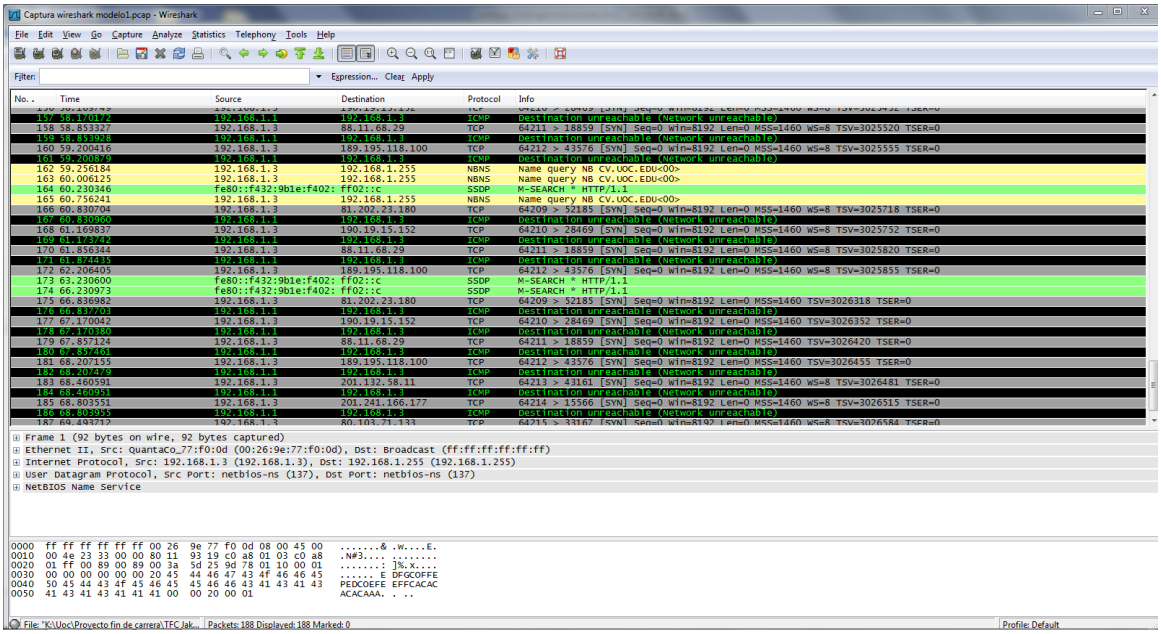


Fig.82 Captura de tráfico con wireshark en las pruebas con el modelo 1

Se pueden observar un total de 188 paquetes transmitidos y recibidos. También se ha realizado otro tipo de prueba con el software MIB Browser de iReasoning, que ha permitido separar el tráfico entrante y el saliente para poder comprobar de esa manera la respuesta del servidor. Se toman muestras cada minuto durante 10 minutos. Se ha utilizado una hoja de cálculo para incluir los datos a partir de los cuales se genera una gráfica.

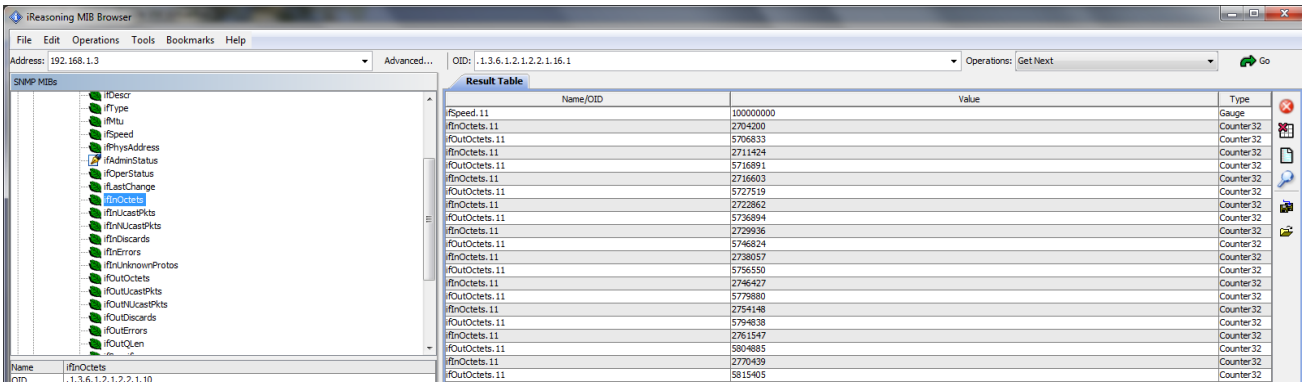


Fig.83 Toma de muestras con MIB Browser modelo 1

Delta tiempo	Tiempo total	Input total	Output total	Delta Input	utilización input %	Delta Output	utilización output %	utilización %
0	0	2704200	5706833	7224		10058		
60	60	2711424	5716891	5179	6,905333333	10628	14,17066667	21,076
59	119	2716603	5727519	6259	8,486779661	9375	12,71186441	21,19864407
60	179	2722862	5736894	7074	9,432	9930	13,24	22,672
61	240	2729936	5746824	8121	10,6504918	9726	12,75540984	23,40590164
59	299	2738057	5756550	8370	11,34915254	23330	31,63389831	42,98305085
62	361	2746427	5779880	7721	9,962580645	14958	19,30064516	29,26322581
65	426	2754148	5794838	7399	9,106461538	10047	12,36553846	21,472
60	486	2761547	5804885	8892	11,856	10520	14,02666667	25,88266667
60	546	2770439	5815405					

Fig.84 Tabla con los datos recogidos con MIB Browser modelo 1

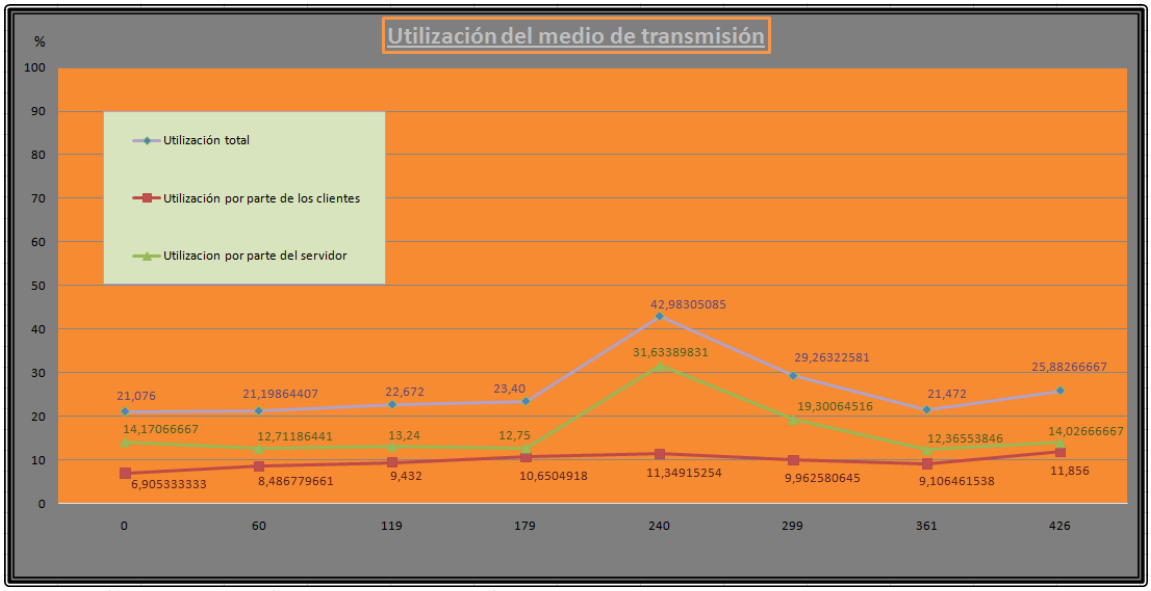


Fig.85 Gráfico de la utilización del medio de transmisión modelo 1

Uno de los datos más relevantes de esta prueba es la utilización del medio de transmisión por parte del servidor, que en ésta ocasión alcanza un valor máximo del 31,6 %.

A continuación se muestran los resultados de las pruebas anteriores obtenidos para el modelo 2.

En primer lugar obtenemos un total de 235 paquetes capturados con Wireshark durante 1 minuto.

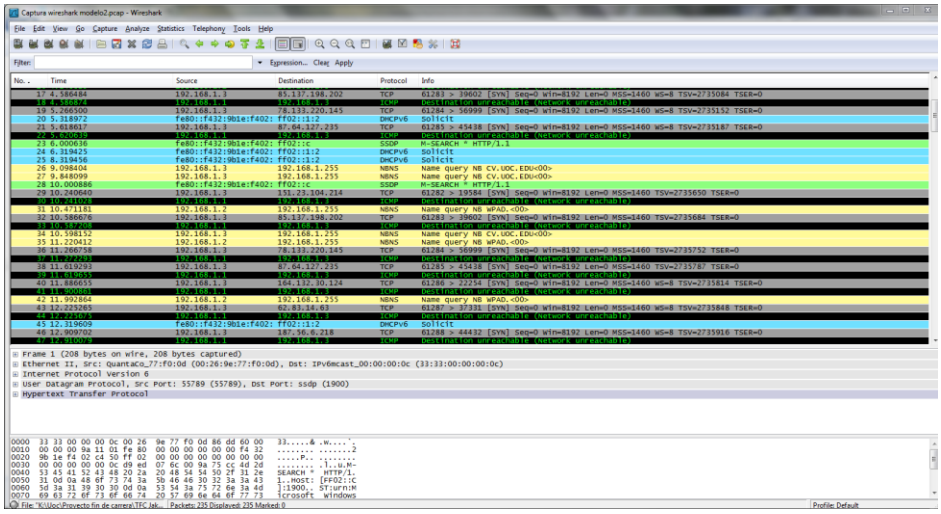


Fig.86 Captura de tráfico con wireshark en las pruebas con el modelo 2

Para terminar con las pruebas del comportamiento de las aplicaciones colaborativas se muestran los resultados obtenidos a partir de MIB Browser para el modelo 2.

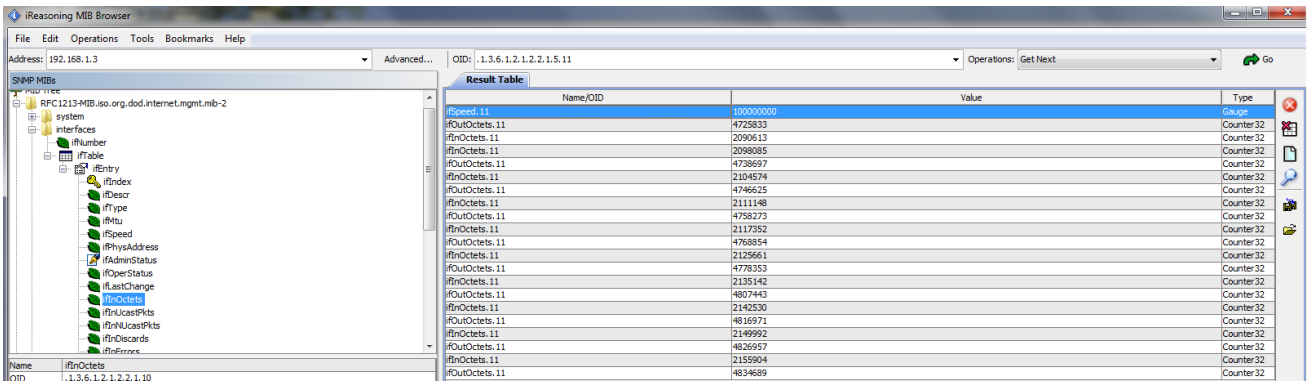


Fig.87 Toma de muestras con MIB Browser modelo 2

Delta tiempo	Tiempo total	Input total	Output total	Delta Input	utilización input %	Delta Output	utilización output %	utilización %
0	0	2090613	4725833	7472		12864		
60	60	2098085	4738697	6489	8,652	7928	10,57066667	19,22266667
59	119	2104574	4746625	6574	8,913898305	11648	15,79389831	24,70779661
60	179	2111148	4758273	6204	8,272	10581	14,108	22,38
61	240	2117352	4768854	8309	10,89704918	9699	12,72	23,61704918
59	299	2125661	4778553	9481	12,85559322	8890	12,05423729	24,90983051
62	361	2135142	4787443	7388	9,532903226	9528	12,29419355	21,82709677
65	426	2142530	4796971	7462	9,184	9986	12,29046154	21,47446154
60	486	2149992	4806957	5912	7,882666667	7732	10,30933333	18,192
60	546	2155904	4814689					

Fig.88 Tabla con los datos recogidos con MIB Browser modelo 2

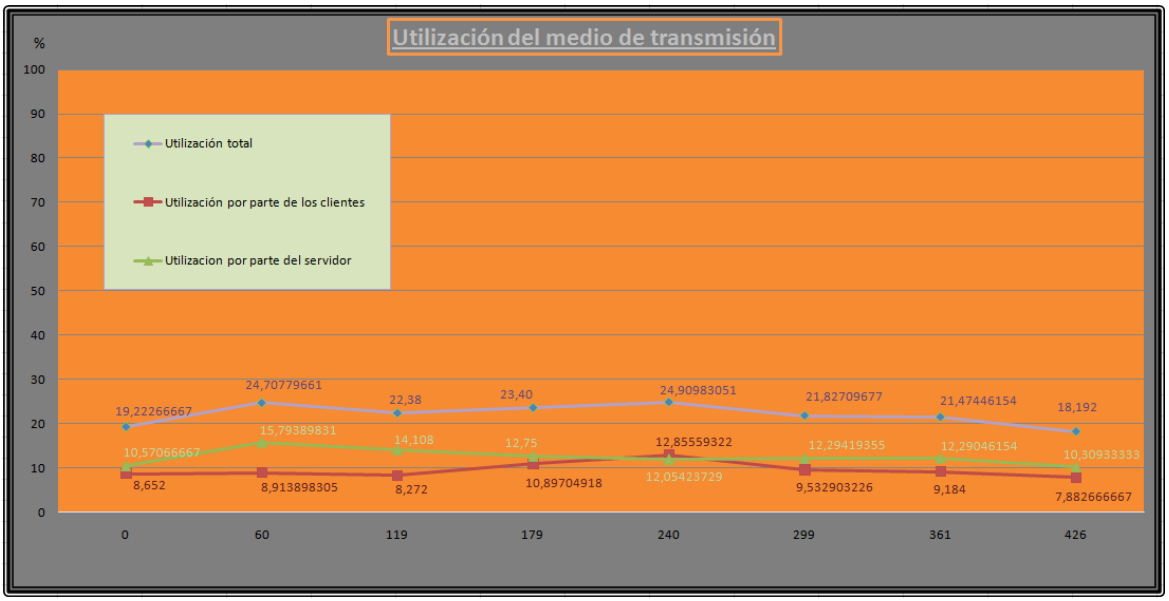


Fig.89 Gráfico de la utilización del medio de transmisión modelo 2

En ésta ocasión se aprecia que el valor máximo de utilización del medio de transmisión por parte de uno de los servidores es del 15,8%.

Se ha terminado con las pruebas sobre las aplicaciones colaborativas, a continuación se mostrarán los resultados obtenidos a partir de las simulaciones de los dos modelos.

Llegados a este punto se continua con las pruebas realizadas sobre las simulaciones. Se da una explicación de la configuración de los gráficos para el modelo 1 que nos servirá también para el modelo 2.

En primer lugar durante la explicación en el apartado de las simulaciones, podíamos observar cómo se iba creando un registro de cada evento durante la ejecución de éstas. Cuando se finalizaba la ejecución se llamaba a la finalización de todos los módulos y de esta manera se crean dos tipos de fichero output o de resultados, los resultados vectoriales y los resultados escalares. Disponemos entonces de tres tipos de ficheros a partir de los cuales generaremos unas gráficas que nos ayudarán a analizar los resultados.

A continuación se muestra un detalle de cómo se configuran estos ficheros para mostrar en los gráficos los datos deseados.

Se crea un filtro para el registro de eventos que nos permite seleccionar el servidor y uno sólo de los clientes, ya que si mostramos los eventos de todos los clientes se genera demasiada información que no podríamos interpretar.

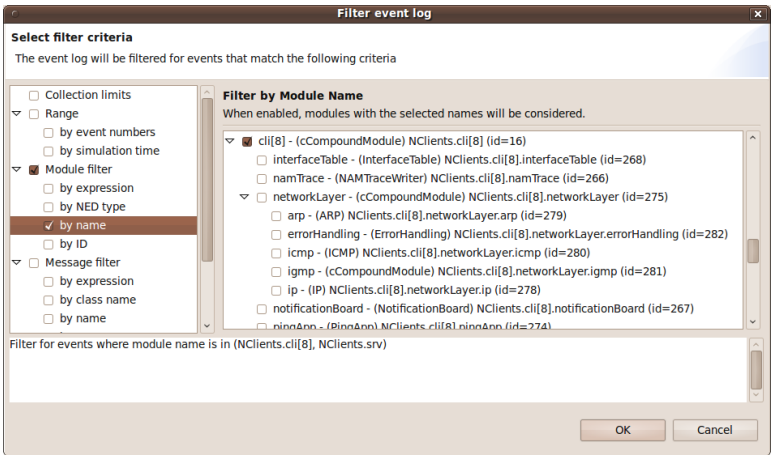


Fig.90 Filtro del log de eventos

Es importante conocer el total de eventos registrados durante la ejecución de la simulación.

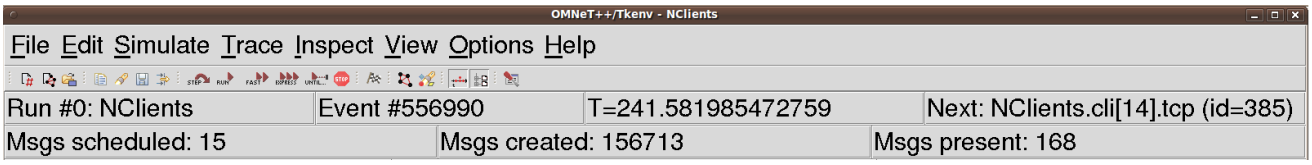


Fig.91 Número de eventos totales modelo 1

Al final obtenemos un Timeline o gráfico en la línea del tiempo en la que se observa la comunicación entre el cliente y el servidor.

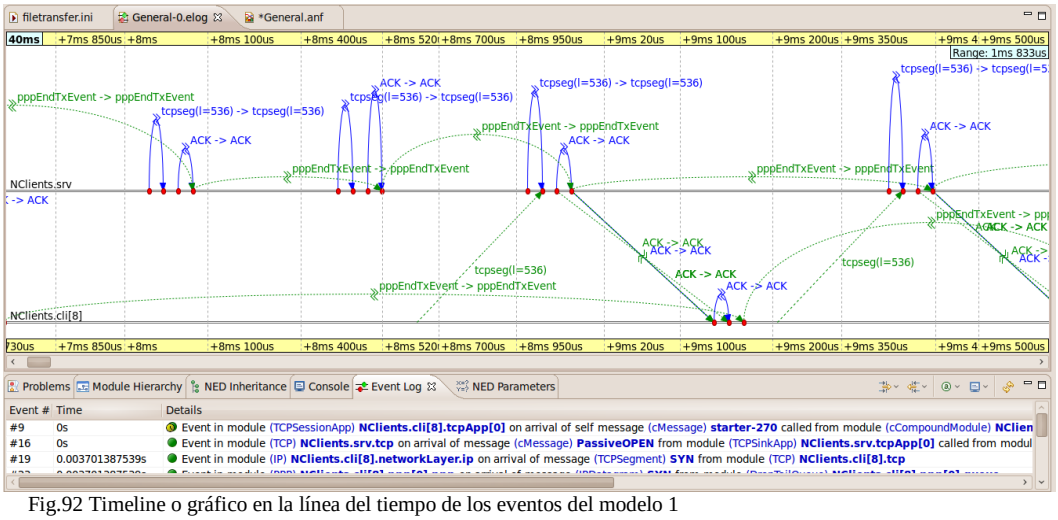


Fig.92 Timeline o gráfico en la línea del tiempo de los eventos del modelo 1

Una vez analizado el registro de eventos procedemos a configurar los gráficos obtenidos a partir de los archivos vectoriales y escalares con ayuda de la herramienta de análisis de resultados.

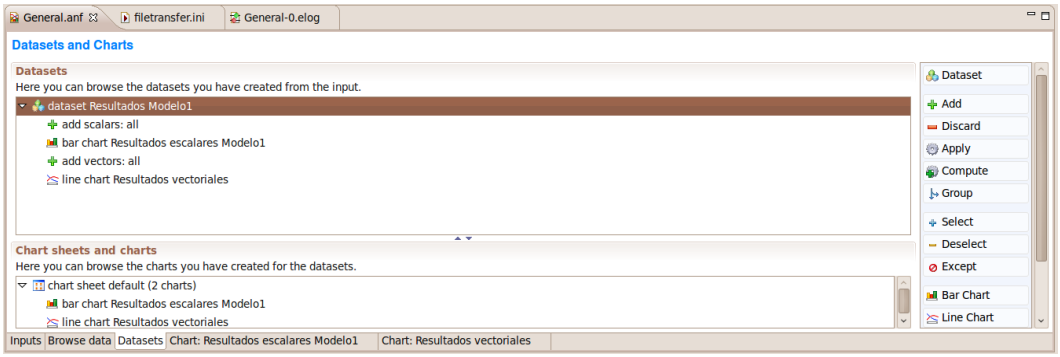


Fig.93 Timeline o gráfico en la línea del tiempo de los eventos del modelo 1

Configuramos también cada gráfico para que se muestre únicamente la información que nos interesa, en nuestro caso es el número de bytes enviados y recibidos por parte de los clientes y el servidor.

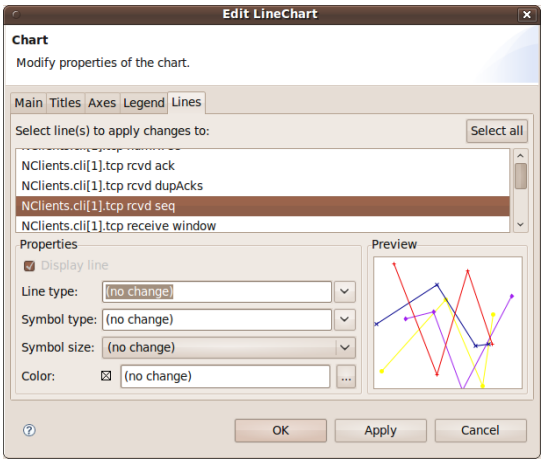


Fig.94 Configuración de los archivos escalar y vectorial para obtener los gráficos

Obtenemos por tanto los gráficos escalar y vectorial. En el caso del modelo 1 podemos observar en el gráfico escalar la cantidad de datos recibidos por el servidor y la densidad de las transmisiones en el gráfico vectorial.

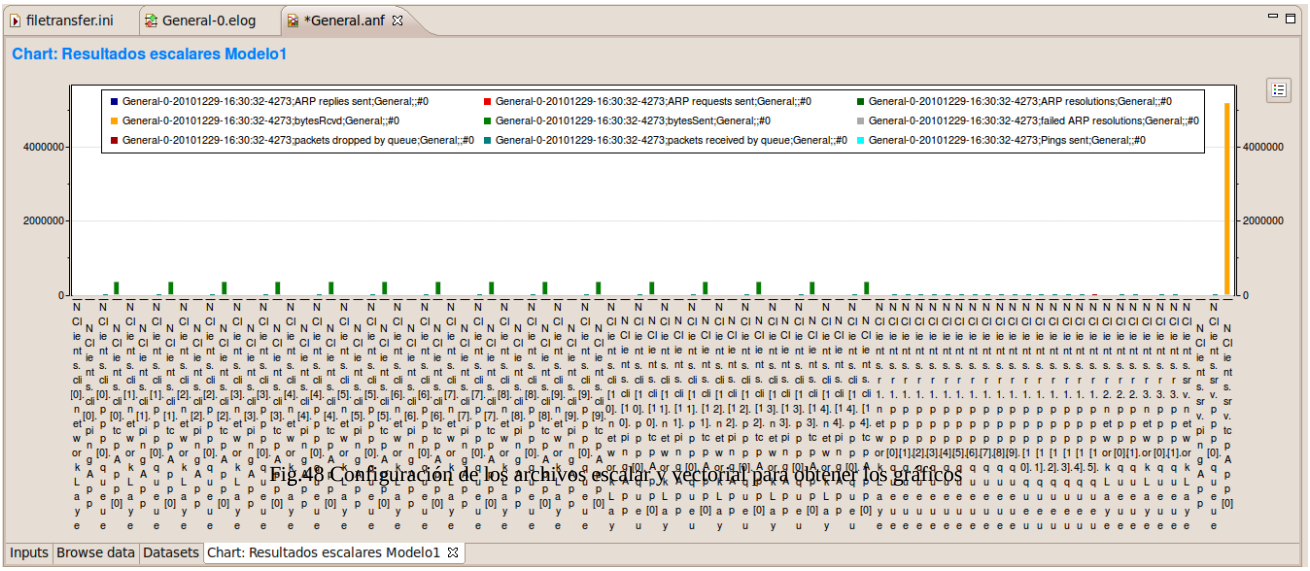


Fig.95 Gráfico escalar del modelo 1

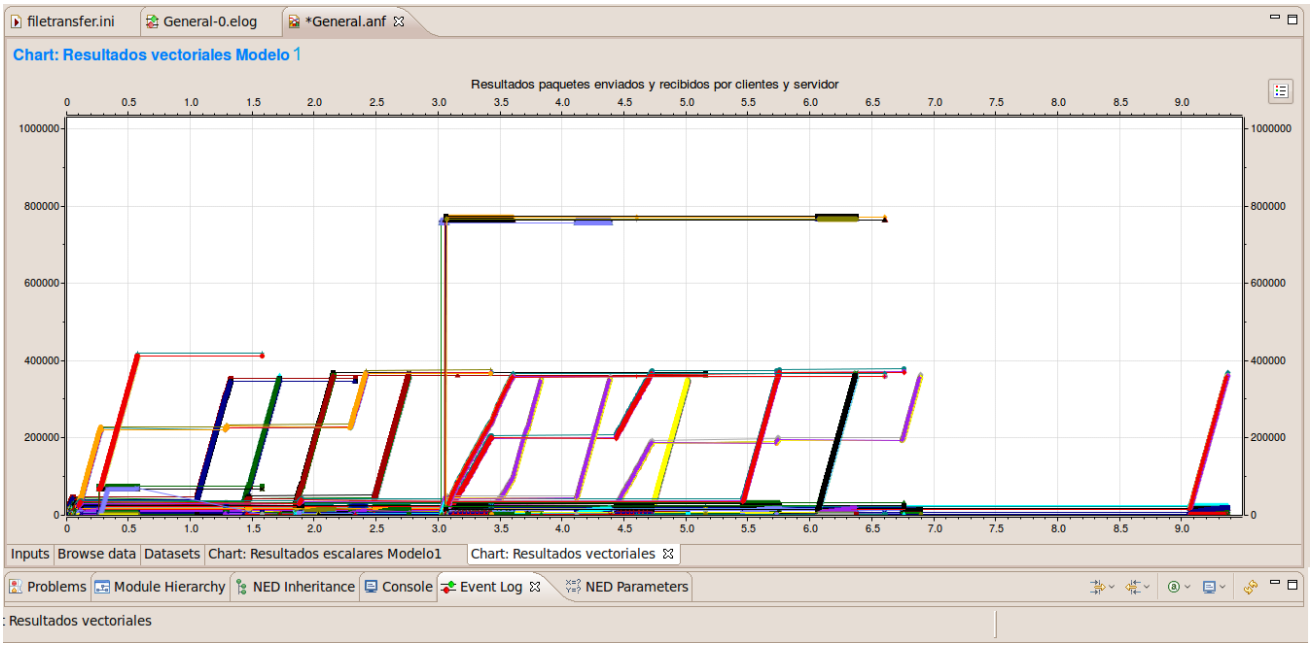


Fig.96 Gráfico vectorial del modelo 1

En las páginas siguientes se muestran los resultados obtenidos para el modelo 2 sin dar ningún tipo de explicación ya que se han dado para el modelo 1. Estos resultados se analizarán en el último apartado de conclusiones.

Modelo 2 Servidor A

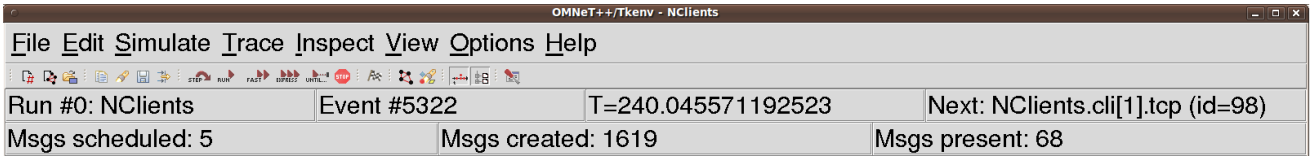


Fig.97 Eventos totales del modelo 2a

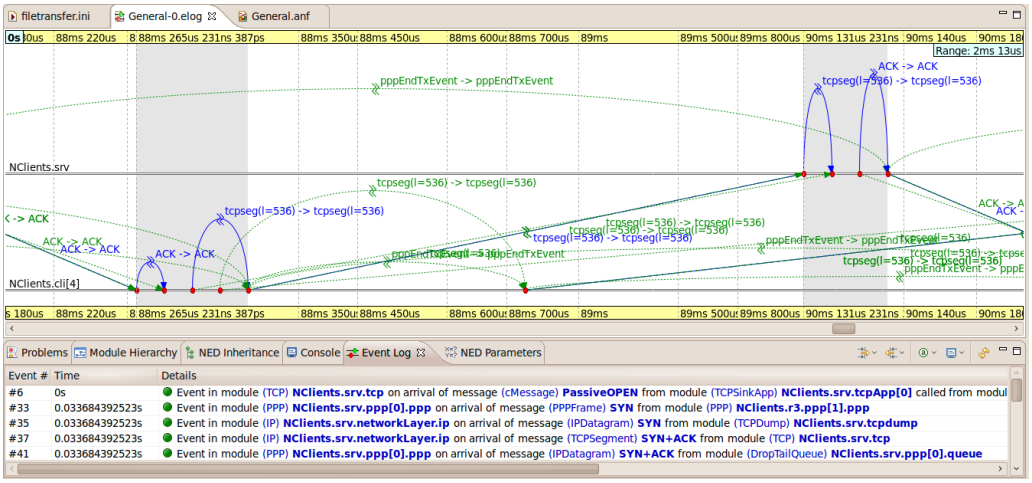


Fig.98 Timeline con los eventos del modelo 2a

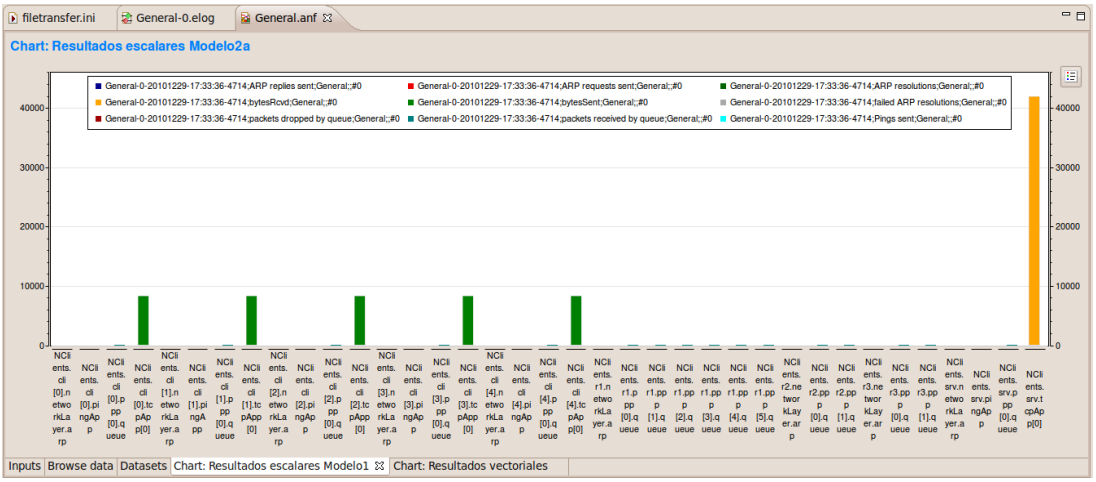


Fig.99 Gráfico escalar del modelo 2a

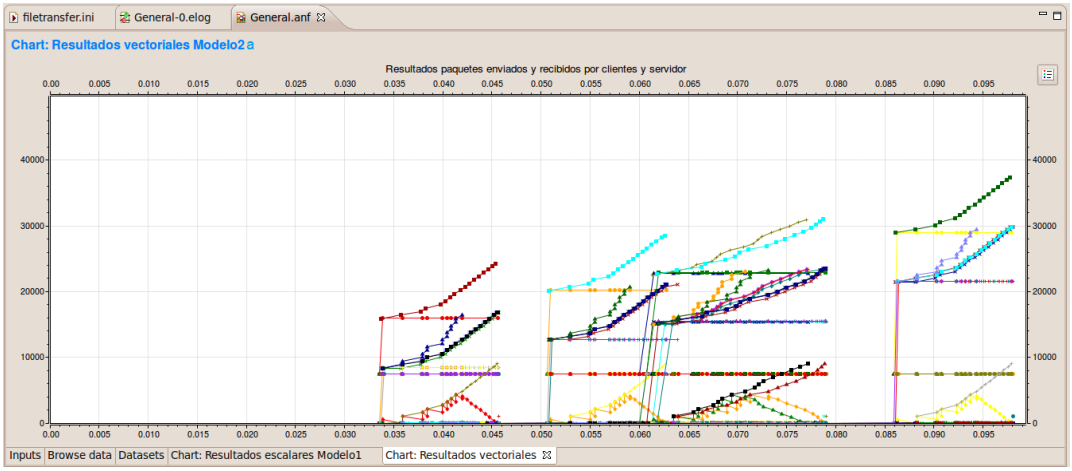


Fig.100 Gráfico vectorial del modelo 2a

Modelo 2 Servidor B

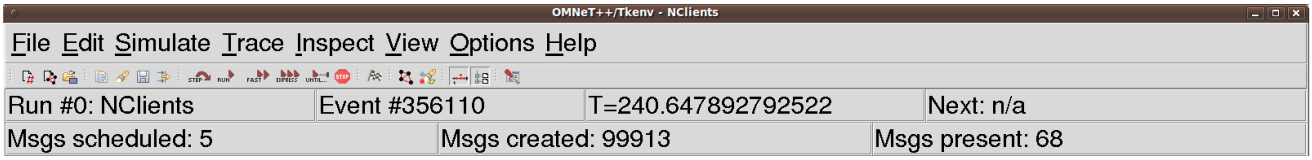


Fig.101 Eventos totales del modelo 2b

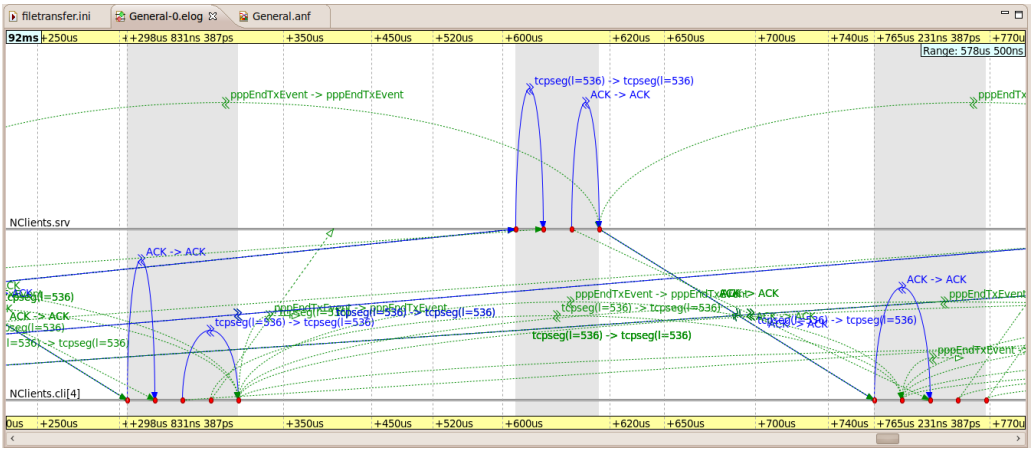


Fig.102 Timeline con los eventos del modelo 2b

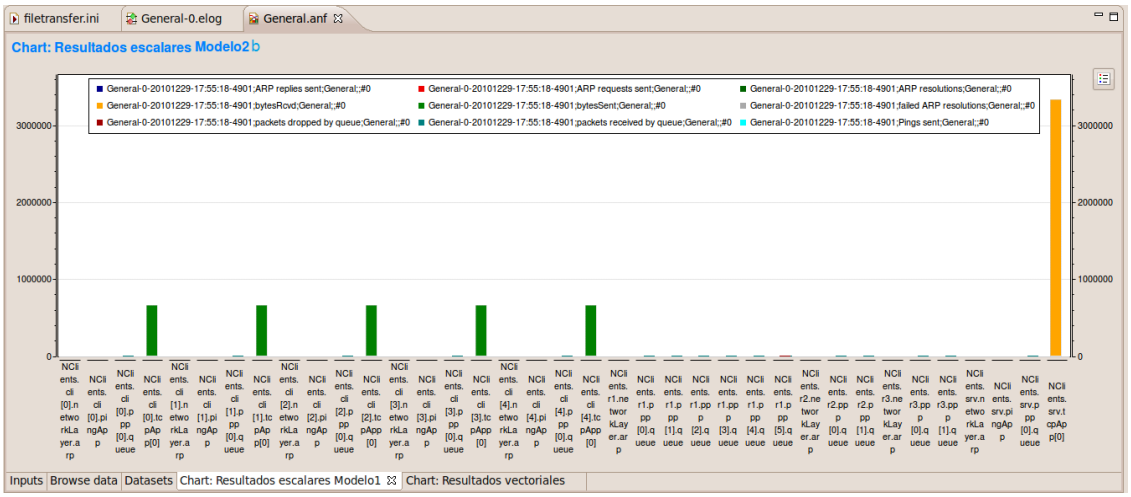


Fig.103 Gráfico escalar del modelo 2b

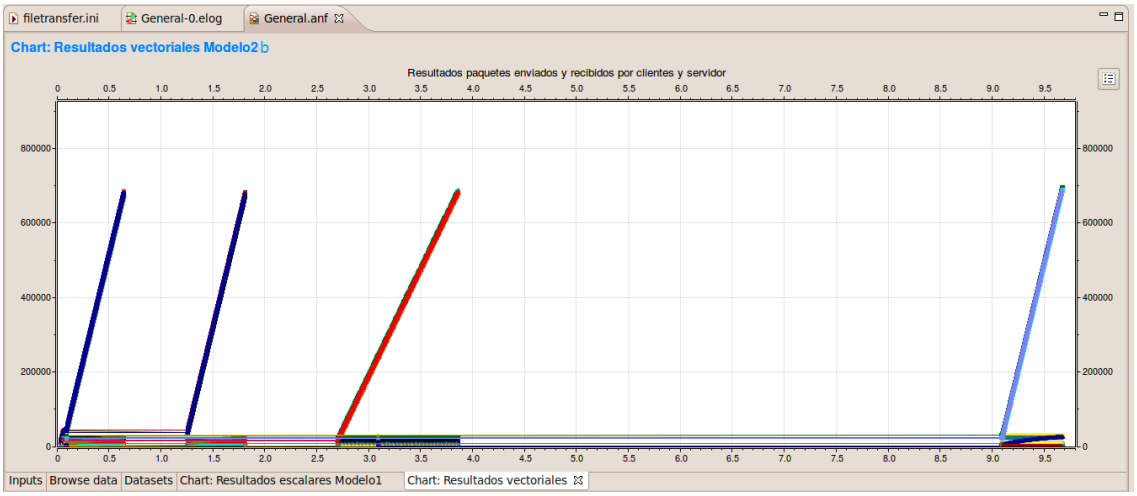


Fig.104 Gráfico vectorial del modelo 2b

Modelo 2 Servidor C

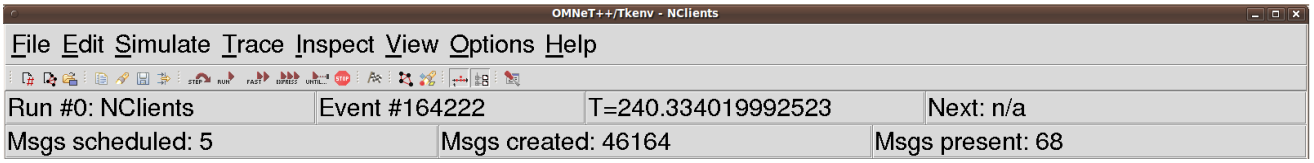


Fig.105 Eventos totales del modelo 2c

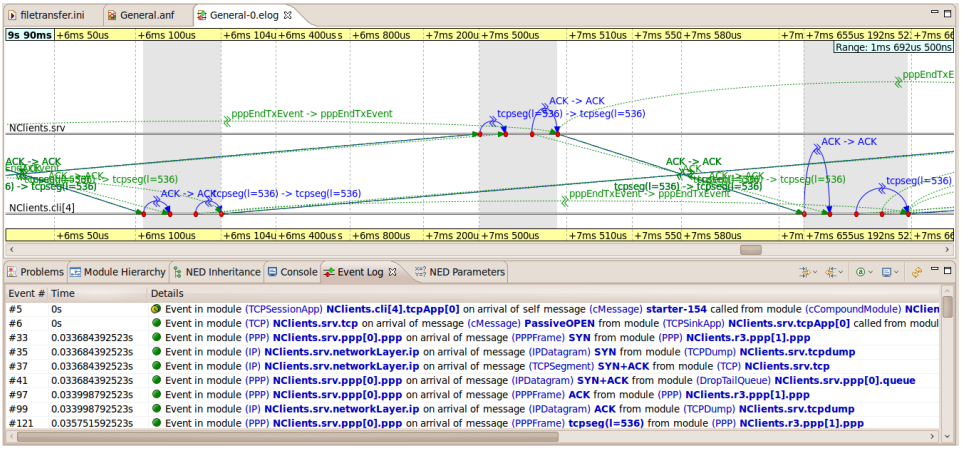


Fig.106 Timeline con los eventos del modelo 2c

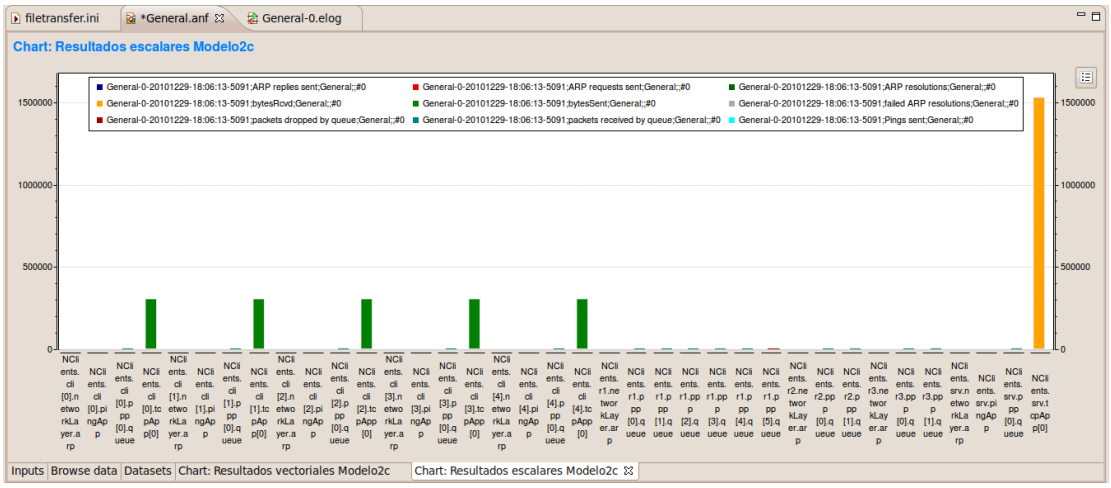


Fig.107 Gráfico escalar del modelo 2b

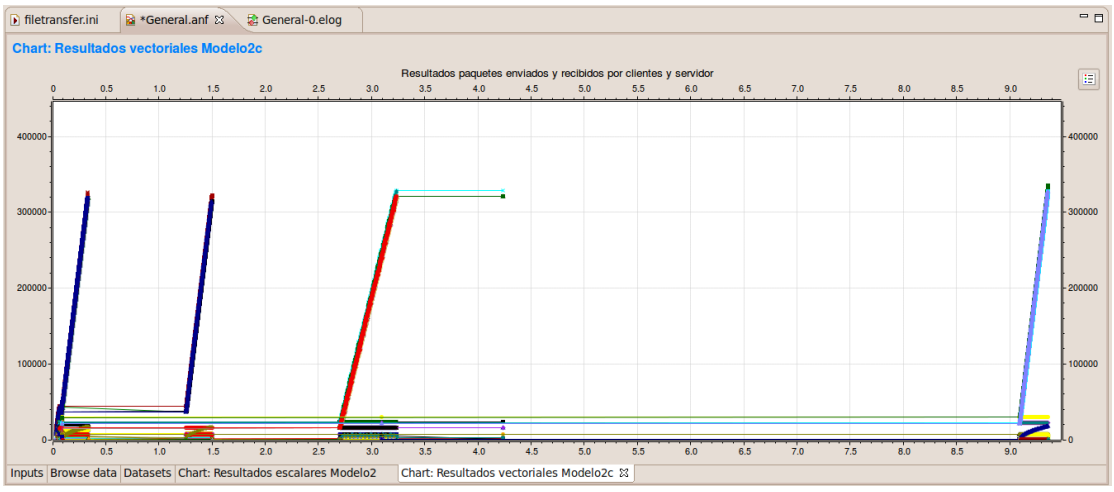


Fig.108 Gráfico vectorial del modelo 2c

Conclusiones

Se realizará un resumen de los resultados obtenidos para posteriormente llegar a unas conclusiones de la comparación de los 2 modelos analizados durante el proyecto.

De las pruebas de comportamiento del entorno de Habanero obtenemos los siguientes resultados:

- Modelo 1 (Servidor único)
    - 188 paquetes capturados con wireshark.
    - 31,63% de utilización del medio de transmisión por parte del servidor.
- Modelo 2 (Múltiples instancias del servidor)
    - 235 paquetes capturados con wireshark.
    - 15,8% de utilización del medio de transmisión por parte del servidor.

De las pruebas realizadas en las simulaciones con omnetpp obtenemos los siguientes resultados:

- Modelo 1 (Servidor único)
    - 556990 eventos registrados en 10 segundos.
    - 5,2 MB recibidos por parte del servidor.
    - Mayor densidad en el gráfico vectorial.
- Modelo 2 (Múltiples instancias del servidor)
    - 5322, 356110 y 164222 eventos registrados en cada prueba con cada tipo de servidor, 525654 eventos en total.
    - 42 KB, 3,2 MB y 1,5 MB recibidos por parte de los tres servidores, 4,74 MB totales recibidos.
    - Menor densidad de tráfico en los gráficos vectoriales.

A la vista de los resultados se pueden obtener las siguientes conclusiones. Teniendo en cuenta el límite de 30 clientes que puede soportar el servidor de aplicaciones a partir del cual se satura y deja de responder bloqueando el sistema, podemos afirmar que es necesario repartir la carga en diferentes servidores, para esto sería necesario utilizar múltiples instancias del servidor Tomcat.

Al utilizar el balanceo de carga repartiendo los clientes dependiendo del tipo de carrera que están estudiando, podemos observar que el rendimiento no mejora de forma lineal, es decir, si observamos los totales del modelo 1 y del modelo 2, en el segundo caso tenemos valores más pequeños, lo que significa que mejora aún más el rendimiento general del sistema con el mismo número de clientes.

Volviendo al símil utilizado al principio del proyecto, en el que hacíamos una comparación con un supermercado, en el primer modelo tendríamos a todos los clientes en la cafetería con los camareros saturados sin capacidad de reaccionar y en el segundo modelo tendríamos todos los clientes repartidos por zonas en el supermercado y todo funcionando correctamente.

Llegados a éste punto se han alcanzado todos los objetivos planteados para el proyecto. Sólo queda el último apartado en el que se aplicarán algunas medidas de seguridad ya que son importantes y es recomendable aplicarlas en cualquier sistema informático.

Seguridad del servidor

Contraseñas shadow

En entornos multiusuario es muy importante utilizar *contraseñas shadow* también conocido como oscurecimiento de contraseñas, (proporcionadas por el paquete shadow-utils). Haciendo esto se mejora la seguridad de los archivos de autenticación del sistema.

Se pueden observar los dos archivos que contienen las contraseñas, el que está sin encriptar, en el que ya no aparecen las contraseñas, y el archivo que las contiene, pero en el que se ven encriptadas.

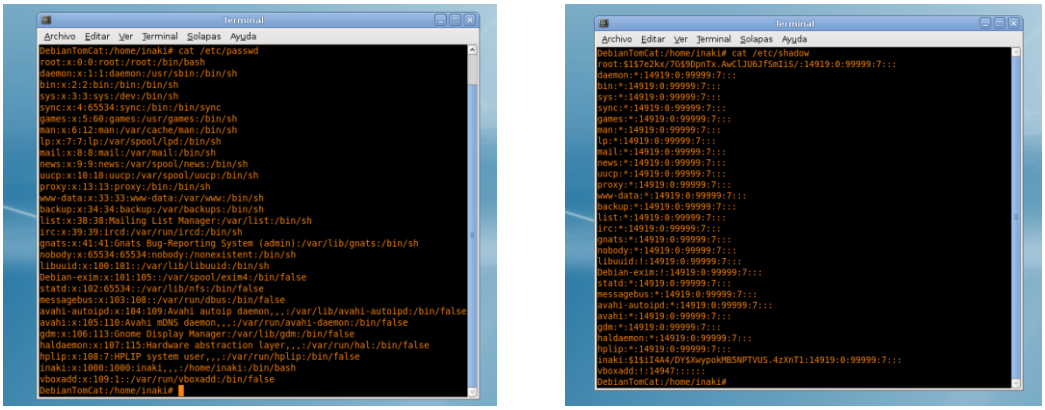


Fig.109 Archivos passwd y shadow. Contienen las contraseñas de usuario

Bit de inmutabilidad

El bit pegajoso (sticky bit) significa que un usuario solo podrá modificar y eliminar ficheros y directorios subordinados dentro de un directorio que le pertenezca. Los directorios a los cuales se les ha establecido bit pegajoso restringen las modificaciones de los usuarios sobre los archivos. Esta característica puede ser útil para que nadie pueda modificar por ejemplo una página web. En éste caso se puede apreciar que se aplica el bit de inmutabilidad a los servlets que aparecen en la página inicial de Tomcat.

Valor	Permiso	Decripción
0	-	Nada
1	x	Ejecución
2	w	Escritura
3	wx	Escritura y ejecución
4	r	Lectura
5	rx	Lectura y Ejecución
6	rw	Lectura y Escritura
7	rwx	Lectura, Escritura y Ejecución

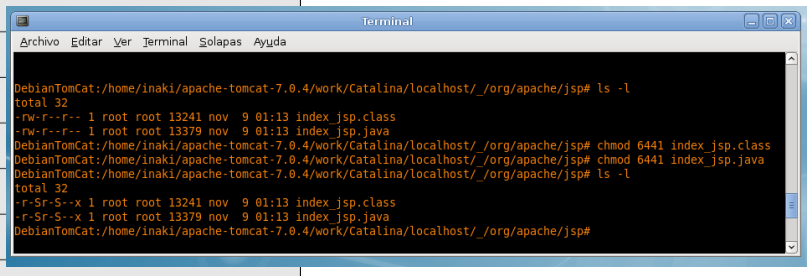


Fig.110 Aplicación del bit de inmutabilidad

Registro de eventos del sistema. Logging con sysklogd

El programa sysklogd es un demonio, o servicio del sistema operativo, que se encarga de registrar los eventos que van sucediendo en el sistema. Es posible configurarlo para que envíe alarmas cuando se registra un determinado tipo de evento. Estos eventos se ordenan dependiendo de que parte lo genere, por ejemplo puede generar un evento el núcleo del sistema, y también tienen una escala de relevancia.

A continuación podemos ver el sencillo proceso de instalación, mediante un único comando y el archivo de configuración.

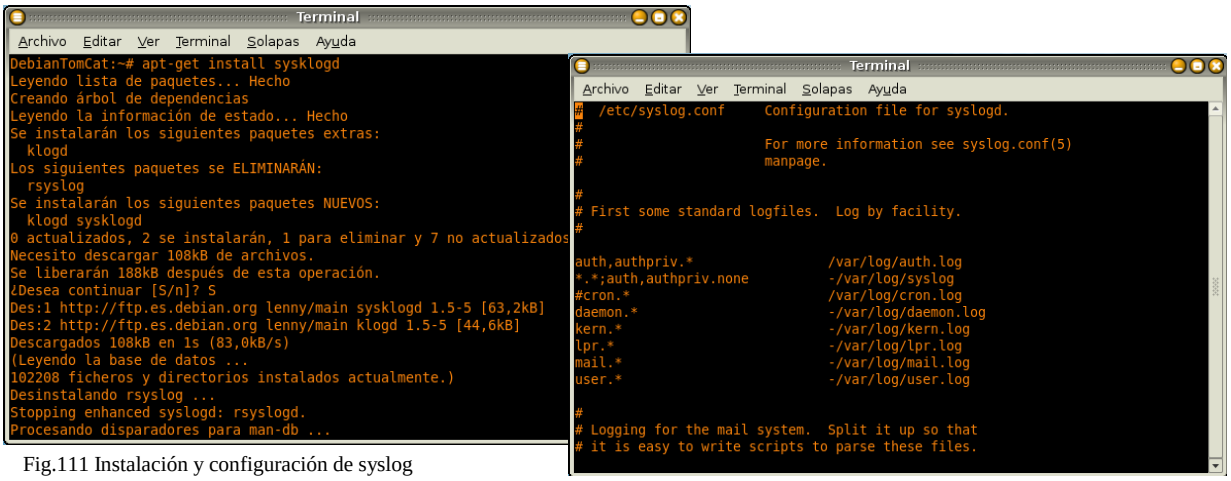


Fig.111 Instalación y configuración de syslog

Una vez instalado y configurado syslog, se procede a arrancarlo manualmente, o en el arranque del sistema operativo. Se puede ver el archivo de log en tiempo real, cada vez que sucede un evento se muestra en pantalla. Para poder hacerlo, tenemos que abrir una nueva terminal, por ejemplo mediante Ctrl-alt-f2, y logearnos como administrador, ya que es el único que puede acceder a este archivo. Después mostramos el archivo que se irá actualizando en pantalla mediante el comando tail.

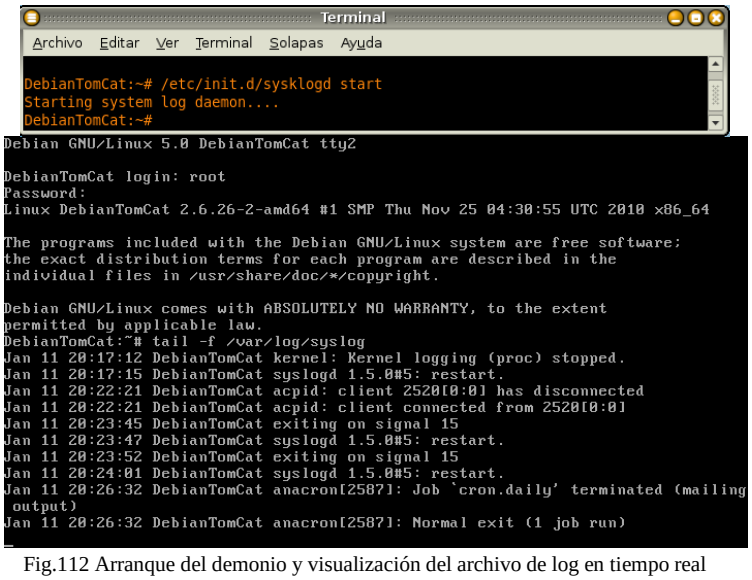


Fig.112 Arranque del demonio y visualización del archivo de log en tiempo real

IPTABLES (Cortafuegos en el servidor)

Ahora se va a presentar un cortafuegos, apto para el servidor Tomcat que nos permitirá bloquear todos los puertos excepto los que utiliza el propio servidor.

En primer lugar se crea el archivo /etc/network/if-pre-up.d/firewall con un editor.

 # vim /etc/network/if-pre-up.d/firewall

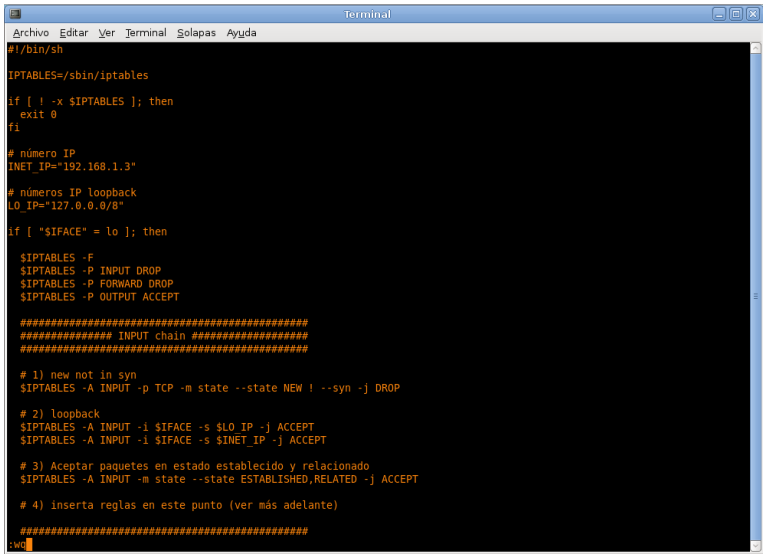


Fig.113 Configuración de las reglas del cortafuegos

Todo *script* en el directorio /etc/network/if-pre-up.d/ se corre antes de iniciar la interfaz. Debemos dar a éste archivo permisos de ejecución

chmod 755 /etc/network/if-pre-up.d/firewall

Después se crea el siguiente archivo

vim /etc/network/if-up.d/firewall

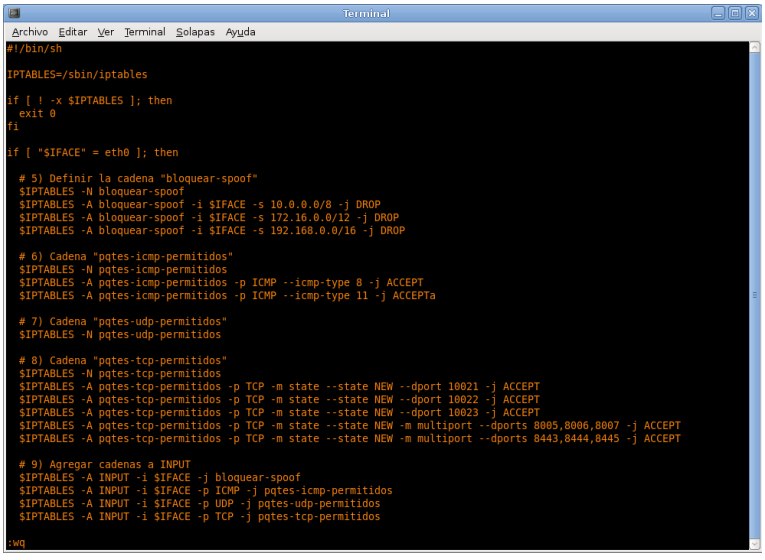


Fig.114 Configuración específica de los puertos de conexión permitidos

En este otro archivo es donde se definen las reglas. Este script se inicia al levantar la interfaz de red. Contiene las reglas del cortafuegos que se aplicarán. Es necesario dar permisos de ejecución también a éste archivo.

chmod 755 /etc/network/if-up.d/firewall

Se define la cadena "bloquear-spoof" para descartar paquetes TCP en la interfaz externa cuya fuente son números IP de redes privadas. Estos números son reservados para redes internas y no deberían circular en forma externa.

Se define la cadena "pqtes-icmp-permitidos" con reglas relacionadas al protocolo ICMP. La primera regla acepta paquetes ICMP de tipo 8 (echo request) para permitir que la red externa reciba una respuesta con ping. La segunda regla acepta paquetes ICMP de tipo 11 (time exceeded).

Se define la cadena "pqtes-udp-permitidos" con reglas relacionadas al protocolo UDP.

Se define la cadena "pqtes-tcp-permitidos" con reglas relacionadas al protocolo TCP. En las tres primeras reglas, se abren los puertos 10021, 10022 y 10023 a requerimientos de todo número IP por la interfaz de la red externa, para permitir la conexión al servidor. Las dos siguientes reglas permiten conectar el servidor Tomcat con el servidor Habanero. Para poder especificar múltiples puertos es necesario cargar la condición "multiport" explícitamente.

Se insertan las nuevas reglas de la cadena INPUT, a partir de la última regla insertada en el *script* anterior. La primera de ellas hace saltar todo paquete que llega a la interfaz externa a la cadena "bloquear-spoof" para descartar todo paquete obviamente malicioso. Las próximas tres reglas hacen saltar todo paquete que llega a la interfaz externa, de los protocolos ICMP, UDP y TCP, a las cadenas "pqtes-icmp-permitidos", "pqtes-udp-permitidos" y "pqtes-tcp-permitidos", respectivamente.

Por último es necesario reiniciar la red para que se apliquen los cambios realizados con iptables.

/etc/init.d/networking stop

/etc/init.d/ifupdown-clean restart

/etc/init.d/networking start

Certificados digitales y encriptación de las conexiones

El último paso para crear un entorno seguro de comunicaciones entre los clientes y el servidor será encriptar las conexiones para que los datos viajen cifrados y estén protegidos.

A continuación se muestra el esquema en el que se aprecian los clientes que se conectan al servidor a través de un túnel ssh, en el que los datos viajan seguros. Por otra parte el servidor Tomcat se encuentra en una DMZ, o zona desmilitarizada, a la cual se puede acceder desde internet, pero la conexión entre éste y los servidores de aplicaciones colaborativas sólo es posible desde la red interna. Además ésta conexión entre servidores también está protegida por un túnel ssh.

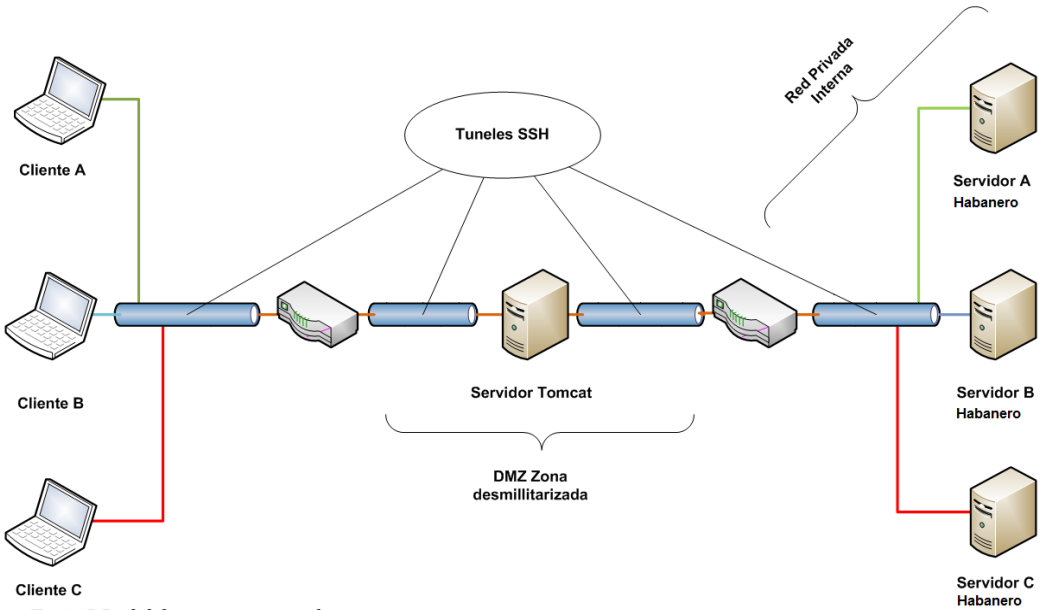


Fig.115 Red del sistema encriptado

Para poder crear los túneles ssh y encriptar de esa manera las conexiones, necesitamos certificados digitales. Para crear estos certificados y poderlos auto-firmar como si fuéramos una autoridad certificadora debemos instalar openssl mediante el siguiente comando.

```
apt-get install openssl
```

Una vez instalada la herramienta openssl lo primero que hay que hacer es crear la CA, que es la que nos valida y confirma que nuestro certificado es válido. Creamos un directorio en el que guardar los certificados que vamos creando y ejecutamos el siguiente comando.

```
openssl req -x509 -newkey rsa:2048 -keyout cakey.pem -days 3650 -out
cacert.pem
```

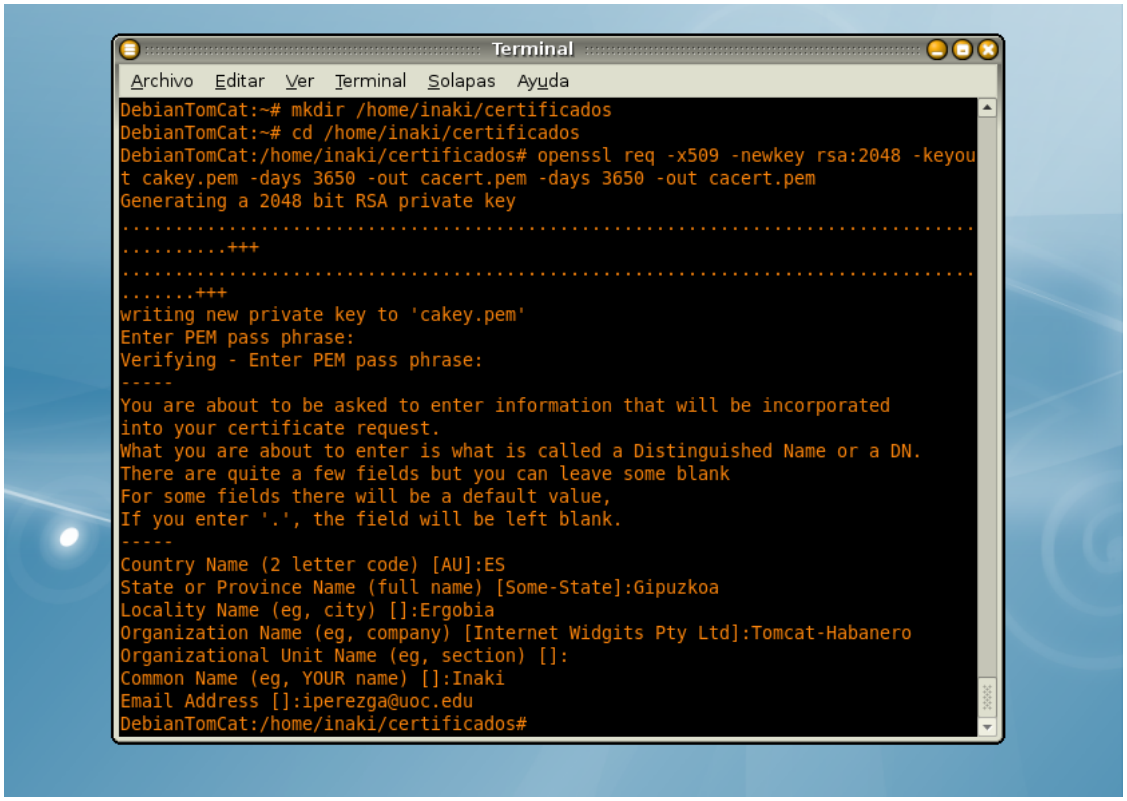


Fig.116 Creación de la autoridad certificadora

Con este comando creamos un CA para certificados X509 con algoritmo de encriptación rsa de 2048 bytes. Con el -keyout le indicamos que la clave privada de nuestra CA se almacene en el fichero cakey.pem y la clave publica -out en el cacert.pem.

Seguidamente nos pedirá un password y una serie de datos como por ejemplo país, nombre de empresa, que nos identifica como CA.

Ahora crearemos un certificado digital , es decir con nuestra CA creada vamos a crearnos un certificado.

Primero generamos la clave privada del que será nuestro certificado digital:

openssl genrsa -des3 -out serv-priv.pem -passout pass:mipassword 2048

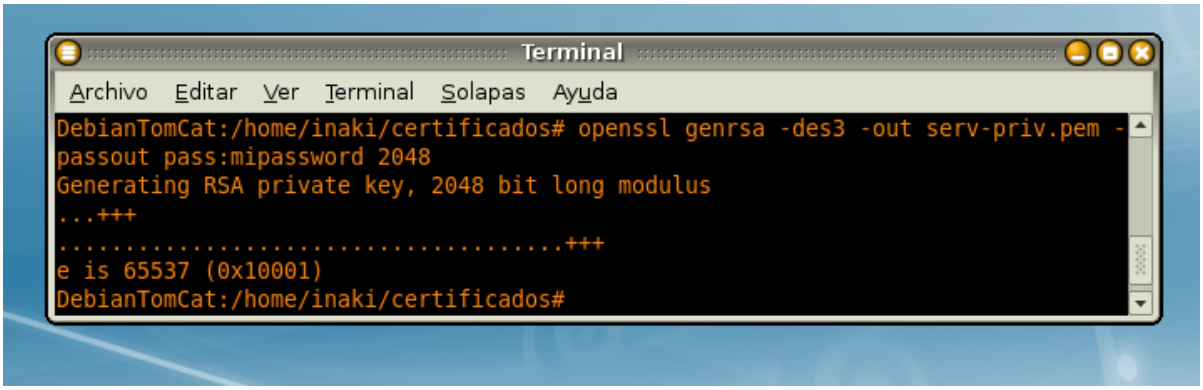


Fig.117 Creación de la clave privada

Con esto generamos la clave privada la cual tendrá un algoritmo de cifrado triple des (-des3) de 2048 bits y se almacenara en el fichero serv-priv.pem.

Ahora vamos a generar la petición del certificado, antes de hacer un certificado , hay que hacer una petición donde se define el propietario del certificado.

openssl req -new -subj "/DC=shadowsland.com/OU=com/CN=shadowsland" -key serv-priv.pem -passin pass:mipassword -out petic-certificado-serv.pem

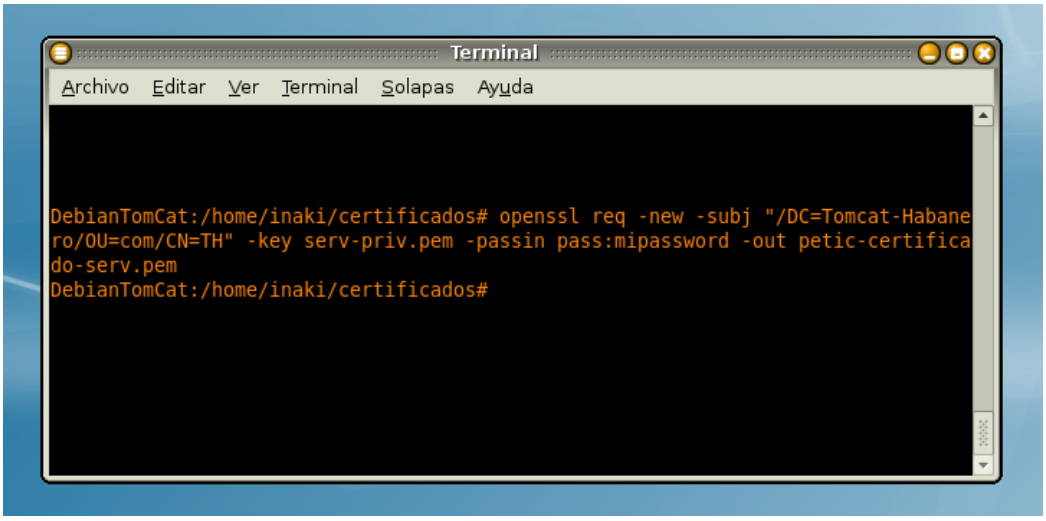


Fig.118 Petición de certificado

Finalmente ya podemos emitir el certificado.

openssl x509 -CA cacert.pem -CAkey cakey.pem -req -in petic-certificado-serv.pem -days 3650 -extfile config1.txt -sha1 -CAcreateserial -out servidor-cert.pem

Indicamos que será un certificado del tipo x509 cuya CA está definida en el fichero cacert.pem y que usa como clave privada el fichero cakey.pem y que el certificado a generar tendrá las especificaciones definidas en el apartado anterior las cuales están en el fichero de petición petic-certificado-serv.pem. El certificado tendrá una validez de diez años (-days 3650).

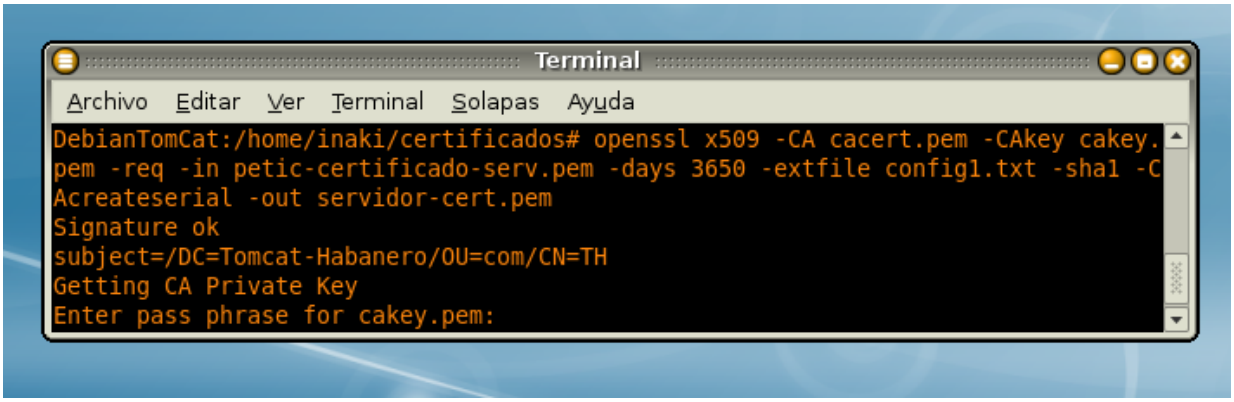


Fig.119 Creación del certificado digital firmado

Siguiendo todos los pasos anteriores obtenemos los siguientes archivos de entre los cuales cabe destacar servidor-cert.pem, que es el que se aplicaremos a Tomcat y al servidor de aplicaciones colaborativas.

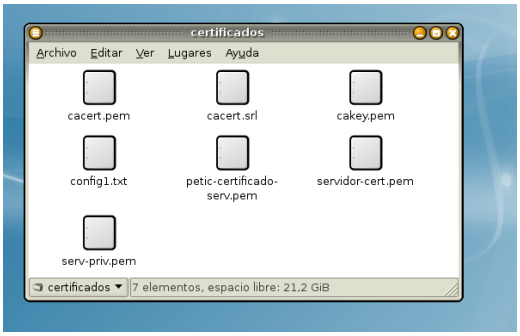


Fig.120 Archivos relacionados con el certificado digital

Por último sólo nos queda configurar Tomcat y Habanero para que utilicen SSL (Secure Socket Layer) y podamos tener comunicaciones seguras en nuestro sistema.

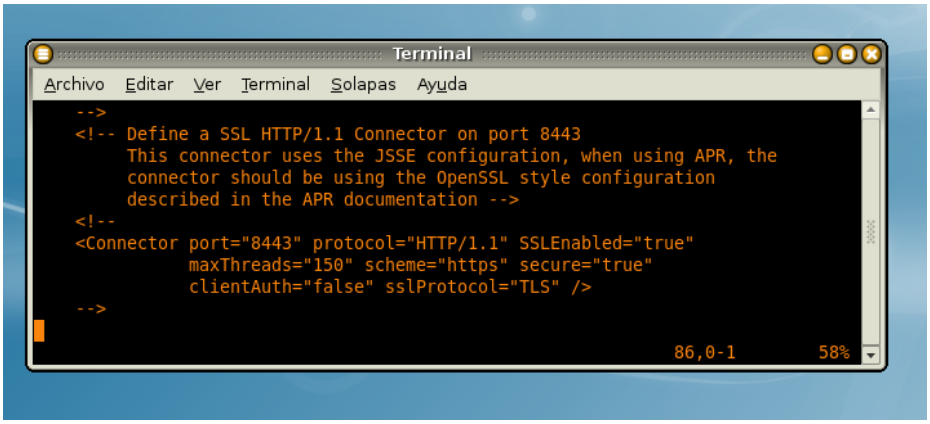


Fig.121 Aplicación de SSL en Tomcat

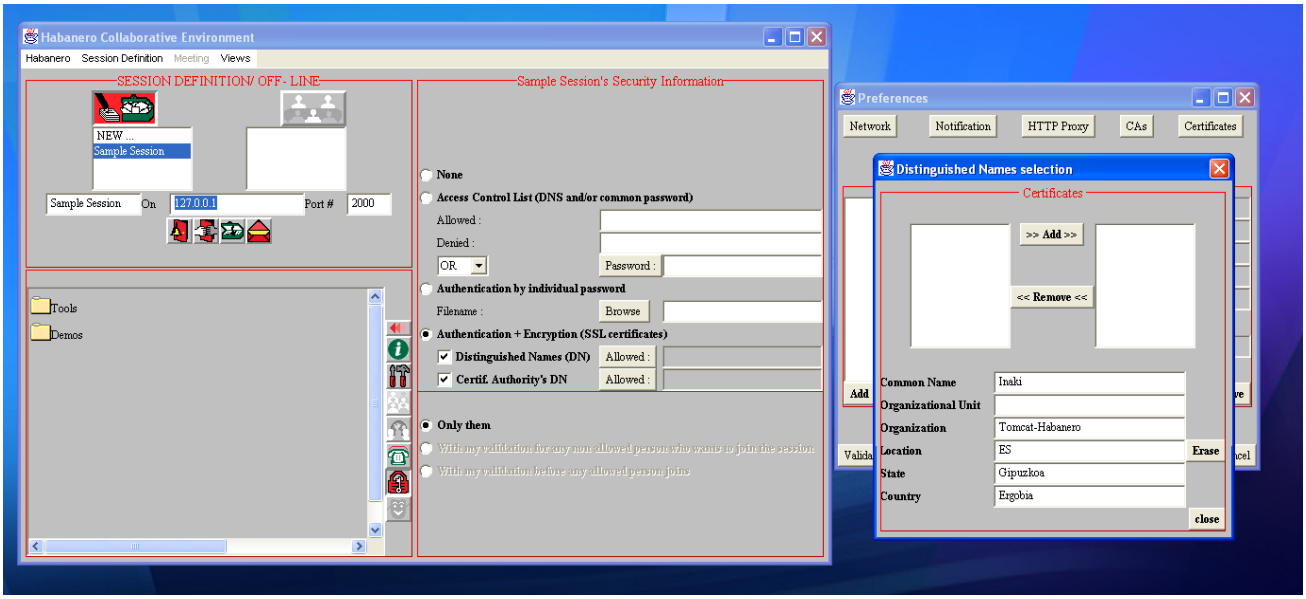


Fig.122 Aplicación de SSL en Habanero

Anexo A. Planificación del proyecto

El proyecto constará de cuatro etapas bien diferenciadas.

✓ Plan de trabajo

Se ha pretendido elaborar una temporización acorde a la dificultad de las distintas etapas y teniendo en cuenta el corto espacio de tiempo durante el que se desarrollará.

✓ Investigación y preparación del entorno de trabajo

Durante esta etapa se ha querido diferenciar dos vertientes del proyecto que se trabajarán en paralelo, la parte de simulación y la parte de software del servidor Tomcat.

✓ Modelado y simulación

En este punto se pretende hacer un modelado del problema a resolver y es también la etapa en la que convergen las dos vertientes descritas anteriormente. Si el tiempo lo permite se intentará abordar la temática de la seguridad informática.

✓ Memoria y documentación

Para terminar se redactará la memoria final y se realizará una prentación con diapositivas.

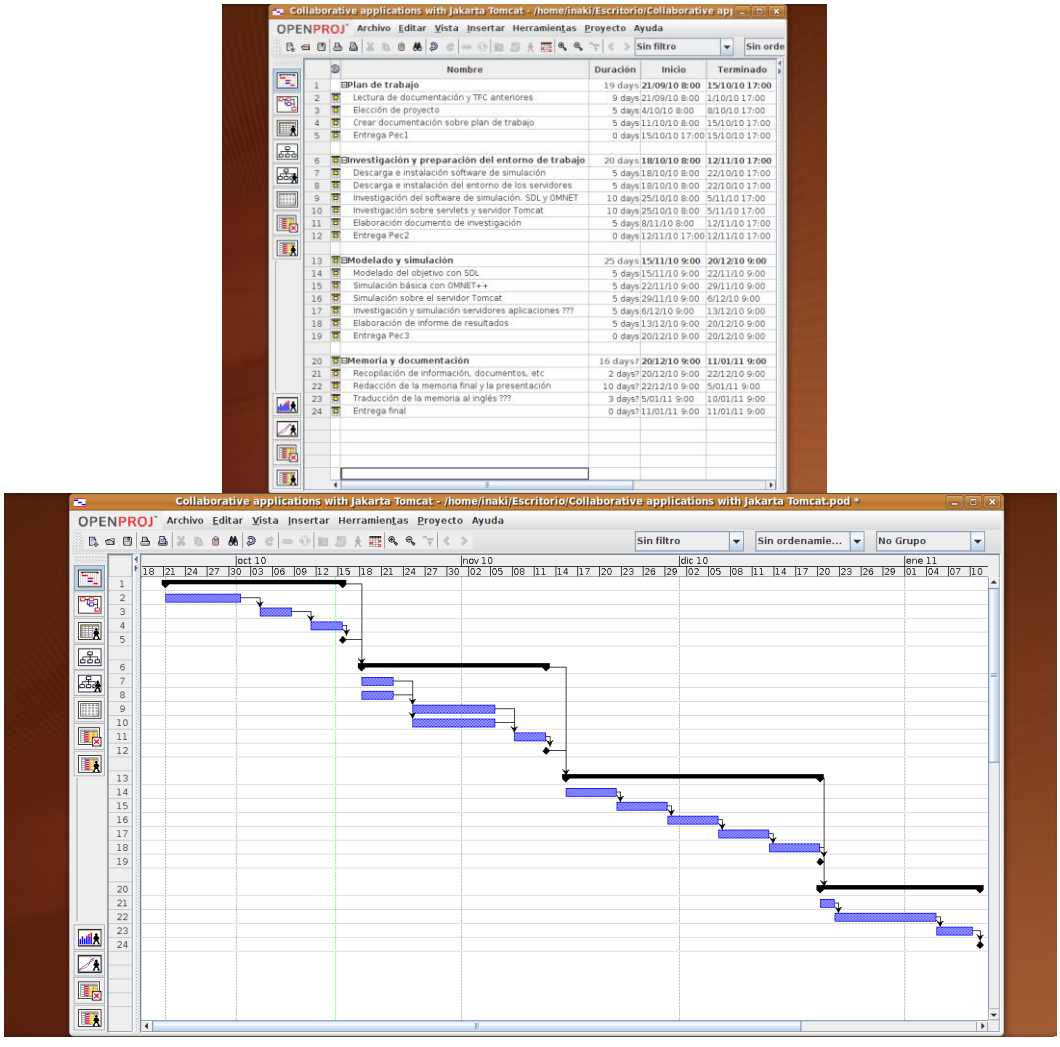


Fig.123 Planificación del proyecto y gráfico de Gantt

Anexo B. Indice de figuras

- Fig.1 Organización de un supermercado 5
- Fig.2 Cafetería del supermercado 5
- Fig.3. Estructura del lenguaje SDL 11
- Fig.4 Sandrila y Visio 12
- Fig.5 Configuración de Visio para incluir Sandrila 12
- Fig.6 Librerías de símbolos 12
- Fig.7 Diseño de sistema en SDL 13
- Fig.8 Canales, síncronos y asíncronos 13
- Fig.9 Especificación de un bloque en SDL 14
- Fig.10 Especificación de un proceso en SDL 14
- Fig.11 Simbología comúnmente utilizada en SDL 15
- Fig.12 Ejemplo de emisor en SDL 15
- Fig.13 Logo Ubuntu 16
- Fig.14 Configurando y compilando OMNeT++ 17
- Fig.15 Logo OMNeT++ 4.0 18
- Fig.16 Workbench de OMNeT++ 18
- Fig.17 Importando los proyectos Inet y ReaSE 18
- Fig.18 Configurando y compilando los proyectos Inet y ReaSE 19
- Fig.19 Configurando la ejecución de una simulación 19
- Fig.20 Parámetros de inicialización de una simulación 20
- Fig.21 Topología de la simulación MixedLan 20
- Fig.22 Configuración de los parámetros de los componentes de la simulación 20
- Fig.23 Ejecución de la simulación. Se puede observar una animación 21
- Fig.24 Listado de eventos 21
- Fig.25 Configuración mediante archivos .ini 21
- Fig.26 Archivo de salida, resumen de los resultados obtenidos 22
- Fig.27 Inicialización de parámetros. Número de hosts 22
- Fig.28 Topologías de 10 y 50 hosts y ejecución con animación de 10 hosts 22
- Fig.29 Topología de la simulación del servidor externo 23
- Fig.30 Configuración de la simulación. Detalle de configuración de dirección ip del servidor. 23
- Fig.31 Distintos modos de ejecución de la simulación 24
- Fig.32 Error con librerías pcap 24
- Fig.33 Arranque de Debian 5.05 y logo 25
- Fig.34 Arrancando Tomcat 26
- Fig.35 Página principal del servidor Tomcat 7.0.4 27
- Fig.36 Configuración de usuarios 27
- Fig.37 Login en Tomcat Manager Application 27
- Fig.38 Gestor de aplicaciones 28
- Fig.39 Directorio de ejemplos 28
- Fig.40 Estado del servidor 28
- Fig.41 Configuración de puertos 29
- Fig.42 Ejemplo sencillo de servlet con parámetros 29
- Fig.43 Ejemplo de servlet con conexión base de datos JDBC 30
- Fig.44 Logo Eclipse 30
- Fig.45 Edición del código de un servlet con Eclipse 30
- Fig.46 Ejemplos de aplicaciones colaborativas del proyecto Habanero 31
- Fig.47 Diseño del modelo 1 en SDL 35
- Fig.48 Diseño del modelo 2 en SDL 36
- Fig.49 Procedimientos de cada tipo de estudiante 37
- Fig.50 Procedimientos de cada tipo de servidor 37
- Fig.51 Procesos del servidor que recoge los resultados 38
- Fig.52 Escenario simple y configuración de la simulación extServer 39
- Fig.53 Error con librerías pcap. 39
- Fig.54 Solución del error de las librerías pcap 39
- Fig.55 Avance en la ejecución de la simulación extServer 40
- Fig.56 Nuevo error con tabla de rutas 40
- Fig.57 Fichero de configuración de la tabla de rutas del router 40
- Fig.58 Error en la interpretación de las tramas que se envían desde un navegador 41
- Fig.59 Interconexión entre la simulación y la parte real 41
- Fig.60 Configuración de los puertos de conexión y dirección del servidor externo 41
- Fig.61 Configuración del número de interfaces del router 42
- Fig.62 Escenario del modelo 1 (Servidor único) 42
- Fig.63 Escenario del modelo 2 (Múltiples instancias del servidor) 42
- Fig.64 Tamaño de los archivos a manejar por las aplicaciones 43
- Fig.65 Configuración del archivo de inicialización para el modelo 1 43
- Fig.66 Escenario ejecutándose del modelo 1 44
- Fig.67 Registro de los eventos de la simulación 44
- Fig.68 Límite de tiempo de las simulaciones 44
- Fig.69 Escenario ejecutándose del modelo 2 45
- Fig.70 Configuraciones para las simulaciones del modelo 2 45
- Fig.71 Fin de las simulaciones 46
- Fig.72 Estructura de carpetas para múltiples instancias del servidor Tomcat 47
- Fig.73 Configuración de los puertos de cada servidor 47
- Fig.74 Arranque de múltiples servidores 48

- Fig.75 Varios servidores funcionando simultáneamente 48
- Fig.76 Estructura de directorios del contenedor de servlets....49
- Fig.77 Archivo que ejecutarán los clientes cuando conecten al servlet....49
- Fig.78 Varios ejemplos de aplicaciones colaborativas....51
- Fig.79 Ejecución del cliente de Habanero junto a WhiteBoard. Modelo 1 (Servidor único) 51
- Fig.80 Configuración de puertos en el script de inicio del cliente 52
- Fig.81 Varios clientes conectados a varios servidores. Modelo 2....52
- Fig.82 Captura de tráfico con wireshark en las pruebas con el modelo 1 53
- Fig.83 Toma de muestras con MIB Browser modelo 1 53
- Fig.84 Tabla con los datos recogidos con MIB Browser modelo 1 54
- Fig.85 Gráfico de la utilización del medio de transmisión modelo 1....54
- Fig.86 Captura de tráfico con wireshark en las pruebas con el modelo 2 54
- Fig.87 Toma de muestras con MIB Browser modelo 2 55
- Fig.88 Tabla con los datos recogidos con MIB Browser modelo 2 55
- Fig.89 Gráfico de la utilización del medio de transmisión modelo 2....55
- Fig.90 Filtro del log de eventos....56
- Fig.91 Número de eventos totales modelo 1....56
- Fig.92 Timeline o gráfico en la línea del tiempo de los eventos del modelo 1....57
- Fig.93 Timeline o gráfico en la línea del tiempo de los eventos del modelo 1....57
- Fig.94 Configuración de los archivos escalar y vectorial para obtener los gráficos....57
- Fig.95 Gráfico escalar del modelo 1....58
- Fig.96 Gráfico vectorial del modelo 1....58
- Fig.97 Eventos totales del modelo 2ª....59
- Fig.98 Timeline con los eventos del modelo 2ª 59
- Fig.99 Gráfico escalar del modelo 2ª....59
- Fig.100 Gráfico vectorial del modelo 2ª....59
- Fig.101 Eventos totales del modelo 2b....60
- Fig.102 Timeline con los eventos del modelo 2b 60
- Fig.103 Gráfico escalar del modelo 2b....60
- Fig.104 Gráfico vectorial del modelo 2b....60
- Fig.105 Eventos totales del modelo 2c....61
- Fig.106 Timeline con los eventos del modelo 2c....61
- Fig.107 Gráfico escalar del modelo 2b....61
- Fig.108 Gráfico vectorial del modelo 2c 61
- Fig.109 Archivos passwd y shadow. Contienen las contraseñas de usuario 64
- Fig.110 Aplicación del bit de inmutabilidad....64
- Fig.111 Instalación y configuración de syslog 65
- Fig.112 Arranque del demonio y visualización del archivo de log en tiempo real....65
- Fig.113 Configuración de las reglas del cortafuegos....66
- Fig.114 Configuración específica de los puertos de conexión permitidos 66
- Fig.115 Red del sistema encriptado....68
- Fig.116 Creación de la autoridad certificadora....69
- Fig.117 Creación de la clave privada 69
- Fig.118 Petición de certificado....70
- Fig.119 Creación del certificado digital firmado....70
- Fig.120 Archivos relacionados con el certificado digital....71
- Fig.121 Aplicación de SSL en Tomcat....71
- Fig.122 Aplicación de SSL en Habanero....71
- Fig.123 Planificación del proyecto y gráfico de Gantt....72

Anexo C. Bibliografía

- ✓ Aprenda Java como si estuviera en primero
  - ➢ Autores: J. Garcia, J.I.Rodriguez, I. Mingo
  - ➢ Editorial: Escuela Superior de Ingenieros de la Universidad de Navarra
- ✓ Aprenda servlets de Java como si estuviera en segundo
  - ➢ Autores: J. Garcia, J.I.Rodriguez, A.Imaz
  - ➢ Editorial: Escuela Superior de Ingenieros de la Universidad de Navarra
- ✓ Learning Debian GNU/Linux
  - ➢ Autor: Bill McCarty
  - ➢ Editorial: O'Reilly & Associates
- ✓ Apache práctico
  - ➢ Autor: Ken Coar/Rich Bowen
  - ➢ Editorial: Anaya Multimedia/O'Reilly
- ✓ Desarrollo web con PHP, Apache y MySQL
  - ➢ Autores: M.K. Glass, E. Naramore, G. Mailer
  - ➢ Editorial: Anaya multimedia
- ✓ Apache Jakarta-Tomcat
  - ➢ Autor: James Goodwill
  - ➢ Editorial: Apress
- ✓ Seguridad en servidores Linux
  - ➢ Autor: Bauer, Michael D.
  - ➢ Editorial: Anaya Multimedia
- ✓ Integrating real world applications into OMNeT++
  - ➢ Autores: Christoph P. Mayer, Thomas Gamer
  - ➢ Editorial: Universit at Karlsruhe, Institut fur Telematik

- ✓ Miquel Angel Gallego Borras/Jose Miguel Vera Roa (2010): Modelaje y simulación del sistema informático que da soporte al Campus Virtual - Utilización de agentes inteligentes para modelar perfiles de usuario con Omnet++.

- ✓ Jose V. de la Cruz Chumillas (2009): Simulación del sistema informático de la UOC-Balanceadores de carga con distribuciones basadas en datos reales.

Referencias en Internet

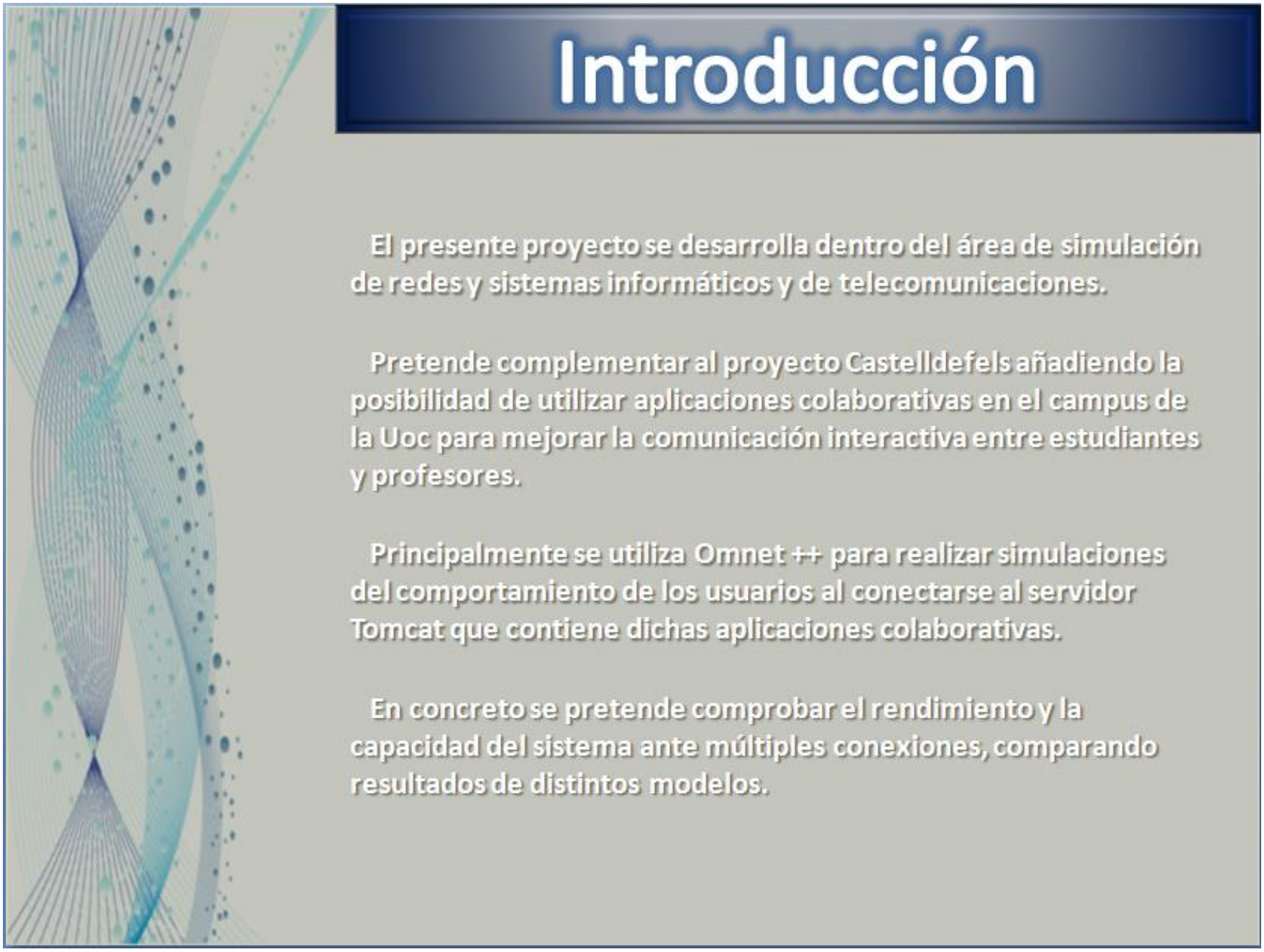
- www.apache.org
- jakarta.apache.org
- tomcat.apache.org
- www.debian.org
- www.servlets.com/jservlet2/examples/
- http://flanagan.ugr.es/docencia/2005-2006/2/servlets/ejemplos.html
- http://www.dosideas.com/noticias/java/547-la-revolucion-de-la-web-asincronica.html
- http://www.javahispano.org/contenidos/es/_google_y_jetty_implementaran_comet_en_gwt_/
- www.omnetpp.org/
- www.sdl-rt.org/
- www.eclipse.org/
- http://dpcs.uoc.edu/joomla/index.php/academics/104-computer-mas
- http://wiki.canaima.softwarelibre.gob.ve/wiki/index.php/Servidor_Tomcat
- http://www.bit-net.org/java/
- http://www.sintef.no/time/elb40/html/elb/sdl/sdl_t03.htm
- http://www2.ing.puc.cl/~iic3562/clases/clase04/index.html
- http://seguridadyredes.nireblog.com/post/2010/06/09/xplico-analizando-e-interpretando-nuestras-capturas-pcap
- http://www.isrl.illinois.edu/isaac/Habanero/
- http://man-es.debianchile.org/cortafuego.html

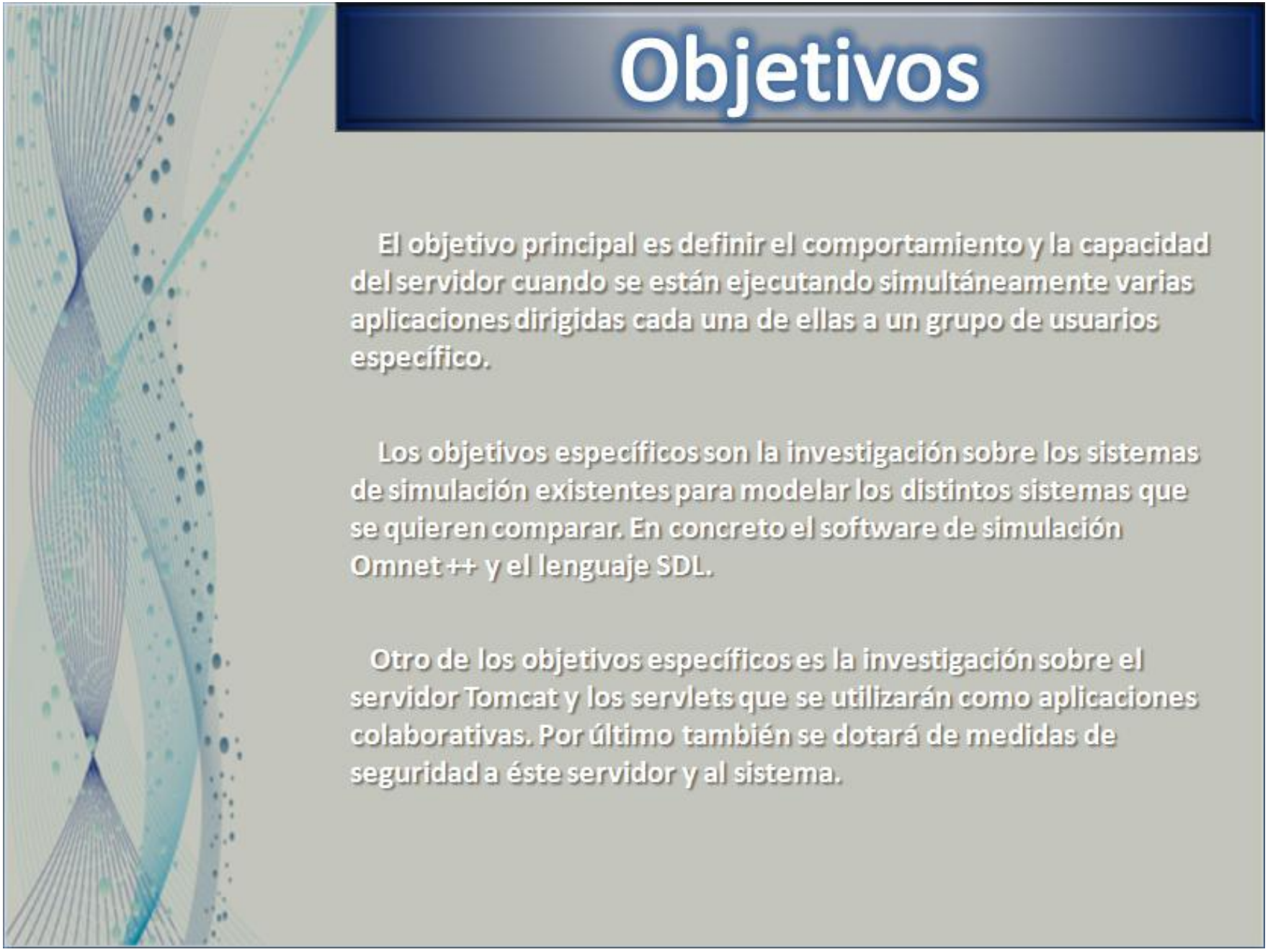
Defensa del proyecto.

Presentación virtual con diapositivas.

Junto con la memoria se ha adjuntado un dvd con el que se puede visualizar la presentación. En ese dvd también se puede encontrar el presente documento en diversos formatos y manuales y software utilizados en él.

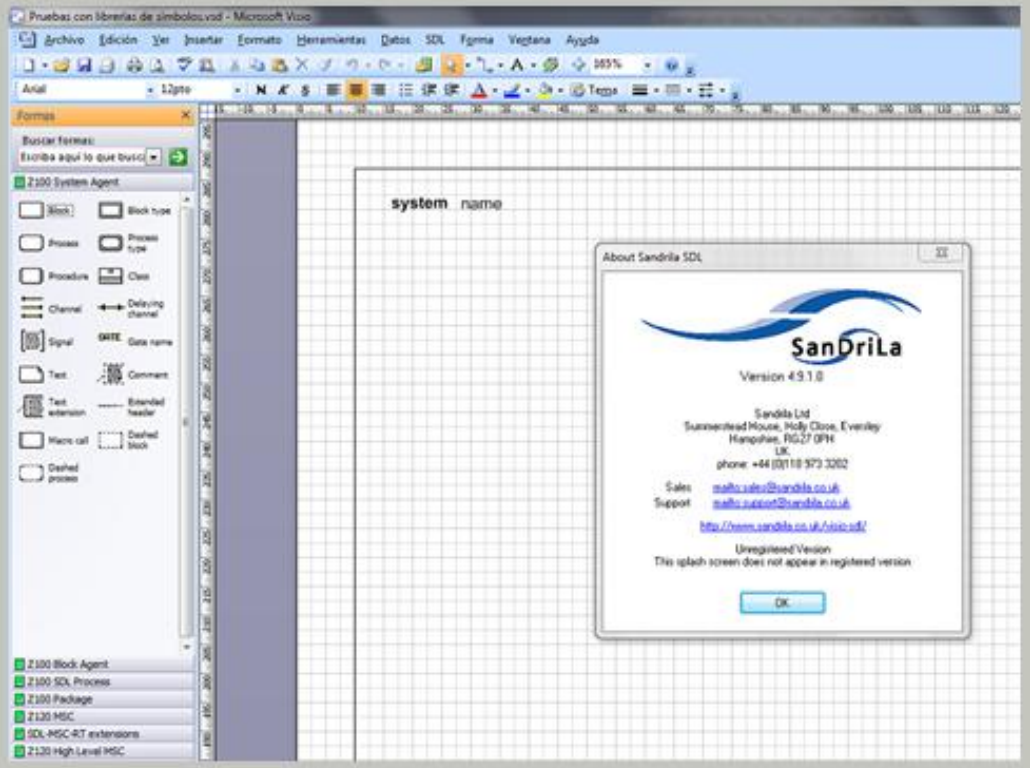

Iñaki Pérez García
TFC - Simulación de aplicaciones colaborativas sobre un servidor Tomcat
Ing. Téc. Telecomunicaciones
Esp. Telemática


Introducción
El presente proyecto se desarrolla dentro del área de simulación de redes y sistemas informáticos y de telecomunicaciones.
Pretende complementar al proyecto Castelldefels añadiendo la posibilidad de utilizar aplicaciones colaborativas en el campus de la Uoc para mejorar la comunicación interactiva entre estudiantes y profesores.
Principalmente se utiliza Omnet ++ para realizar simulaciones del comportamiento de los usuarios al conectarse al servidor Tomcat que contiene dichas aplicaciones colaborativas.
En concreto se pretende comprobar el rendimiento y la capacidad del sistema ante múltiples conexiones, comparando resultados de distintos modelos.


Objetivos
El objetivo principal es definir el comportamiento y la capacidad del servidor cuando se están ejecutando simultáneamente varias aplicaciones dirigidas cada una de ellas a un grupo de usuarios específico.
Los objetivos específicos son la investigación sobre los sistemas de simulación existentes para modelar los distintos sistemas que se quieren comparar. En concreto el software de simulación Omnet ++ y el lenguaje SDL.
Otro de los objetivos específicos es la investigación sobre el servidor Tomcat y los servlets que se utilizarán como aplicaciones colaborativas. Por último también se dotará de medidas de seguridad a éste servidor y al sistema.

SDL Básico

Los primeros pasos que se han dado han tenido que ver con SDL, un lenguaje de descripción que nos permite modelar los sistemas a simular. El software que se ha utilizado es Sandrila, una extensión de Visio.

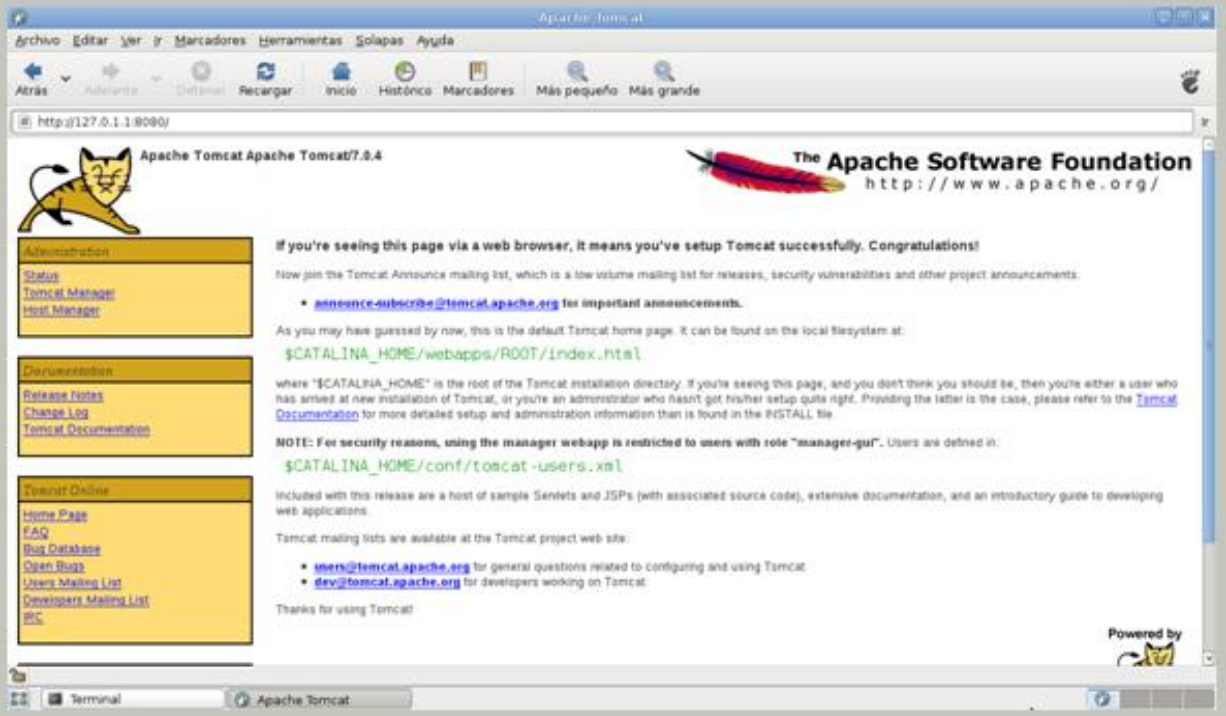


Omnet ++ Básico

Otra de las herramientas que se ha utilizado ha sido Omnet++. Este software de simulación es la utilidad principal en el proyecto, ya que es el que nos ha permitido obtener los resultados a partir de los cuales sacar las conclusiones.

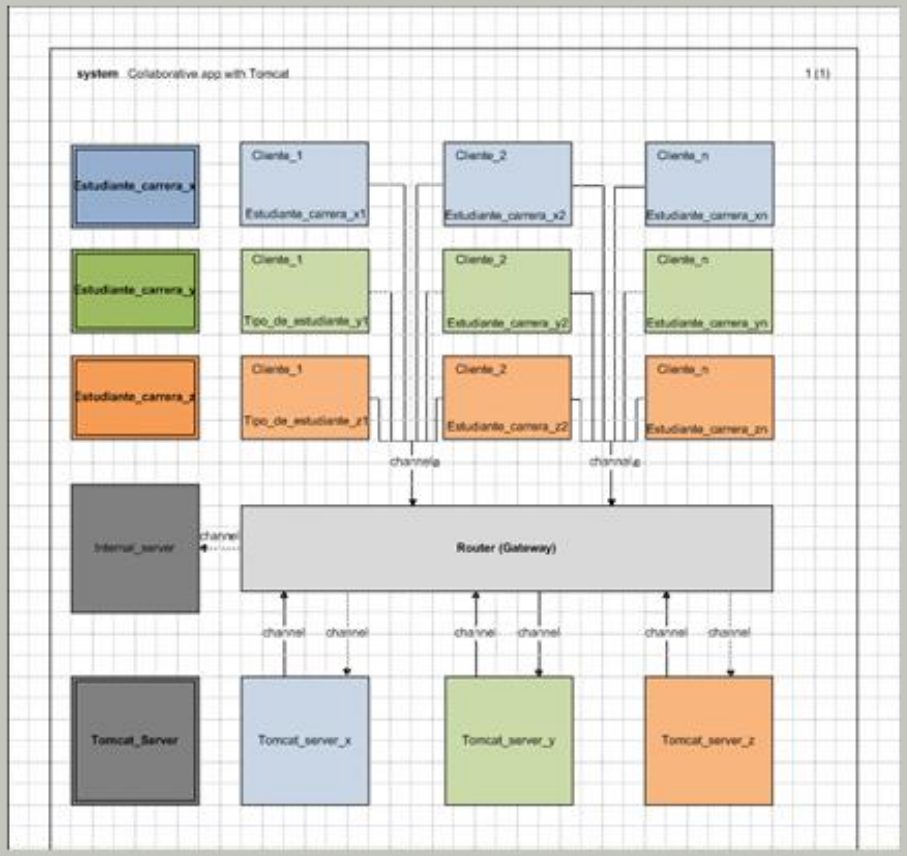
Servidor Tomcat Básico

Por último observamos el correcto funcionamiento del servidor y algunos ejemplos sencillos de servlets. Los conceptos aprendidos son los mismos que con las otras aplicaciones, instalación, configuración y ejecución de ejemplos.



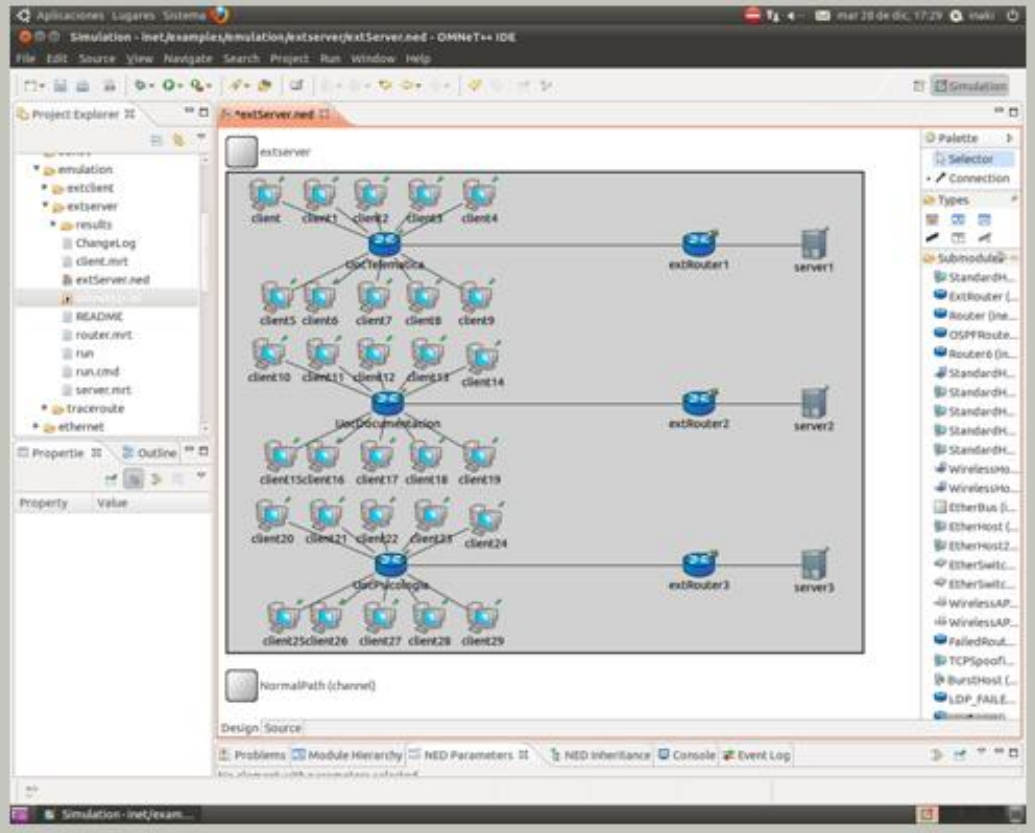
SDL avanzado

En la segunda vemos como existen tres servidores y cada grupo de clientes se conecta al servidor que le corresponde.



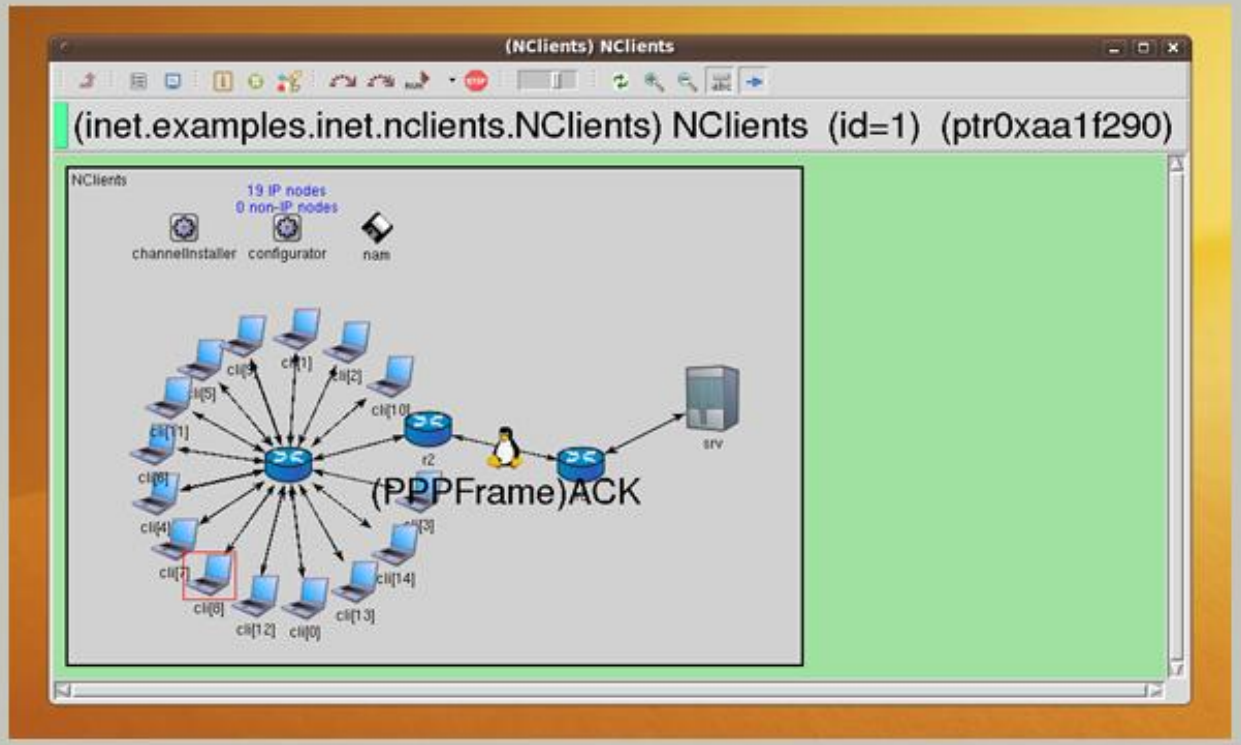
Omnet ++ avanzado

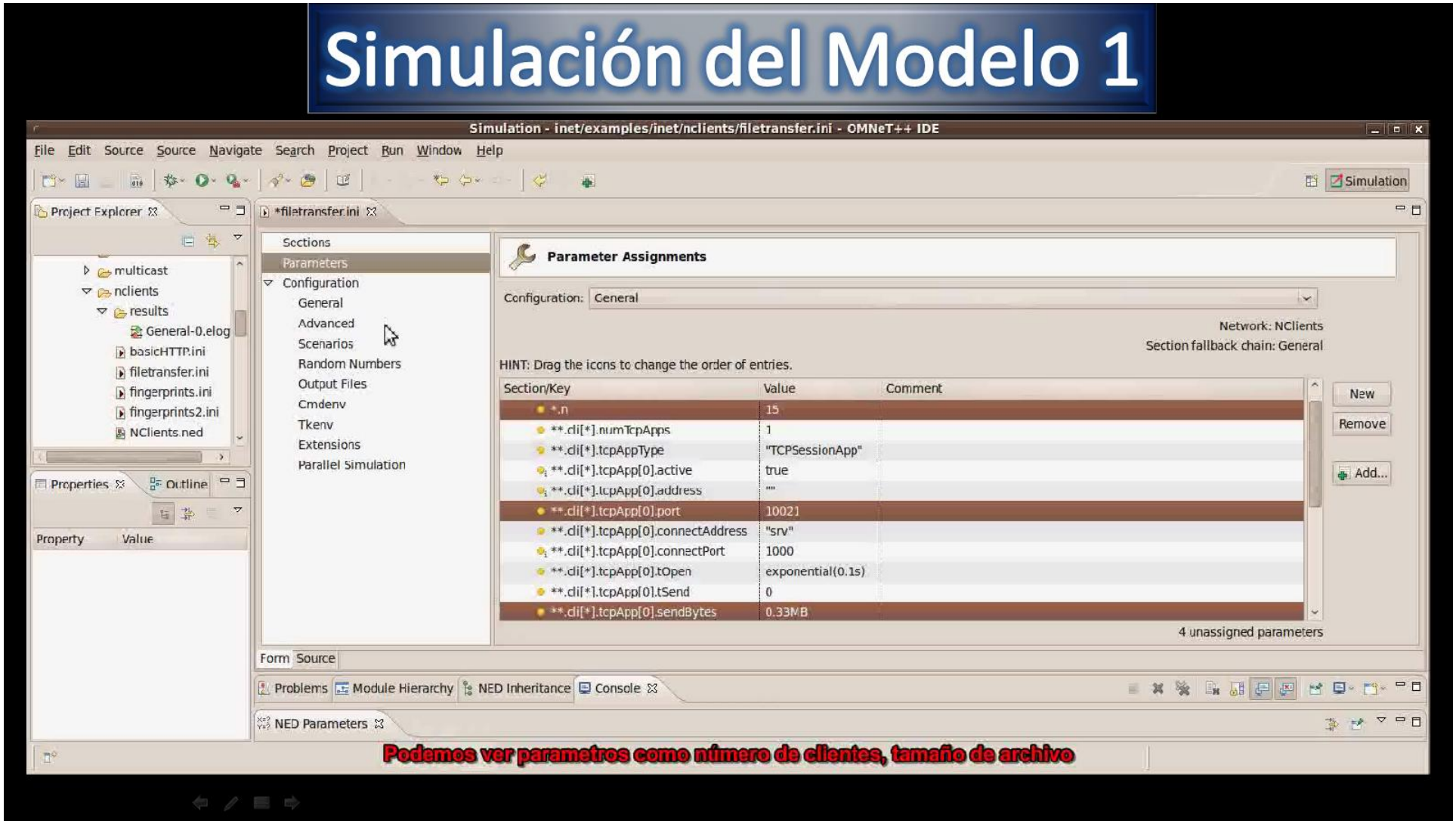
Podemos observar tanto el escenario para el modelo 1 con un único servidor como para varios servidores.

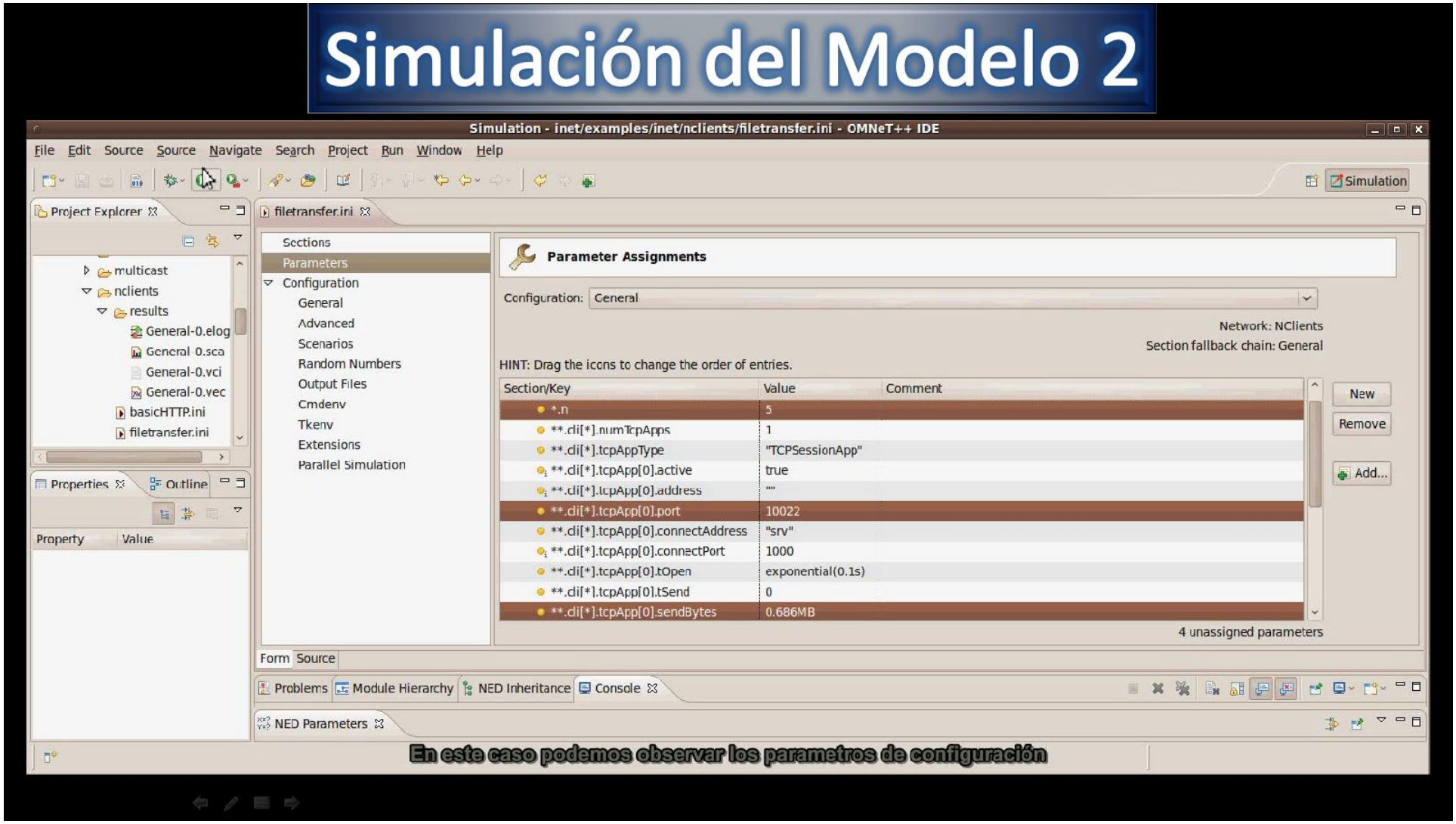


Omnet ++ avanzado

También tenemos un escenario para cada modelo, en el que varía el nº de clientes, puerto, tamaño de transferencia como se puede apreciar en sus configuraciones.

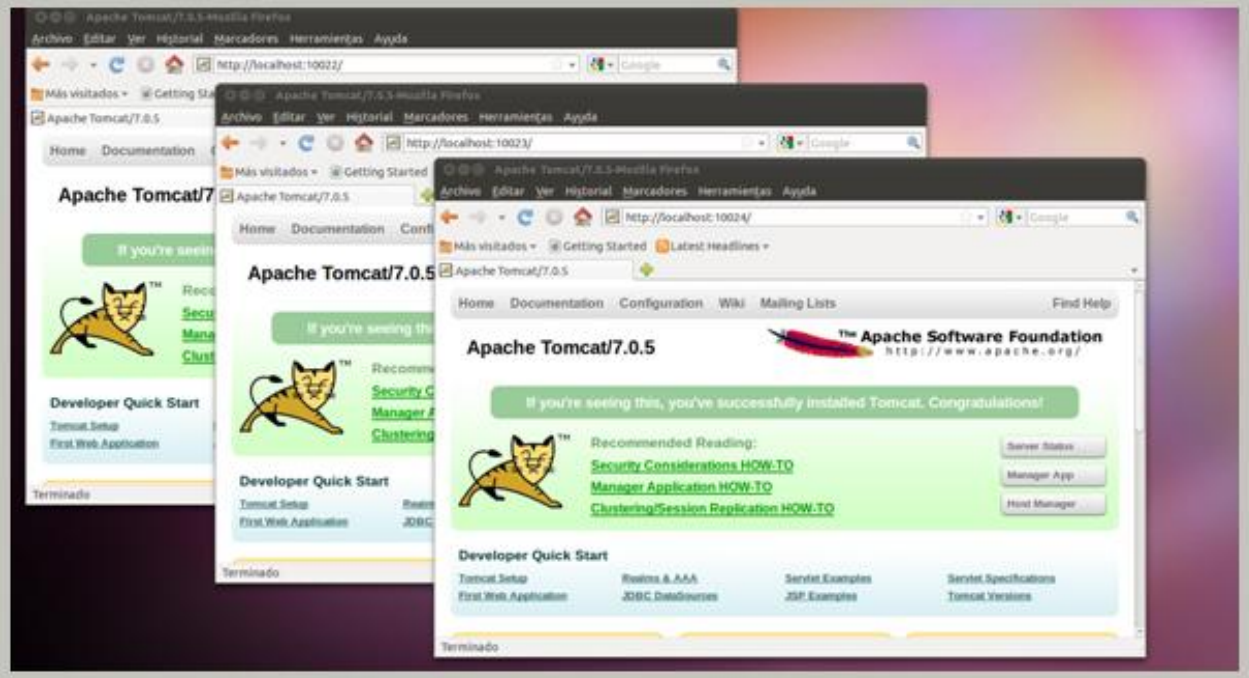


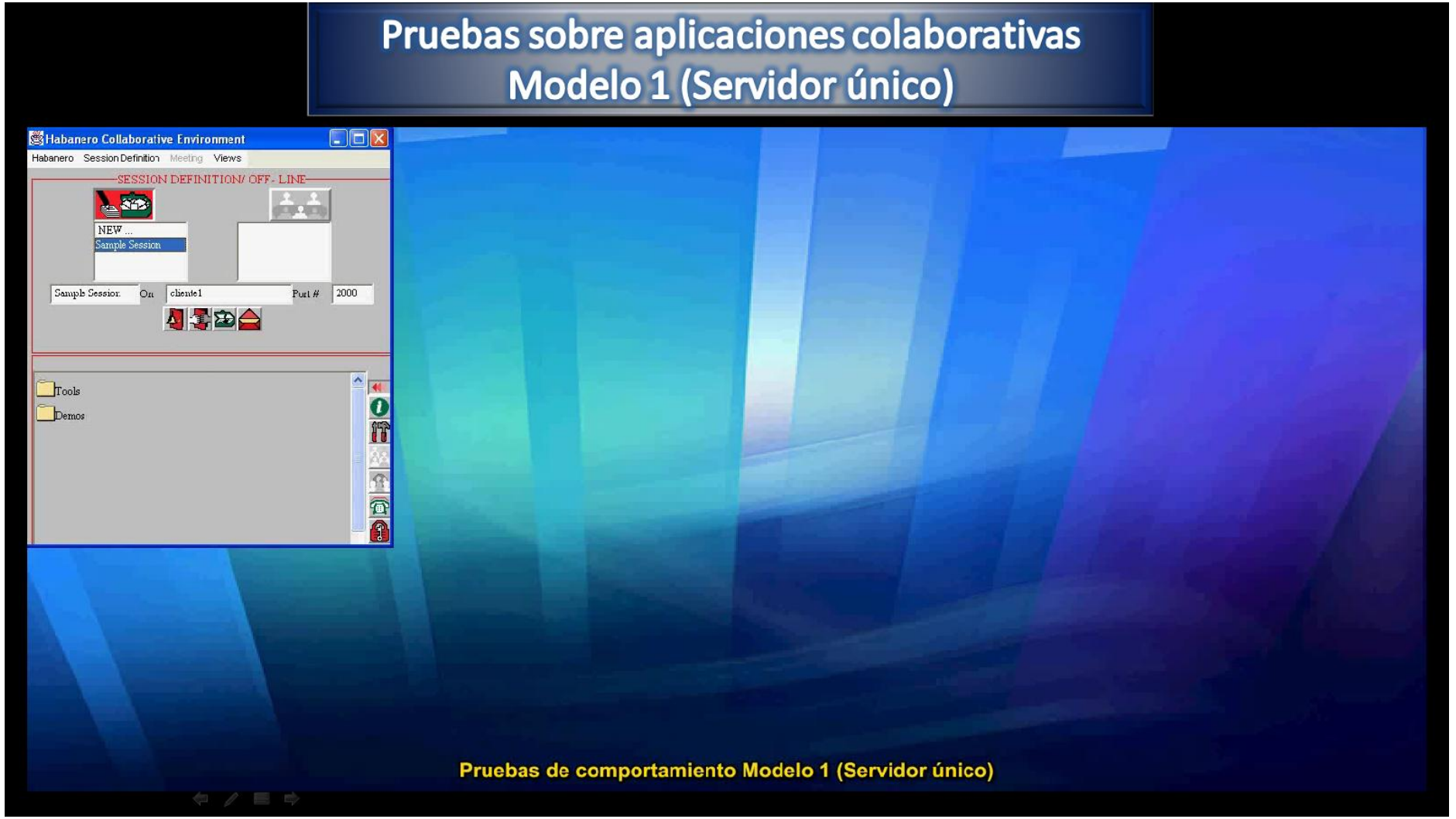

Simulación del Modelo 1
Simulation - inet/examples/inet/nclients/filetransfer.ini - OMNeT++ IDE
File Edit Source Source Navigate Search Project Run Window Help
Simulation
Project Explorer
multicast
nclients
results
General-0.elog
basicHTTP.ini
filetransfer.ini
fingerprints.ini
fingerprints2.ini
NClients.ned
Properties
Outline
Property
Value
*filetransfer.ini
Sections
Parameters
Configuration
General
Advanced
Scenarios
Random Numbers
Output Files
Cmdenv
Tkenv
Extensions
Parallel Simulation
Parameter Assignments
Configuration: General
Network: NClients
Section fallback chain: General
HINT: Drag the icons to change the order of entries.
Section/Key
Value
Comment
*.n
15
**.cli[*].numTcpApps
1
**.cli[*].tcpAppType
"TCPSessionApp"
**.cli[*].tcpApp[0].active
true
**.cli[*].tcpApp[0].address
""
**.cli[*].tcpApp[0].port
10021
**.cli[*].tcpApp[0].connectAddress
"srv"
**.cli[*].tcpApp[0].connectPort
1000
**.cli[*].tcpApp[0].tOpen
exponential(0.1s)
**.cli[*].tcpApp[0].tSend
0
**.cli[*].tcpApp[0].sendBytes
0.33MB
New
Remove
Add...
4 unassigned parameters
Form Source
Problems Module Hierarchy NED Inheritance Console
NED Parameters
Podemos ver parametros como número de clientes, tamaño de archivo

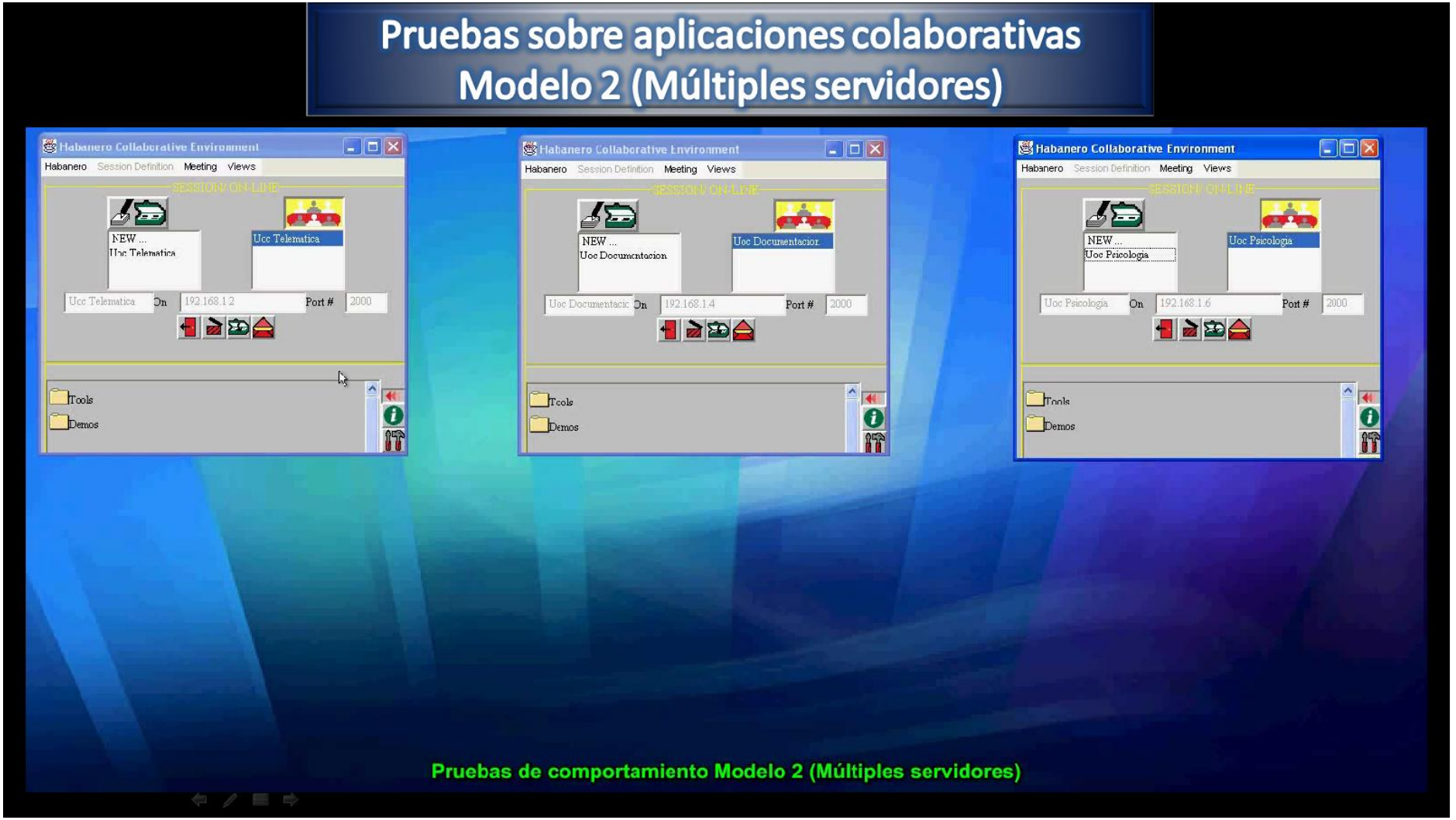

Simulación del Modelo 2
Simulation - inet/examples/inet/nclients/filetransfer.ini - OMNeT++ IDE
File Edit Source Source Navigate Search Project Run Window Help
Simulation
Project Explorer
multicast
nclients
results
General-0.elog
General-0.sca
General-0.vci
General-0.vec
basicHTTP.ini
filetransfer.ini
Properties
Outline
Property
Value
filetransfer.ini
Sections
Parameters
Configuration
General
Advanced
Scenarios
Random Numbers
Output Files
Cmdenv
Tkenv
Extensions
Parallel Simulation
Parameter Assignments
Configuration: General
Network: NClients
Section fallback chain: General
HINT: Drag the icons to change the order of entries.
Section/Key
Value
Comment
*.n
5
**.cli[*].numTcpApps
1
**.cli[*].tcpAppType
"TCPSessionApp"
**.cli[*].tcpApp[0].active
true
**.cli[*].tcpApp[0].address
""
**.cli[*].tcpApp[0].port
10022
**.cli[*].tcpApp[0].connectAddress
"srv"
**.cli[*].tcpApp[0].connectPort
1000
**.cli[*].tcpApp[0].tOpen
exponential(0.1s)
**.cli[*].tcpApp[0].tSend
0
**.cli[*].tcpApp[0].sendBytes
0.686MB
New
Remove
Add...
4 unassigned parameters
Form
Source
Problems
Module Hierarchy
NED Inheritance
Console
NED Parameters
En este caso podemos observar los parametros de configuración

Tomcat avanzado

Habrá que configurar cada uno de ellos de forma diferente para que se puedan ejecutar todos a la vez.

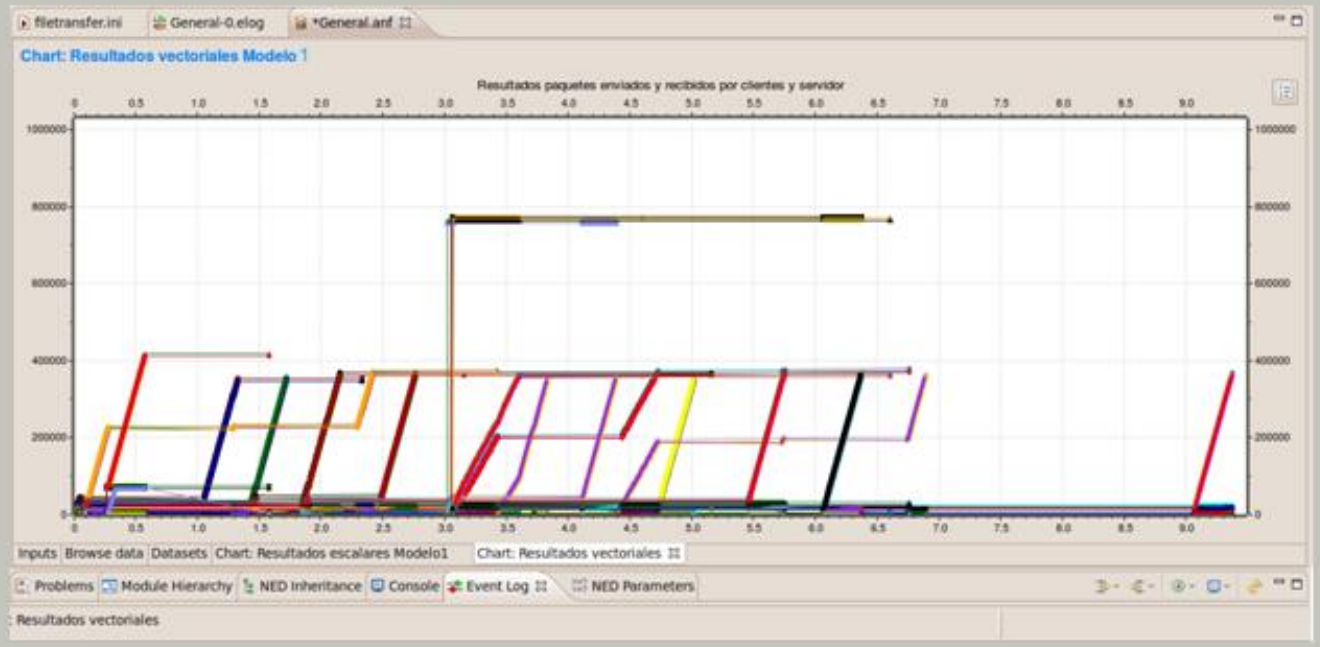



Pruebas sobre aplicaciones colaborativas
Modelo 1 (Servidor único)
Habanero Collaborative Environment
Habanero Session Definition Meeting Views
SESSION DEFINITION/ OFF - LINE
NEW ...
Sample Session
Sample Session: On cliente1 Port # 2000
Tools
Demos
Pruebas de comportamiento Modelo 1 (Servidor único)


Pruebas sobre aplicaciones colaborativas
Modelo 2 (Múltiples servidores)
Habanero Collaborative Environment
Habanero Session Definition Meeting Views
SESSION ON-LINE
NEW ...
Ucc Telematica
Ucc Telematica On 192.168.1.2 Port # 2000
Tools
Demos
Habanero Collaborative Environment
Habanero Session Definition Meeting Views
SESSION ON-LINE
NEW ...
Uoc Documentacion
Uoc Documentacior
Uoc Documentacic On 192.168.1.4 Port # 2000
Tools
Demos
Habanero Collaborative Environment
Habanero Session Definition Meeting Views
SESSION ON-LINE
NEW ...
Uoc Psicologia
Uoc Psicologia
Uoc Psicologia On 192.168.1.6 Port # 2000
Tools
Demos
Pruebas de comportamiento Modelo 2 (Múltiples servidores)

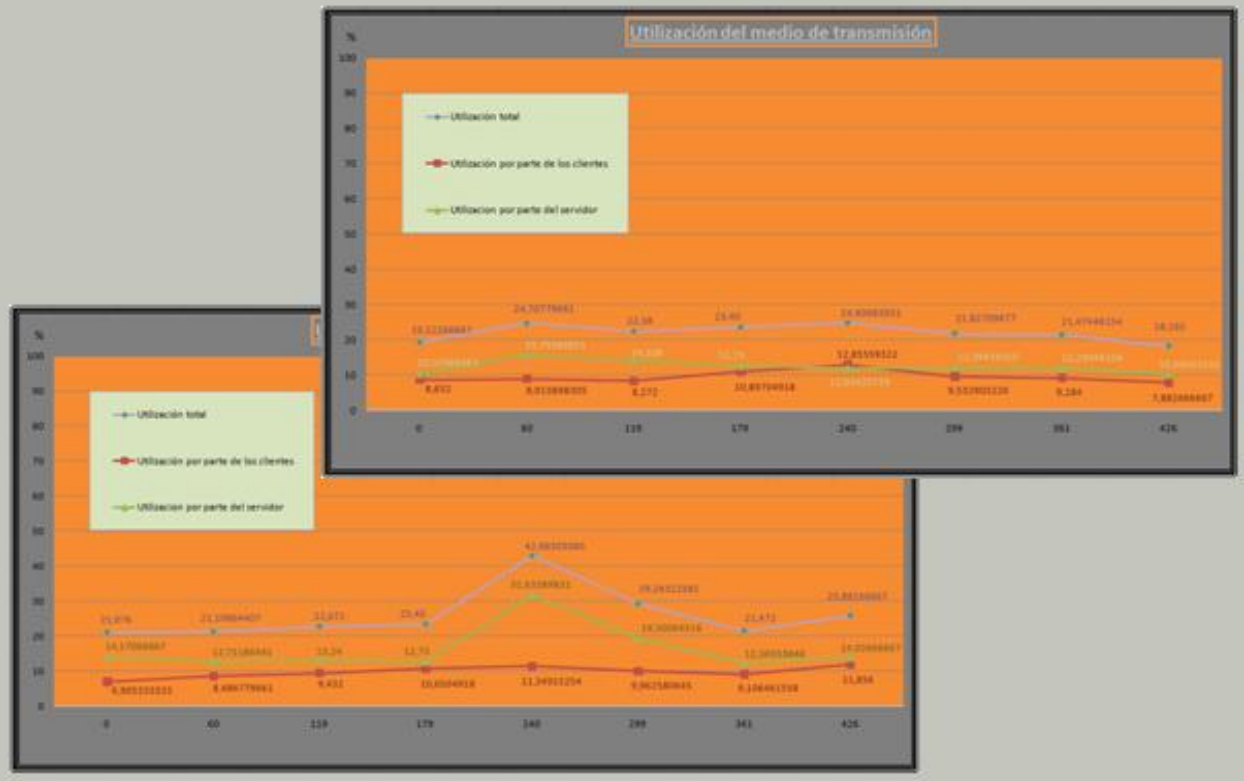
Resultados de las simulaciones

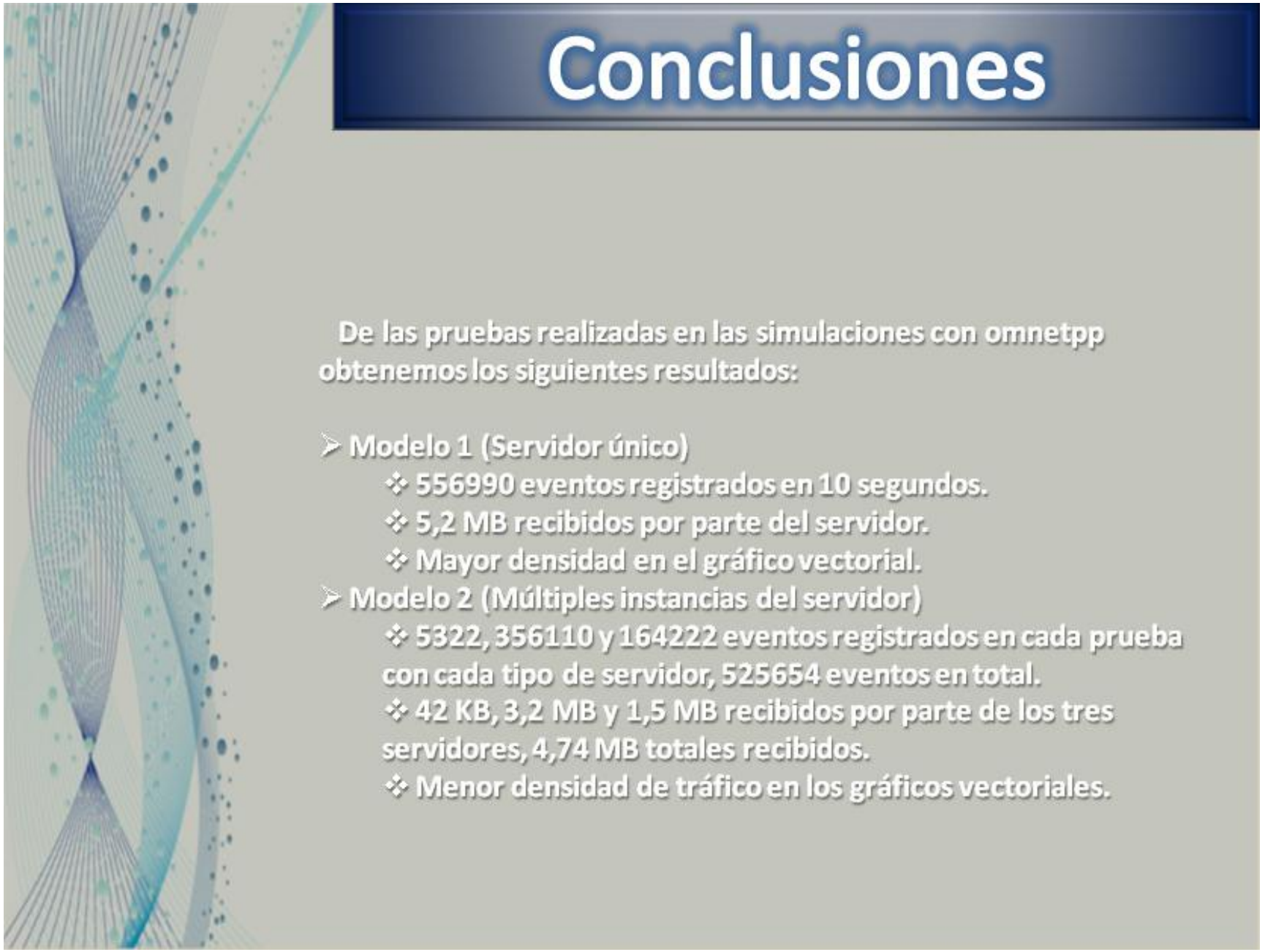
A partir de los eventos registrados creamos una línea del tiempo en la que los representamos gráficamente. Después también obtenemos las gráficas a partir de dos tipos de archivos de resultado, resultados escalares y resultados vectoriales



Resultados de las aplicaciones

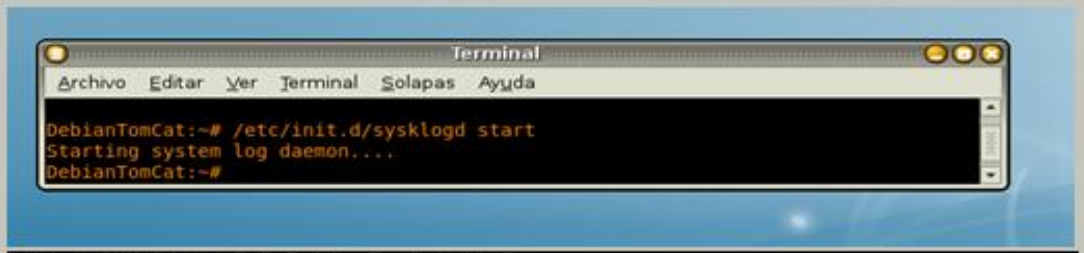
Por último se pueden observar las tablas con los datos obtenidos y las gráficas generadas a partir de éstos que serán comparadas.




Conclusiones
De las pruebas realizadas en las simulaciones con omnetpp obtenemos los siguientes resultados:
Modelo 1 (Servidor único)
556990 eventos registrados en 10 segundos.
5,2 MB recibidos por parte del servidor.
Mayor densidad en el gráfico vectorial.
Modelo 2 (Múltiples instancias del servidor)
5322, 356110 y 164222 eventos registrados en cada prueba con cada tipo de servidor, 525654 eventos en total.
42 KB, 3,2 MB y 1,5 MB recibidos por parte de los tres servidores, 4,74 MB totales recibidos.
Menor densidad de tráfico en los gráficos vectoriales.

Seguridad del servidor

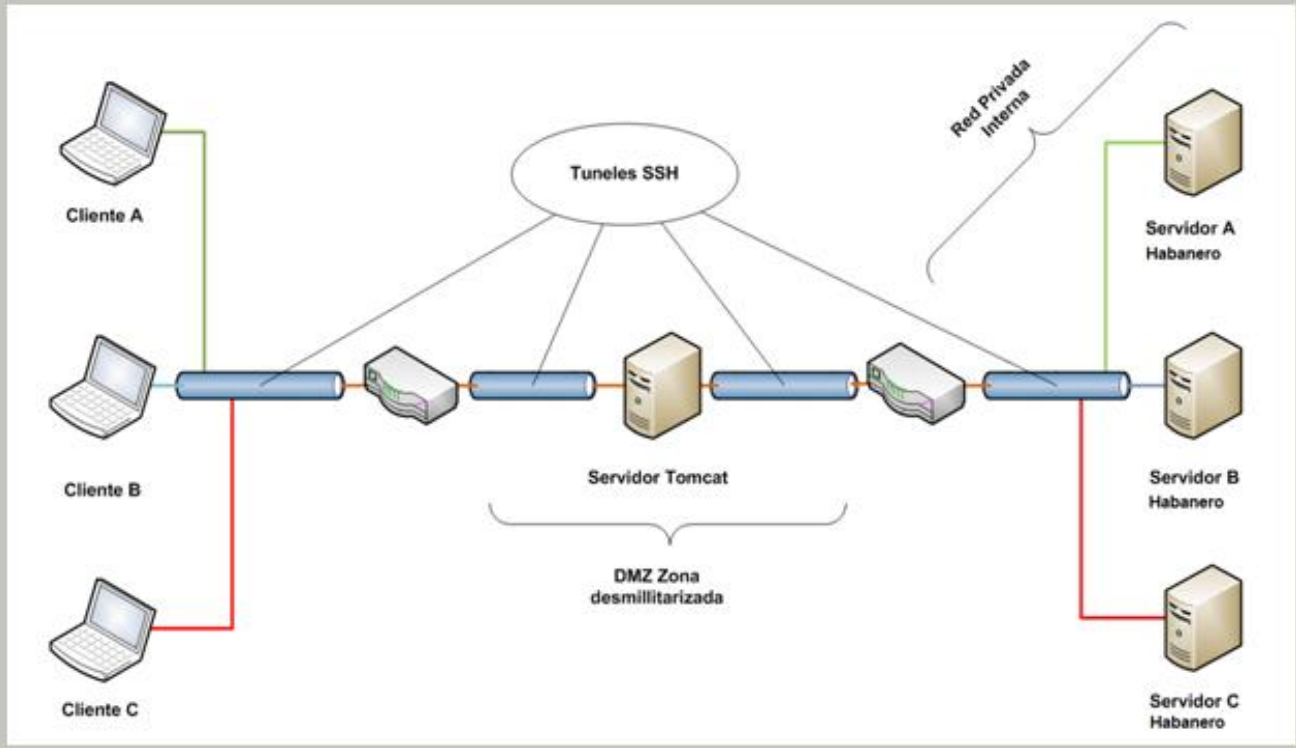
Es importante tener un registro de lo que está pasando en cada momento en nuestro sistema. Para ello disponemos de sysklog, que monitoriza las actividades y permite incluso emitir alarmas por correo electrónico.




```
Debian GNU/Linux 5.0 DebianTomCat tty2

DebianTomCat login: root
Password:
Linux DebianTomCat 2.6.26-2-amd64 #1 SMP Thu Nov 25 04:38:55 UTC 2010 x86_64

The programs included with the Debian GNU/Linux system are free software:
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
DebianTomCat:~# tail -f /var/log/syslog
Jan 11 20:17:12 DebianTomCat kernel: Kernel logging (proc) stopped.
Jan 11 20:17:15 DebianTomCat syslogd 1.5.0#5: restart.
Jan 11 20:22:21 DebianTomCat acpid: client 2520[0:0] has disconnected
Jan 11 20:22:21 DebianTomCat acpid: client connected from 2520[0:0]
Jan 11 20:23:45 DebianTomCat exiting on signal 15
Jan 11 20:23:47 DebianTomCat syslogd 1.5.0#5: restart.
Jan 11 20:23:52 DebianTomCat exiting on signal 15
Jan 11 20:24:01 DebianTomCat syslogd 1.5.0#5: restart.
Jan 11 20:26:32 DebianTomCat anacron[2587]: Job `cron.daily' terminated (mailing
 output)
Jan 11 20:26:32 DebianTomCat anacron[2587]: Normal exit (1 job run)
_
```

Encriptación de las conexiones

Se ha conseguido tener un sistema que podrá ofrecer la funcionalidad buscada y que a pesar de que la seguridad total no existe nos permitirá estar algo más tranquilos.




Habanero
debian
uburtu
WIRESHARK
OMNeT++ 4
eclipse
INET


Aplicaciones Lugares Sistema
jue 13 dic ene, 23:12 maki
FIN