



**Curs 2004/2005**

**Quadrimestre de Tardor**

# **OPNET:**

## **Manual de usuario**

**Departament d'Enginyeria Telemàtica  
Secció d'Enginyeria Telemàtica de l'EPSEVG**

**Setembre 2004**

# Contenido

---

**Prólogo .....**

**Capítulo 1. Introducción a los Simuladores.....**

- 1.1. Introducción**
- 1.2. Tipos de simulación**
- 1.2. Pasos a seguir para una correcta simulación**
- 1.4. Fiabilidad de los resultados obtenidos**
- 1.5. Tiempo de simulación y número de eventos**

**Capítulo 2. Simulador OPNET .....**

- 2.1. Introducción**
- 2.2. Que es OPNET?**
- 2.2. Como funciona OPNET Modeler**
- 2.4. Fases para la realización de una simulación en OPNET**
- 2.5. Componentes del OPNET Modeler**
- 2.6. Simulación DES**
- 2.7. Simulaciones en OPNET**

# Capítulo 1

---

## Introducción a los simuladores

---

### 1.1.- Introducción

### 1.2.- Tipos de simulación

### 1.2.- Pasos a seguir para una correcta simulación

### 1.4.- Fiabilidad de los resultados obtenidos

### 1.5.- Tiempo de simulación y número de eventos

### 1.1.- Introducción

Definimos simulación como una técnica que imita el comportamiento de un sistema del mundo real conforme evoluciona en el tiempo. Por lo tanto, se podrá analizar y observar características, sin la necesidad de acudir al sistema real.

Surgen pues, de éste concepto, dos nuevas definiciones:

- Modelo de simulación que se refiere al conjunto de hipótesis acerca del funcionamiento del sistema expresado como relaciones matemáticas y/o lógicas entre los elementos del sistema.
- Proceso de simulación que será la ejecución del modelo a través del tiempo en un ordenador para generar nuestras representaciones del comportamiento del sistema.

En resumen, podemos decir que el modelo hace referencia a la representación del sistema real que vamos a analizar, las condiciones de su funcionamiento, y las variables que emplea. En cambio, el proceso hace referencia a una ejecución concreta, con unos valores asociados a las variables que se pueden ajustar en el modelo, y que se realiza para obtener los resultados referidos a ciertos parámetros que especifican el comportamiento del sistema.

La simulación se limita a informar de cuál sería el comportamiento del sistema analizado en las condiciones que se indiquen para un proceso determinado.

Otro punto a tener en cuenta son los resultados obtenidos por el simulador. El proceso se basa en el muestreo aleatorio, es decir, los resultados que de ella se extraigan, están sujetos a variaciones aleatorias y por este motivo los resultados obtenidos han de ser examinados. Por lo tanto los resultados tendrán que ser evaluados y comprobar si éstos resultados son fiables o no conforme a las previsiones que se tenían antes de realizar dicho proceso.

Otra definición muy importante corresponde a la de sistema. Un sistema es un conjunto de elementos que, actúan e interactúan para lograr algún fin lógico.

Cuando hablamos de estado del sistema hacemos referencia al conjunto de variables necesarias para describir el estado del sistema en un determinado instante de tiempo.

Entre todas estas variables tenemos que distinguir las entradas y las salidas. Las salidas serán los objetivos de nuestro estudio, expresados mediante valores numéricos.

Las entradas serán los valores numéricos que permitan iniciar la simulación y obtener las salidas. En estas entradas, se incluyen:

- Las condiciones iniciales: Valores que expresan el estado del sistema al principio de la simulación.
- Datos determinísticos: Valores conocidos necesarios para realizar los cálculos que producen las salidas.
- Datos probabilísticos: Cantidades cuyos valores son inciertos pero necesarios para obtener las salidas del sistema. Los valores específicos de estos datos deben conocerse a través de una distribución de probabilidad.

## 1.2.- Tipos de simulaciones

Existen varias características de las cuales podemos clasificar las simulaciones. En este apartado se muestran algunas de estas posibles clasificaciones:

- Simulación estática: Se denomina modelo de simulación estática a la representación de un sistema en un instante de tiempo determinado.

- Simulación dinámica: Se denomina modelo de simulación dinámica a la representación de un sistema cuando evoluciona con el tiempo.

- Simulación determinista: Se denomina modelo de simulación determinista a la representación de un sistema que no contiene absolutamente ninguna variable aleatoria.

- Simulación aleatoria: Se denomina modelo de simulación aleatoria a la representación de un sistema que contendrá variables aleatorias.

- Simulación continua: Se denomina modelo de simulación continuo a la representación de un sistema donde su comportamiento cambia de forma continua en el tiempo.

- Simulación Discreta: Se denomina modelo de simulación discreto a la representación de un sistema donde su comportamiento cambia únicamente en instantes de tiempo concretos, eventos.

## 1.2.- Pasos a seguir para una correcta simulación

Cuando nos disponemos a realizar una simulación que nos informe de lo que realmente nos interesa acerca del modelo y garantice una cierta fiabilidad de los resultados, tenemos que seguir una serie de procedimientos determinados:

- 1.- En primer lugar, enunciar explícitamente los objetivos que se pretenden: los interrogantes que se nos plantean, las hipótesis que se quieren demostrar, y las distintas posibilidades a considerar.
- 2.- A continuación se ha de proceder con la creación del modelo. En nuestro caso, el diseño de la red que hemos de analizar.
- 2.- Posteriormente, se tendrá que diseñar un programa de ordenador para el modelo.
- 4.- Debemos de verificar el programa y validar el modelo.
- 5.- En este momento, ya estamos en disposición de utilizar el modelo para experimentar y contestar a las preguntas que inicialmente se nos planteaban.
- 6.- Finalmente, tendremos que reunir, procesar y analizar los datos generados como soluciones del modelo y en términos de validez y fiabilidad estadística.

#### **1.4.- Fiabilidad de los resultados obtenidos.**

Como es lógico, podemos decir que una sola simulación proporciona un único valor de entre muchos posibles resultados, que puede ser distinto cada vez, como consecuencia del carácter aleatorio de algunas de sus variables. Por ello, tenemos que acudir a las técnicas de inferencia estadística para establecer el número de simulaciones requerido con el objetivo de proporcionar un nivel deseado de confianza. Estas técnicas tienen que ser utilizadas también a la hora de reunir los datos que sirven como entradas.

Expresaremos la confianza en el resultado como la probabilidad de que el valor del sistema real esté comprendido en un intervalo que tenga por centro el valor estimado. Este intervalo se denomina intervalo de confianza. La determinación de estos intervalos de confianza en las simulaciones no resulta sencillo debido a que las salidas no suelen ser independientes (por ejemplo, si en una red consideramos como salidas el rendimiento (throughput) y el retardo, un aumento en el primero vendrá acompañado de un incremento en el segundo), y las condiciones iniciales pueden influir considerablemente en los resultados.

#### **1.5. Tiempo de simulación y número de eventos.**

El tiempo y el número de eventos son parámetros propios de la simulación y están íntimamente ligados a la fiabilidad con que podemos aceptar los resultados que ésta nos devuelve. El tiempo de simulación como su propio nombre indica, permite especificar cuánto tiempo vamos a suponer que está en funcionamiento nuestro sistema, para observar su comportamiento a lo largo de ese periodo, lo cual implicará si estamos calculando valores medios, por ejemplo, calcular la media del valor que toma ese parámetro a lo largo de todo el periodo de simulación.

Gracias a estas técnicas, resulta posible estudiar el comportamiento que tendría un sistema en un largo periodo de tiempo empleando para ello un tiempo considerablemente menor. De manera intuitiva, se puede decir que la fiabilidad de un resultado que haga referencia a la media de una cierta variable, aumentará con el tiempo de simulación si el resto de entradas se mantienen constantes.

De igual forma, el número de eventos serán el número de llegadas o salidas que se producen durante la simulación. Se establece que a mayor número de eventos, en iguales condiciones, la fiabilidad se ve incrementada.

Será, pues, de especial importancia, prestar atención a que el número de eventos que se sucedan en una determinada simulación sea suficiente para garantizar que los resultados son estadísticamente fiables.



# Capítulo 2

---

## Simulador OPNET

---

- 2.1. Introducción**
- 2.2. Que es OPNET?**
- 2.2. Como funciona OPNET Modeler**
- 2.4. Fases para la realización de una simulación en OPNET**
- 2.5. Componentes del OPNET Modeler**
- 2.6. Simulación DES**
- 2.7. Simulaciones en OPNET**

### 2.1.- Introducción

Las simulaciones de sistemas utilizando equipos informáticos son en la actualidad de gran aplicación en el ámbito de la ingeniería. En ellas se puede observar la evolución del sistema, sus características, propiedades..., existiendo únicamente en la memoria de un ordenador. El objetivo que busca todo simulador es recrear un modelo lo más fiable posible a la realidad, al menos en cuanto a las características a estudiar, para poder extrapolar los resultados obtenidos mediante la simulación.

En el campo de las redes de telecomunicaciones se ha experimentado un crecimiento exponencial a nivel mundial, esto ha dado lugar a la necesidad de su sofisticación. Por ello se prioriza disponer de un simulador de red que ofrezca herramientas potentes con el objetivo de diseñar modelos, simular datos y analizar la redes.

Opnet Modeler es capaz de simular una gran variedad de redes. El flujo de mensajes de datos, paquetes perdidos, mensajes de flujo de control, caídas de los enlaces, son algunas de las opciones que nos permite estudiar este simulador; proporcionando a las universidades e ingenieros la forma más efectiva para demostrar los diferentes tipos de redes y protocolos.

OPNET proporciona librerías y gracias a ellas se consigue la formación de redes de comunicaciones y se facilita el estudio del desarrollo de los modelos mediante la conexión de diferentes tipos de nodos, utilizando diferentes tipos de enlaces, etc...

### 2.2.- Qué es OPNET

Es un lenguaje de simulación orientado a las comunicaciones. Proporciona acceso directo al código fuente siendo esto una gran ventaja para los nuevos programadores que se aventuren a programar con OPNET.

Actualmente es utilizado por grandes empresas de telecomunicaciones, por ejemplo para desarrollar proyectos gubernamentales y del ejército, etc...

Para más detalle disponemos de su página oficial <http://www.opnet.com> donde podemos encontrar toda la información referente a cómo descargarnos el software necesario, qué es OPNET, etc...



Figura 2.1. Encabezado del Simulador OPNET

## 2.2.- Cómo funciona OPNET modeler

Como veremos a continuación OPNET es un simulador que posee una interfaz muy seductora para los usuarios. Esto es debido a que incluye varias librerías de modelos. El código fuente de estas librerías es accesible si se dispone de la versión OPNET Modeler y esto consigue que el programador se pueda familiarizar más rápidamente con toda la jerarquía interna del programa.

Para ser utilizado, primero el usuario tiene que comprender la jerarquía que se utiliza para poder plantear las simulaciones.

Esta jerarquía de diseño se muestra en la figura 2.2.

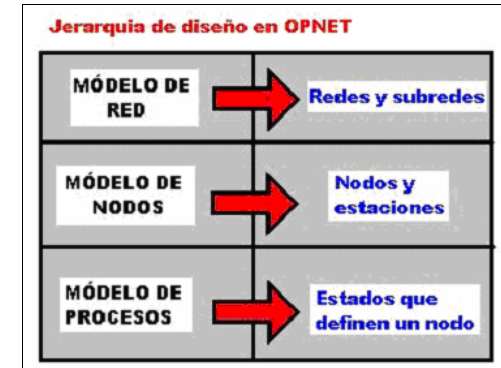


Figura 2.2. Jerarquía de diseño en OPNET

Como podemos observar en la figura 2.2, tenemos un modelo de red donde irán definidas las redes y subredes de la simulación. Seguidamente disponemos de un modelo de nodos donde se define la estructura interna de éstos y por último tenemos el modelo de procesos donde se definen los estados que definen un nodo.

En la correcta simulación de un proyecto se debe haber entendido los aspectos anteriores; puesto que de lo contrario la simulación saldría errónea y no serviría.

Para un mayor entendimiento profundizaremos en este aspecto.

Como hemos mencionado con anterioridad, poseemos un Modelo de nodos. Este modelo de nodos funciona como muestra la figura 2.3:

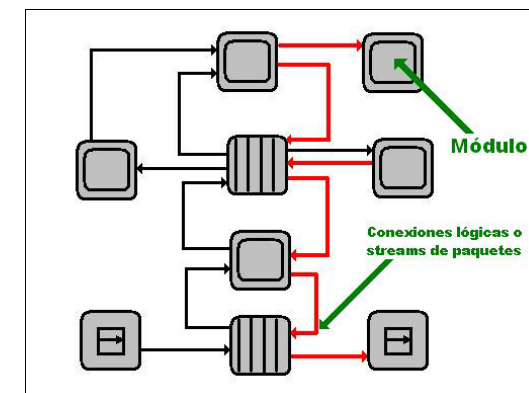


Figura 2.2. Funcionamiento del modelo de Nodos.

Un nodo puede tener en su interior varios módulos. Éste tiene una función definida en su interior, así, un módulo llamado receptor tendrá la función de recibir los paquetes de otro.

Para realizar esta parte de la simulación necesitamos tener el Node editor que podremos encontrar en el OPNET Modeler.

Los módulos que nos ofrece el node editor son los siguientes:

|  |   |
|--|---|
|  | Icono estándar para un módulo de procesado. Su función principal es construir bloques de modelo de nodo.  |
|  | Icono estándar usado para un módulo de cola. Lo que le hace diferente del resto de módulos es que el módulo de colas tiene recursos adicionales internos llamados subcolas. |
|  | Icono usado para un módulo de transmitir. Transmisor Punto a Punto.   |
|  | Icono usado para un módulo de recibir. Receptor Punto a Punto.  |
|  | Icono usado para un módulo de transmitir. Transmisor tipo bus.  |
|  | Icono usado para un módulo de recibir. Receptor tipo bus.   |
|  | Icono usado para un módulo de transmitir. Transmisor por radiofrecuencia.   |
|  | Icono usado para un módulo de recibir. Receptor de radiofrecuencia.   |
|  | Icono estándar usado para las comunicaciones vía satélite.  |
|  | Icono estándar para un módulo esys.   |

Tabla 2.4. Módulos del Node Editor

Todos estos módulos se relacionan directamente con los procesos mediante la opción Process Model en su edit attributes.

Por otra parte, las conexiones lógicas que nos ofrece el node editor son las siguientes:

|  |  |
|--|--|
|  | Packet stream: Son conexiones que llevan los paquetes desde un módulo fuente a un módulo destino.  |
|  | Statistic wires: Transportan datos de un módulo fuente a un módulo destino. Sirven como interface para que un módulo fuente pueda compartir datos con un módulo destino, y proporcionar información respecto de su estado. |
|  | Logical associations: Su misión es indicar qué relación existe entre dos módulos de la simulación. Por tanto, no trasportan datos entre módulos.   |

Tabla 2.5. Conexiones del Node Editor.

Además disponemos de un modelo de procesos donde se define lo realizado en cada uno de los módulos de un nodo. El gráfico 2.6 muestra el funcionamiento explicado anteriormente.

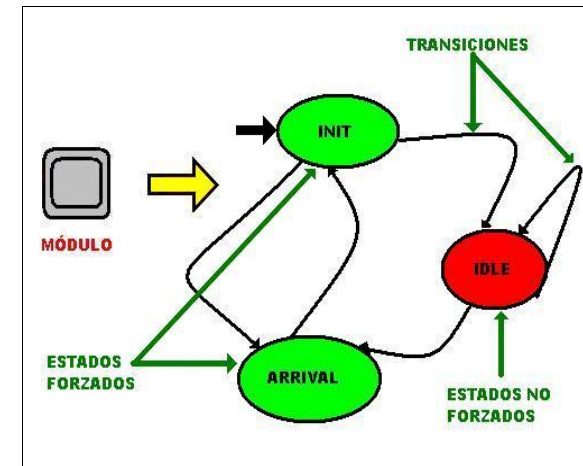


Gráfico 2.6. Funcionamiento del módulo de nodos.

La funcionalidad de cada módulo se define a través de modelos de proceso, que se representan mediante máquinas de estados finitos (FSM). Las transiciones entre

estados pueden ser condicionales o incondicionales. El funcionamiento interno tanto de estados como de transiciones implica la programación de código C/C++.

Para crear estos modelos de procesos, el software ofrece un editor llamado process editor. En éste se pueden definir los estados y las transiciones de los estados. Además, en él se programa el grueso de la simulación en lenguaje C++. OPNET, que contiene un manual de aproximadamente 1500 hojas, donde explica todas las funciones y ejemplos para el aprendizaje de la programación en el simulador. Una vez se ha creado el modelo de proceso, el siguiente paso es la compilación y la verificación. Para este paso se debe albergar en el ordenador un compilador de C++ ( en nuestro caso hemos instalado el visual C++ de Microsoft ).

En la tabla 2.7 disponemos de los iconos de configuración del process model.

En el momento de realizar una simulación el programador tendrá que efectuar los siguientes pasos:

- Descripción del modelo
- Modelo de red
- Modelo de nodos
- Modelo de procesos
- Resultado de la simulación

Para una mejor comprensión de los conceptos, ilustramos un ejemplo de simulación con OPNET, al creer de gran importancia la comprensión de la jerarquía de la programación .











|   |   |
|---|---|
|  | Create State: Crea un estado dentro del process model.  |
|  | Create Transition: Crea transiciones entre los diferentes estados.  |
|  | Set Initial State: Establece el Estado Inicial.   |
|  | Edit State Variables: Establece los estados variables de la simulación.   |
|  | Edit Temporary Variables: Instauro Estados Temporales para el process model   |
|  | Edit Header Block: Instauro el Header Block donde se declaran las variables, macros, tipos de datos, etc., incluidos en el process model.     |
|  | Edit Function Block: Instauro el Function Block en el cual se crean nuevas funciones en C++ para el process model                             |
|  | Edit Diagnostic Block: Establece el Diagnostic Block que contiene funciones sobre el estado de la simulación.                                 |
|  | Edit Termination Block: Establece el Termination Block donde se establecen unas funciones que son ejecutadas antes de terminar la simulación. |
|  | Compile Process Model: Compilación del código.  |

Tabla 2.7. Opciones del Process Model

Ejemplo: Simulación Aloha (Esta simulación la explicaremos detalladamente más adelante ).

- Modelo de red

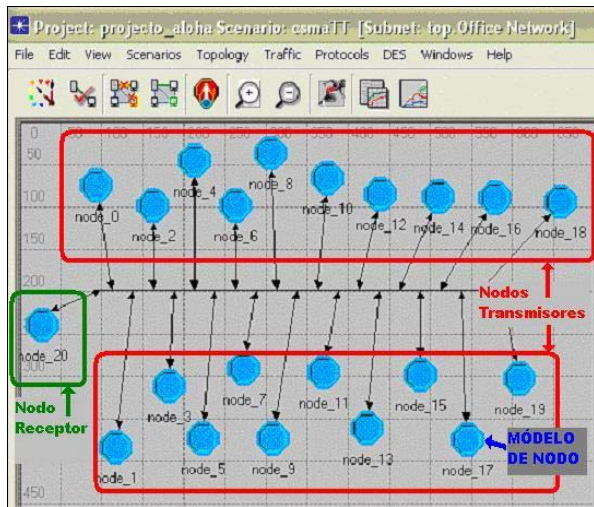


Figura 2.8. Ejemplo de Modelo de red.

En la figura 2.8 disponemos un ejemplo del modelo de red. En este caso tenemos 19 nodos transmisores y 1 nodo receptor. Como se puede apreciar en la figura de arriba, los rombos azules son lo que hemos denominado modelo de nodos. Las flechas que unen los nodos son los enlaces y además OPNET nos ofrece un editor llamado Link editor en el cual podemos definir las características de éste.

- Modelo de nodos

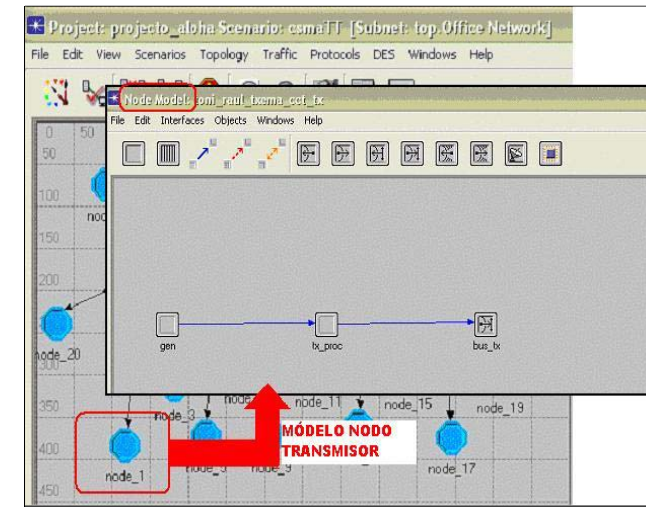


Figura 2.9. Ejemplo de Modelo de nodo Transmisor

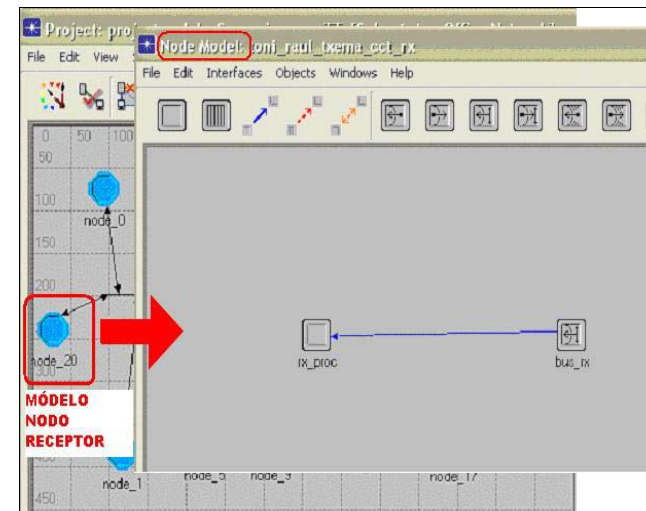


Figura 2.10. Ejemplo de Modelo de nodo Receptor.

La figura 2.9 y 2.10 muestra cómo cada nodo que teníamos en el modelo de red, está definido con un conjunto de módulos. Disponemos de dos clases de nodos, un nodo

transmisor que transmite los paquetes en cuanto le llegan y un nodo receptor hacia el cual se direccionan todos los nodos.

- Modelo de procesos

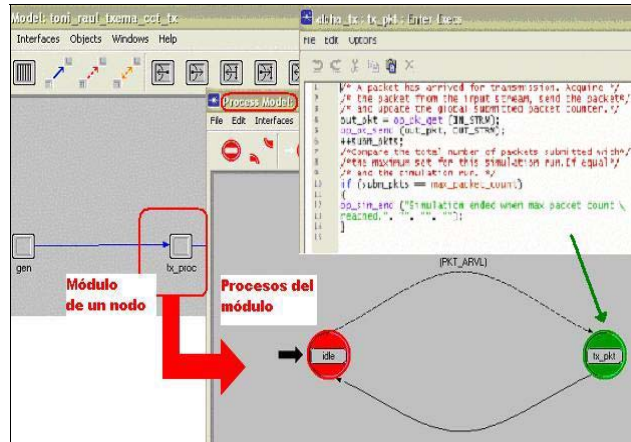


Figura 2.11. Ejemplo de proceso del nodo transmisor

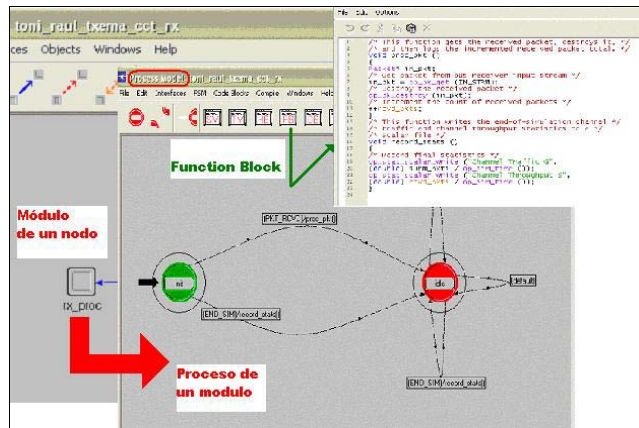


Figura 2.12. Ejemplo Modelo de proceso del nodo receptor.

En las figuras 2.11 y 2.12 se puede observar cómo cada módulo definido en el node model tiene su correspondiente modelo de proceso, el cual define lo que tiene que realizar cada módulo.

- Resultado de la simulación

Una vez se han realizado todos los pasos anteriores ya se puede simular la red. Para ello se han de configurar una serie de parámetros en el project editor. El resultado se muestra en forma de gráfica. Puede presentarse con valores escalares o con valores vectoriales. Un ejemplo de resultado de simulación es el que mostramos en la figura 2.12.

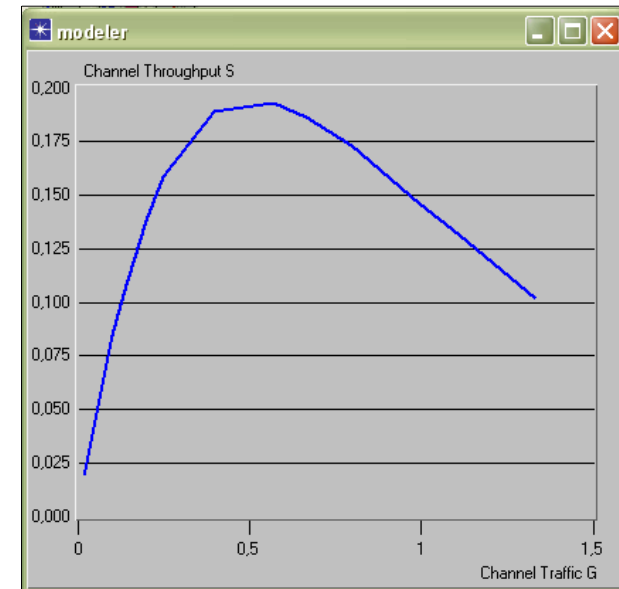


Figura 2.12. Resultado de la simulación.

**2.4.- Fases para la realización de una simulación en OPNET**

Para la realización de una simulación en OPNET se siguen 3 fases:

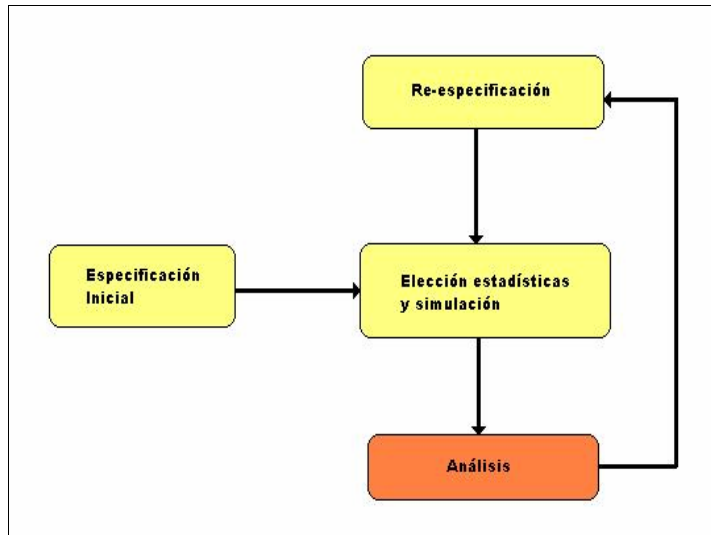


Figura 2.14. Esquema de las fases de la simulación.

Como se puede observar en el esquema superior (Figura 2.14), la especificación del modelo consiste en desarrollar la representación del sistema a estudiar. Para esta fase, disponemos de modelos ya realizados en OPNET y de editores para perpetrar nuestros propios modelos.

Una vez especificado el modelo a simular, el siguiente paso es elegir los datos a recolectar y seguidamente nos dispondremos a realizar el análisis para validar las especificaciones expuestas en el modelo. En el caso de que estos resultados no fuesen los deseados, se tendría que hacer una re-especificación donde se modificarían los aspectos erróneos del modelo simulado.

## 2.5.- Partes de OPNET modeler

A continuación explicamos las diferentes partes de que consta el OPNET Modeler, y cuales vamos a utilizar. Los editores incluidos proporcionan las herramientas necesarias para la creación de topologías de red. Cada editor se encarga de una tarea distinta, inmediatamente explicaremos cada uno de ellos.

### 2.5.1.- Project Editor:

El Project editor es el principal escenario en la creación del entorno de la simulación de la red. Es usado para crear un modelo de red utilizando unos ya existentes que podemos encontrar en la librería estándar, recolectar estadísticas sobre la red, comenzar la simulación y observar los resultados. También podemos crear nodos, construir formatos de paquetes, etc.

Este editor contiene tres tipos básicos de objetos: subredes, nodos y enlaces.

Las paletas (accesibles mediante un icono situado en la parte superior izquierda del editor) ordenan los objetos disponibles en categorías. Por ejemplo, en la paleta ethernet, se encuentran los nodos y enlaces más utilizados para el diseño de este tipo de red.

En este editor como hemos mencionado podemos observar los resultados obtenidos.

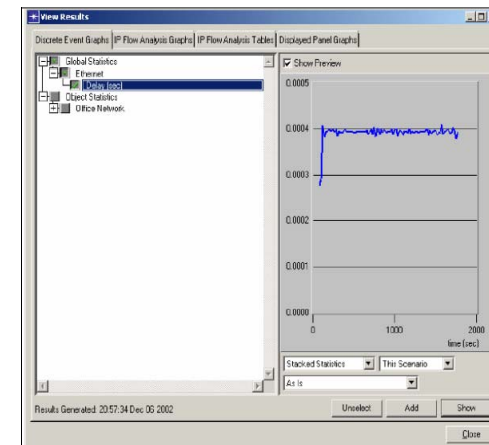


Figura 2.15. Visualización de resultados.

Al seleccionar la opción de ver resultados (view results), aparecen las estadísticas disponibles. Esta opción logramos observarla en la figura 2.15, donde se plasma la

visualización de un resultado de retardo. También podemos distinguir en la zona izquierda de la figura una selección, ésta son los diferentes resultados que nos permite analizar el programa.

El formato que posee el project editor es el mostrado en la figura 2.16.

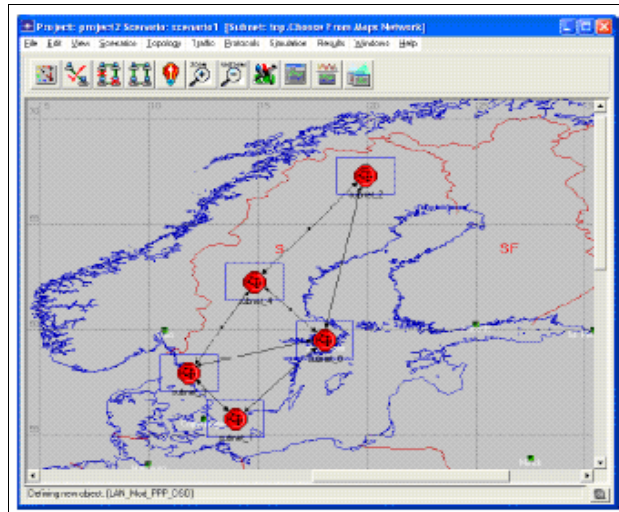


Figura 2.16 Formato Project Editor.

### 2.5.2.- Node Editor:

El Node Editor es un editor que es usado para crear modelos de nodos especificando su estructura interna. Estos modelos son usados para crear nodos en el interior de la red en el Project Editor.

Internamente, los modelos de nodos tienen una estructura modular, al ser definido un nodo como la conexión entre varios módulos con paquetes de streams y cables. Esta conexión permite intercambiar información y paquetes entre ellos. Cada módulo tiene una función específica dentro del nodo, tal como: generar paquetes, encolar los, procesarlos o transmitirlos y recibirlos.

En este editor los elementos se encuentran disponibles como cajas negras, albergando atributos que pueden ser configurados. Cada una de ellas representa una función en el nodo.

Los objetos presentes en este editor son los procesadores. El comportamiento de éstos viene definido en el editor de procesos. Existen modelos ya configurados, tales como fuentes de datos, sumideros, etc...

Los procesadores más comunes son:

- Colas: Poseen distintos atributos para definir el carácter de la misma.
- Transmisores y receptores: Controlan la salida y entrada de paquetes al nodo.
- Stream de paquetes: Lleva el flujo de paquetes entre cajas negras.
- Statistics Wire: Transporta estadísticas.
- Cable de asociación lógica transmisor-receptor: Usado para crear un vínculo entre transmisores y receptores de un mismo elemento.

La estructura de formación de un modelo de nodo se puede visualizar en la figura 2.17, en ella se muestra tanto el editor de nodos como un pequeño ejemplo en él. En ese ejemplo logramos distinguir los diferentes procesadores que anteriormente hemos expuesto.

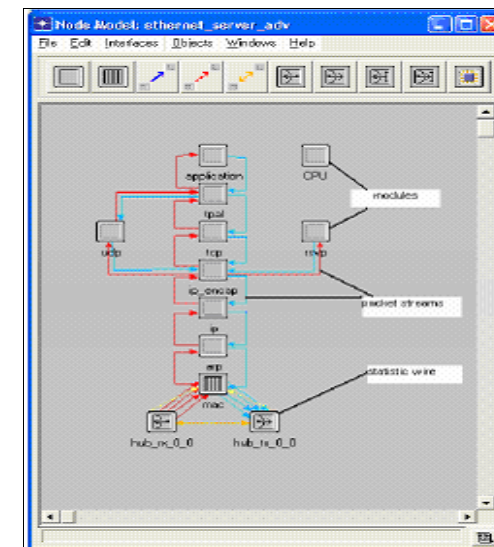


Figura 2.17. Node Editor.

## 2.5.2.- Process Model Editor:

Se utiliza en la creación de modelos de procesos, que a su vez controlan los modelos de nodo creados en el Node Editor. Los Process Model son representados por estados (FSM), y son creados por iconos que representan estos estados y por líneas que representan las transiciones entre ellos.

Las operaciones que realizan cada estado o cada transición se escriben en lenguaje C++. A este tipo de simulación se le denomina simulación DES (Discrete event simulation). Más adelante explicaremos en qué consiste la ésta.

La estructura de este editor se visualiza en la figura 2.18, mediante un ejemplo de un proceso.

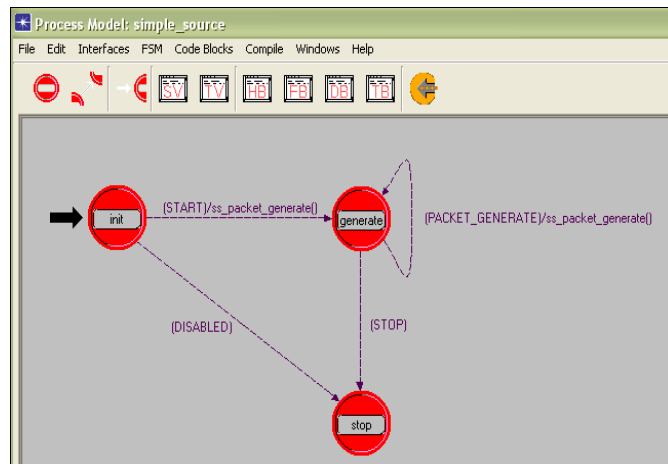
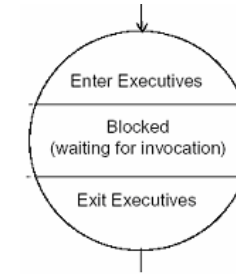


Figura 2.18. Formato del Process Model Editor.

En este editor se colocan los diferentes estados. Todos ellos están compuestos de dos partes: Una parte llamada enter executives donde se alberga la programación ejecutada a los procesos incidentes al estado, y otra parte denominada exit executives en la que como su nombre indica se ejecutan sus comandos cuando el proceso sale del estado hacia una transición.



Existen dos tipos de estados

1.- Estado forzado



Un estado forzado es aquel que cuando le llega un proceso, ejecuta las comandos que tiene su enter executives e inmediatamente ejecuta las comandos que tiene su exit executives. A continuación el proceso saldrá por una transición.

2.- Estado no forzado.



Un estado no forzado permite hacer una pausa entre el enter executive y el exit executive. Cuando un proceso llega a este estado, en primer lugar se ejecuta el enter executives y se forma una pausa hasta que sucede una nueva invocación. Cuando esta invocación se ha producido, se ejecuta el exit executive y el proceso pasa a la transición de salida.

Por lo tanto podemos decir que los objetos más importantes de este editor son:

- Estados: Cada uno de ellos representa un proceso. Se definen en él las funciones a realizar durante su ejecución.

- Transiciones: Marcan la condición que se necesita para pasar de un estado a otro.
- Bloques: Sirven para la programación, la declaración de variables, funciones, etc .....

El botón amarillo con una flecha situado en la parte superior izquierda compila el proceso, que es necesario para su correcto funcionamiento.

### 2.5.4.- Link Model Editor:

Este editor nos ofrece la posibilidad de crear nuevos tipos de objetos link. Cada nuevo tipo de link puede tener diferentes atributos y representaciones. En la figura 2.19 se muestra el editor.

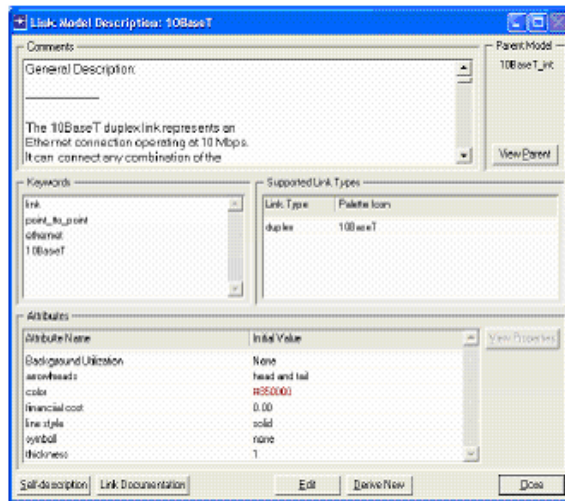


Figura 2.19. Link Model Editor.

Un link model especifica la información siguiente:

- Tipo de enlace soportado: Todos los enlaces que creamos pueden soportar uno o los cuatro s tolerados por el simulador. Estos enlaces son los siguientes: punto a punto duplex, punto a punto simple, bus y taps links.
- Keyword: Sirve para simplificar la paleta del project editor y así facilitar el trabajo al programador.
- Comentarios: Este apartado nos permite añadir comentarios a nuestros enlaces. Es muy útil a la hora de utilizar la versión de evaluación ya que no podremos acceder al editor y poder ver lo que hace. Aquí se pueden comentar la capacidad del enlace, las características, etc...
- Especificación de atributos: Podemos cambiar los valores de los atributos puestos por defecto.

### 2.5.5.- Path Editor:

Es usado para crear nuevos objetos path que sirven para definir un traffic route. En la figura 2.20 se muestra este editor.

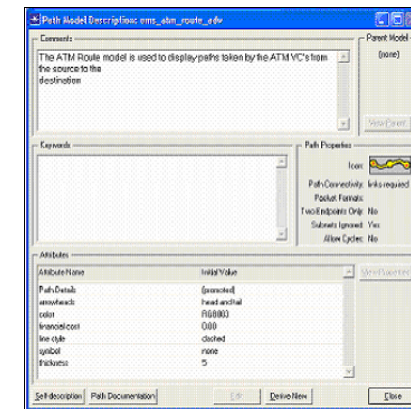


Figura 2.20. Path Editor.

Debemos tener en cuenta que ningún protocolo que use conexiones lógicas o circuitos virtuales puede usar paths para routes de tráfico.

### 2.5.6.- Packet Format Editor:

Este editor es utilizado en la definición de la estructura interna de un paquete como un conjunto de campos. Para cada uno de los campos el packet format especifica un único nombre, tipo de datos, valor por defecto, tamaño en bits, comentarios opcionales, etc...

Los packet formats son atributos de los módulos de transmisión y recepción del modelo de nodos. El formato de un paquete contiene uno o más campos, representados en el editor como cajas rectangulares.

El tamaño de la caja es proporcional al número de bits específicos del campo. En la figura 2.21 podemos contemplar tanto el editor como un ejemplo de un paquete.

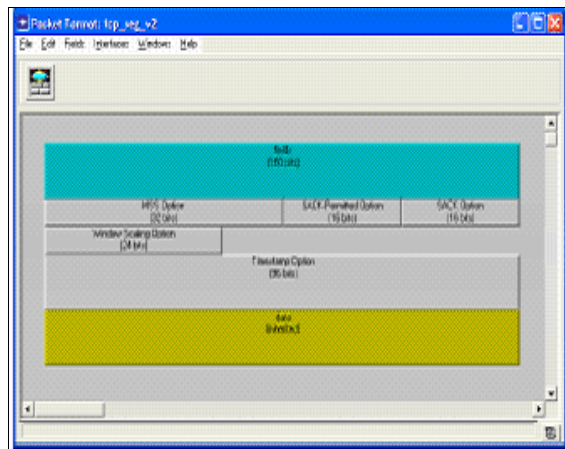


Figura 2.21. Packet Format Editor.

### 2.5.7.- Probe Editor:

Probe editor es usado para especificar las estadísticas que van a ser recopiladas. Pueden ser de diferentes tipos como: estadísticas globales, de enlaces, de nodos, de atributos, etc. Este editor tiene una representación que podemos contemplar en la figura 2.22.

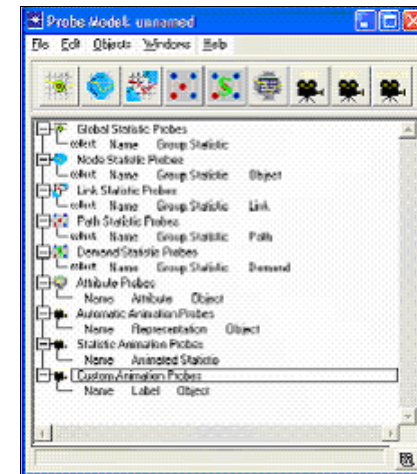


Figura 2.22. Probe Editor.

### 2.5.8.- Simulation Sequence Editor:

En este editor podemos añadir valores adicionales específicos tales como el control del tiempo de simulación, la velocidad, etc. En la figura 2.23 lo podemos observar.

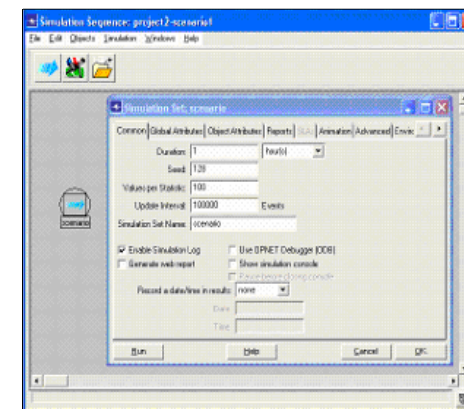


Figura 2.22. Simulation Sequence Editor.

## 2.5.9.- Analysis Tool:

El analysis tool se emplea para crear gráficos escalares de parámetros a estudiar, definir estadísticas de datos, y contemplar los resultados de la simulación. Nos podemos referir a la figura 2.24 para contemplar un ejemplo.

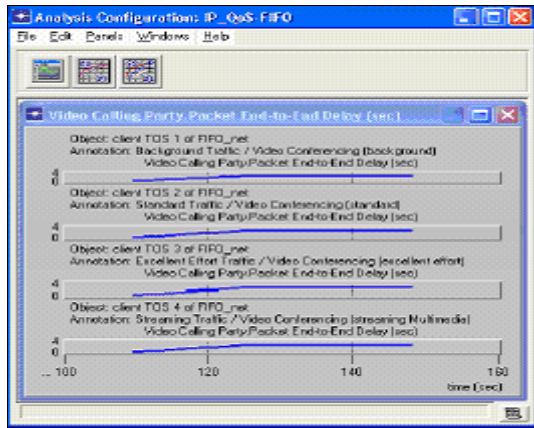


Figura 2.24. Análisis Tool.

## 2.6.- Simulación DES

Se ha comentado anteriormente que el entorno de simulación OPNET utiliza la tecnología DES ( Discrete event simulation ). Ahora nos disponemos a exponer de que trata esta tecnología.

En una simulación DES se utilizan eventos para describir sucesos o acciones que tienen lugar en un determinado momento. Cada uno de estos eventos tiene un instante de incidencia puntual en la escala temporal. Esta escala, al igual que el resto de magnitudes, es discreta.

Durante la simulación, se puede distinguir entre el tiempo real y el tiempo de simulación, por ello se utilizan variables contador para representar el momento actual y las cantidades de tiempo. También se utilizan varias variables de estado para representar la fase del sistema simulado. Por lo que se refiere al sistema, evoluciona en la memoria

del ordenador, produciéndose diferentes eventos que edifican las variables de estado, y éstas a su vez determinan los futuros eventos.

Cada evento tiene un instante de incidencia puntual, es decir, pueden ser representados sobre la escala temporal discreta de la simulación ocupando una única posición. De este modo, dichos eventos logran ser ordenados cronológicamente, según su instante de incidencia, para ser procesados. En este tipo de simulación, el evento es la unidad de ejecución. Cada uno describe una acción, y el resultado de ésta es la modificación de las variables de estado. Esta característica está especialmente soportada por los lenguajes OOP (programación orientada a objetos).

Así pues, un simulador DES debe tener un bloque que inicialice todas las variables de estado del sistema simulado, un procesador que ejecute eventos, un scheduler que sincronice los bloques asegurando que los eventos se ejecutan en el orden adecuado y un recolector de datos estadísticos que tome nota de lo ocurrido. Por último, al finalizar la simulación o de manera dinámica durante su ejecución, se podrán procesar los datos recogidos para extraer la información deseada con la posibilidad de representarlos de forma gráfica.

Entretanto, la simulación se repite de forma continuada en un diagrama de bloques que en cada iteración recoge el primer evento a procesar. A continuación se ejecuta éste y modifican sus variables de estado, después avanza el contador de tiempo hasta el momento de incidencia del próximo evento, y se reinicia el bucle. El proceso mencionado lo encontramos en la figura 2.26.

Como hemos mencionado anteriormente, el contador de tiempo no avanza de manera constante, sino que salta directamente el tiempo restante desde el primer instante actual hasta la incidencia del próximo evento, siendo este periodo aleatorio y determinado por la secuencia de eventos.

Durante la actualización del contador de tiempo y la modificación de las variables de estado, pueden generarse nuevos eventos que se insertarán en la lista de eventos a procesar o modificarán los atributos de los ya existentes.

El simulador OPNET proporciona un capítulo extenso que contiene las DES ya diseñadas. En la *anexo1* exponemos las que consideramos más importantes para llevar a cabo la realización de la simulación de una red EPON.

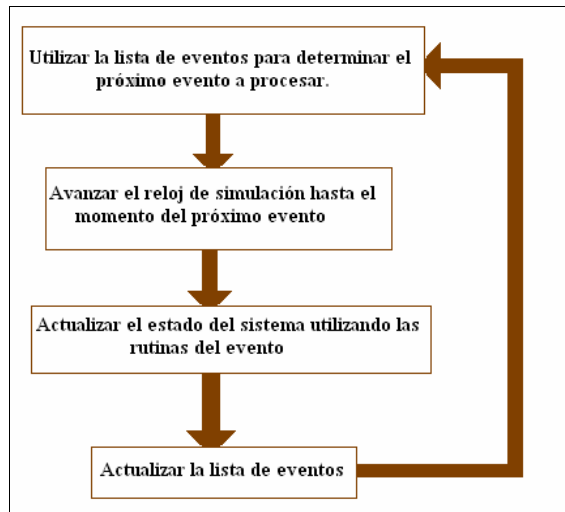


Figura 2.26. Proceso de simulación.

## 2.7.- Simulaciones con OPNET

En este apartado explicamos la realización de varias simulaciones con el simulador OPNET. Como se podrá observar hemos profundizado en el modo de realización para que el lector note las diferencias que se aprecian en comparación con el simulador NS explicado en el capítulo anterior.

### 2.7.1.- Protocolo de control de acceso al medio Aloha

Aloha es una técnica de control de acceso al medio para acceso múltiple. Este protocolo fue desarrollado para redes de paquetes de radio, siendo igualmente aplicable a cualquier medio de transmisión compartido. El funcionamiento del protocolo Aloha consiste en identificar el momento en que una estación desea transmitir hasta que los realiza. Esto lo convierte en un protocolo extremadamente sencillo y debido a esta sencillez presenta algunos puntos débiles. Dado que el número de colisiones crece muy

rápidamente cuando aumenta la carga, la utilización máxima del canal es sólo del 18%. Esto es lo que nosotros vamos a demostrar mediante la simulación siguiente. Desde el punto de vista académico el estudio de las redes Aloha es muy interesante ya que su relativa sencillez le permite extraer conclusiones sobre las prestaciones de protocolos más complejos.

#### 2.7.1.1.- Introducción.

Para la realización de la simulación necesitamos el software OPNET Modeler 10.5. El protocolo Aloha es un control de Acceso al Medio aleatorio lo que nos indica que la transmisión se realiza de forma aleatoria, es decir, sin tiempo establecido o predecible para que las estaciones transmitan. Su funcionamiento es el más simple posible, siendo posible que cuando una estación obtiene una trama para ser enviada ésta se envíe. Seguidamente se atiende el medio durante un tiempo igual al máximo retardo posible, que es dos veces el tiempo de propagación entre las dos estaciones más distantes más un pequeño incremento de tiempo fijado. En este tiempo de escucha debe recibirse una confirmación de trama por parte de la otra estación. Si no fuera así esta trama se retransmitiría. En el caso de que hubiera varias retransmisiones se desistiría en el envío de la trama.

La sencillez del Aloha hace que sufra muchas colisiones al aumentar la carga, consiguiendo una utilización máxima del canal aproximadamente el 18%.

El procedimiento de elaboración de esta simulación se compone de diferentes partes que uniremos para la realización de la simulación. Para esta simulación necesitamos varios componentes: una unidad transmisora, otra receptora y un enlace que será el medio por el cual las comunicaremos. Las unidades transmisoras y receptoras son nodos que utilizamos en nuestra paleta de proyectos para definir un escenario. Para comprender un nodo utilizamos el Node Model. Su funcionamiento lo podemos encontrar detalladamente explicado en la sección que trata del funcionamiento del OPNET en apartados anteriores de este mismo capítulo. Los módulos de proceso de que consta el node model tienen asignados procesos que en algunas ocasiones no están todavía implementados. Por este motivo, tuvimos que realizar nosotros estos procesos

mediante el Process Model. A continuación explicamos detalladamente los pasos realizados para obtener dichos modelos.

### 2.7.1.2.- Unidad transmisora

Primeramente, definimos el proceso que debe realizar la unidad transmisora, todo ello con el Process Model. El proceso que debe seguir un transmisor Aloha es recibir únicamente los paquetes del generador de paquetes y enviarlos por al transmisor. Por lo tanto, nuestro process model tiene dos estados denominamos Idle y tx\_pkt, en función de lo que hacen. El primero, Idle, es un estado no forzado ( unforced ), esto significa que devuelve un control al ejecutar sus ejecutables.

La simulación comienza en el estado Idle donde espera los paquetes entrantes. Al ser no forzado el proceso necesita una interrupción de la simulación para poder comenzar.

En el comienzo de la simulación, el FSM ejecuta el estado Idle, después permanece esperando hasta la llegada de la transición del primer paquete que le llegue.

El estado tx\_pkt es un estado forzado. La única interrupción esperada es el packet arrival interrupt , que indica la llegada de paquetes generados.

Para su realización hicimos lo siguiente. Primero abrimos una nueva página de Process Model y en ella situamos dos estados que denominamos Idle y tx\_pkt.

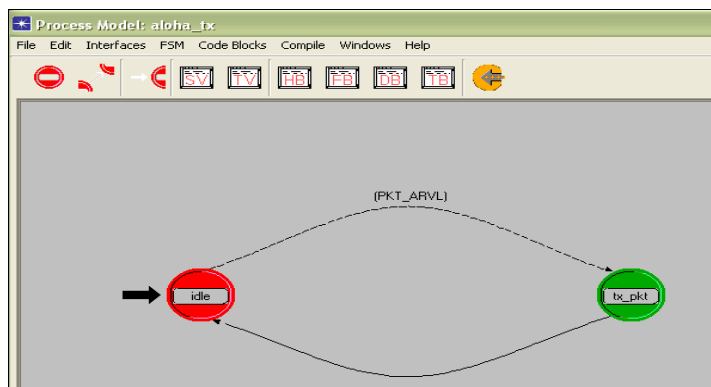


Figura 2.27. Process Model de la unidad transmisora.

Observando la figura 2.27 podemos observar los dos estados, el estado Idle es unforced como podemos distinguir al ver su color rojo, y el tx\_pkt es forced señalizándose con el estado en color verde. La atribución del nombre y estado forzado o no forzado se hizo en el Edit Attributes de cada uno.

Una vez tuvimos ambos estados detallados, nos dispusimos a desarrollar el código del Process Model en el Header Block. En este bloque lo que especificamos son macros para sustituir las expresiones más complicadas de las condiciones y ejecutables de transición. Lo que conseguimos con el uso de estos macros es simplificar la tarea de interpretar un diagrama FSM y también una reducción considerable de espacio.

Para introducir el código abrimos la ventana del Header Block (HB) e introducimos el código apropiado. La figura 2.28 muestra el contenido del Header Block de esta simulación.

```

1  /* Input stream from generator module */
2  #define IN_STRM 0
3  /* Output stream to bus transmitter module */
4  #define OUT_STRM 0
5  /* Conditional macros */
6  #define PKT_ARVL (op_intrpt_type () == OPC_INTRPT_STRM)
7  /* Global variable */
8  extern int subm_pkts;
9
  
```

Figura 2.28. Header Block

Ahora realizaremos un estudio de la reacción que produce este código sobre nuestro proceso.

Definimos IN\_STRM y OUT\_STRM a cero para que el proceso sepa por donde conseguir los paquetes y enviarlos. Para ello le asignamos un flujo, en nuestro caso el flujo cero. Existen un total de 9 flujos posibles (de 0 a 8). Mediante estas líneas de código, nuestro proceso, mediante IN\_STRM conoce la entrada de paquetes, y mediante el OUT\_STRM la salida de paquetes.

Inmediatamente encontramos la condición de macro PKT\_ARVL, ésta determina si una interrupción de paquete se ha producido. Comparando el valor que nos retorna la

función `op_intrpt_type()` con la constante que tiene OPNET de `OPC_INTRPT_STRM` sabrá si se ha producido la interrupción. Si el resultado de esta comparación es `True`, esto nos revela que la interrupción producida es a causa de la llegada de un paquete por el flujo entrante. Como hemos definido anteriormente este análisis sólo lo realiza o afecta al flujo habilitado, en este caso el flujo cero.

Finalmente definimos una variable global que tiene como misión contabilizar el número de paquetes utilizados.

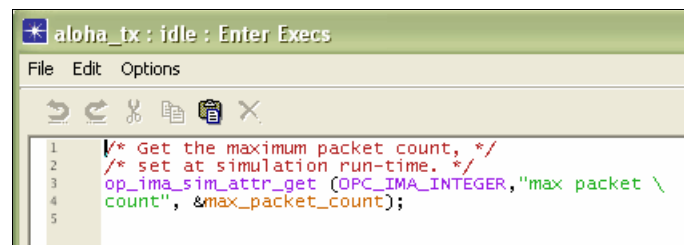
En la figura 2.27 podemos ver la unión de los estados. Estas uniones son transiciones que tuvimos que definir mediante el botón de creación de transiciones. A la transición que une `Idle` con `tx_pkt`, le asignamos la condición `PKT_ARVL` mediante su `Edit Attributes`. Esta condición es la del macro que anteriormente hemos definido en el `Header Block`. A la transición de `tx_pkt` a `idle` no asignamos ninguna condición.

Una vez configuramos nuestras transiciones, creamos los ejecutables que necesitaba el FSM. El código que necesitamos para configurar nuestros ejecutables se denomina `Proto-C`. Hay tres lugares prioritarios donde utilizarlos, éstos son:

- Enter Executive: Es el código ejecutado al entrar en un estado.
- Exit Executive: Es el código ejecutado al salir de un estado.
- Transition Executive: Es el código ejecutado al realizarse una transición.

Para introducir el código del `Enter executive` debemos hacer un doble clic en la parte superior del estado, y en la parte inferior para el `Exit Executive`.

Primero definimos el `Enter Executive` del estado `Idle`. El código que le asignamos es el siguiente, quedando la ventana surgiente como muestra la figura 2.29.



```

1  /* Get the maximum packet count, */
2  /* set at simulation run-time. */
3  op_ima_sim_attr_get (OPC_IMA_INTEGER, "max packet \
4  count", &max_packet_count);
5

```


Figura 2.29. Enter Executive.

Como podemos observar en la figura 2.29, utilizamos la función `op_ima_sim_attr_get (attr_type, attr_name, value_pointer)`. Esta función tiene tres argumentos: el primero `attr_type` nos indica el tipo del atributo. Tenemos los siguientes tipos posibles: `OPC_IMA_INTEGER`, `OPC_IMA_DOUBLE`, `OPC_IMA_TOGGLE`, `OPC_IMA_STRING`. En nuestro caso será un entero ya que le hemos asignado la correspondiente a los enteros.

El siguiente atributo es `attr_name`, en el que se atribuye un nombre. En nuestro caso es `max_packet \count`.

El tercer y último atributo `value_pointer` especifica un puntero a la variable que se llenará con los valores de los atributos.

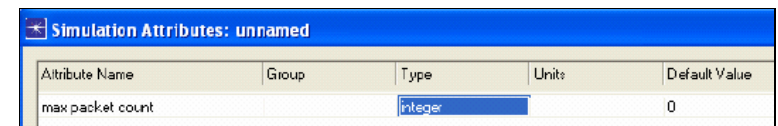
La variable `max_packet_count` que hemos utilizado en `Enter Executive` todavía no está definida. Esta variable tiene como función definir el número máximo de paquetes que se procesarán en toda la simulación. Para declararla utilizamos el **state variable block**, ya que el valor de la variable debe conservarse entre invocaciones. Para realizar este apartado abrimos el `SV (state variable)` e introducimos las características de la variable quedando como se muestra en la figura 2.30.



| Type | Name             | Comments                              |
|------|------------------|---------------------------------------|
| int  | max_packet_count | /* Número de paquete4s a procesar. */ |

Figura 2.30. State Variable.

Consecutivamente, lo definimos en los atributos de la simulación, en **Interface → Simulation Attributes**. La forma de definirlo se observa en la figura 2.31.



| Attribute Name   | Group | Type    | Units | Default Value |
|------------------|-------|---------|-------|---------------|
| max packet count |       | integer |       | 0             |

Figura 2.31. Simulation Attributes.

Ya tenemos definido completamente el estado Idle. Posteriormente realizaremos lo mismo en el estado tx\_pkt.

En primer lugar introducimos el código en el Enter Executive del estado correspondiente. Este código lo presentamos a continuación.

```

/* A packet has arrived for transmission. Acquire */
/* the packet from the input stream, send the packet */
/* and update the global submitted packet counter. */
out_pkt = op_pk_get (IN_STRM);
op_pk_send (out_pkt, OUT_STRM);
++subm_pkts;
/* Compare the total number of packets submitted with */
/* the maximum set for this simulation run. If equal */
/* end the simulation run. */
if (subm_pkts == max_packet_count)
{
op_sim_end ("max packet count reached.", "", "", "");
}

```

Examinando este código encontramos la función `op_pk_get( IN_STRM )`, cuya misión es devolver un puntero al paquete según su flujo de entrada. Si el flujo de entrada no contiene paquetes se devuelve el valor `OPC_NIL`. Una vez hemos obtenido el paquete con la función anterior, le toca el turno a la función `op_pk_send (out_pakt, OUT_STRM)` A ésta le asignamos el puntero de paquetes `out_pkt`. Esta variable es temporal ya que su única misión es adquirir paquetes del generador y los transmite inmediatamente, por esta razón no se necesita guardar la variable.

Prosiguiendo con el análisis del Enter Executive del `tx_pkt` nos encontramos con la función `op_pk_send ( packet_pointer, outstream_index )`, que contiene dos argumentos. El primero especifica el puntero que utilizamos y el segundo especifica el flujo de salida que en nuestro caso es el cero, indicada y definida como `OUT_STRM`.

El paso consecutivo es definir la variable `subm_pkts`. Esta variable se declara en el Header Block y su valor aumenta cada vez que se envía un paquete, es decir, cada vez que se ejecute el código del Heather block su valor se incrementará en uno.

En resumen, el código realiza la obtención del paquete, el envío de éste y la actualización del contador total de paquetes. Después compara el número de paquetes que lleva con el máximo establecido y si es igual se termina la simulación.

Para concluir se utiliza la siguiente función: **op\_sim\_end (line0, line1, line2, line3)**.

Una vez terminamos la realización del código del Heather Block creamos la variable temporal que hemos comentado anteriormente ( `out_pkt` ). Para definir una variable temporal hicimos Click sobre el botón TV (**temporary variable block**), y escribimos el código siguiente dentro de ella: La acción se muestra en la figura 2.32.

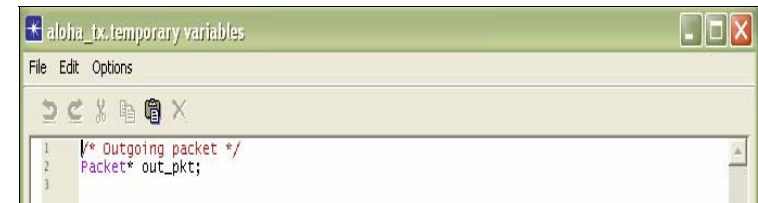


Figura 2.32. Temporary variable Block.

Como posteriormente muestra la figura 2.33 modificamos el **Process Interfaces** para acabar con la definición de esta variable.

| Attribute Name   | Status | Initial Value |
|------------------|--------|---------------|
| begsim intrpt    | hidden | enabled       |
| doc file         | hidden | nd_module     |
| endsim intrpt    | hidden | disabled      |
| failure intrpts  | hidden | disabled      |
| intrpt interval  | hidden | disabled      |
| priority         | hidden | 0             |
| recovery intrpts | hidden | disabled      |
| subqueue         | hidden | (...)         |
| super priority   | hidden | disabled      |

Figura 2.32. Edit Atributes del módulo gen.

Habiendo realizado todos los procesos correctamente compilamos el proceso. Una vez realizada la compilación sin ninguna señal de error podemos guardar el proceso con el nombre `aloha_tx`. Éste lo utilizamos en la realización del módulo transmisor en el Node model.

Para concluir el modelo transmisor realizamos su Node Model, para ello, obviamente abrimos un terminal del mismo. Seguidamente cogimos dos módulos de proceso y los situamos seguidos uno detrás de otro. A continuación colocamos un módulo bus transmitter. Todos estos módulos están unidos entre ellos con el **packet streams**. Quedando la pantalla como la figura 2.34.:

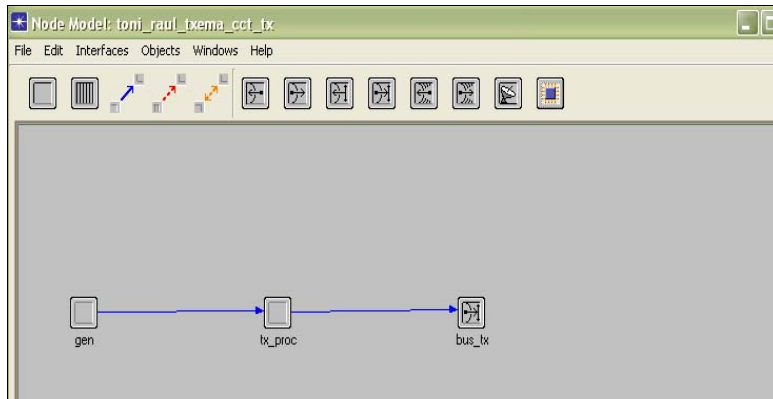


Figura 2.34. Node Model transmisor

Ahora es momento de definir el primer proceso, el del módulo gen. Para ello utilizamos su Edit Attributes, que lo hallamos pulsando sobre éste con un clic en el boton derecho del ratón. Una vez dentro del edit attributes le asignamos un proceso colocando el nombre en el campo nombrado process model. Con esto lo que hacemos es decirle al módulo que realice las acciones descritas en el process model que le hemos asignado. También modificamos el campo con el nombre **Packet Interarrival Time**. Este campo lo colocamos promoted para que se pueda asignar diferentes valores en el tiempo de la simulación.

La figura 2.35 muestra el Edit Attributes modificado:

| Attribute                | Value           |
|--------------------------|-----------------|
| name                     | gen             |
| process model            | simple_source   |
| icon name                | processor       |
| Packet Format            | NONE            |
| Packet Interarrival Time | promoted        |
| Packet Size              | constant (1024) |
| Start Time               | 10.0            |
| Stop Time                | Infinity        |

Figura 2.35. Edit Attributes del módulo gen.

El process model que le hemos asignado es una simple\_source. Éste se conoce vulgarmente como una fuente simple. OPNET nos proporciona ya este process model. La figura 2.36 muestra cómo es el process model de la fuente simple.

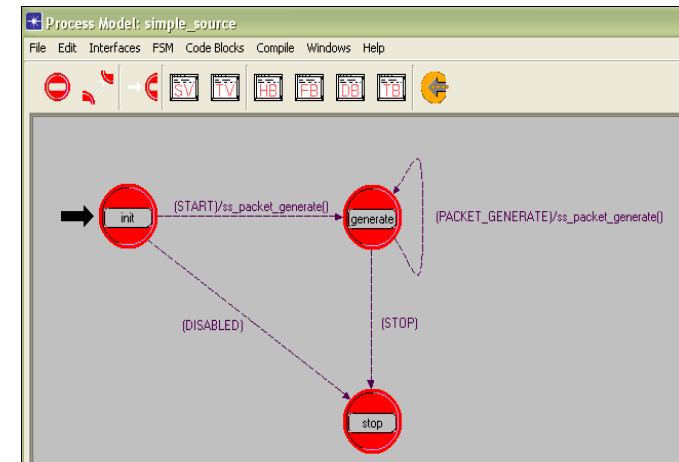


Figura 2.36. Process Model de la fuente simple.

Al siguiente módulo le nombramos tx\_proc y le asignamos el process model que anteriormente hemos realizado y que le nombramos aloha\_tx. Una vez realizado esto, el edit attributes del módulo tx\_proc se mostrará como la figura 2.37.

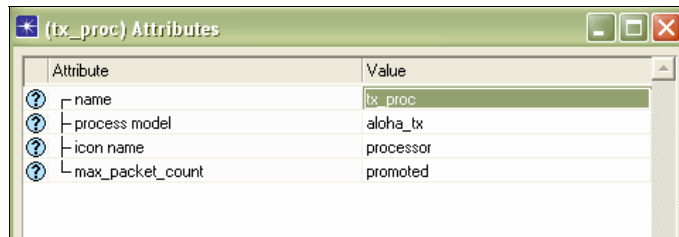


Figura 2.37. Edit Attributes de tx\_proc.

El bus transmitter es nombrado bus\_tx.

A continuación modificamos el Node interfaces para que el nodo no fuera móvil ni satélite y cambiamos los estados de todos los atributos a hidden excepto el de gen.packet Interarrival Time que lo pusimos promoted.

Para finalizar guardamos el modelo de nodo con el nombre de módulo \_ transmisor.

### 2.7.1.2.- Unidad receptora

La unidad receptora de nuestro aloha es responsable de utilizar los paquetes recibidos con el objetivo de conseguir una estadística final. Es decir, su función es la de contar los paquetes y conseguir las diferentes estadísticas con objetivo de supervisar el buen funcionamiento de la red y comprobar si los paquetes están siendo transmitidos de acuerdo con el protocolo Aloha.

Paso principal para crear el receptor es la realización del Process Model. Este proceso registra el rendimiento del procesamiento del canal y los valores para el análisis en la simulación final.

Por ello, abrimos una ventana de Process Model y añadimos dos estados que nombramos init y idle. El estado Init es un estado forzado y el idle no forzado. En la ventana debe ser dispuesta como muestra la figura 2.38.

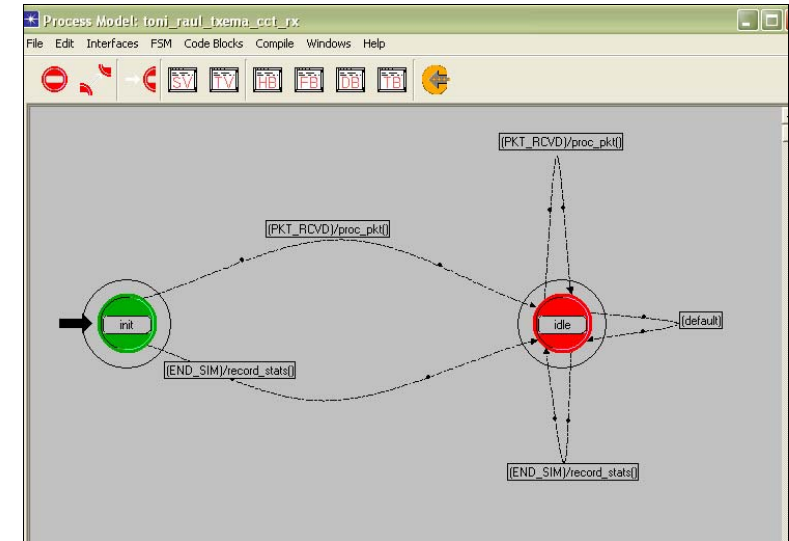


Figura 2.38. Process Model del receptor.

Posteriormente configuramos su Header Block introduciéndole el código siguiente:

```

/* Input stream from bus receiver */
#define IN_STRM 0
/*Conditional macros */
#define PKT_RCVD (op_intrpt_type () == OPC_INTRPT_STRM)
#define END_SIM (op_intrpt_type () == OPC_INTRPT_ENDSIM)
/* Global variable */
int subm_pkts = 0;

```

Logramos observar la definición de un flujo de entrada que es el cero, para coincidir con el del transmisor. El macro **PKT\_RCVD** sirve para detectar los paquetes recibidos. El macro **END\_SIM** determina si la interrupción recibida es la de finalizar la simulación.

Por último declaramos una variable global subm\_pkts que la inicializamos a cero.

Esta variable coincide con la del módulo transmisor.

Seguidamente, establecimos el Enter Executive del proceso INIT. En éste, solamente se inicializa el acumulador y por lo tanto el Enter Executive debe cumplimentarse de la forma expuesta a continuación.

```
/* Initialize acumulador */
rcvd_pkts = 0;
```

Rcvd\_pkts es una variable global y por lo tanto hay que declararla como vimos en el anterior apartado con el SV.

La figura 2.39 muestra como queda el SV:

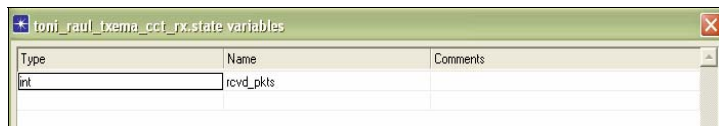


Figura 2.39. State Variables.

Una vez hemos realizado los pasos anteriores, tenemos que definir dos funciones necesarias para el funcionamiento del sistema. Una primera que reciba los paquetes y una segunda que escriba estadísticas en un archivo. Para ello utilizamos el Function Block y colocamos en él el siguiente código.

```
/* This function gets the received packet, destroys */
/* it, and logs the incremented received packet total*/
static void proc_pkt (void)
{
Packet* in_pkt;
FIN (proc_pkt());
/* Get packet from bus receiver input stream */
in_pkt = op_pk_get (IN_STRM);
/*Destroy the received packet */
op_pk_destroy (in_pkt);
/* Increment the count of received packet */
++rcvd_pkts;
```

```
FOUT;
}
/* This function writes the end-of-simulation channel */
/* traffic and channel throughput statistics to a */
/* scalar file */
static void record_stats (void)
{
double cur_time;
FIN (record_stats());
cur_time = op_sim_time();
/* Record final statistics */
op_stat_scalar_write ("Channel Traffic G",
(double) subm_pkts / cur_time);
op_stat_scalar_write ("Channel Throughput S",
(double) rcvd_pkts / cur_time);
FOUT;
}
```

En la primera parte del código correspondiente a la función **proc\_pkt()** conseguimos el paquete recibido, lo destruimos y después incrementamos el registro de paquetes recibidos. En la segunda parte del código correspondiente a la función **record\_stats()** escribimos las estadísticas del rendimiento del canal.

Inmediatamente unimos mediante transiciones los estados, quedando como se muestra en la figura 2.38.

La primera transición es desde el estado INIT al estado idle, y le asignamos la condición **PKT\_RCVD** y el executive attribute **proc\_pkt()**. Esto significa que cada vez que se ejecute esta transición se ejecutará la función **proc\_pkt()** realizada anteriormente en el Function Block (FB).

La segunda transición tiene el mismo origen y destino. En el Edit Attributes de la transición asignamos la condición **END\_SIM** y el executive attribute **record\_stats()**.

Seguidamente realizamos 3 autoalimentaciones en el estado idle. En la primera le asignamos la condición **PKT\_RCVD** y el executive attributes **proc\_pkt()**, en la siguiente le asignamos la condición default y en la última su condición es **END\_SIM** y su executive attribute **record\_stats()**.

Para finalizar definimos el Process Interfaces, que debe visualizarse de forma que nos expone la figura 2.40.

| Attribute Name   | Status | Initial Value |
|------------------|--------|---------------|
| begsim intrpt    | hidden | disabled      |
| doc file         | hidden | nd_module     |
| endsim intrpt    | hidden | enabled       |
| failure intrpts  | hidden | disabled      |
| intrpt interval  | hidden | disabled      |
| priority         | hidden | 0             |
| recovery intrpts | hidden | disabled      |
| subqueue         | hidden | (...)         |
| super priority   | hidden | disabled      |

Figura 2.40. Process Interfaces.

Para concluir este Process Model compilamos y guardamos con el nombre toni\_raul\_txema\_cct\_rx.

Una vez realizado el proceso del receptor, tenemos que producir el nodo receptor apropiado en el Node Model. Éste consta de un módulo genérico y un módulo bus receiver. Unimos los nodos con un packet stream. A continuación les cambiamos el nombre para que fuera más entendible. Al primero le denominamos rx\_proc y su process model es el generado anteriormente, es decir, el toni\_raul\_txema\_cct\_rx. Al bus receiver le denominamos bus\_rx. La forma con la que le asignamos el Process Model y lo nombramos es mediante la configuración del Edit Attributes. Teniendo una muestra en la figura 2.41.

| Attribute     | Value                  |
|---------------|------------------------|
| name          | rx_proc                |
| process model | toni_raul_txema_cct_rx |
| icon name     | processor              |

Figura 2.41. Edit Attributes de rx\_proc.

La pantalla del Node model debe presentarse como se muestra en la figura 2.42.

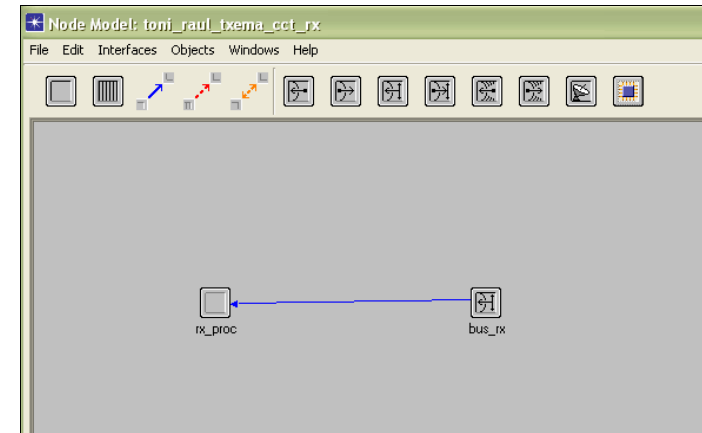


Figura 2.42. Node Model receptor.

Para concluir modificamos el Node Interfaces como mejor nos convenga y acorde a nuestras necesidades.

Para ello deshabilitamos las opciones de nodo móvil o satélite y sólo permitimos la opción de un nodo fijo. A continuación cambiamos todos los estados de sus atributos a hidden. Estos pasos se ven reflejados en la figura 2.42.

| Node Type | Supported | Default Icon |
|-----------|-----------|--------------|
| fixed     | yes       | fixed_comm   |
| mobile    | no        |              |
| satellite | no        |              |

| Attribute Name    | Status | Initial Value               |
|-------------------|--------|-----------------------------|
| TIM source        | hidden | none                        |
| altitude          | hidden | 0.0                         |
| altitude modeling | hidden | relative to subnet-platform |
| condition         | hidden | enabled                     |
| financial cost    | hidden | 0.00                        |
| phase             | hidden | 0.0                         |
| priority          | hidden | 0                           |
| user id           | hidden | 0                           |

Figura 2.42. Modificaciones de los atributos

Una vez hemos realizado los cambios oportunos en el Node interface ya podemos guardar este Node Model con un nombre que posteriormente logramos identificar como tal. Nosotros le denominamos módulo \_ receptor.

### 2.7.1.4.- Link Model

En este apartado vamos a definir un enlace acorde a nuestras necesidades. Este enlace será el que una las unidades transmisoras con la unidad receptora.

Para ello, abrimos un Link Model y en éste modificamos los apartados referentes a los tipos de enlace soportables:

Sólo soporta los tipos de enlace bus y bus tap, de esta forma nos cercioramos de que sólo se conectara a este tipo de enlace, bus.

El resto de especificaciones las dejamos tal y como están por defecto.

Las especificaciones quedaron de la siguiente forma que muestra la figura 2.44.

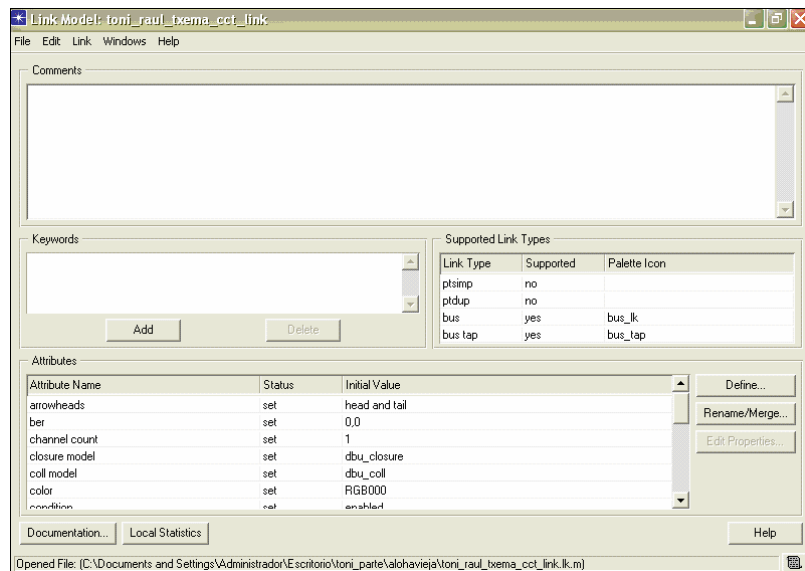


Figura 2.44. Link Model.

Después de realizar los cambios pertinentes guardamos este enlace generado por nosotros con un nombre que nos facilite su identificación. Nosotros lo guardamos como toni\_raul\_txema\_cct\_link.

### 2.7.1.5.- Creación de la red mediante el Network Model

El objetivo de este proceso es el análisis del protocolo, que lo conseguimos definiendo los nodos anteriormente construidos. La introducción de los paquetes en el aloha se realiza de una forma exponencial en el tiempo. Nosotros pusimos un número de nodos finito, este número de nodos que se halla en el bus no debe ser pequeño para así completar un buen análisis.

Para realizar la simulación comenzamos abriendo una nueva parte del OPNET llamada Project. A continuación guardamos el archivo con el nombre del proyecto y un nombre del escenario. Nosotros le denominamos toni\_raul\_txema\_cct\_network y el nombre del escenario es aloha.

Posteriormente entramos en un Wizard para la configuración del escenario y colocamos los siguientes valores de forma progresiva.

| Dialog Box Name      | Value                                       |
|----------------------|---|
| Initial Topology     | Default value: <b>Create Empty Scenario</b> |
| Choose Network Scale | <b>Office</b> (Usamos unidades métricas)    |
| Specify Size         | <b>700 m x 700 m</b>                        |
| Select Technologies  | <b>None</b> (las introducimos nosotros)     |
| Review               | Verificamos y <b>OK</b>                     |

Posteriormente configuramos la paleta con los nodos y enlaces a utilizar, que en nuestro caso son los nodos y enlace anteriormente realizados.

Para ello efectuamos los siguientes pasos: **Configure Palette -> Clear** para borrar los actuales.


Hacemos click en **Node Models** e incluimos los realizados por nosotros.

Posteriormente hacemos click en **Link Model** e incluimos los enlaces necesarios para la simulación.

Después de hacer esto ya tenemos nuestra paleta configurada, la guardamos, y proseguimos nuestra simulación.

Para colocar los nodos en el escenario utilizamos una configuración rápida de red que encontramos en **Topology** → **Rapid Configuration**. Escogemos Bus, ya que es la topología que utilizamos, también seleccionamos la opción node model del nodo receptor que nosotros realizamos y la opción link model análogamente.

Tras aceptar, nos apareció el bus con sus nodos transmisores. Posteriormente, desde la paleta seleccionamos nuestro nodo receptor y lo colocamos a la izquierda del bus. Ahora sólo falta unir los nodos transmisores con el nodo receptor. Éstos los unimos

mediante el enlace siguiente  desde el bus al receptor. El escenario final es el que muestra la figura 2.45:

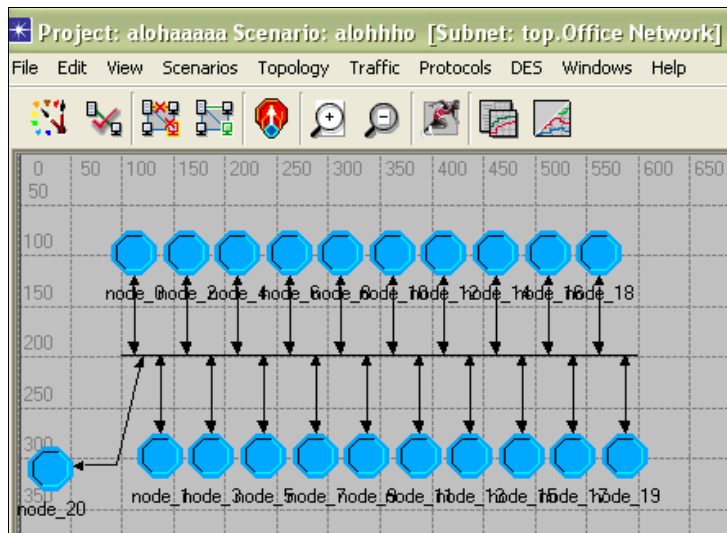


Figura 2.45. Escenario final.

Concluidos todos los pasos, nos disponemos a guardar el proyecto con el nombre que surgía por defecto y que era el que habíamos dispuesto al principio.

### 2.7.1.6.- Realización del análisis

Sin haber cerrado el archivo de proyecto que hemos configurado en el apartado anterior, nos dispusimos a configurar la simulación.

Desde **Simulation** → **Configure Discrete Event Simulation (Advanced)**

Nos surge una nueva pantalla donde se realiza el análisis, en ella aparece este símbolo



A continuación hacemos clic en el Botón derecho y seleccionamos Edit Attributes.

Inmediatamente aparece una pantalla de configuración de la simulación donde se pueden definir las variables de simulación, los archivos de salida, etc... La figura 2.46 muestra como es esta pantalla.

Colocamos la duración de la simulación a 20000 segundos.

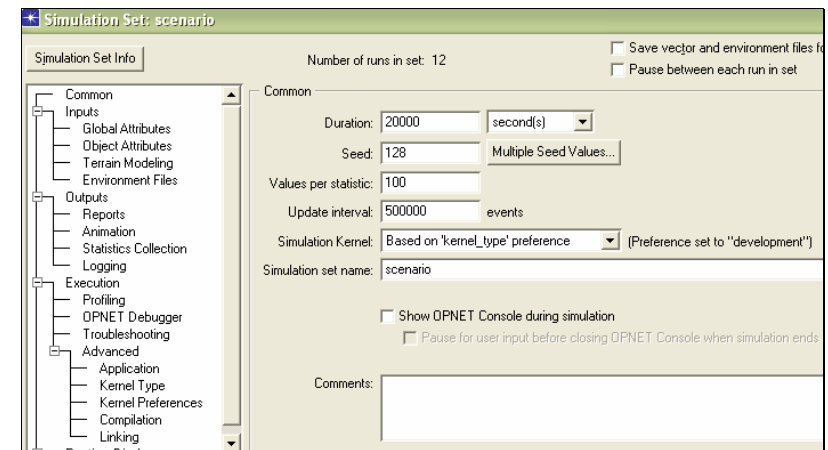


Figura 2.46. Configuración de la simulación.

Dentro de la opción **Global Attributes** asignamos el valor 1000 al max packet count de la forma indicada por la figura 2.47. Esto hace que el número máximo de paquetes de la simulación sea 1000.

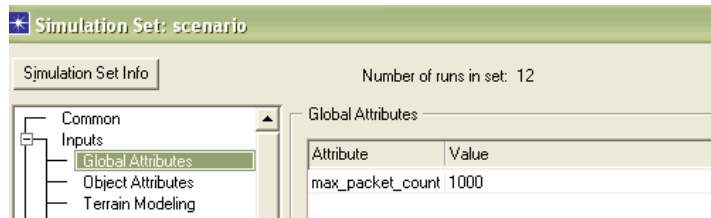


Figura 2.47. Global Attributes.

Posteriormente en Object Attributes adjuntamos el Office Network. \*.gen.Packet Interarrival Time, la mitología para adjuntarlo, se muestra en la figura 2.48. A éste le asignamos el valor en exponencial(1000), y asignamos más valores al mismo 200, 150, 100, 80, 50, 35, 30, 25, 20, 18 y 15. Quedando de la forma que se reproduce en la figura 2.49:

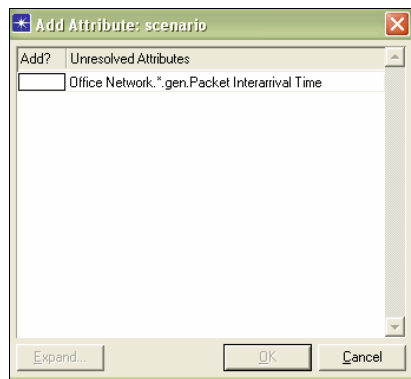


Figura 2.48. Adquisición de Office Network \*.gen.Packet

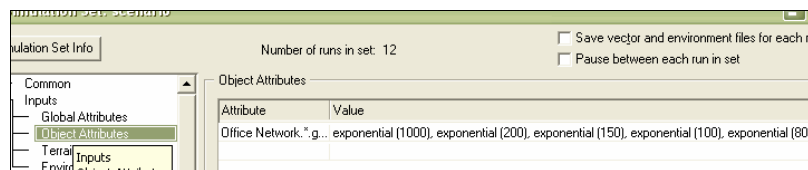


Figura 2.49. Asignación de los valores.

Por último, en la opción Advanced asignamos la realizada por nosotros, en Probe file NONE y en Scalar file añadimos el nombre del archivo de salida aloha\_a. Este último archivo es el que genera la simulación y fue el que abrimos para observar la gráfica posteriormente. Todas las características se ven reflejadas en la figura 2.50.

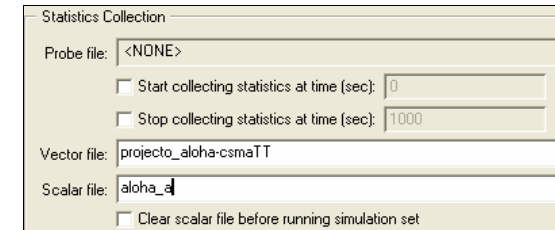


Figura 2.50. Statics Collection.

Antes de simular borramos las simulaciones anteriores en **File → Model Files → Delete Model Files**, escogemos los outputs scalars y los borramos.



En este momento estamos en disposición de ejecutar la simulación pulsando **Analysis Configuration**.



Dentro del nuevo programa, seleccionamos **Create graph of two scalars file** y asignamos al Canal de tráfico G a la horizontal y el canal Throughput a la vertical. Hacemos click en aceptar y nos aparece la gráfica deseada, ésta se muestra en la figura 2.51.

Una vez obtenida la gráfica, vamos a estudiar los resultados obtenidos con el fin de poder evaluar si la simulación es correcta o por el contrario no lo es.

El rendimiento del sistema Aloha según su tráfico lo podemos calcular con la fórmula siguiente:  $S = Ge^{-2G}$ . Siendo S el rendimiento del proceso y G la función del tráfico. Con ella podemos calcular un rendimiento máximo del canal de 0.18, ya que viendo la gráfica obtenida consideramos que el máximo rendimiento (Max Throughput) es de  $G = 0.5$  aprox., un 18%, tal y como la teoría nos indicaba antes de realizar la simulación.

Esto nos indica que en los valores bajos de tráfico las colisiones no se suceden continuamente, y en los niveles altos, este canal se recarga, las colisiones se producen en exceso y evita que los paquetes no sean recibidos.

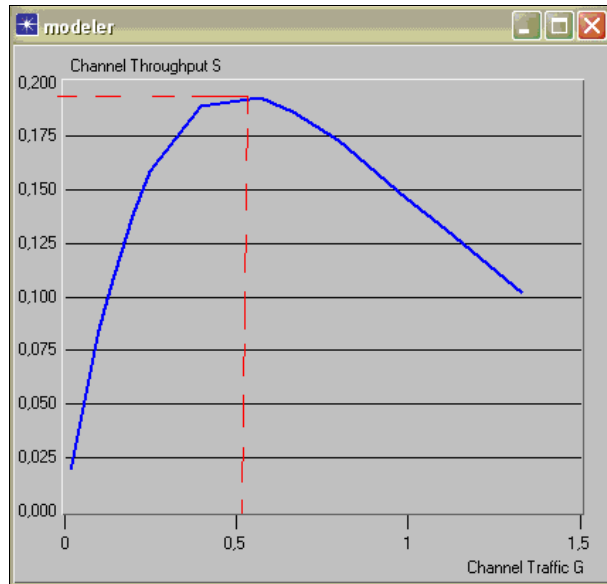


Figura 2.51. Resultado del rendimiento.

Los resultados teóricos asumen esencialmente un número infinito de fuentes para eliminar los efectos del buffering que emergerían en una red real. El modelo analítico también asume que el sistema está en una condición ideal del estado constante. Cualquier diferencia en el funcionamiento medido de este modelo y los modelos analíticos se puede atribuir a las particularidades del seed del número random seleccionado para las simulaciones individuales.

## 2.7.2.- Expansión de la intranet de una pequeña compañía

### 2.7.2.1.- Introducción

En este ejemplo proponemos la simulación de una oficina de dos plantas. Hipotéticamente consideramos que la empresa se expande y quiere ampliarse conllevando un aumento también de sus oficinas. Esta planta es insuficiente para las necesidades que requiere la empresa, ya que tiene un alto rendimiento. En este apartado se propone la incorporación de una segunda planta y cómo afecta ésta al funcionamiento de la intranet. Primero analizamos la primera planta, que es donde actualmente tienen una red tipo estrella con 29 terminales. Seguidamente construiremos la red de la posterior planta, las uniremos y contemplaremos los cambios experimentados.

Como podemos observar, esta simulación es parecida al planteamiento de una actividad típica de una clase de XALO, donde el objetivo era la ampliación de redes. Lo que conseguimos en este ejemplo es ver los resultados de estas modificaciones y observar el impacto que puede ocasionar el incremento de éstas, antes de realizarlo, para facilitar entonces el estudio del impacto de estas ampliaciones.

### 2.7.2.2.- Realización de la primera red

Comenzamos como anteriormente hemos mencionado analizando la primera planta. Para ello abrimos un proyecto nuevo y lo nombramos Toni\_Raul\_SM\_Int. Al escenario le ponemos el nombre de Primera\_Planta.

Seguidamente aparece un Wizard para poder configurar de forma rápida y sencilla nuestro escenario. Para realizar este Wizard elegimos **Create empty scenario**, el tipo de escenario que viene referido por la escala será el de **Office**. Utilizamos las unidades métricas, 100m x 100m, y elegimos la tecnología que utilizamos en la implementación de la planta. Esta tecnología es **Sm\_Int\_Model\_List**. A diferencia de la simulación hecha en el apartado anterior, en ésta, los modelos ya están implementados en OPNET y nosotros sólo tenemos que colocar estos modelos en el escenario y configurarlos.

Podemos revisar las características del Wizard una vez introducido los elementos para comprobar que todo sea correcto. La figura 2.52 muestra este wizard:

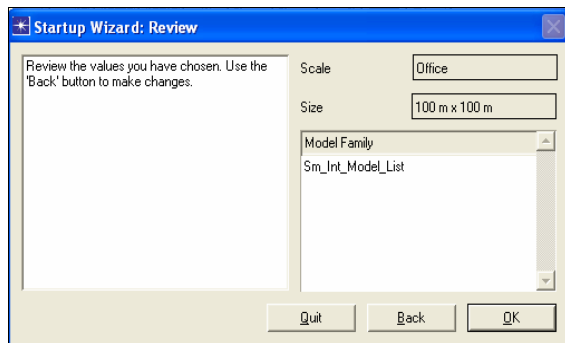


Figura 2.52. Wizard de configuración del escenario.

Para crear la primera red utilizamos la configuración rápida. La configuración de la red es en estrella. Los modelos que hemos colocado en la simulación son los siguientes: el modelo del nodo central es un **3C\_SSII\_1100\_3300\_4s\_ae52\_e48\_ge3**, el modelo de los nodos periféricos es **Sm\_Int\_wkstn**, y el modelo de enlace que hemos utilizado es el **10BaseT**. Nosotros hemos colocado 30 nodos. El emplazamiento que utilizamos es el siguiente: **X** e **Y** igual a **25** y un **radio** de **20 m**. Con esto conseguimos situarlo en el borde superior izquierdo.

La figura 2.53 muestra la pantalla de rapid configuration que hemos explicado anteriormente:

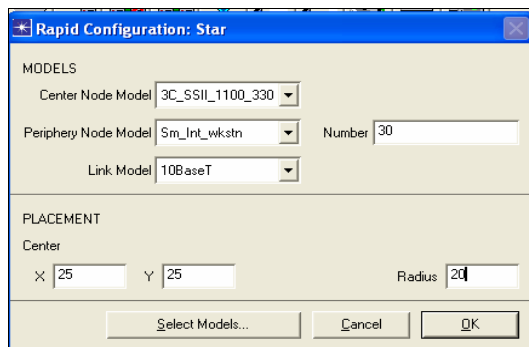


Figura 2.52. Rapid Configuration.

Anteriormente, hemos utilizado una nomenclatura para definir los modelos que seguidamente utilizamos en nuestra simulación. Pero antes explicaremos un poco la nomenclatura y el funcionamiento de los susodichos.

#### - Ethernet2\_bridge\_int:

La nomenclatura utilizada es la consiguiente: primero se colocan los protocolos soportados, que pueden ser uno o varios. A continuación, viene la función general del modelo y acaba mostrando el nivel de derivación.

La forma generalizada que estila OPNET para asignar los nombres a los modelos alcanza la siguiente estructura:

#### **Protocolo1\_ .Protocolo2\_ ..... \_ProtocoloN\_Función\_mod**

Analizando el modelo nombrado anteriormente respecto a esta estructura podemos distinguir los siguientes campos:

**Int** significa intermedio.

**Ethernet2** significa derivación de un segundo puerto Ethernet.

**Bridge** quiere decir que el modelo hace la función de puente.

#### - **3C\_SSII\_1100\_3300\_4s\_ae52\_e48\_ge3:**

Este es nuestro modelo del nodo central.

El nodo pila que contiene dos 3Com SuperStack II 1100 y dos Superstack II 3300 chasis → **3C\_SSII\_1100\_3300**.

También podemos observar que tiene 4 slots → **4s**.

Tiene 52 puertos Ethernet auto-sensados → **ae52**.

Tiene 48 puertos Ethernet → **e48**.

Finalmente tiene 3 puertos Gigabit Ethernet → **ge2**.

El enlace que utilizamos es el 10BaseT, este enlace trabaja bajo una topología estrella o en árbol. Por lo tanto es idóneo para nuestro escenario.

La longitud máxima de los segmentos es de 100 metros y utiliza cables de pares trenzados de categoría 3 y 5.

Una vez realizada la configuración rápida de red, la topología general estará construida y nos aparece en la parte superior izquierda de nuestro escenario. El siguiente paso es la colocación de un servidor que encontramos en la paleta. Lo colocamos a la derecha de la topología es el **Sm\_Int\_Server**.

Para conectar el servidor colocado a nuestra red en estrella utilizamos el enlace **10BaseT** comentado anteriormente. Una vez situado configuramos las características del sistema tales como el tráfico. Para ello añadimos unas aplicaciones de

configuración. Los objetos de las aplicaciones que hemos elegido son los siguientes: **Sm\_Application\_Config** y **Sm\_Profile\_Config**. Estos modelos los podemos hallar en la paleta de modelos. La primera aplicación sirve para la configuración de la aplicación y la segunda configura el perfil de la aplicación. No les someteremos a ninguna modificación, ya que sus valores por defecto son los requeridos.

Situamos estas aplicaciones justo debajo del servidor quedando el escenario creado como se muestra en la figura 2.54.

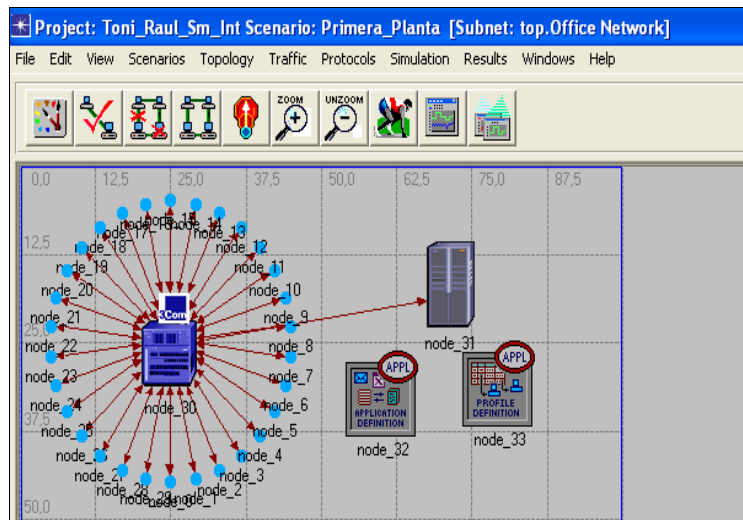


Figura 2.54. Escenario LAN.

A continuación es el turno de la configuración de las estadísticas que deseamos adquirir. Éstas se pueden subdividir en si requerimos un análisis de un nodo en concreto (**object statistics**), o por lo contrario de la red al completo (**global statistics**).

Para una correcta simulación, estas estadísticas que deseamos obtener nos deben solucionar las dudas que nos pueden surgir al ampliar la red. Estas dudas o cuestiones a analizar son si el servidor puede soportar otra red adjunta o no y si el retraso total con la segunda red es aceptable.

Opnet nos facilita una fácil obtención de la solución a nuestros análisis. Para obtener la carga que va a soportar el servidor necesitamos obtener una estadística de Server Load. Esta estadística es de objeto servidor ( **object statistic** ), ya que como

hemos mencionado anteriormente es el estudio sobre un único nodo. En cambio, para el estudio del retardo, que es global para toda la red, se complementa con un global statistic, siendo éste el Ethernet Delay.

Para obtener el object statistic nos situamos sobre el nodo del servidor, y efectuamos click en el botón derecho. Inmediatamente nos aparecen varias opciones, entre las cuales está la **Choose Individual Statistics**. Dentro de este apartado hemos elegido **Ethernet** y dentro de este **Load (bits/sec)**. La figura 2.55 muestra el proceso explicado anteriormente:

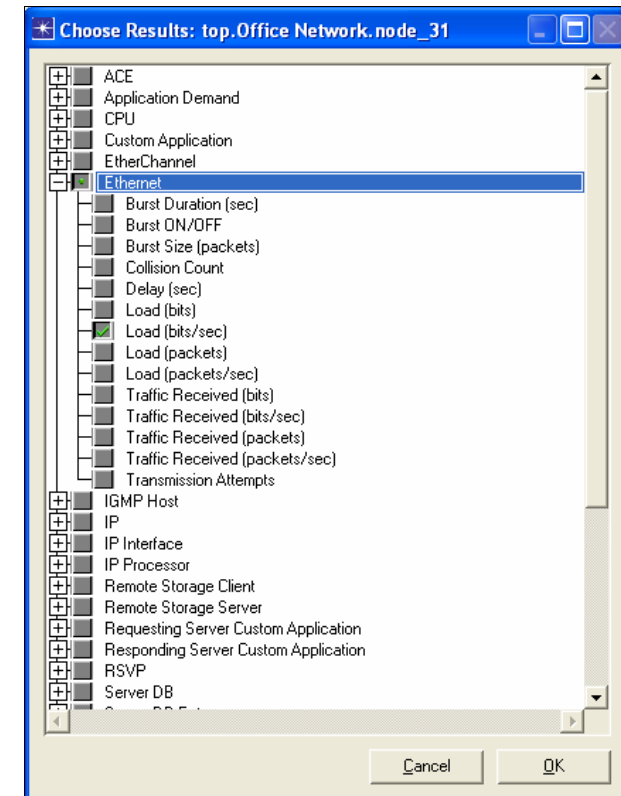
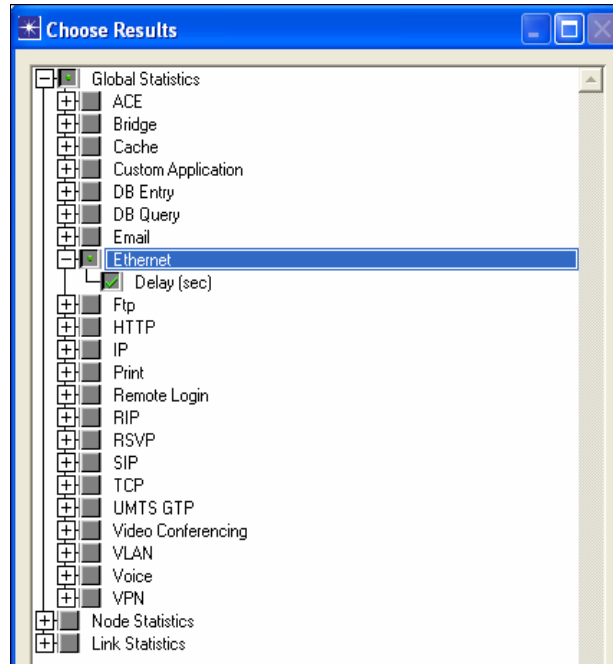


Figura 2.55. Estadísticas elegidas.

Una vez realizado hemos concluido la configuración del simulador para que nos facilite la gráfica con la carga del servidor. El siguiente paso es la asignación de las

estadísticas globales. La global statistic se realiza en un punto libre de nodos o enlaces. Pulsamos el botón derecho y elegimos Choose Individual Statistics. Seguidamente hemos elegido Global Statistic, seleccionamos Ethernet y por último Delay. La figura siguiente muestra este proceso:



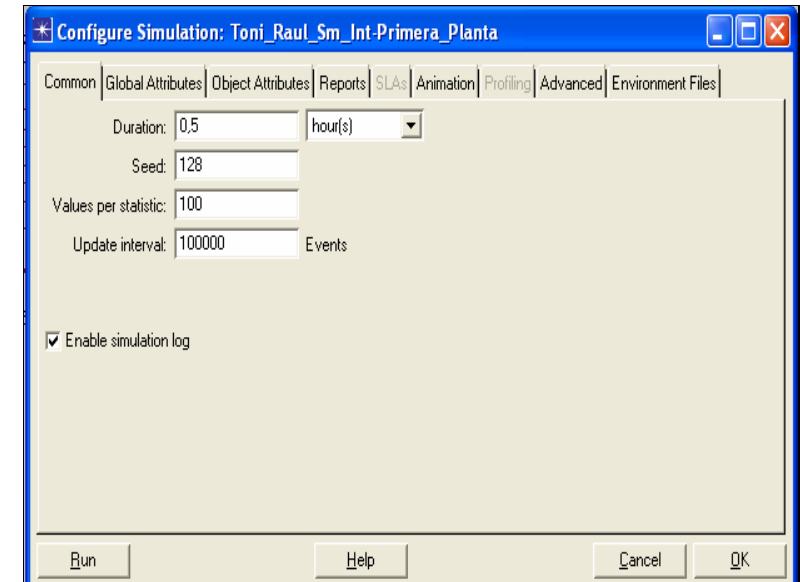
**Figura 2.56.** Elección de la estadística del Retardo.

Una vez realizado todo esto, guardamos nuestro escenario y podemos realizar ya la simulación.

### 2.7.2.2.- Simulación de la primera red

Para realizar la simulación abrimos Simulation → Configure Discrete Event Simulation....

El único parámetro que modificamos será la duración que disponemos a media hora, quedando la pantalla de simulación como mostramos en la figura 2.57.

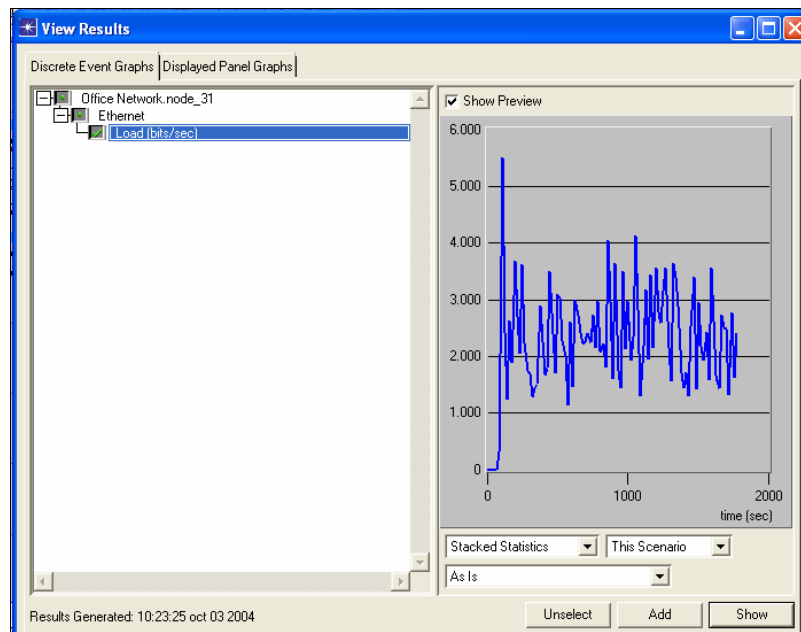


**Figura 2.57.** Configurate Discrete Event Simulation.

Una vez realizado, accionamos el botón Run para que OPNET comience la simulación. Sucesivamente surge una pantalla de su estado. Cuando finaliza, cerramos la ventana y nos dirigimos otra vez a nuestro proyecto desde donde observamos los resultados obtenidos.

Los resultados los observamos de una forma directa. Para el Server Load, nos dirigimos al nodo del servidor, hacemos click en el botón derecho y en **View Results**. En la pantalla que surge, habilitamos la opción Ethernet y seguidamente la Load (bits/s).

Haciendo esto aparece una gráfica como la que mostramos en la figura 2.58.

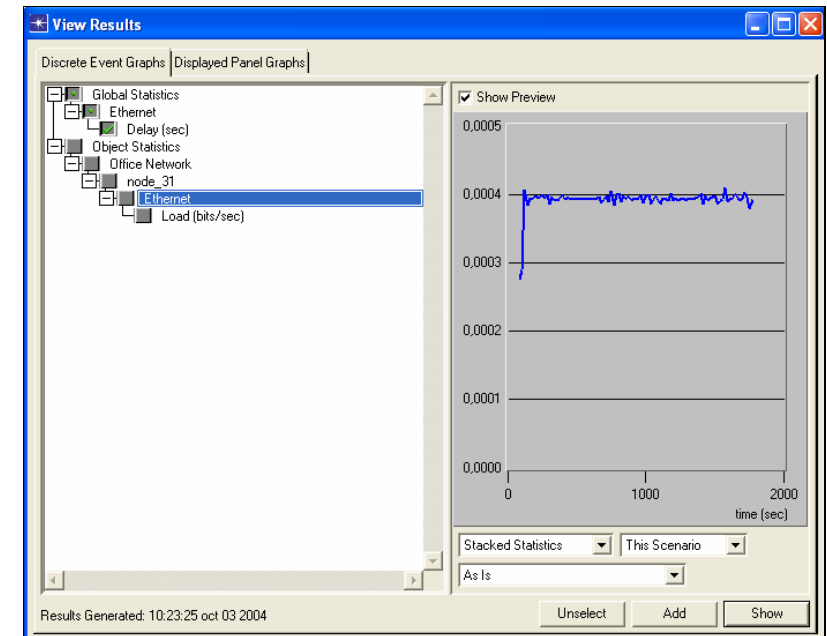


**Figura 2.58.** Resultado de la Carga.

Esta gráfica muestra la carga a la que se somete el servidor estudiado. En la gráfica, podemos observar un pico al principio y seguidamente se estabiliza dentro de unos límites. Estos límites rondan entre los 2000 bits y 4000 bits. La carga máxima a la que puede ser sometido este servidor es de 10M y como podemos observar en el resultado obtenido no se sobrepasa este valor en ningún momento. Esto nos indica el buen uso de este servidor, y su correcta utilización.

Para visualizar la otra gráfica del retardo, nos colocamos sobre alguna zona libre de objetos dentro del escenario, pulsamos botón derecho y escogemos **View Results**. Entonces elegimos **Global Statistics, Ethernet y Delay**.

Una vez realizado nos proporciona una gráfica con el resultado obtenido en la simulación que se muestra en la figura 2.59.



**Figura 2.** Resultado del retardo.

En la gráfica proporcionada por OPNET, podemos observar un establecimiento del retardo en la red con un valor que es máximo. El valor máximo del retardo es 0.4 milisegundos.

### 2.7.2.4.- Realización de la expansión de la red existente

Una vez realizada la simulación de la red que actualmente tendría la oficina, nos debemos disponer a realizar la red completa con todas sus ampliaciones. El objetivo de esta simulación es demostrar que el aumento de las instalaciones no limita a éstas en su funcionamiento. En este caso observamos si la carga (Load) a la que se somete el servidor no es excesiva y si el retardo total que encontramos en la red no aumenta.

Para la realización de este apartado, cogemos de base el escenario Primera\_Planta que hemos realizado anteriormente. Este escenario lo duplicamos para no perder su análisis individual y para que la simulación sea más rápida. Para duplicar el escenario

efectuamos click sobre la opción **Scenarios → Duplicate Scenario** y lo nombramos **Expansión**.

La expansión constituye un aumento de las oficinas y consiste en la adquisición de una planta superior en la cual construiremos una nueva red que consta de 14 terminales más. El servidor que administra esta nueva red es el mismo que administra la red ya existente y por lo tanto un dato importante es saber si este servidor puede seguir trabajando de forma óptima a pesar de la expansión de la red.

Nuestro primer paso a seguir para el análisis de la expansión es construir esta nueva red. Para ello utilizamos la herramienta que anteriormente hemos utilizado para la realización de la anterior red, es decir, usamos la configuración rápida de red **Topology → Rapid Configuration**.

Los modelos que utilizamos en esta nueva simulación son los mismos, exceptuando el número de terminales que esta vez son **15** y su situación es **X 75** e **Y 62,5**. Estos parámetros lo que hacen es situarlos en la zona inferior izquierda de nuestro escenario. Dichos modelos no cambian ya que no habrá un cambio de protocolos, ni de configuración de la red.

La configuración rápida se muestra en la figura 2.60.

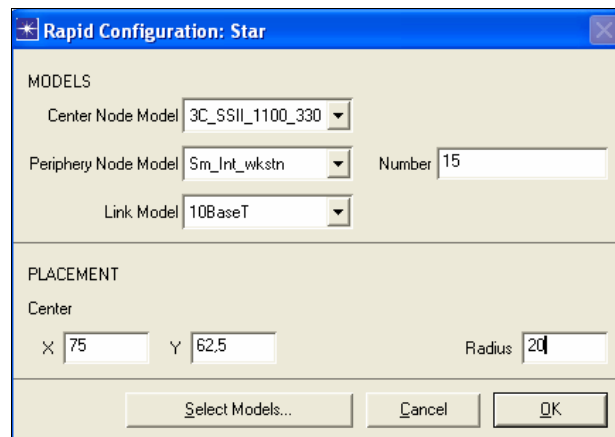


Figura 2.60. Configuración rápida para la expansión.

Una vez realizado, disponemos de dos redes en nuestro escenario. Para unir las necesitamos un nuevo componente llamado router. Por lo tanto, seleccionamos de nuestra paleta un router de la marca Cisco 2514.

Concretamente el modelo que hemos seleccionado se llama CS\_2514\_1s\_e2\_sl2.

Este modelo corresponde a un router de la marca CISCO, que representa un router que puede soportar 2 interfaces hub ethernet y 2 interfaces serial IP. Los paquetes IP llegan al router y son enrutados hacia la salida dependiendo de la dirección IP del paquete.

Este modelo lo situamos en una zona intermedia entre las dos redes y las unimos con un enlace 10BaseT. Los terminales a unir son los nodos centrales, en nuestro caso el nodo 30 y el nodo 49. Éstos nodos son los nodos centrales que administran sus redes, podemos decir que son Switchs que trabajan a nivel 2 y que ofrecen conectividad a la red.

El escenario completo de la simulación es como el que muestra la figura 2.61.

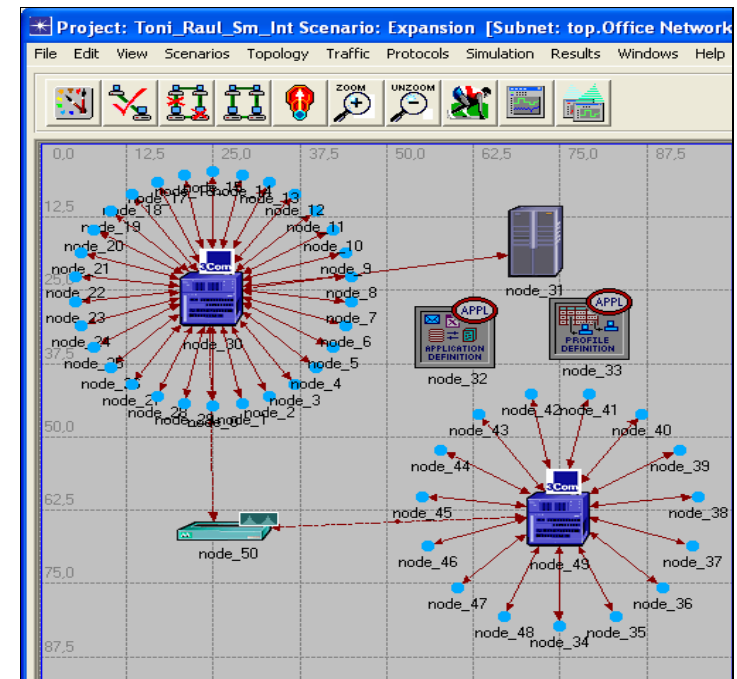


Figura 2.61. Escenario final.

### 2.7.2.5.- Simulación de la expansión de la red existente

Una vez tenemos completada la fase de la expansión, realizamos el análisis. No hemos modificado ninguna condición ya que cuando hemos duplicado el escenario se han guardado las inicializaciones para realizar las estadísticas. Por lo tanto ya podemos realizar la simulación. Hacemos click sobre **Simulation → Configure Discrete Event Simulation....** y **Run**.

El último paso para concluir la simulación es la comparación de los resultados de las dos simulaciones realizadas, y la conclusión sobre si es satisfactoria la expansión con los parámetros estudiados.

Por lo que se refiere a la carga del servidor, que tenemos ilustrada en la gráfica Load, nos situamos sobre el servidor, pulsamos el botón derecho sobre éste y elegimos **Compare Results**. El resultado de la comparación es el que podemos observar en la figura 2.62.

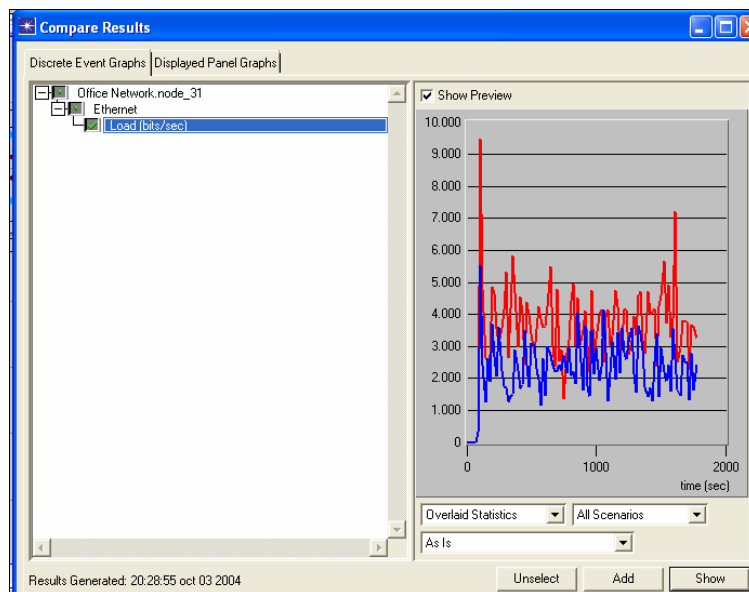


Figura 2.62. Comparación de la carga (Load).

Podemos observar en la comparativa que como era esperado, la carga a que se somete el servidor es mayor si aumentamos la red a la cual ofrece su servicio. Pero aún así, no parece que el aumento de la red haya sobrecargado en exceso el servidor. Se pueden observar un pico de carga elevado al principio de la simulación pero este pico tiende a estabilizarse a medida que avanza. Podemos decir que el rango de la carga en esta segunda simulación es de 2000 bits a 6000 bits.

Por lo que se refiere al retardo de la red, pulsamos con el botón derecho en una zona sin ningún objeto y elegimos **Compare Results**. El resultado ofrecido por OPNET se observa en la figura 2.62.

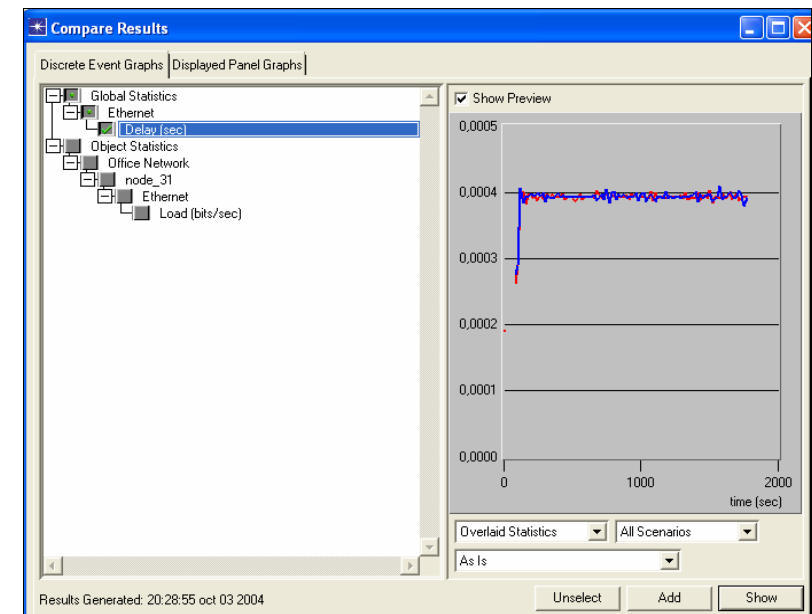


Figura 2.62. Comparativa del retardo.

Como podemos observar en la gráfica, el retardo total de la red no se ha visto afectado por el aumento de la misma.

Como conclusión a esta simulación y después de haber estudiado los resultados obtenidos, podemos afirmar que la expansión de la red en la empresa no afectará al rendimiento de los equipos ya instalados en la empresa y por tanto se puede realizar sin ningún tipo de problema.

## 2.7.2.- Simulación de una red Ethernet mediante el simulador OPNET

Como hemos visto en las diferentes simulaciones realizadas en este capítulo, OPNET nos ofrece un conjunto de librerías de gran utilidad para realizar nuestras simulaciones sin tener que realizar nosotros los modelos.

Como muestra de ello, en este apartado, nos centramos en uno de los protocolos más importantes que existe hoy en día como es el protocolo ethernet.

### 2.7.2.1.- Ethernet en OPNET

OPNET nos ofrece una gran variedad de modelos para poder simular redes ethernet con gran facilidad.

Accedemos a ellos abriendo un nuevo proyecto y cuando nos demanda seleccionar la topología para la simulación nos aparecen los siguientes modelos. La figura 2.64 muestra el wizard de seleccionar tecnología.

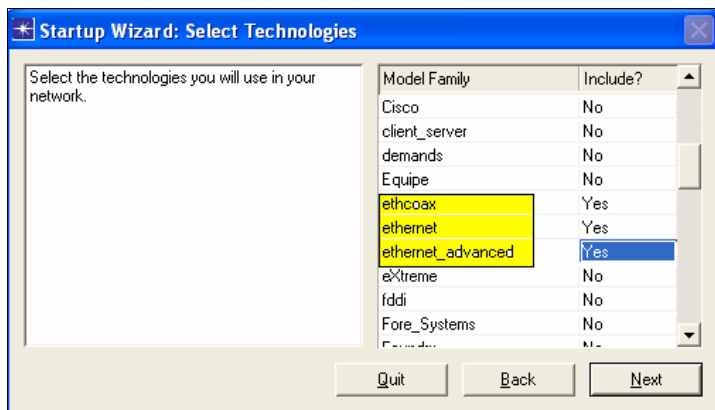


Figura 2.64. Selección de tecnologías.

Una vez seleccionadas las tecnologías deseadas podemos comenzar a realizar el escenario de la simulación.

Ahora la paleta del proyecto contiene los modelos ethernet que hemos seleccionado, que son los que muestra la figura 2.65.

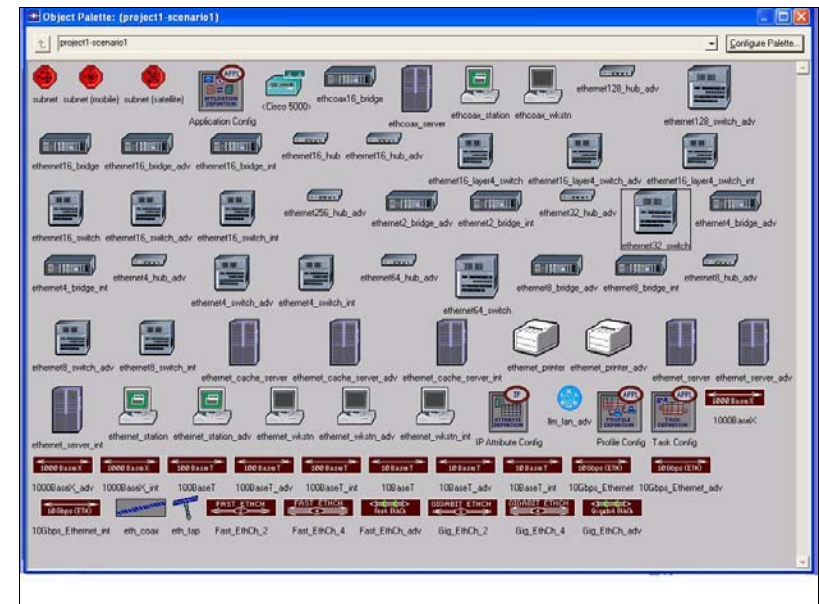


Figura 2.65. Modelos Ethernet.

En la paleta podemos observar que aparecen todo tipo de componentes que utilizan la tecnología ethernet (servidores, bridges, estaciones de trabajo, switch, enlaces, impresoras, etc...). Gracias a estos modelos podemos estudiar el comportamiento de una red ethernet sin la necesidad de realizar ésta físicamente.

### 2.7.2.2.- Realización del escenario

El objetivo de esta simulación es enseñar las diferentes gráficas que OPNET nos proporciona en el momento de evaluar una red que utilice la tecnología ethernet.

Para ello abrimos un nuevo proyecto. Creamos una oficina donde albergar los equipos a analizar. Las dimensiones que situamos son de 100 x 100 metros. Una vez realizado,

habilitamos el tipo de tecnología deseada, en nuestro caso utilizamos las tecnologías ethernet que hemos comentado anteriormente.

Ahora ya estamos en disposición de realizar el escenario oportuno.

Para ello colocamos cuatro estaciones ethernet que actúan como transmisoras de datos. Éstas pueden ser conectadas a:

- Otra estación ethernet
- Un hub, bridge o switch ethernet

El protocolo soportado es el IEEE 802.3 (Ethernet, Fast Ethernet, Gigabit Ethernet).

Una vez colocadas las estaciones transmisoras, las conectamos todas a un switch (ethernet16\_switch)

El switch utilizado puede soportar 16 estaciones ethernet conectadas a él. Los protocolos que soporta son el Spanning Tree Bridge Protocol (IEEE 802.1D) y Ethernet (IEEE 802.3).

A continuación colocamos un servidor, en concreto el ethernet\_server. Éste representa un nodo servidor que trabaja con aplicaciones TCP/IP y UDP/IP. Los protocolos soportados por éste son RIP, UDP, IP, TCP, Ethernet, Fast Ethernet, Gigabit Ethernet y OSPF.

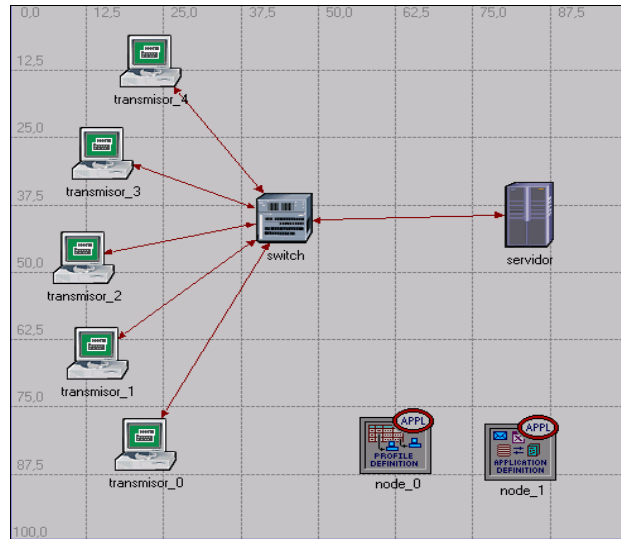


Figura 2.66. Escenario obtenido.

Por último ubicamos dos tipos de aplicaciones como hicimos en la simulación de la expansión de la red de una compañía. Estas aplicaciones son Profile conf. y task conf. Todos los enlaces de nuestra red son de 100Mbps pero podríamos colocar perfectamente enlaces de 10, 100 o 1000 Mbps sin ningún tipo de problemas.

La figura 2.66 muestra el escenario de la simulación.

### 2.7.2.2.- Simulación del escenario

Una vez hemos realizado el escenario de la simulación, el consecuente paso es configurar los resultados que deseamos obtener.

Esto lo conseguimos en **Simulation** → **Choose individual statistics**.

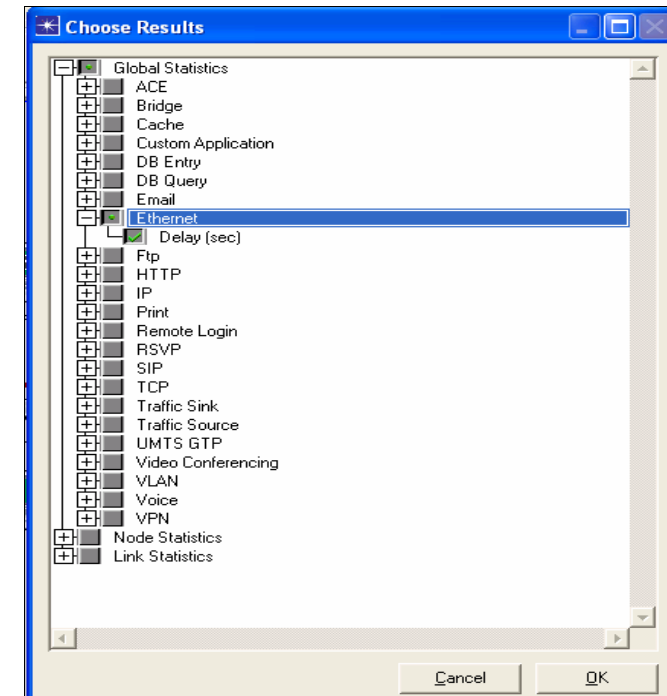


Figura 2.67. Estadísticas seleccionadas.

Al realizarlo nos aparece una ventana donde vemos todas las estadísticas que el programa nos ofrece. Las marcadas con un punto verde son las estadísticas que nosotros seleccionamos para que OPNET recolecte durante la simulación.

Estas estadísticas se dividen en:

- Estadísticas globales: Que son las estadísticas válidas para todo el escenario, es decir, para todos los modelos.

Como podemos observar en la figura 2.67 nosotros hemos seleccionado la estadística global de retardo de la red.

Para marcar las estadísticas de las estaciones ethernet, del servidor y de los enlaces nos colocamos encima de estos elementos, con el botón derecho abrimos un menú donde aparece la opción choose individual statistics y hacemos click en esta opción.

Una vez marcadas las estadísticas deseadas nos disponemos a realizar la simulación. Por lo tanto abrimos el menú de su configuración y colocamos un tiempo de 2 horas. Posteriormente hacemos clic en run.

A continuación analizaremos los resultados obtenidos en esta simulación.

- Retardo total de la red mostrado en la figura 2.68.:

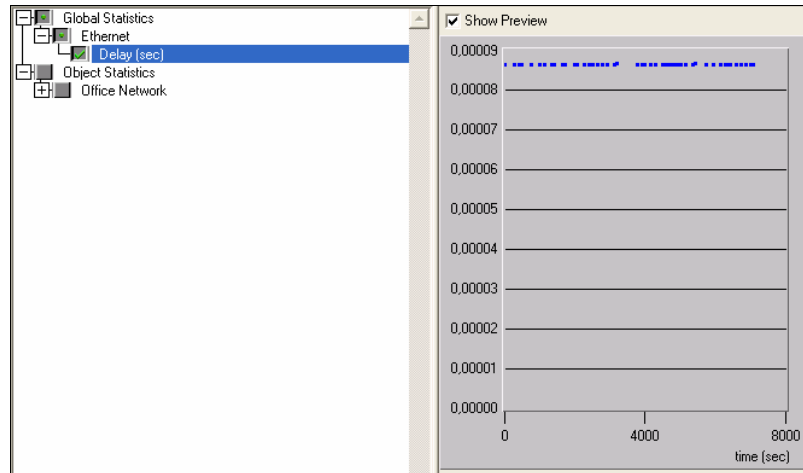


Figura 2.68. Retardo de la red.

Como podemos observar el retardo producido en la red es de 85µs aprox.

- Estaciones transmisoras que podemos observar en la figura 2.69:

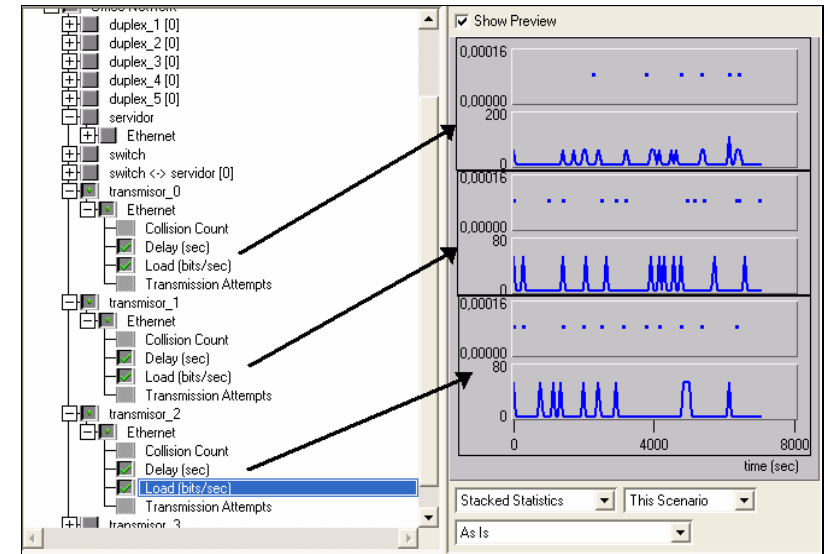


Figura 2.69. Retardo y carga de las estaciones transmisoras.

En esta gráfica observamos la carga y el retardo de los transmisores 0, 1 y 2, y también el retardo que se produce en cada transmisor, que es de 80µs aprox. Este retardo se aproxima al obtenido para toda la red, que era de 85µs.

Además podemos observar la carga en cada transmisor que es aproximadamente de 80 bits/s.

- Switch:

La figura 2.70 muestra el tráfico recibido y enviado por el switch durante la simulación.

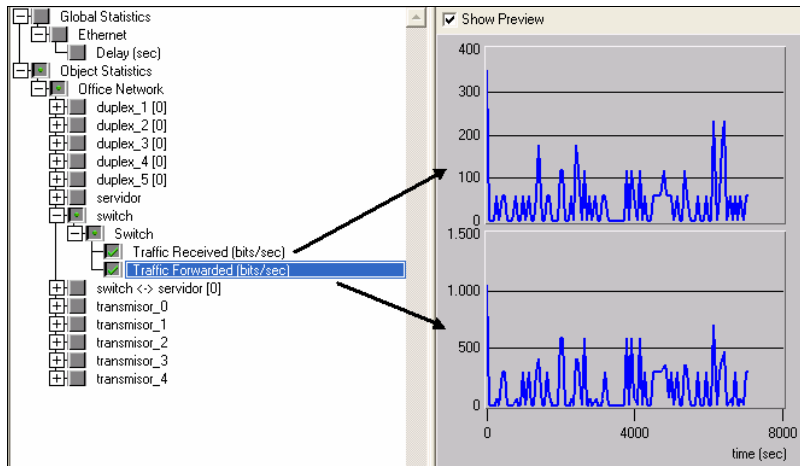


Figura 2.70. Tráfico soportado por el Switch.

En las gráficas visualizadas en la figura 2.71 observamos cuándo está el enlace ocupado o no, mediante la grafica de arriba y la de abajo, y el throughput de cada enlace.

Como hemos podido ver OPNET nos ofrece una herramienta de simulación potente, ya que dispone de una gran variedad de estadísticas resultantes ya realizadas.

En este ejemplo podemos cerciorarnos de que el programa OPNET se nos ofrece como una herramienta potente y cómoda para analizar redes. En este caso, esta comodidad mencionada se refleja en hallar todos los modelos y necesidades para la realización de la simulación.

- Enlaces de la red:

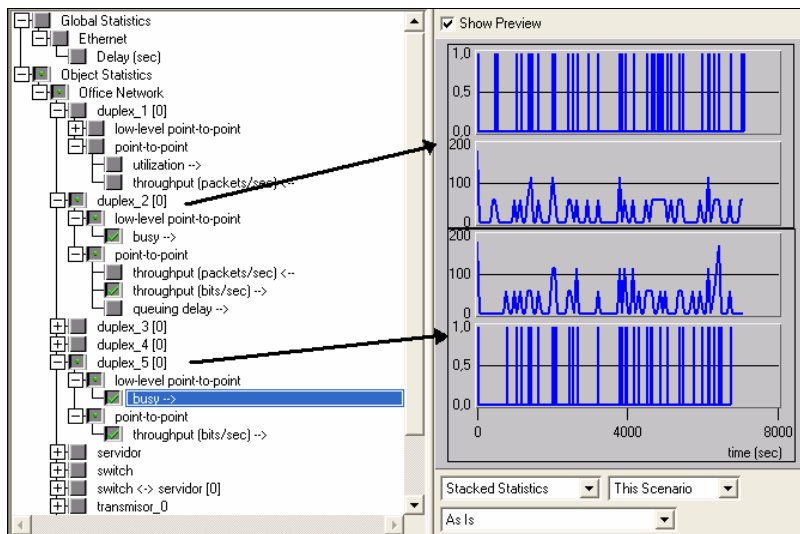


Figura 2.71. Rendimiento y ocupación de los enlaces de la red.