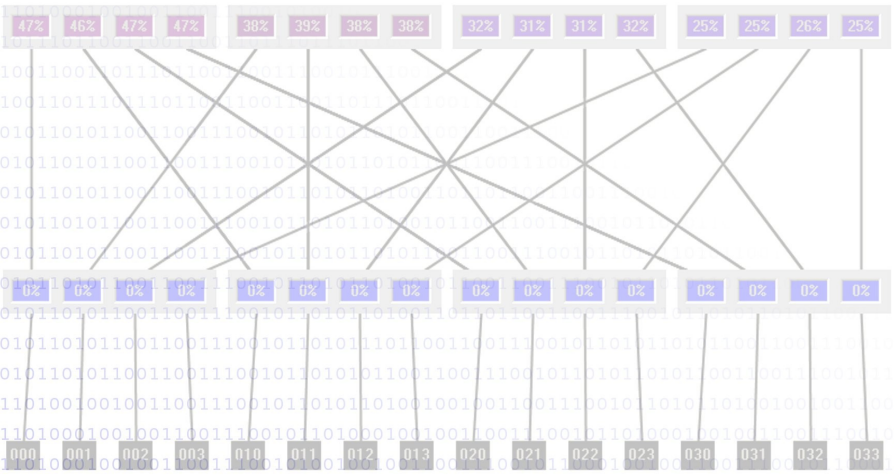
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

REPRESENTACION GRAFICA DEL FUNCIONAMIENTO DE REDES DE INTERCONEXION FAT-TREE



CÓDIGO P.F.C.: DISCA-171
Alumno: Xavier León Civera
Director: Marina Alonso Díaz
Codirector: Vicente Santonja Gisbert

Septiembre 2012

Índice

Índice Ilustraciones

Introducción

Introducción

Objetivo

El objetivo de este documento es exponer el trabajo desarrollado para la realización del proyecto final de carrera, que consiste en una aplicación de escritorio que ayudará a profesores de la Escuela Técnica Superior de Ingeniería Informática de la UPV en una de sus investigaciones sobre el ahorro de energía en las redes de interconexión Fat Tree.

A lo largo del documento se van a exponer conceptos sobre las redes de interconexión Fat Tree, y el trabajo de desarrollo de dicha aplicación, así como su funcionamiento .

Alcance

Entra en el alcance de este documento la introducción a las redes de interconexión Fat Tree para poder comprender las diferentes posibilidades del desarrollo de la aplicación, además de dar todas las nociones teóricas antes de que un desarrollador pase a leer el código.
El documento también contempla una explicación extensa del uso de la aplicación describiendo paso a paso su utilización.

No entra dentro del alcance un estudio profundo del comportamiento de dichas redes.

Audiencia

El presente documento va destinado a aquellas personas que estén interesadas en las arquitecturas de redes de interconexión, los usuarios de la aplicación o aquellas personas que quieran saber más de la aplicación tanto para ampliarla como para resolver algún problema similar.

La información expuesta acerca de la implementación de la aplicación requiere de conocimientos técnicos para su comprensión, no es así para la comprensión de su uso ya que se ha pretendido hacer una guía fácil de interpretar para que el usuario pueda hacer uso de la aplicación de una forma rápida y sencilla.

Organización de este documento

La memoria se va a dividir en dos partes fundamentales.

- La primera parte enfocada a el análisis del problema y el desarrollo de la aplicación.
  En este apartado se explicará todo lo necesario para que el lector comprenda como se ha razonado y desarrollado las soluciones y los problemas que han ido apareciendo a lo largo del desarrollo de la aplicación.

- En la segunda parte el documento se desarrolla una guía para que el usuario aprenda el uso de la aplicación de forma rápida y sencilla.

Objetivos del proyecto

Con este Proyecto se pretende dar una respuesta a las necesidades de algunos profesores, que están investigando sobre distintas topologías de red en el departamento DISCA de la Escuela Técnica Superior de Ingeniería Informática de la UPV.

Este proyecto interpreta y representa la información facilitada por los ficheros de la salida producida por un simulador de redes con la topología *Fat-tree k ary-n tree*.

El simulador imita situaciones reales y el resultado es volcado en ficheros de texto que en la actualidad se analizan manualmente.

Los ficheros proporcionados por el simulador son:

- Fichero "KaryNtree.out" que posee la información de carga y estado de cada nodo, la carga media de la red durante la simulación, además de otros muchos datos que no son necesarios para el objetivo que persigue la aplicación.

- Fichero "OnOff.out" que contiene información acerca de la potencia consumida, el tráfico y latencia de paquetes.

El objetivo es agilizar el trabajo de análisis del comportamiento de la red gracias a la utilización de esquemas y gráficas.

Para ello se ha desarrollado una aplicación gráfica que representa e interpreta de forma sencilla, la información comentada anteriormente generada por el simulador de redes.

Concretamente la red que se va a analizar tiene una topología *Fat-tree* con dos configuraciones:
4-ary 2-tree y *4-ary 3-tree*.

La aplicación consta de un entorno gráfico que represente estas topologías, describiendo los diferentes nodos, *switches* y los enlaces que contiene el sistema concreto.

La información de la red a mostrar es el estado de los enlaces de subida, los enlaces de bajada, y las gráficas con las estadísticas de la actividad global de la simulación.

La aplicación es capaz de reproducir toda la actividad registrada por el simulador, y representarla mediante una animación gráfica, facilitando el avance y retroceso en el tiempo.

Así mismo, el avance y retroceso se puede hacer a diferentes velocidades. La aplicación también es capaz de parar o retomar la reproducción de la simulación para poder analizar un momento concreto antes de seguir observando el comportamiento de la red.

Se representan los puertos y los enlaces haciendo visible el estado y utilización de cada uno.

Para controlar la reproducción, la aplicación tiene una barra de reproducción como la de los reproductores de multimedia que permite al usuario avanzar o retroceder en el tiempo.
Esta barra contiene una gráfica de tráfico generado durante la simulación, de modo que se nos

permite ver en todo momento la relación entre el instante de reproducción y la información que estemos visualizando.

Software utilizado

Los programas que se van a utilizar se han obtenido en su mayoría de la plataforma PoliDotNet.

PoliDotNet es una plataforma que ofrece a los alumnos de la Escuela Técnica Superior de Informática de la UPV licencias para poder utilizar en cada curso las herramientas de Microsoft.

El desarrollo del proyecto estará basado en un Sistema operativo Windows XP donde la herramienta para aplicación sera el IDE de Microsoft VisualStudio 2008 (ambos productos obtenidos de PoliDotNet).

Para desarrollo del resto del proyecto se utilizarán herramientas con licencia GPL.
Para la memoria se utilizará la aplicación OpenOffice como editor de textos, la herramienta GIMP para la manipulación de imágenes, y la herramienta Argo para los diagramas UML.

Tecnología utilizada

El proyecto se va a desarrollar con el lenguaje de programación C# .

La elección de la tecnología sobre la que se va a desarrollar el proyecto se ha basado principalmente en los lenguajes de aplicación estudiados durante la carrera y las herramientas disponibles para cada lenguaje, además se ha tenido en cuenta la plataforma sobre la que correrá la aplicación: Windows XP o superior con .NET 3.5.

Requisitos mínimos

Los requisitos mínimos para el uso de la aplicación son tener una maquina Pentium II o superior con 500MB de memoria RAM y Windows XP con .NET 3.5.

Red Fat Tree

¿Qué es una red Fat tree?

Las redes Fat Tree emplean una topología en forma de árbol.

Las topologías en árbol adolecen de un inconveniente: la raíz puede concentrar el tráfico procedente de muchos nodos, por lo que en ella puede producirse un cuello de botella.

Una modificación es lo que se llama una estructura de árbol grueso en que los nodos interiores son switches, y no elementos de proceso, que se comportan como pequeñas centrales telefónicas.

La ventaja de esta topología, sobre el árbol convencional, radica en que los conmutadores están diseñados para admitir varios mensajes simultáneamente procesando un mayor número de mensajes.

Esto elimina el inconveniente de los árboles, ya que el número de enlaces se mantiene constante según nos acercamos a la raíz. Este tipo de red, ya que mantiene la forma de árbol, esta es una red de interconexión estática porque los conmutadores mantienen fijos los enlaces entre los nodos .

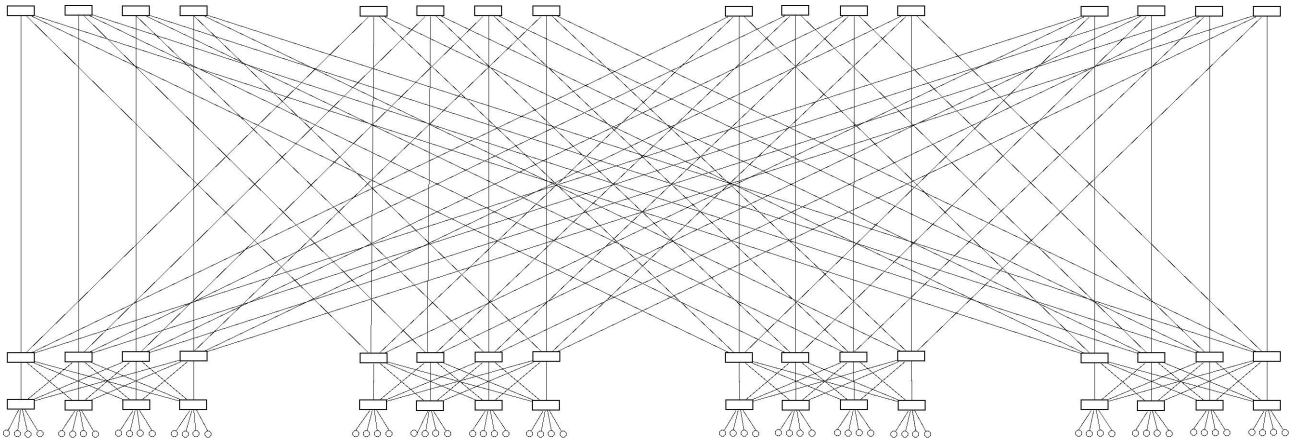


Figura 1: Esquema Fat Tree

¿Cómo está estructurada una red Fat Tree?

Las redes Fat Tree tiene una estructura **k**-ary **n**-tree siendo **k** la aridad y **n** el número de niveles que tiene la red.

Las topologías de redes Fat-tree (**k**-ary **n**-tree) están formadas por dos tipos de vértices: nodos de procesamiento($N = \mathbf{k}^{\mathbf{n}}$) y switches ($S=\mathbf{n}\mathbf{k}^{\mathbf{n}-1}$) de aridad **k**.

En la imagen siguiente vemos una red Fat-tree 4-ary 3-tree, donde se puede apreciar las dimensiones de las redes y la estructura general; de arriba a abajo de la imagen se aprecian 3 niveles de switches (enmarcados en rojo) y 1 nivel de nodos nodos de procesamiento (enmarcados en azul).

Los niveles están numerados de arriba a abajo desde 0 hasta **n**-1.

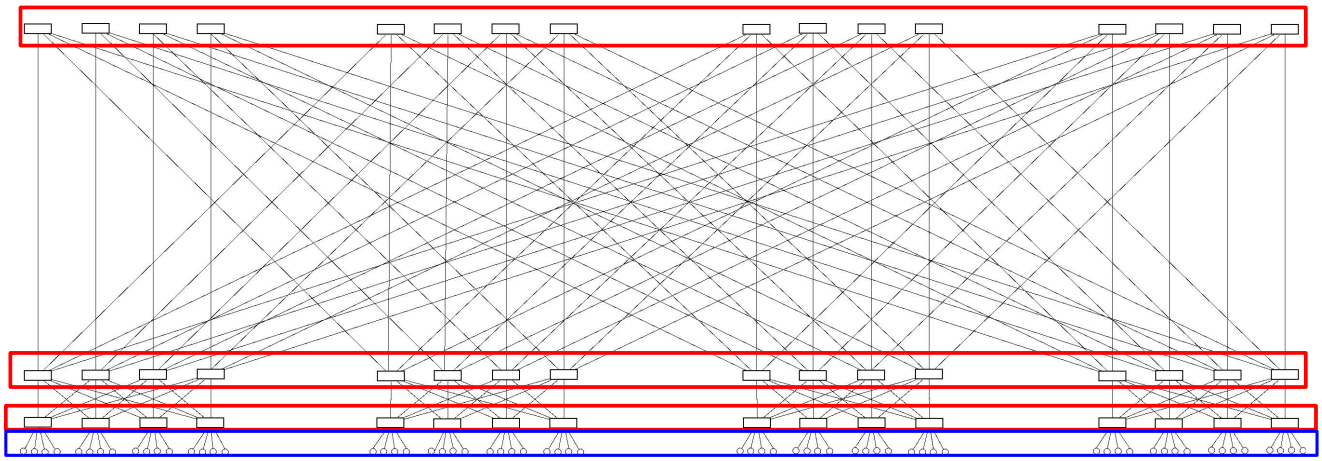


Figura 2: Esquema Fat Tree niveles

Este tipo de red incorpora un mecanismo de reducción de consumo, esto se consigue mediante la conexión y desconexión de enlaces.

Cuando la carga de la red es baja se van deshabilitando enlaces para reducir el consumo, los enlaces se volverán a activar en los switches que lo necesiten y a medida que el tráfico de la red aumente .

Suponiendo que el tráfico de la red fuese casi nulo, se deshabilitarían todos los enlaces a excepción de los que pertenecen al árbol mínimo.

El árbol mínimo es aquel que asegura que todos los nodos de procesamiento van a estar conectados entre si siempre.

A continuación podemos ver una imagen de la red cuando solo está habilitado el árbol mínimo.

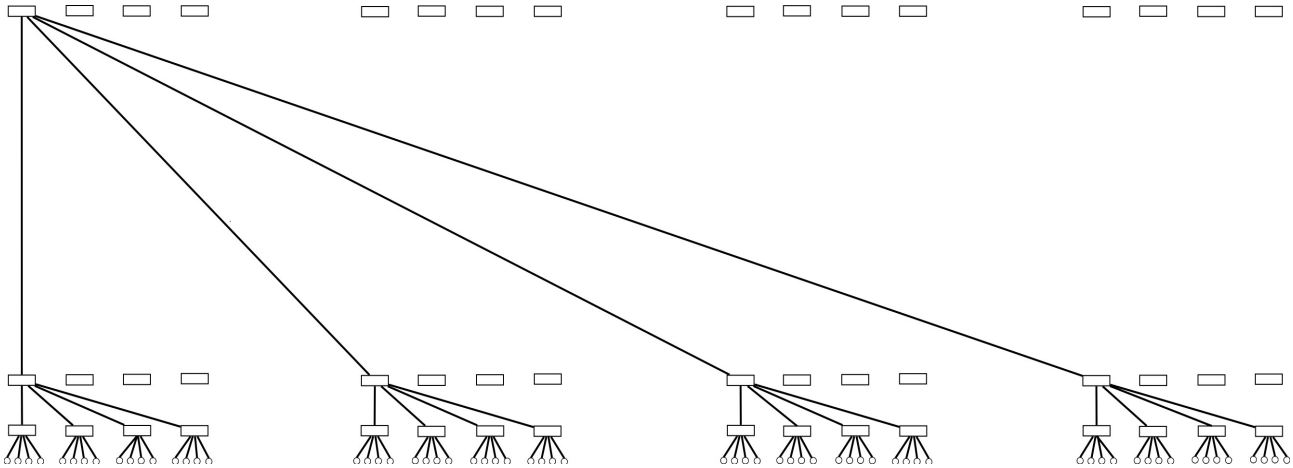


Figura 3: Diagrama árbol mínimo

Los enlaces tiene cuatro estados de funcionamiento, pueden estar habilitados (**On**), deshabilitados (**Off**), pero también existen dos estados intermedios que indican si el enlace esta siendo habilitado (**TurningOn**) o deshabilitado (**TurningOff**).

Guía de Desarrollo

Desarrollo de la aplicación

Estrategia de resolución del problema

Se ha dividido el funcionamiento de la aplicación en dos partes bien diferenciadas.

- **Tratamiento y análisis de datos del simulador**

  Esta parte se encargará de leer y analizar los archivos de datos del simulador, comprobando si la información es la correcta y guardando los datos en las colecciones de objetos correspondientes, de forma óptima para el funcionamiento de la aplicación.

  - El funcionamiento específico de la arquitectura es el que se encarga de leer los ficheros de datos proporcionados por la salida del simulador y de toda la información que contiene extrae solo la relevante para el análisis.

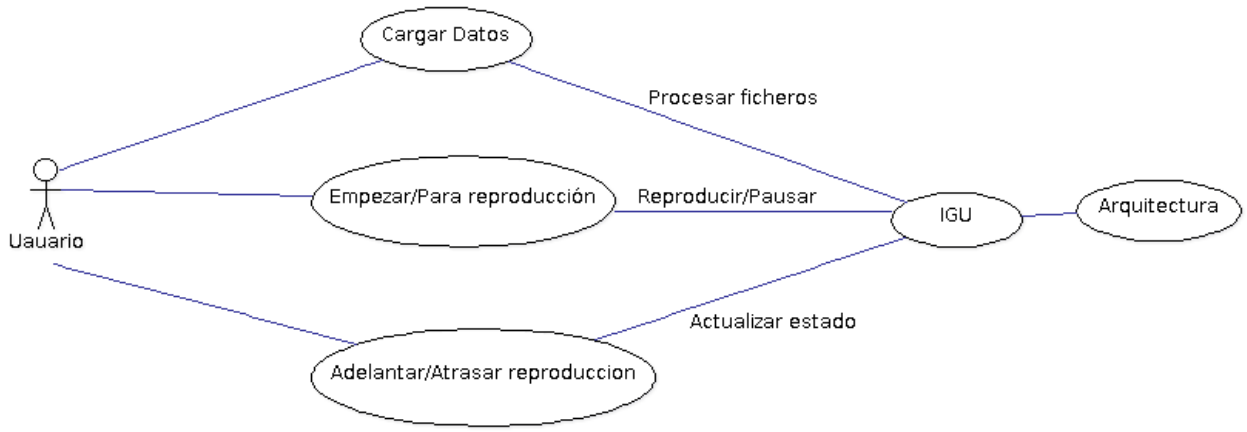
- **Interfaz gráfico (IGU)**

  Con el que interactuará el usuario, representa la información almacenada en la arquitectura de forma sencilla y visual para que el usuario la comprenda.

  El interfaz gráfico consulta cuál es la información a representar para un instante de tiempo y un elemento concreto actualizando la información que se muestra al usuario.

De esta forma se asegura un funcionamiento independiente entre el interfaz gráfico y la arquitectura permitiendo así la posible inclusión de más arquitecturas de red en el futuro.

En el siguiente esquema podemos ver cómo interactúan las partes de la aplicación entre sí y con el usuario.



llustration 4: Diagrama de usos

Tratamiento y análisis de datos del simulador

Los datos están ubicados en dos ficheros de texto que son generados por el simulador, nuestra aplicación no calcula ningún dato, toda la información mostrada es obtenida de los ficheros.

- Fichero "***KaryNtree.out***" que contiene la carga de los switches para cada puerto en cada instante de tiempo de la simulación, además posee el promedio de carga de toda la arquitectura en cada instante de tiempo.

- Fichero "***OnOff.out***" que contiene información acerca de la potencia consumida el tráfico generado y la latencia de los paquetes. Esta información es la empleada para realizar las gráficas.

Los ficheros contienen más información aparte de la especificada anteriormente, pero se omite la información que no es necesaria para representar lo exigido.

Descripción de la arquitectura

Las redes Fat Tree tiene una estructura **k**-ary **n**-tree siendo **k** la aridad y **n** el número de niveles que tiene la red.

En las especificaciones de los nodos para la arquitectura Fat-Tree se han establecidos los parámetros grupo, columna, y nivel.

Grupo se refiere al índice **g** $\in \{0,1,...,k^{n-2}-1\}$ para $0 \leq g \leq n-1$.

Columna se refiere al índice **c** $\in \{0,1,...,k-1\}$.

Nivel al que pertenece un switch, para **l** $\in \{0,1,...,n-1\}$.

Esto se ha hecho para poder identificar y conectar los switches entre si más fácilmente.

En la figura siguiente se muestra una red Fat-Tree 4-ary 3-tree, se puede aprecian grupos verticales de switches (rectángulos rojos) que contienen 3 niveles de switches con 4 switches en cada nivel, cada uno de esos switches son las columnas (rectángulos azules) definidas anteriormente (se identifican 4 columnas dentro de cada grupo).

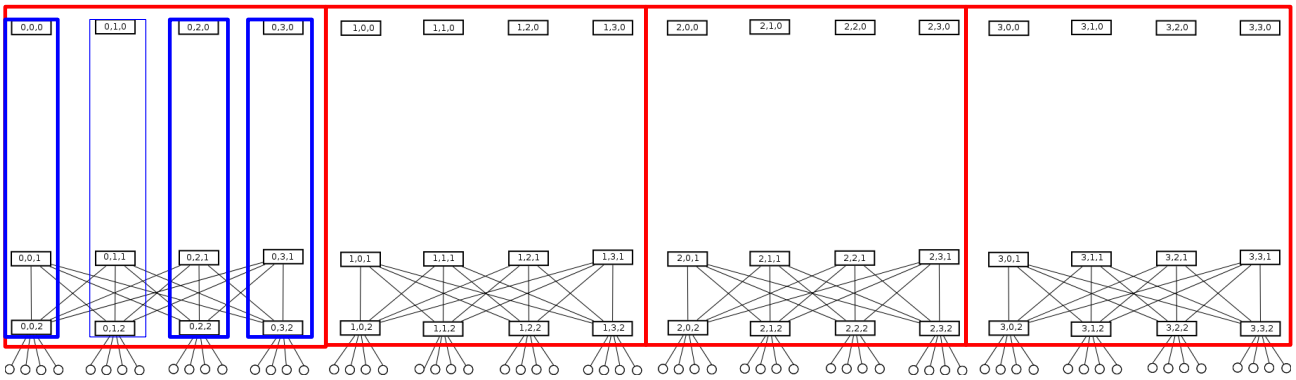


Figura 5: Esquema ordenación Fat Tree

Los switches son numerados de arriba hacia bajo y de izquierda a derecha.

De este modo los switches se nombran por la siguiente expresión Switch (**g**, **c**, **l**) de modo que el primer switch será el (0,0,0), el segundo el (0, 0, 1), (0, 0, 2), (0, 1, 0) …

Además de por la expresión anterior los switches se identifican por un número identificador **ID** $\in \{0,..., k^n-1\}$.

A partir de la expresión Switch (**g**, **c**, **l**), se puede deducir su identificador gracias a la siguiente expresión

ID = (**g** * k * n) + (**c** * n) + n.

Especificación

En el diagrama de clases que se presenta a continuación vamos a ver una explicación simplificada de como se han relacionado los distintos elementos de la red Fat-Tree.

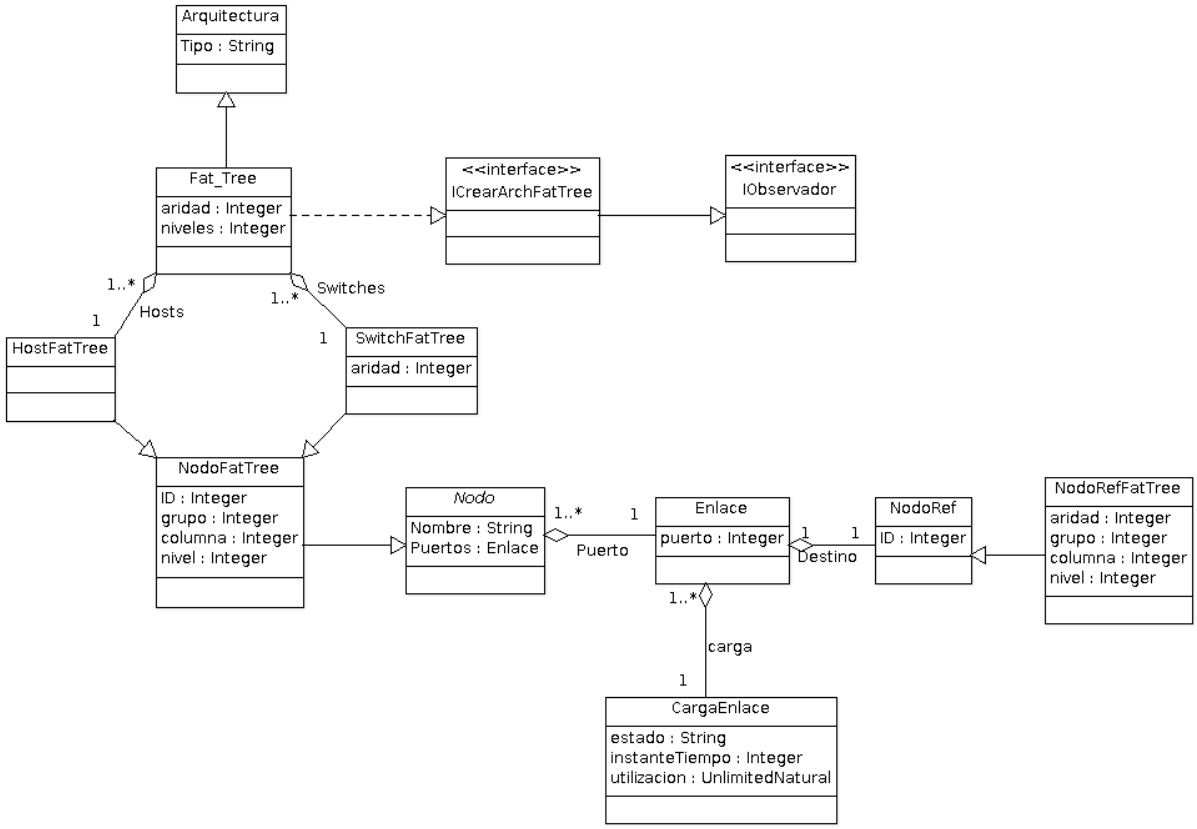


Figura 6: Diagrama de clases fat-Tree

En el diagrama se puede ver 3 elementos principales.

El principal es la arquitectura de tipo Fat-Tree que contiene una colección de Switches y otra de Hosts (se ha llamado hosts a los nodos de procesamiento)

Los switches están organizados en un diccionario ya que será una colección grande en la que se desea un acceso no secuencial.

Los hosts están almacenados en una lista ya que se consulta menos veces con acceso secuencial.

Clases Genéricas

Para facilitar el trabajo futuro se han creado algunos elementos que van a proporcionar atributos comunes a todas las redes unificando el comportamiento, y ahorrando tiempo y trabajo en el futuro.

Estas clases genéricas son:

- **Arquitectura**, esta clase permite que todas las define la colección de switches y lo establece como el elemento sujeto del patrón de diseño *Observer* empleado para comunicar la arquitectura con el IGU
- **Nodo**, define las propiedades de cualquier nodo de la red.
- **NodoRef**, define una referencia de un nodo para así evitar hacer copias continuas de un elemento
- **Enlace**, define los atributos de los enlaces de una red (Nodo destino, puerto origen, y carga).
- **CargaEnlace**, define una clase que contiene el estado y la carga de un puerto.

Clases arquitectura FatTree

Para la arquitectura Fat Tree se han definido las clases con los atributos y característica especificas de esta arquitectura.

- **NodoFatTree**, define los atributos específicos de esta arquitectura como son el nombre del nodo que en esta arquitectura tiene características especiales.
- **NodorefFatTree**, la diferencia con su clase genérica esta clase facilita mas información del nodo.

Descripción del entorno gráfico

El interfaz gráfico se ha planteado como una capa entre el usuario y los datos.

Esta permitirá al usuario cargar los ficheros de datos iniciar o pausar la reproducción, adelantar o atrasar en el tiempo, y aumentar o reducir la velocidad de reproducción.

La aplicación va a constar de un entorno gráfico que represente las topologías Fat Tree 4-ary 2-tree y 4-ary 3-tree, describiendo los diferentes nodos, switches y enlaces que contiene el sistema concreto, y gráficas con la información de la simulación.

La información se mostrará en 3 pestañas dentro de una ventana principal.
En cada una de estas pestañas se representa el estado de los enlaces de subida, los enlaces de bajada, y las gráficas con las estadísticas de la actividad global de la simulación.

Esta aplicación se plantea como un reproductor gráfico de la actividad registrada por el simulador y almacenada en sus ficheros de salida, estos ficheros son la fuente de información para representar la evolución de la red.

La aplicación es capaz de reproducir toda la actividad registrada por el simulador, y representarla mediante una animación gráfica, facilitando el avance y retroceso en el tiempo.

Así mismo, el avance y retroceso se puede hacer a diferentes velocidades, también es capaz de parar o retomar la reproducción de la simulación para poder analizar un momento concreto antes de seguir observando el comportamiento de la red.

La aplicación se iniciará sobre la pestaña de estadísticas que representa un resumen de lo ocurrido a lo largo de toda la simulación. De este modo el usuario puede ver en primera instancia todos los datos de la simulación, pudiendo observar a un primer golpe de vista si hay información especialmente interesante.
Las estadísticas representadas son:

- La potencia consumida en cada instante de la simulación.
- Promedio de carga de toda la red en cada instante de simulación.
- Latencia de los paquetes generados.

La información de las gráficas está toda en los ficheros, nuestra aplicación no hace ningún cálculo se limita a interpretar los datos proporcionados por el simulador.

En la pestaña de subida y en la pestaña de bajada se mostrarán los puertos y los enlaces igual que en los esquemas de la red mostrados anteriormente.

Los puertos activos se representarán con una gama de colores de verde a rojo según la utilización del enlace: fluido o saturado, respectivamente. Los puertos no activos se serán representados en un tono gris. Además de los enlaces también se verá representados con una línea de color negra gruesa para los enlaces activos, y con una línea gris de menor grosor para los enlaces no activos.

Para indicar si un switch está deshabilitado todos sus puertos se dibujarán de color gris.

Se representará el consumo medio de los enlaces, su utilización media , y la latencia de los paquetes Para controlar la reproducción la aplicación tendrá una barra de reproducción como la de los reproductores de vídeo que permite al usuario avanzar o retroceder en el tiempo.

Esta barra contiene una gráfica de tráfico generado durante la simulación, de modo que se nos permite ver en todo momento la relación entre el instante de reproducción y la información que estemos visualizando en la pestaña.

Especificación

A continuación se presenta un diagrama en el que se puede ver como se ha estructurado el diseño del interfaz gráfico estableciendo un paralelismo en las estructuras gráfica e interna, para que el interfaz gráfico simplemente sea una capa que represente los datos de la red.

Cada uno de los elementos mostrados en el diagrama solo contendrá los datos del instante de tiempo en el que se encuentra y una referencia del objeto de la red al que representa.

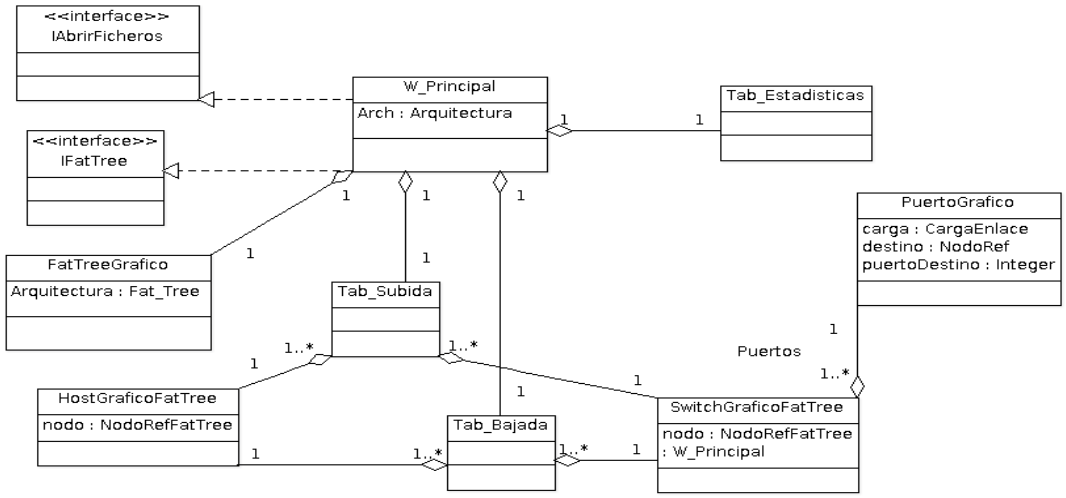


Figura 7: Diagrama de clases IGU

La ventana principal de la aplicación "***W_Principal***" tiene que trabajar con controles de ventana y los agrega en el contenedor que le corresponde.

El encargado de crear los controles y enviárselos a la ventana principal para que los agregue es la clase "***FatTreeGrafico***". Esto se ha hecho así para simplificar el código permitiendo la inclusión de otras arquitecturas independientemente de la arquitectura FatTree.

A continuación se va hacer una breve descripción de los elementos empleados para representar la información.

- La ventana se ha dividido en dos paneles principales, uno para las pestañas con la información y otro para los elementos de reproducción.

- El elemento de pestañas es el control "***TabPages***" *que nos permite tener tantas pestañas como queramos especificándolo en su propiedad "****TabPages****"*.

- Para representar las gráficas se han utilizado el elemento "***Chart***" que pertenece a la librería de elementos de VisualBasic.

- Para los Switches se han creados controles personalizados llamados "***SwitchGraficoFatTree***" basados en el control "***GroupBox***" el cual incluye propiedades como una referencia al nodo al que hace referencia y una colección de puertos controles "***PuertoGrafico***" basados en "***Labels***".

- Los "***PuertoGrafico***" que tienen una referencia al nodo destino y la carga del puerto al que refiere.
- Los nodos de procesamiento están realizados con el control "***HostGraficoFatTree***" basados en un "***GroupBox***".

- Los enlaces están implementados mediante el elemento de VisualBasic "***LineShape***" que son lineas dibujadas punto a punto con las coordenadas de los controles "***PuertoGrafico***".

Los elementos representados en la IGU son elementos del tipo "*Control*" C#.

Los controles están organizados como un diccionario dentro del "*Control*" que los contiene y para acceder a ello hay que invocarlos con los mecanismos del lenguaje refiriéndose como clave con

- ***Switches*** , **ID** del switch
- ***Hosts***, identificados con su **nombre**.
- ***Enlaces***, *identificados* con el **ID y puerto** del nodo origen. Por ejemplo el nodo 0 puerto 1 sería su clave sería la 01.

Todo lo referente al control de elementos gráfico se encuentra en la clase "***FatTreeGrafico"*** ya que la ventana lo único que hace es coger los controles que esta clase le proporciona dibujando los en la ventana de la aplicación.

Carga de la información

La información requerida por la aplicación para poder representar la simulación son dos ficheros de texto con la extensión .out creados por el simulador.

La aplicación recorrerá los dos ficheros extrayendo la información necesaria y pasándosela a la arquitectura la cual se encargara de crear los objetos necesarios, con la información contenida en los ficheros.

Al mismo tiempo que la arquitectura crea los hosts y switches ordena al interfaz gráfico que los construya y muestre.

En el siguiente diagrama se puede ver como se comunican la arquitectura con el interfaz gráfico para cargar los datos.

Para realizar esta unión se ha utilizado el modelo de diseño *Observer* que define una dependencia uno a muchos entre objetos de forma que cuando uno cambie se notifique automáticamente a todos los que de el dependen.

De esta forma vemos que el elemento central *Sujeto* del que desciende *Arquitectura* provocará que cuando haya algún cambio en ella se notificará a la IGU.

De la misma forma se ha comunicado la *Arquitectura* con los clases que cargan los datos para que estas le envíen toda la información actualizando la conforme leen los ficheros de datos.

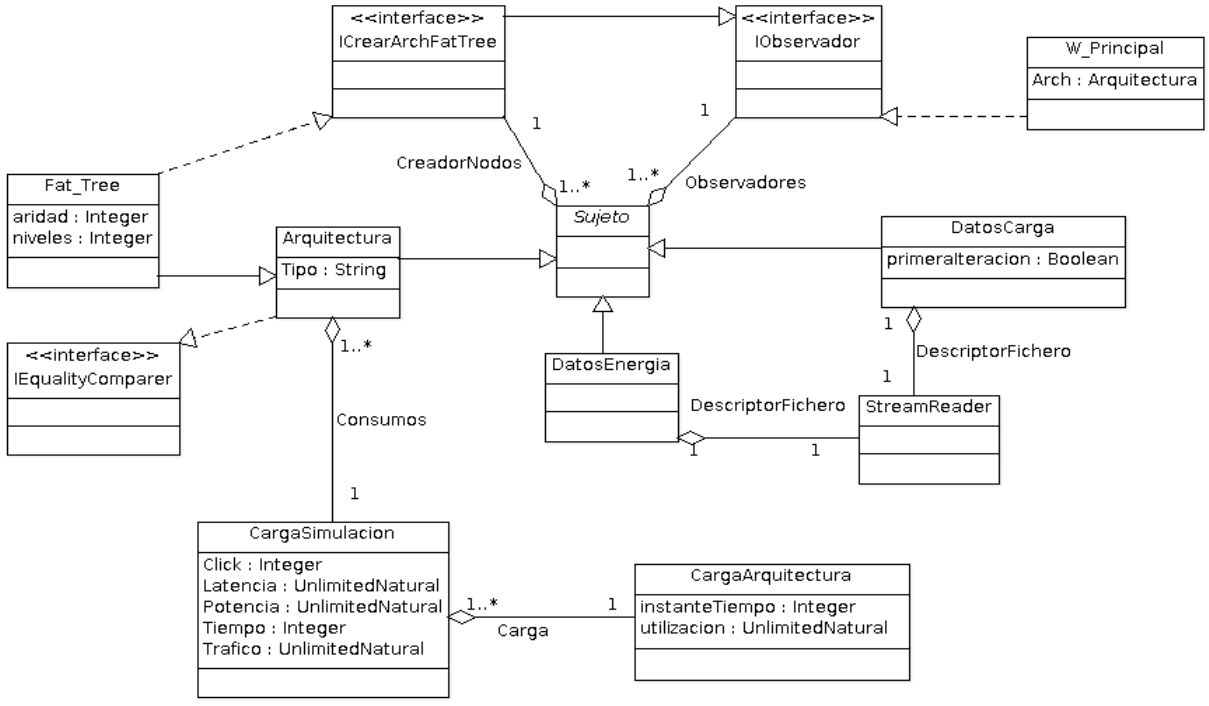


Figura 8: Diagrama clases carga ficheros

Cuando la aplicación tiene los archivos de datos se procesa en primer lugar el fichero que contiene los datos referentes a la carga de los nodos de la red, y después procede a leer los datos del fichero que tiene los datos de consumo tráfico de la red etc...

Mientras la aplicación lee el fichero *KaryNtree.out* que contiene la carga de los nodos se procede a crear los nodos y asignarles sus correspondientes cargas y estados para cada instante de tiempo de simulación.

Al procesar las líneas del fichero se tienen en cuenta las líneas que definen el número de nodos de procesamiento de datos para comprobar que la arquitectura seleccionada coincide con el fichero introducido para leer, si no coincide devuelve un mensaje de error y no carga los datos.

Si el fichero es el correcto se prosigue leyendo las lineas que empiezan con el símbolo “@”, que son las líneas que contienen la información que nos interesa.

La primera vez que lee las cargas de cada nodo, se crea el nodo correspondiente añadiéndolo al diccionario donde se van a acumular los switches, y definiendo las cargas que tiene cada puerto del switch para el instante inicial, de este modo las siguientes cargas que se leen simplemente se agregan a una lista de cargas que tiene el puerto correspondiente.

En el instante en que el switch se crea y se almacena en el diccionario se crea su representación en el interfaz gráfico asociándole a este una referencia del nodo al cual representa.

Si el switch es del último nivel se crean y asocian los hosts con los que estará conectado, y estos se almacenarán en una lista donde se contendrán todos los hosts.

De esta forma cuando se terminan de procesar la información de los ficheros todos los elementos de la red ya están creados relacionados y representados, y se puede proceder a la reproducción del simulador.

Lectura del fichero KaryNtree.out

Para procesar este fichero la aplicación recorre todas las lineas del fichero comprobando si cumple alguna de las siguientes características:

- La Línea contiene "Procesadores :". Esto indica el número de nodo de procesamiento de datos, comprobando con el número de nodos que teóricamente debería contener la arquitectura seleccionada obtenemos si el fichero se corresponde con el esperado.
- "@Utilizacion de enlaces" a partir de aquí empieza a cargar la utilización para cada enlace.
- "@Utilizacion Media:" nos ofrece el promedio de carga para un instante de tiempo predeterminado.

Si durante la lectura del fichero se produce algún error ya sea de lectura, creando o dibujando algún elemento saltará una excepción que interrumpirá la carga y mostrara un mensaje de error al usuario.

Modificación del fichero KaryNtree.out

Si en el futuro se desea cambiar este fichero de salida, se pueden eliminar o añadir todas aquellas

líneas que no empiecen con el carácter “@”, ya que estas son las que lee nuestra aplicación.

La línea que hace referencia al número de procesadores no es crítico pero convendría dejarla como está ya que es la encargada de comprobar que el fichero se corresponde con la arquitectura seleccionada.

Si fuese necesario aplicar algún cambio que afecte alguna de las líneas procesadas por esta aplicación recomiendo leer el fichero “*DatosCarga.cs*” para saber como alterar el fichero de salida o actualizar el programa.

Lectura del fichero OnOff.out

La lectura de este fichero es mucho más sencilla.

Para este fichero se procesan todas las líneas guardando solo los datos que necesitamos.

Modificación del fichero OnOff.out

Ya que en este fichero se leen todas las líneas se recomienda no hacer ningún cambio en este fichero.

Si fuese necesario aplicar algún cambio recomiendo leer el fichero “*DatosEnergia.cs*” para saber como alterar el fichero de salida o actualizar el programa.

Representación de la información

La información a representar se ha dividido en los mismos apartados que en el interfaz de usuario, información referente a la carga general de la simulación e información específica de cada nodo.

En la información general tenemos los datos referentes a la carga de la red, latencia de paquetes, tráfico generado y promedio de carga de la red.

Estos datos se almacenan en una clase llamada “**CargaSimulacion**” y se emplean para hacer las gráficas de la simulación.

En los datos específicos de cada nodo se almacena su información identificadora como el nombre compuesto por el grupo, columna y nivel al que pertenece, y un identificador (que refiere el orden de creación). Además contiene una colección con los puertos en los cuales se refleja los enlaces y la carga de estos en cada instante de tiempo.

Para almacenar estos datos se han creado dos clases:

- “**Enlace**” contiene una referencia al nodo y puerto destino, además de la carga (lista de *CargaEnlace*) de ese enlace en cada instante de simulación.
- “**CargaEnlace**” que contiene la carga y el estado del enlace para un instante de tiempo.

Es importante entender la información que se plantea a continuación ya que representa las principales colecciones de datos de la aplicación. Esta información ayuda a comprender como están organizados y como se accede a los estados de cada enlace cuando se actualizan durante la

ejecución de la aplicación.

A continuación se presenta como se han organizado las principales colecciones de datos en las cases que las contienen.

Arquitectura:

Diccionario<Switch> Tiene tantos elementos como Switches hay en la topología.

Lista<Host> Tiene tantos elementos como Hosts hay en la topología.

Lista<CargaSimulacion> Tantos elementos como instantes de tiempo de simulación.

Switch:

Lista<Enlace> Habrá tantos elementos como puertos tiene el Switch

Enlace:

Lista<CargaEnlace> Habrá tantos elementos como instantes de tiempo de simulación

Las colecciones planteadas anteriormente se contiene en la parte de la arquitectura no en la IGU.

Con este planteamiento cuando es necesario representar o actualizar tanto un enlace como el puerto que sujeta la carga simplemente hay que pasarle la referencia del nodo y el instante de tiempo del que deseamos obtener la información y la arquitectura responderá con la información deseada para que la IGU la pueda representar.

Organización de los directorios

A continuación se presenta la imagen de la estructura del proyecto.

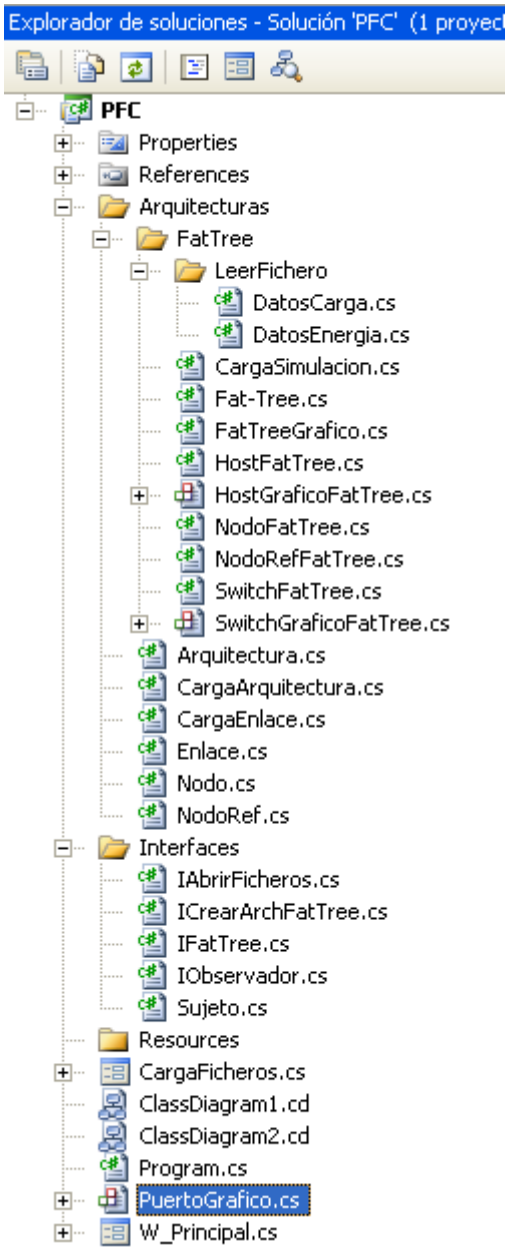


Figura 9: Captura estructura directorios

Se pueden observar diferentes niveles de directorios con elementos que están organizados de más genéricos a más específicos.

En el primer nivel tenemos los ficheros genéricos de la solución con la ventana principal y la de carga de ficheros.

En este primer nivel observamos las carpetas de Interfaces y Arquitecturas.

En la carpeta Interfaces están implementados los interfaces que facilitan la comunicación entre las distintas clases.

En la carpeta ***Arquitecturas*** tenemos los elementos comunes a todas las redes de distintas arquitecturas como son los nodos, los enlaces y sus cargas, Además está definida la carpeta *FatTree.*

En la carpeta ***FatTree*** están definidas las clases concretas para generar la arquitectura. Es decir las especificaciones con los cambios que hacen falta para implementar los Switches y hosts y la arquitectura *FatTree.*

Por último tenemos la carpeta ***LeerFichero*** donde hay dos clases la clase DatosCarga encargada de leer el fichero *KarNtree.out* y la clase *DatosEnergia encargada de leer los datos del fichero* OnOff.out

Ampliaciones futuras

Como se ha especificado durante el documento esta aplicación admite varias ampliaciones.

Una de ellas es la inclusión de nuevas arquitecturas como podrían ser la de toro o malla.

Ampliación de la arquitectura Fat Tree añadiendo otras topologías, por ejemplo: 2ary 2tree

Además es posible que con el tiempo de uso de la aplicación vayan saliendo nuevas necesidades que no se habían contemplado hasta la fecha, como podría ser incluir nuevas pestañas con otro tipo de información acerca de la simulación, etc...

Manual de Usuario

Manual de usuario

Introducción

En este apartado se ha realizado una guía fácil de seguir para que cualquier usuario pueda hacer uso de la aplicación.

En primer lugar se presenta el proceso de instalación de la aplicación, que aunque es un proceso sencillo y estándar, es conveniente explicar por si algún usuario tuviese alguna duda.

Se ha creado un fichero de instalación para poder distribuir la aplicación sin necesidad de compilarla, además de que es mucho más fácil hacer funcionar ya que si algunos de los requisitos necesarios para que la aplicación funcione no se encuentran instalados en el equipo el asistente de instalación se encarga de comunicarlo al usuario.

También se presenta el procedimiento de desinstalación de la aplicación por si algún usuario lo requiriese.

Además se presenta el uso y análisis de la aplicación para facilitar la comprensión de cada apartado a todos aquellos usuarios que no hayan formado parte del desarrollo de esta como ha sido el caso de los directores del proyecto, los cuales han estado directamente implicados en el desarrollo del Interfaz Gráfico de Usuario (IGU).

La aplicación es una interfaz gráfica para representar la información generada por un simulador de redes Fat-Tree.

La idea es poder analizar los datos de la simulación con mayor facilidad, mostrando estadísticas de consumo de potencia, tráfico generado, latencia de los paquetes, y la carga media de la red. Además se mostrará la carga y estado de cada enlace durante la simulación.

Por supuesto el uso de la aplicación va enfocado a gente que entienda el funcionamiento de las arquitecturas de red que comprende la aplicación.

Requisitos mínimos

Los requisitos mínimos para el uso de la aplicación son tener una maquina Pentium II o superior con 500MB de memoria RAM y Windows XP con .NET 3.5.

Instalación

A continuación se detallan los pasos a seguir para la instalación de la aplicación.

Se recomienda hacer una instalación estándar como la que se presenta a continuación.

Para instalar la aplicación se puede hacer desde el fichero “*Setup.msi”* o “*Setup.exe”*, haciendo doble click sobre cualquiera de los dos. A continuación el S.O. solicita confirmación para ejecutar la aplicación, a la que obviamente debemos permitir.

Una vez autorizada la ejecución del instalador se abrirá una ventana de bienvenida como la que se presenta a continuación:

Figura 10: Bienvenida instalación

Dando a siguiente damos paso a la siguiente fase de la instalación.

En esta fase se define la carpeta donde se desea instalar la aplicación, además de especificar si se desea permitir el uso por todos los usuarios del equipo o sólo por el usuario que lo instala.

En este apartado se recomienda dejar los parámetros por defecto, aunque si hay más de un usuario en el equipo puede ser bueno permitir el acceso a todos ellos.

A continuación se presenta la imagen con el ejemplo estándar.

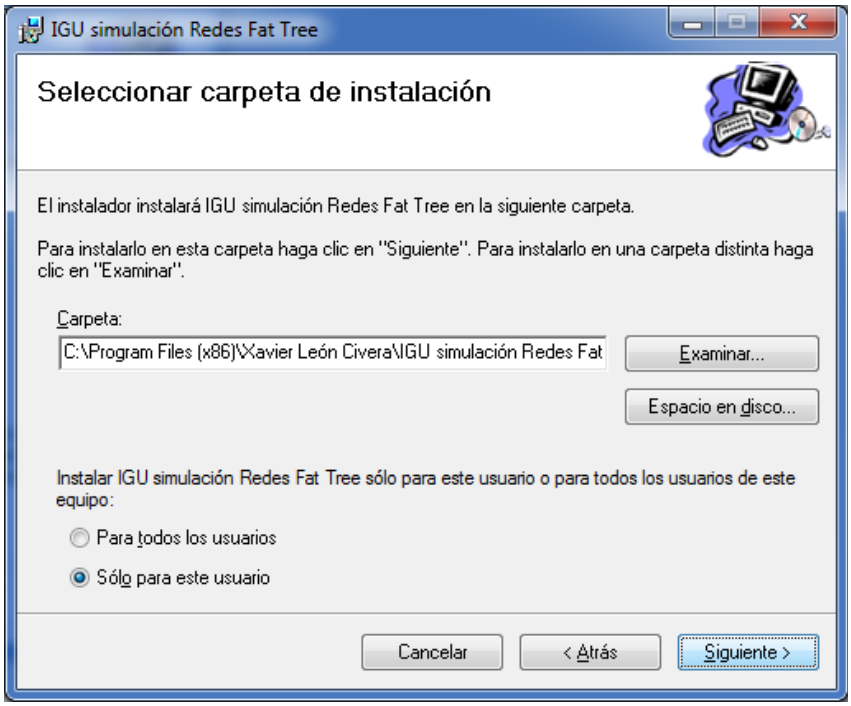


Figura 11: Instalación Selección carpeta destino

La siguiente pantalla que se muestra es la confirmación de que se desea instalar.

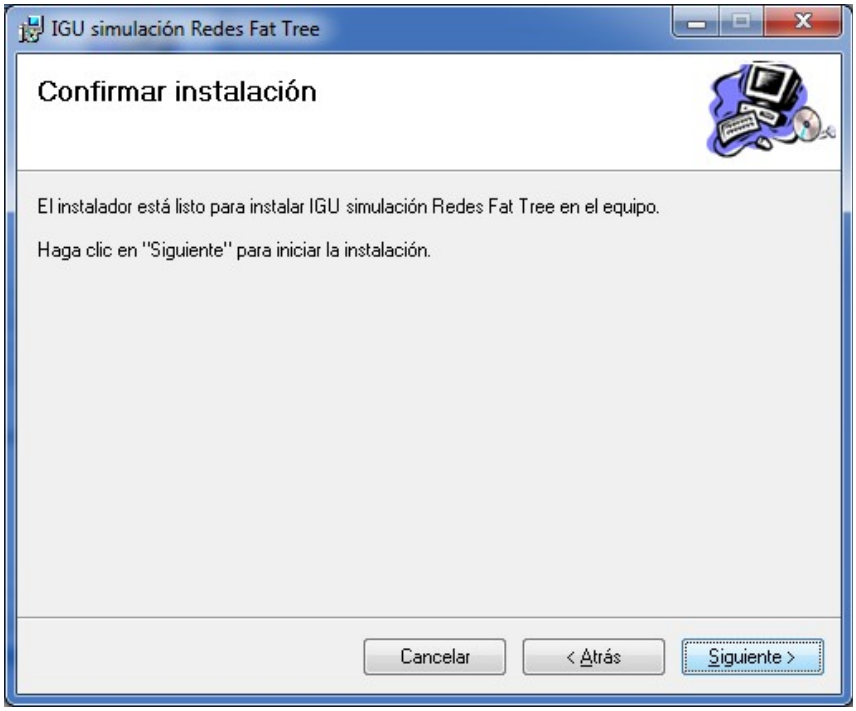


Figura 12: Instalación pantalla confirmación

Una vez confirmada la instalación, se muestra la pantalla que nos informa sobre el estado de la instalación.

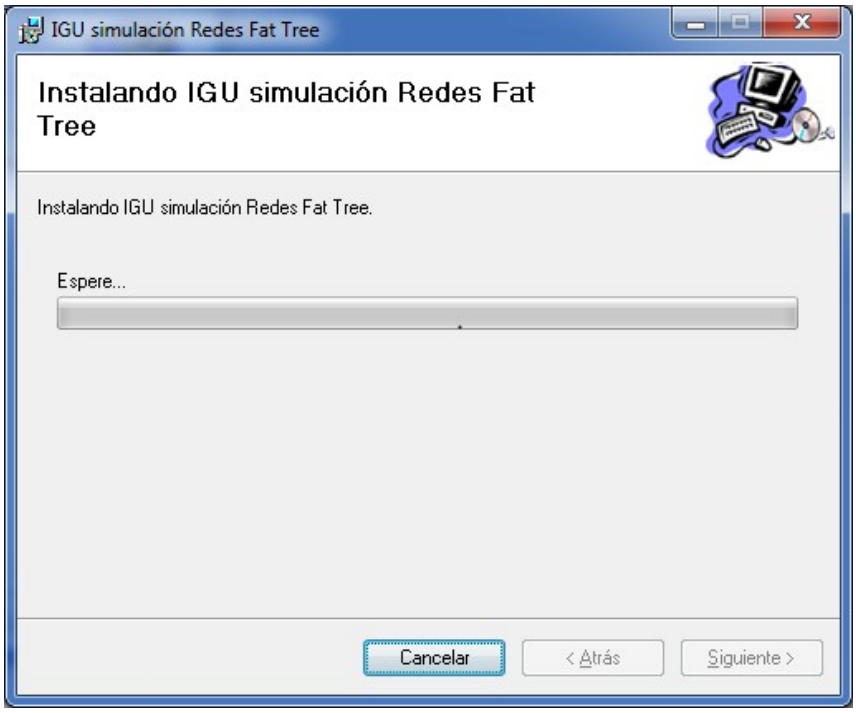


Figura 13: Instalación Proceso Instalación

Finalmente se nos informa de que la aplicación se ha instalado con éxito.

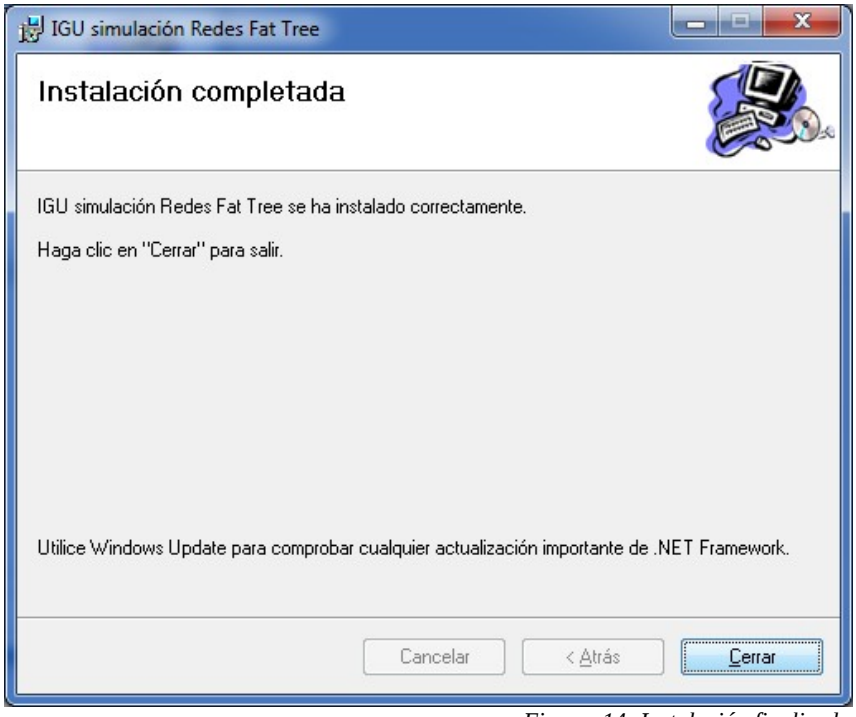


Figura 14: Instalación finalizada

Desinstalación

Aunque el proceso de desinstalar un programa es conocido por todo el mundo en este manual se presenta por si algún usuario tuviese alguna duda.

El primer paso es ir al menú de Windows y seleccionar la opción del "*panel de control*", y dentro de este la opción "*Instalar/Desinstalar Software*".

A continuación se presenta el menú *Inicio* de Windows.

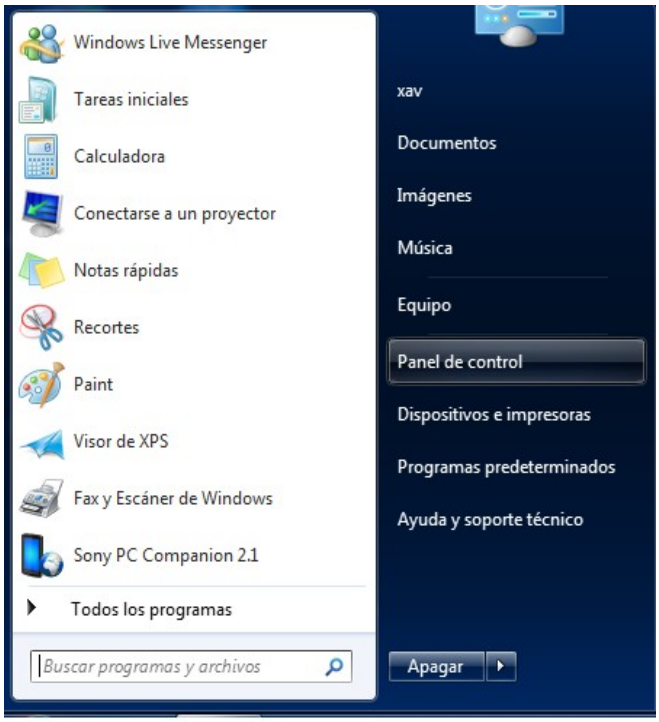


Figura 15: Desinstalación Inicio

A continuación se muestra la ventana del panel de control y la línea que corresponde a esta aplicación donde se puede ver como como nombre el título del proyecto, y editor el nombre del autor de este documento.

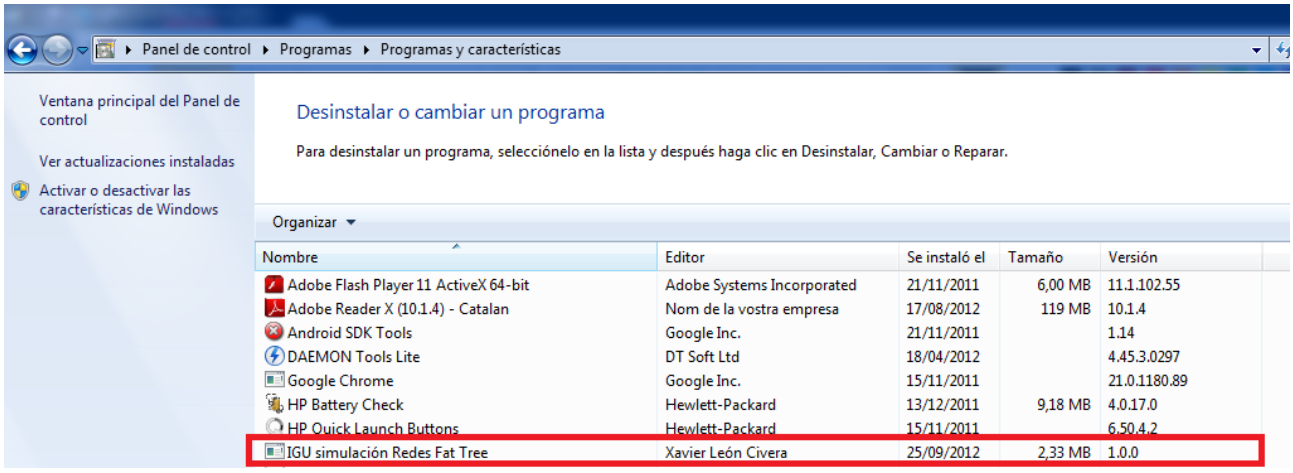


Figura 16: Desinstalación panel de control

Seguidamente el S.O. solicita confirmación del usuario para poder proceder con la desinstalación de la aplicación.

Una vez obtenido dicho permiso se procede a la desinstalación del programa, informando al usuario de la situación del proceso como se puede ver en la siguiente imagen.

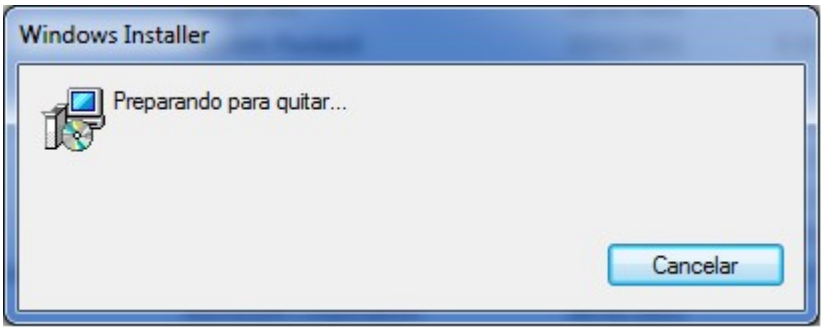


Figura 17: Desinstalación información proceso

Una vez finalizado el proceso desaparecen las ventanas y se puede cerrar panel de control y seguir con el uso del equipo.

Empezando a utilizar la aplicación

Para el uso de esta aplicación es requisito poseer los dos ficheros proporcionados por el simulador, que son el "OnOff.out" que contiene información acerca de la potencia, el tráfico y latencia de los paquetes. El otro fichero es el relacionado con la arquitectura "KaryNtree.out" que posee la información de carga y estado de cada nodo, además de poseer la carga media de la red durante la simulación.

Abrir la aplicación

Una vez instalada la aplicación el usuario tendrá dos opciones de acceso a la aplicación, un acceso directo en el escritorio, y un acceso desde el menú *Inicio* de Windows → *todos los programas* → "*IGU Simulación FatTree*" carpeta en la cual se encuentra el ejecutable.

En la siguiente imagen se puede observar el menú Inicio.

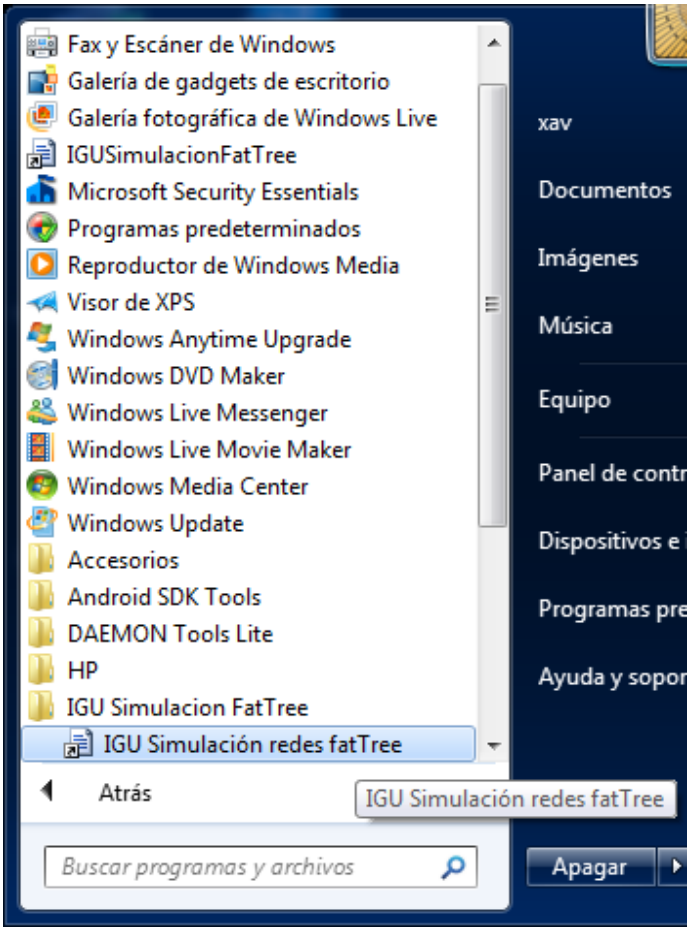


Figura 18: Captura Abrir programa

Una vez ejecutada la aplicación, esta arranca sin ningún dato cargado mostrando la pestaña estadísticas como ventana de inicio.

A continuación se presenta una imagen de la aplicación sin ningún dato cargado, pero podemos observar remarcados en rojo los tres elementos principales.

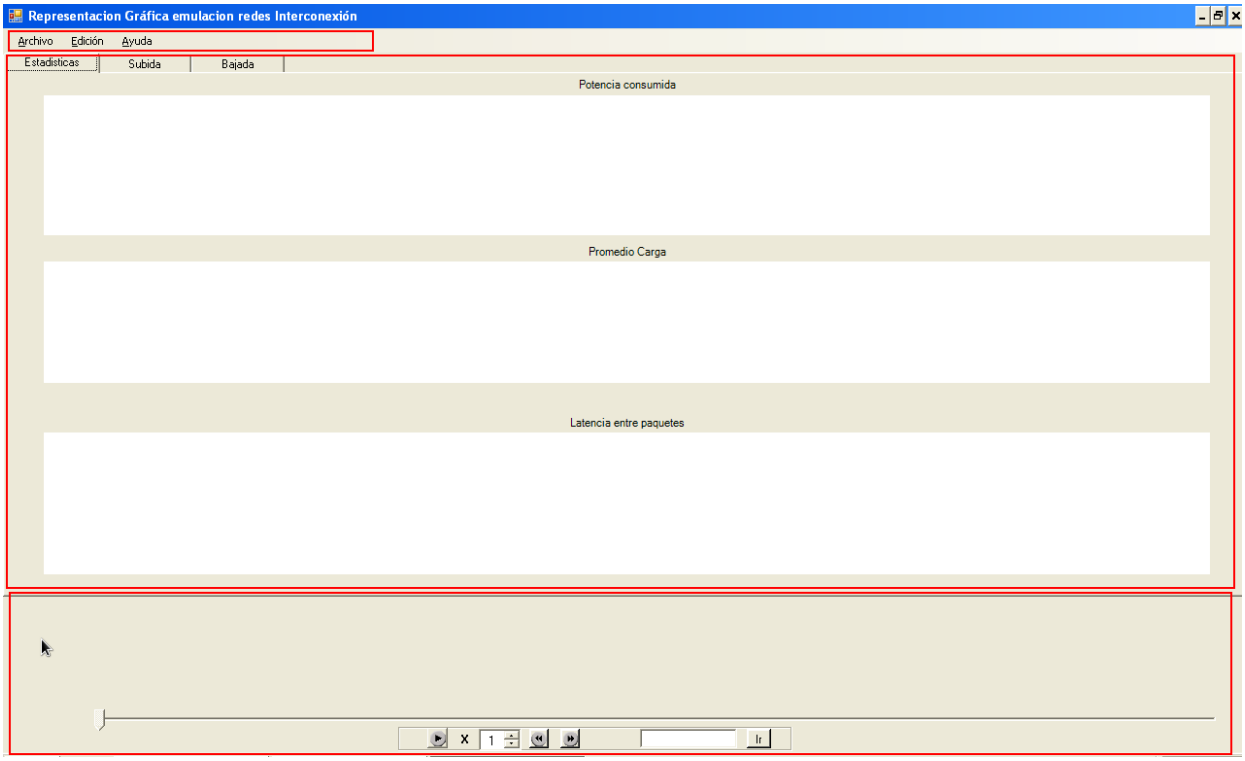


Figura 19: Captura aplicación recién abierto

Para poder representar la carga de cada enlace en cada instante de tiempo la aplicación se ha enfocado como un reproductor multimedia, en el que se puede empezar o detener, adelantar o atrasar en el tiempo, incluso acelerar la velocidad de reproducción o saltar a un instante de tiempo determinado.

Los tres elementos principales, enumerados de arriba abajo son:

Un menú de herramientas con las opciones de la aplicación que contienen todos las aplicaciones.

Un conjunto de pestañas donde se muestran las estadísticas y la información de los enlaces de subida y de bajada.

Una barra de reproducción y unos elementos para adelantar retroceder el aplicación y para mostrar la información de reproducción y arquitectura.

Menú Herramientas

En este menú tenemos las opciones típicas.

En el menú archivo podemos seleccionar abrir una nueva simulación o salir de la aplicación.

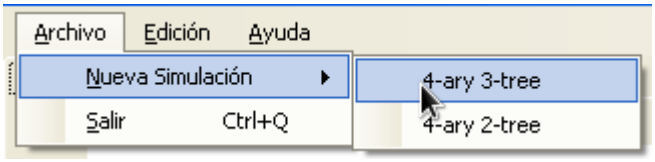


Figura 20: Captura menú archivo

En el menú edición tenemos las opciones de reproducción de la simulación.

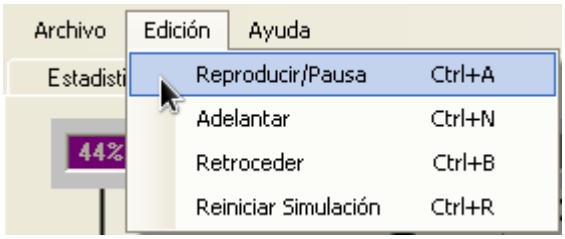


Figura 21: Captura menú edición

En el menú ayuda tenemos la información de la aplicación y el acceso a este manual.

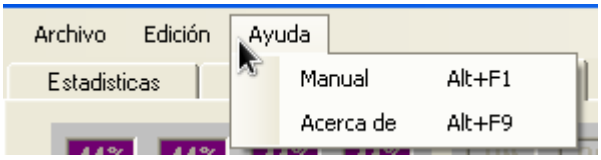


Figura 22: Captura menú ayuda

Primera Simulación.

Para empezar a usar el aplicación hay que dirigirse al menú Archivo y seleccionar el tipo de arquitectura deseado.

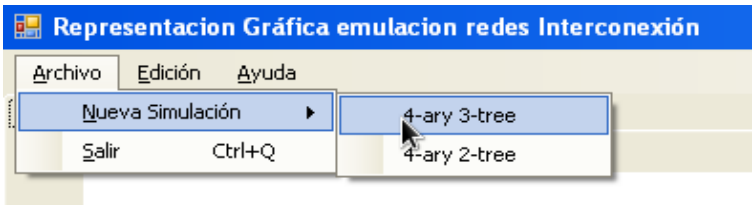


Figura 23: Captura menú nuevo

Una vez seleccionada la arquitectura se abrirá una ventana para seleccionar los ficheros para la simulación.

Daremos a examinar para seleccionar el fichero de información que deseemos.

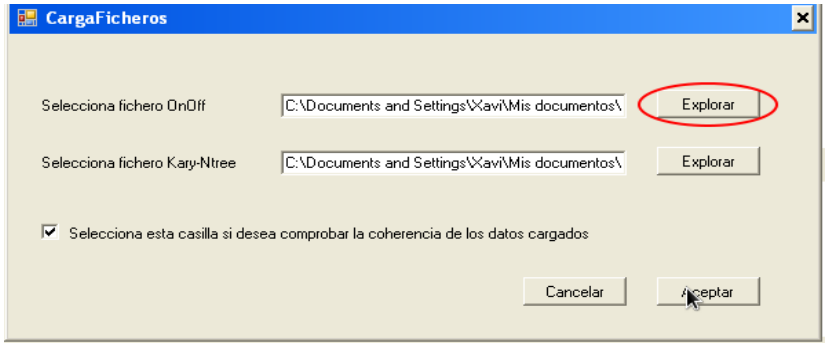


Figura 24: Captura Explorador de ventana

Después de dar a examinar se abrirá una ventana de navegación de Windows para que seleccionemos el fichero que deseemos.

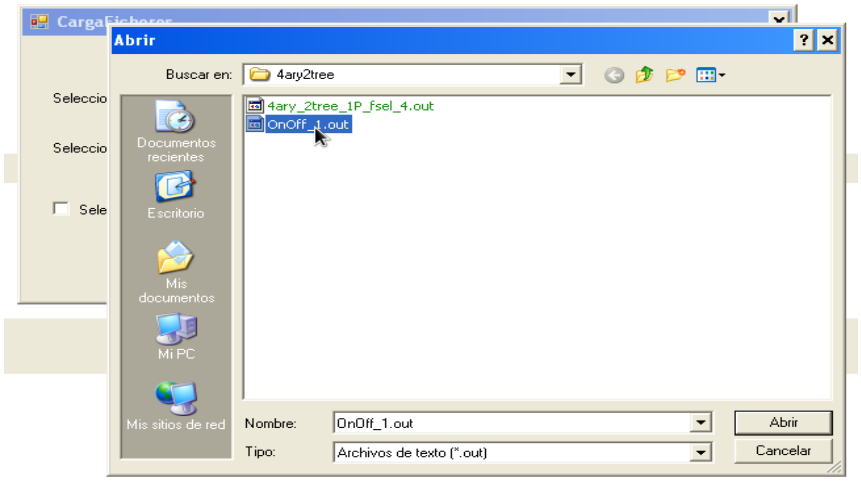


Figura 25: Captura Selección de ficheros

En esta ventana se aprecia una opción que podemos seleccionar para analizar el fichero KaryNtree.out" y crear un fichero de salida con los errores detectados en el fichero.

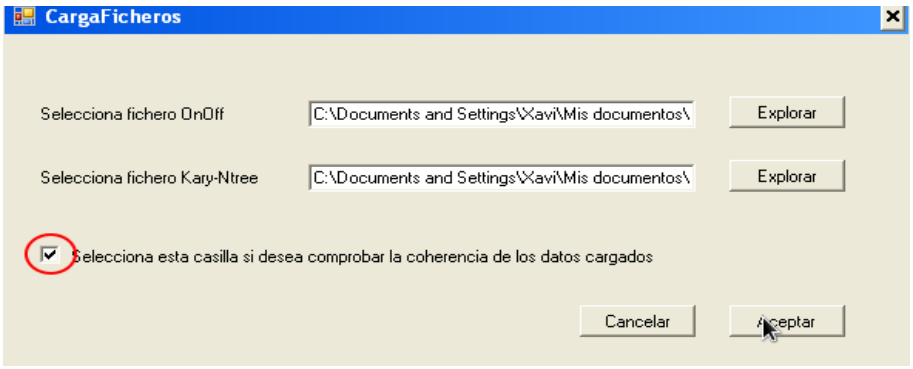


Figura 26: Captura check comprobar fichero

El nombre del fichero de salida será Error seguido del día hora minutos y segundos, y se creará en la carpeta "*Mis Documentos*" del usuario que ejecuto la aplicación.

Analizando los datos

Una vez cargados los datos es tiempo de analizarlos.

La mayoría de los datos se encuentran en la parte central de la ventana, aquí se aprecian 3 pestañas.

La pestaña de Estadísticas que es en la que se inicia la aplicación, aquí se muestran las gráficas de consumo de energía, promedio de carga de los nodos de la red, y la latencia de los paquetes.

Además de la pestaña estadísticas vemos la pestañas subida y bajada. Estas pestañas representan el estado y carga de los enlaces de subida y bajada respectivamente en el instante de la simulación que se desee.

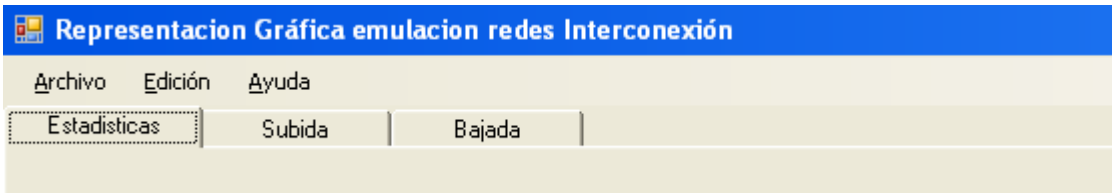


Figura 27: Captura pestañas

Las distintas pestañas se van actualizando en tiempo real conforme la reproducción avanza en el tiempo, así pues se puede cambiar entre las pestañas sin necesidad de detener la reproducción.

Pestaña estadísticas

En esta pestaña tenemos las gráficas que nos representan los datos:

Potencia consumida: representa el total del consumo de los enlaces de la arquitectura durante el tiempo de simulación.

Es una medida relativa. Cada enlace consume **1** cuando está en **ON**, y **0** si está en **Off**. Para los estados de transición **TurningOn** y **TurningOff** se asume el caso más desfavorable, el enlace consume como si estuviese activo pero no está disponible para la transmisión.

Promedio de carga: representa el promedio de carga de todos los enlaces activos durante el tiempo de simulación.

Latencia de los mide el tiempo transcurrido desde que se inyecto el primer bit hasta que el último bit llega a su destino.

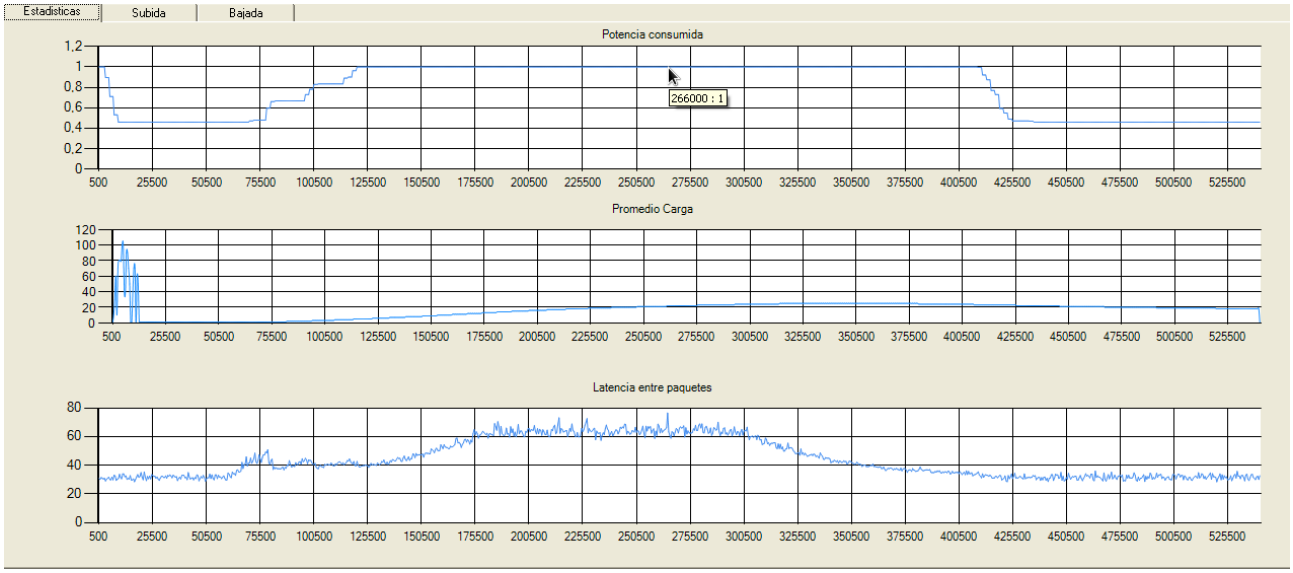


Figura 28: Captura pestaña estadísticas

Al situarnos encima de la línea de la gráfica nos muestra los datos del eje X e Y. Esto permite averiguar el valor que toma en puntas concretas y en qué instante de tiempo exacto sucede.

Pestaña Bajada y Subida

Las pestañas de subida y de bajada muestra el estado de los enlaces de subida y bajada respectivamente.

Como se aprecia en la imagen los enlaces deshabilitados se muestran en color gris con el valor del estado, y los habilitados muestran el tanto por ciento de carga y un fondo de color que varia desde el verde (frío) hasta el rojo (caliente) según la carga que tengan.

En la representación no se tiene en cuenta los estados intermedios TurningOn, TurningOff, se asumen como estados activos del enlace y se muestran con la carga y color de fondo que

corresponde a un enlace activo.

Además del color del puerto, la línea que representa el enlace se ve en un color negro grueso para los enlaces activos y en un gris fino para los deshabilitados.

Por último en esta pestaña se pueden observar los nodos de procesamiento de datos en color negro. Estos nodos la única información que nos proporcionan es el nombre del nodo.

En la figura siguiente podemos ver un ejemplo de los enlaces de bajada. Se aprecia como hay enlaces habilitados con y sin carga, además se aprecian enlaces deshabilitados, y los nodos de procesamiento.

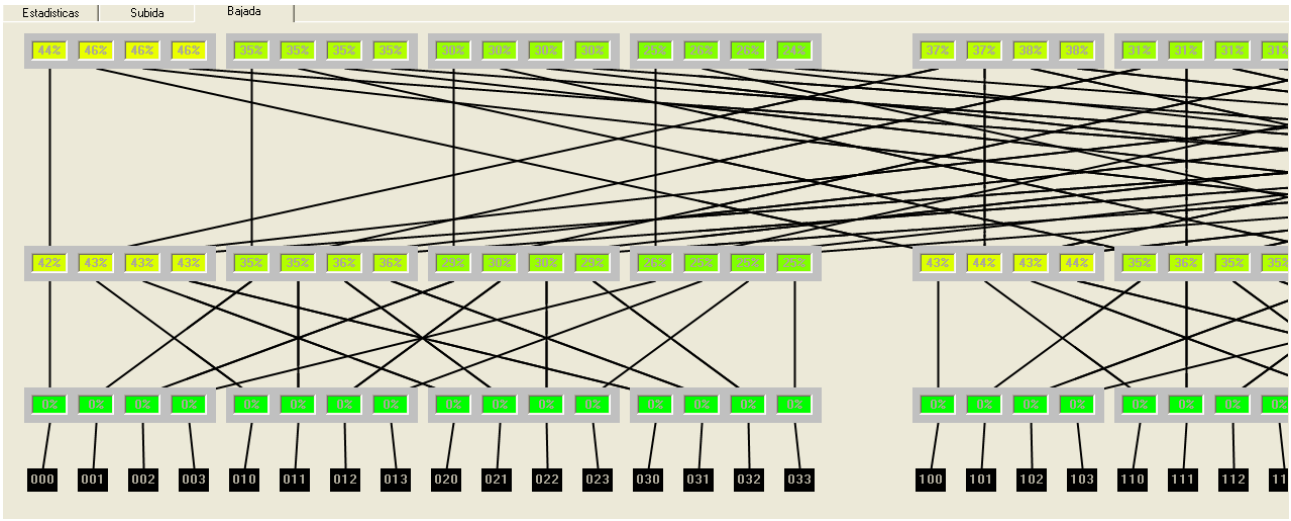


Figura 29: Captura pestaña trafico

Barra de reproducción

Por último podemos observar en la parte inferior de la aplicación la barra de reproducción.

Figura 30: Captura Barra reproducción

Esta barra representa la gráfica de tráfico generado y tiene una barra de desplazamiento que nos permite adelantarnos y atrasar la simulación a un instante deseado.

El tráfico generado se ha representado con una gráfica maciza ya que representa un volumen, y de esta forma es mucho más visual la cantidad de tráfico de la red.

Además en la barra de reproducción tenemos los botones para iniciar, parar o adelantar la

reproducción.

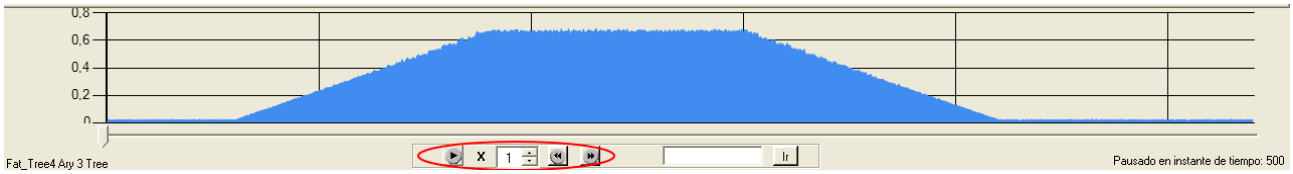


Figura 31: Captura barra reproducción botones

En esta barra se aprecia también una casilla que nos permite aumentar o reducir la velocidad de reproducción.

El valor puede variar de 1 (5 segundos para cada instante de reproducción) hasta 100 (5 milisegundos para cada instante de reproducción).

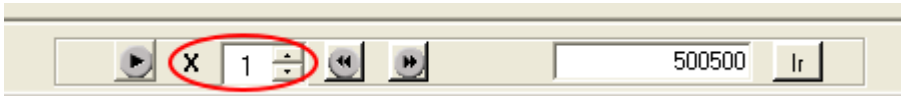


Figura 32: Captura selección velocidad

Además podemos decidir saltar a un instante de tiempo deseado.

Figura 33: Captura Ir a instante tiempo

Por último en la esquina inferior izquierda se muestra la información de la arquitectura seleccionada.

Figura 34: Captura información arquitectura

En la esquina inferior derecha se observa la información de reproducción. Tanto el estado de la reproducción y velocidad como el instante en el que se sitúa la reproducción.

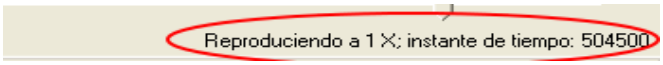


Figura 35: Captura información reproducción

Conclusión

Como se ha presentado en la introducción del documento, la aplicación quiere dar solución a las necesidades reales de los profesores que investigan de las topologías de red Fat Tree.

Por esto el comienzo del proyecto se basó en reuniones con los profesores para analizar sus necesidades para así poder implementar una aplicación práctica y lo más personalizada posible.

Lo siguiente fue presentarles a los profesores el interfaz gráfico para comprobar que había captado todas sus necesidades.

Otra cosa importante en el inicio fue comprender el funcionamiento de las redes Fat Tree, para poder diseñar y separar las especificaciones de la red Fat Tree de las características generales de todas las redes.

Una vez comprobado que tenía claros los objetivos a alcanzar y el funcionamiento de estas redes procedí a diseñar los diagramas de la arquitectura, para poder definir las clases interfaces antes de empezar a programar.

A pesar de haber dedicado un tiempo a diseñar las clases y la relación entre ellas he tenido que ir cambiando el diseño conforme avanzaba con el desarrollo y aumentaban mis conocimientos de las redes Fat tree.

Pese a haber estudiado el lenguaje de programación durante la carrera han surgido imprevistos en cuanto a la hora de posicionar los elementos gráficos en la pantalla, además he tenido dificultades para encontrar solución a mis problemas por lo que me he tenido que limitar a la ayuda ofrecida en la documentación y el foro oficial de Microsoft.

La aplicación cumple todos los objetivos marcados al inicio ofreciendo así una herramienta que va a facilitar el trabajo de análisis de los resultados del simulador de ahorro en el consumo de potencia en las redes de interconexión Fat Tree.

Además de los objetivos iniciales se ha añadido una función extra que comprueba que la información de carga y estado de los enlaces en el fichero "*KaryNtree.out*" es coherente, esta función ha sido incorporada para facilitar la búsqueda de errores y el análisis del propio simulador.

Bibliografía

Libros:

- Patrones de diseño – Pearson.
- PETRINI F., AND VANNESCHI, M. Performance analysis of Wormhole Routed K-ary n-trees. (International Journal on Foundations of Computer Science 9,2 (June 1998), 157-177)

Enlaces:

- http://msdn.microsoft.com/es-ES/