Universidad Pública de Navarra

Área de Ingeniería Telemática

Aquitectura de Redes Sistemas y Servicios

"Conceptos de simulación y OPNET"

ÍNDICE

INTRODUCCIÓN

La herramienta básica que se empleará en el desarrollo de estas prácticas será el simulador de redes Opnet. Antes de comenzar con las prácticas en sí, resulta imprescindible que el alumno cuente con unas nociones básicas acerca de qué implica la utilización de un simulador y qué parámetros habrán de tenerse en cuenta a la hora del diseño de los modelos empleados y de las simulaciones realizadas, para garantizar una fiabilidad de los resultados obtenidos.

1. Modelos y simulación. Qué y para qué.

Definimos simulación como una técnica que imita el comportamiento de un sistema del mundo real conforme evoluciona en el tiempo. Mediante ella podremos, por tanto, analizar y observar ciertas características del sistema, sin necesidad de acudir al sistema real, nos bastará con el análisis del modelo que lo representa.

Surgen, pues, dos nuevas definiciones:

a) 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.

b) Y Proceso de simulación que será la ejecución del modelo a través del tiempo en un ordenador para generar muestras representativas del comportamiento del sistema.

En definitiva, simulación hará referencia a la técnica, modelo de simulación a la representación del sistema real que vamos a analizar, las condiciones de su funcionamiento y las variables que emplea, y el proceso de simulación será una ejecución concreta, con unos valores asociados a las variables que se pueden ajustar en el modelo, que se realizará para obtener los resultados referidos a unos ciertos parámetros que especifican el comportamiento del sistema.

Cabe señalar que la simulación no es una técnica de optimización, esto es, no proporciona el dimensionamiento de las variables del sistema que maximizan el rendimiento o las prestaciones del mismo, sino que únicamente se limita a informar de cuál sería el comportamiento del sistema analizado en las condiciones que se indiquen para el proceso de simulación.

Habrá de tenerse en cuenta que la simulación se basa en el muestreo aleatorio, lo que traerá consigo que la salida de la simulación, los resultados que de ella se extraigan, esté sujeta a variaciones aleatorias y por ello, haya de ser examinada, utilizando para tal fin, pruebas de inferencia estadística, que evalúen si tales resultados son o no fiables y representan fielmente el comportamiento del sistema o no tienen validez alguna.

Continuamente nos estamos refiriendo al término “sistema”. Pero, ¿qué cabe considerar como sistema? Entenderemos por sistema, cualquier colección de elementos que actúan e interactúan para lograr algún fin lógico. De este modo, al tratarse de una

definición poco restrictiva, podremos agrupar bajo este término, objetos tan diversos como: supermercados, hospitales, redes de caminos, empresas, modelos económicos, y lo que centra nuestro interés, redes de computadores.

Asimismo, cuando hablemos de estado del sistema estaremos haciendo referencia al conjunto de variables necesarias para describir el estado del sistema en un determinado instante de tiempo.

Entre estas variables distinguiremos las entradas y las salidas. Las salidas de la simulación 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 las 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.

- Y los datos probabilísticos: cantidades cuyos valores son inciertos pero necesarios para obtener las salidas de la simulación. Los valores específicos de estos datos deben conocerse a través de una distribución de probabilidad.

2. Clasificación y Tipos de Simulación

Existen diversos aspectos en función de los cuales podemos clasificar las distintas simulaciones. En este apartado, se mostrarán algunas de estas posibles clasifica-ciones.

a) Estática vs. Dinámica.

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

Una simulación dinámica, por su parte, es la representación de un sistema cuando evoluciona con el tiempo.

A lo largo de nuestras prácticas, únicamente emplearemos simulaciones dinámicas en Opnet, observando como se comportan las redes que diseñemos con el transcurso del tiempo.

b) Determinista vs. aleatoria.

Un modelo de simulación se dice determinista si no contiene absolutamente ninguna variable aleatoria.

El modelo de simulación estocástico será aquél que sí contenga una o más variables aleatorias.

En nuestras prácticas, el número de variables aleatorias con las que contemos en el diseño del modelo variará de unas a otras, pero en todas ellas existirá alguna varaible aleatoria, sea el tiempo entre llegadas, el tamaño de los paquetes, el tiempo de comienzo de la simulación...

c) Continua vs. Discreta.

Los modelos continuos de simulación serán aquéllos cuyo comportamiento cambia de forma continua con el tiempo. Para describir las interacciones entre los elementos del sistema se suele requerir de ecuaciones diferenciales (existentes debido a la condición de continuidad).

En caso contrario, diremos que el modelo de simulación es discreto, lo que ocurrirá si el comportamiento cambia únicamente en instantes de tiempo concretos, eventos.

A lo largo de nuestras prácticas, comprobaremos como el tipo de simulación que implementa Opnet es una simulación de eventos discretos. Este tipo de simulación se emplea para describir situaciones de colas. Es más, cualquier modelo de eventos discretos está formado por una red de colas interrelacionadas (que en nuestros ejemplos serán las colas de los distintos dispositivos que constituyan la red). Los dos principales eventos son la llegada y la salida (en nuestras prácticas llegadas y salidas de paquetes). La aleatoriedad surge si el intervalo de tiempo entre dos eventos (sea llegada o salida) consecutivos es probabilístico (lo cual puede deberse a tamaños de paquetes o a tiempos entre peticiones aleatorios).

3. Pasos a seguir para una correcta simulación:

Con vistas a conseguir realizar una simulación provechosa que nos informe de lo que realmente nos interesa acerca del modelo y garantice una cierta fiabilidad de los resultados, tendremos que seguir los pasos que se nos indican:

a) 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.

b) A continuación se ha de proceder con la creación del modelo. En nuestro caso, el diseño de la red que hemos de analizar.

c) Posteriormente, se habrá de diseñar un programa de ordenador para el modelo. Aquí, nosotros ya contaremos con nuestro simulador de redes, Opnet.

d) Habremos de verificar el programa y validar el modelo.

e) Ya estamos en disposición de utilizar el modelo para experimentar y contestar a las preguntas que inicialmente se nos planteaban.

f) Finalmente, tendremos que reunir, procesar y analizar los datos generados como soluciones del modelo y en términos de validez y fiabilidad estadística.

4. Fiabilidad de los resultados obtenidos. Intervalos de confianza.

No hemos de olvidar que una sola simulación proporciona un único valor de entre muchos posibles resultados, que puede ser distinto en cada simulación, como consecuencia del carácter aleatorio de algunas de sus variables. Por ello, tendremos que acudir a las técnicas de inferencia estadística para establecer el número de simulaciones requerido para proporcionar un nivel deseado de confianza. Estas técnicas tendrán que utilizarse 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. Intervalo éste, que 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 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.

Aunque la determinación de estos intervalos de confianza resulte complicada, sí resulta evidente la siguiente observación:

Si la variable de salida (léase, por ejemplo, el retardo sufrido por un paquete), se obtiene como la media de un conjunto de salidas obtenido para una determinada muestra, será tanto más fiable cuanto mayor sea el tamaño muestral (esto es, el número de paquetes de los que se ha estudiado el retardo, para calcular el retardo medio). De este modo, en las simulaciones que desarrollaremos en nuestras prácticas tendrán una relevancia considerable parámetros de la simulación tales como tiempo de simulación y número de eventos.

5. Tiempo de simulación y número de eventos.

Estos dos parámetros propios de la simulación estarán 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 de simulación resulta posible simular el comportamiento que tendría un sistema en un largo periodo de tiempo empleando para ello un tiempo considerablemente menor. De manera intuitiva, se puede apreciar como

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 y de igual forma que para el tiempo de 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.

En nuestras prácticas, esto podremos ajustarlo, dando valores adecuados al tiempo que transcurra entre la generación de dos paquetes consecutivos y a la duración de la simulación, de forma que se garantice que se han observado un número suficiente de paquetes (eventos) que haga posible la extracción de conclusiones

Simulador 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...

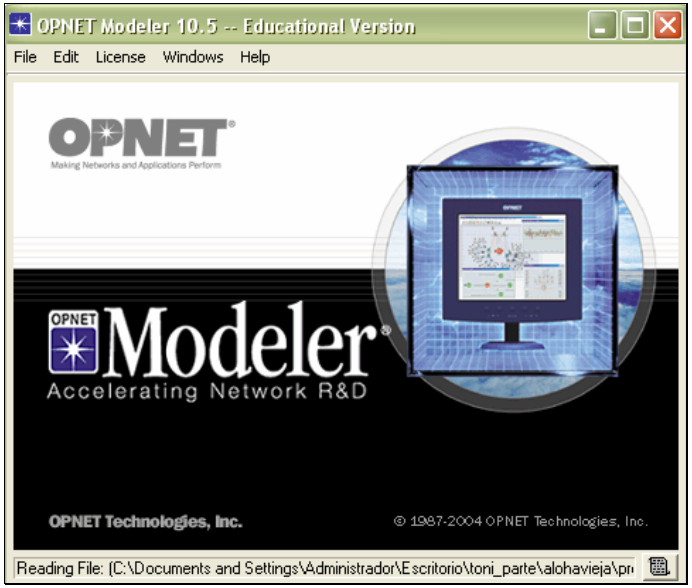


Figura 2.1. Encabezado del Simulador OPNET

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.

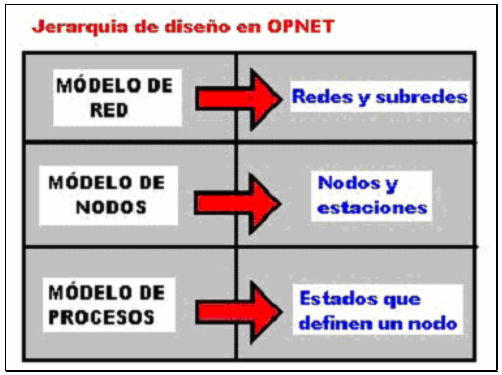


Figura 2.2. Jerarquía de diseño en OPNET

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 figura2.3:

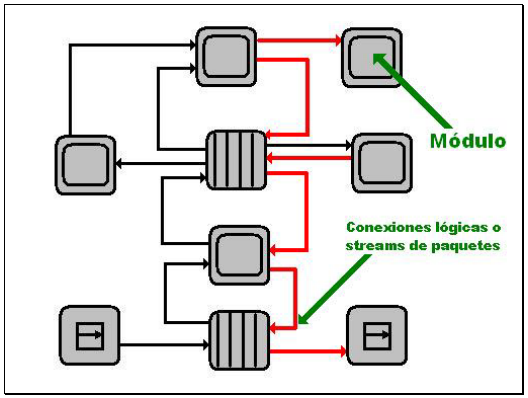


Figura 2.2. Funcionamiento del modelo de Nodos.

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.

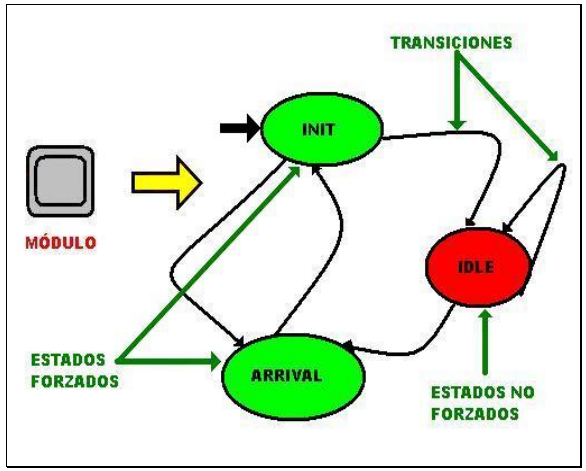


Gráfico 2.6. Funcionamiento del módulo de nodos.

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: Instaura Estados Temporales para el process model
	Edit Header Block: Instaura el Header Block donde se declaran las variables, macros, tipos de datos, etc.., incluidos en el process model.
	Edit Function Block: Instaura el Function Block en el cual se crean nuevas fucniones 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

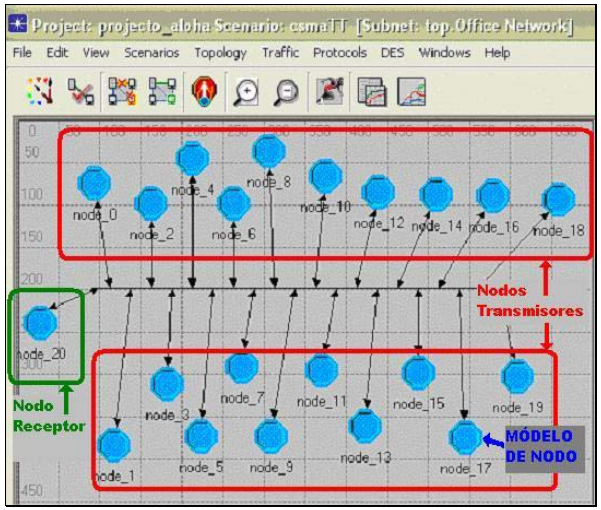


Figura 2.8. Ejemplo de Modelo de red.

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

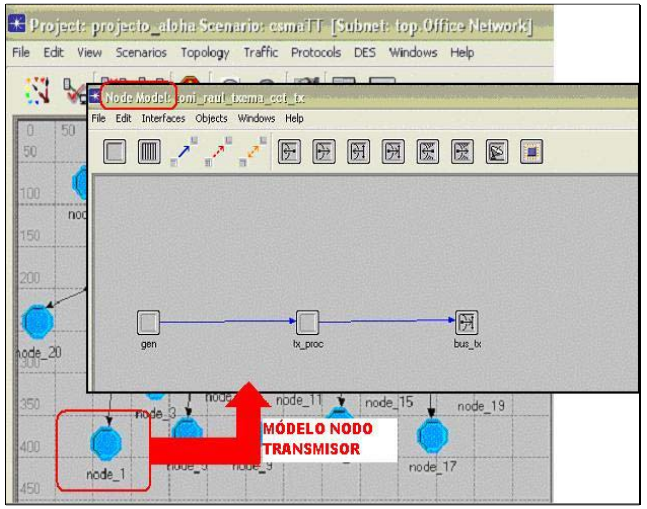


Figura 2.9. Ejemplo de Modelo de nodo Transmisor

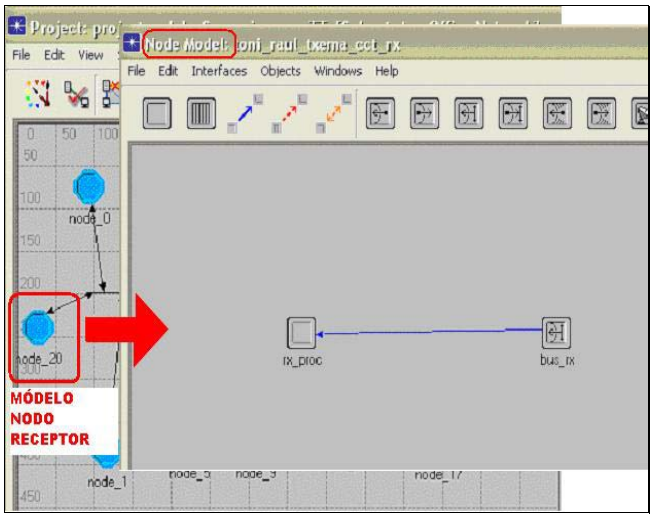


Figura 2.10. Ejemplo de Modelo de nodo Receptor.

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

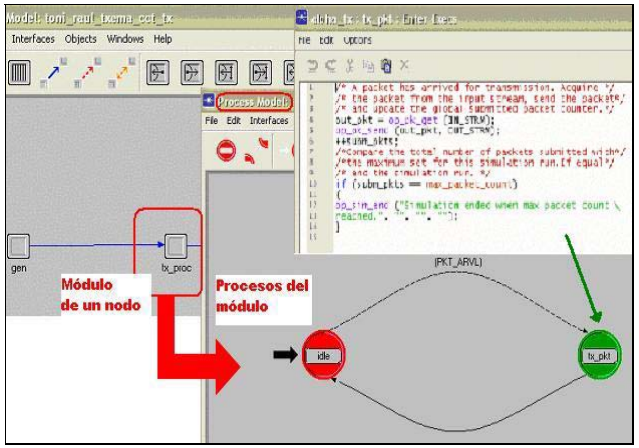


Figura 2.11. Ejemplo de proceso del nodo transmisor

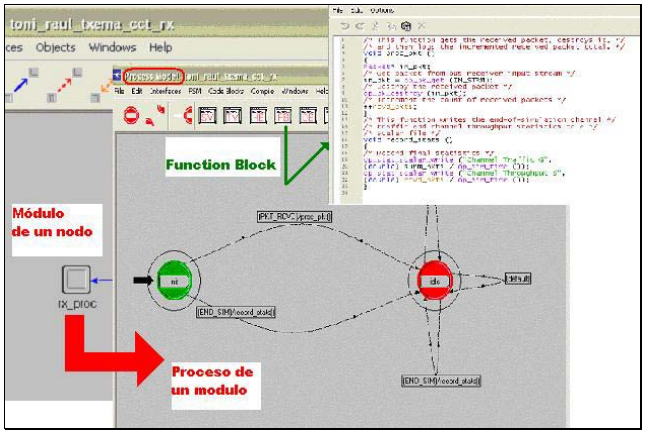


Figura 2.12. Ejemplo Modelo de proceso del nodo receptor.

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.

OPNET: Manual de usuario

- 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.

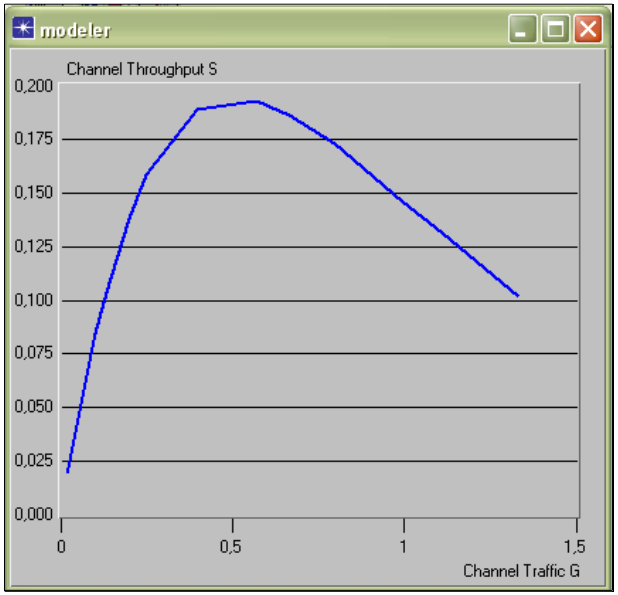


Figura 2.12. Resultado de la simulación.

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

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

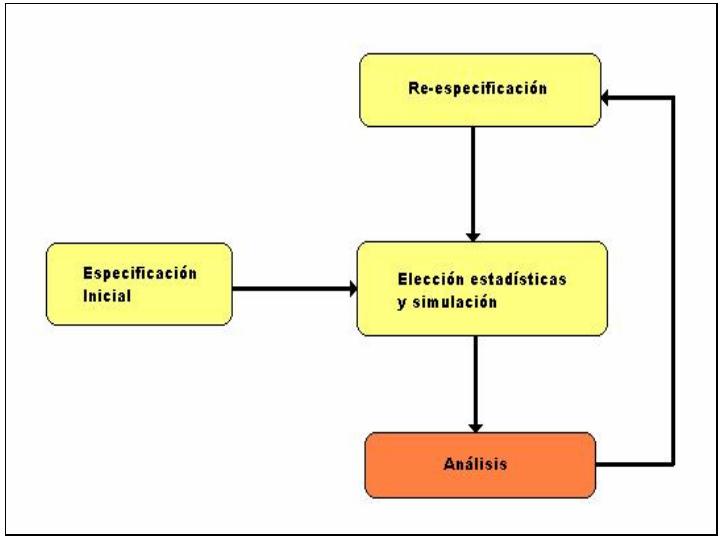


Figura 2.14. Esquema de las fases de la simulación.

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.

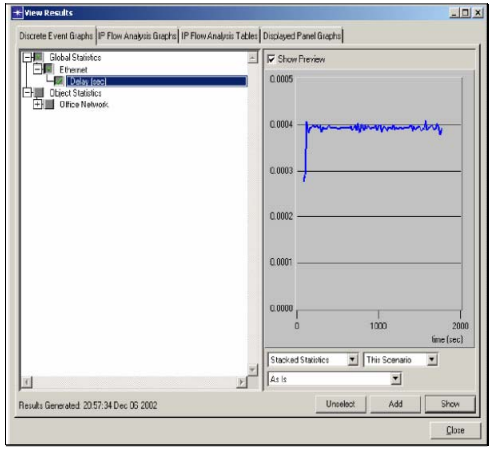


Figura 2.15. Visualización de resultados.

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.

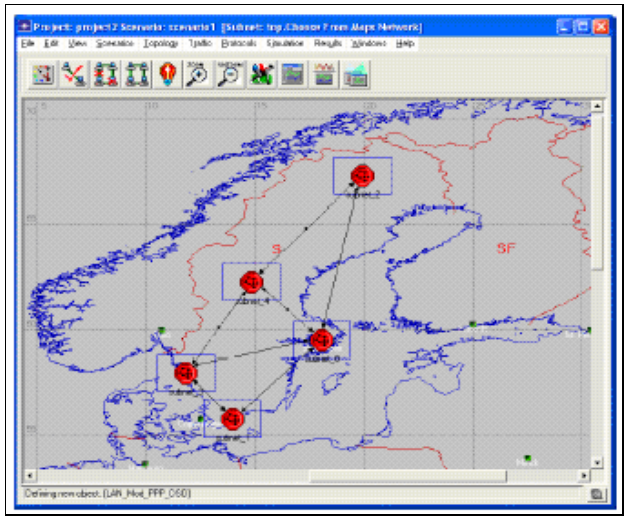


Figura 2.16 Formato Project Editor.

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.

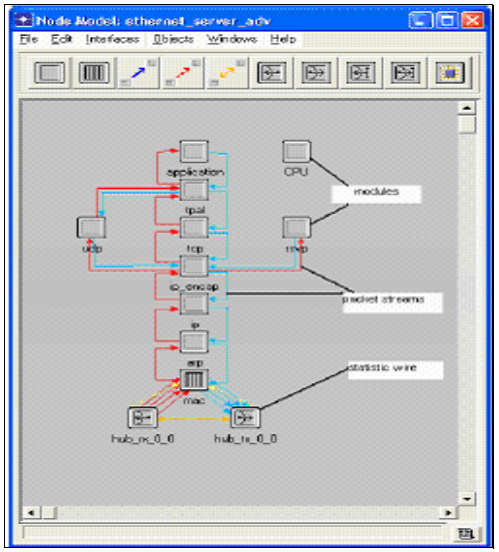


Figura 2.17. Node Editor.

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.

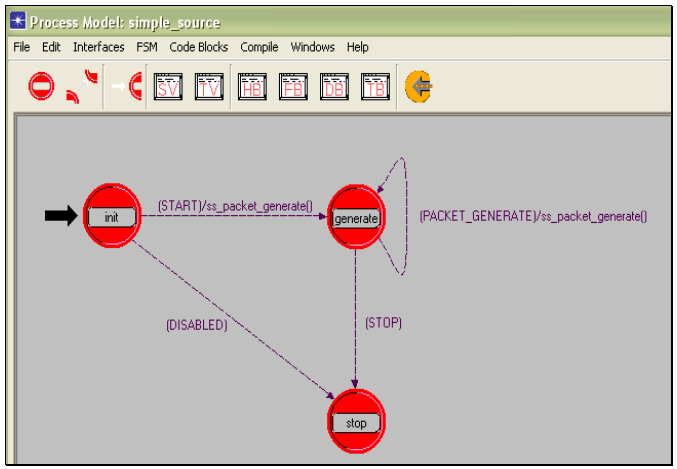
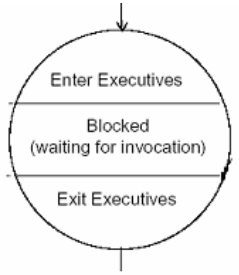


Figura 2.18. Formato del Process Model Editor.

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 comandas 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 comandas que tiene su enter executives e inmediatamente ejecuta las comandas 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.

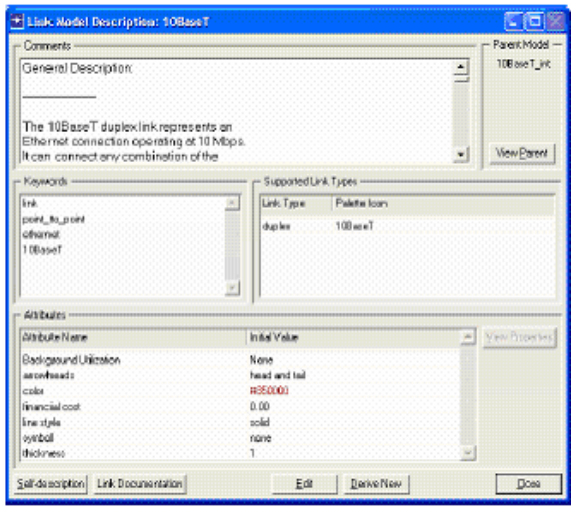


Figura 2.19. Link Model Editor.

Un link model especifica la información siguiente:

-Tipo de enlace soportado: Todos los enlaces que creemos 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.

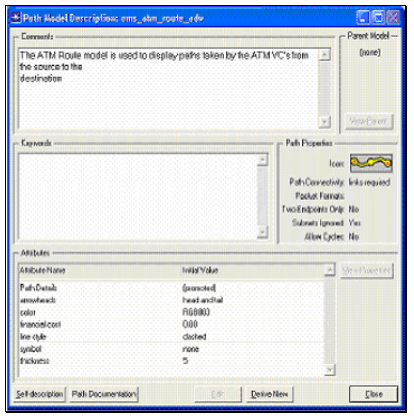


Figura 2.20. Path Editor.

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.

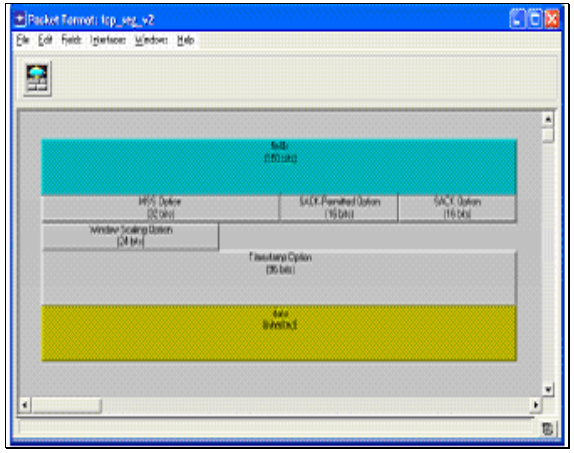


Figura 2.21. Packet Format Editor.

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.

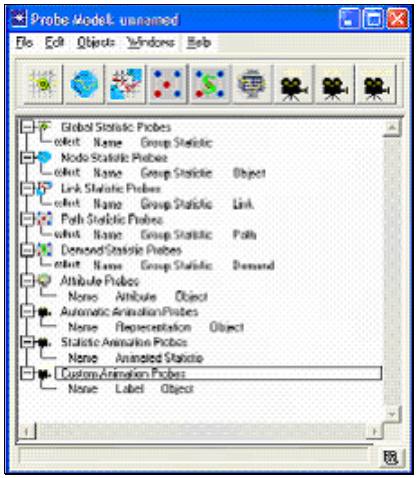


Figura 2.22. Probe Editor.

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.

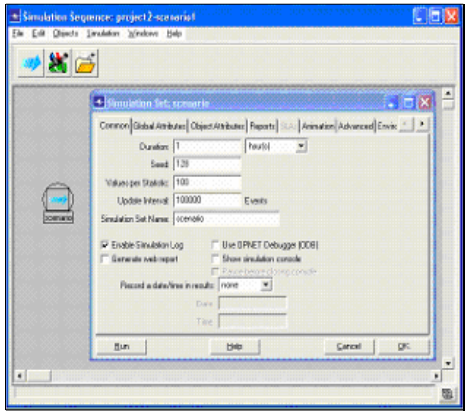


Figura 2.22. Simulation Sequence Editor.

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.

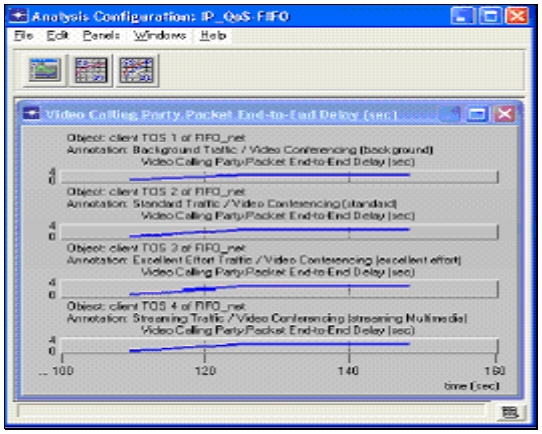


Figura 2.24. Análisis Tool.

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

OPNET: Manual de usuario

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.

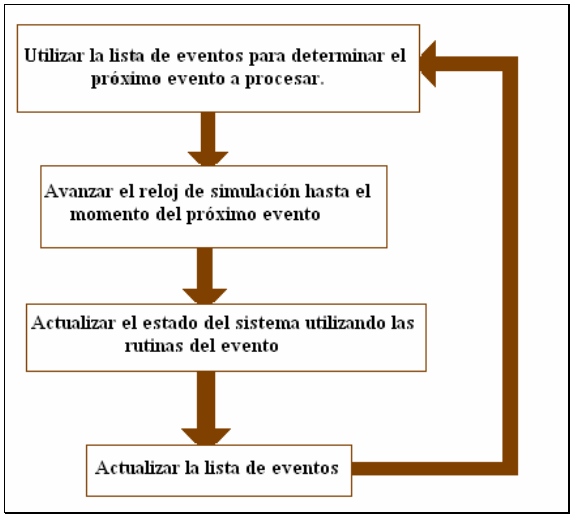


Figura 2.26. Proceso de simulación.

Salidas del simulador y análisis de resultados.

Las estadísticas a estudiar de la simulación pueden ser seleccionadas de distintas formas. Las más sencillas, se pueden encontrar en un menú desplegable. A este menú se accede pinchando con el botón derecho del ratón sobre el espacio del proyecto y eligiendo "Choose Individual Statistics". En este punto se seleccionan las estadísticas a recoger. Estas pueden ser a nivel de nodo o globales. La elección de unas u otras dependerá del estudio que estemos realizando. Se debe poner especial cuidado, ya que puede ocurrir que obtengamos resultados engañosos. Un ejemplo puede ser que se disponga de dos enlaces en una red, uno muy lento y otro muy rápido. Si observamos el retardo global de la red, el resultado será un retardo medio, con lo que podemos pensar que no influyen las distintas velocidades. Por el contrario, si observamos el retardo por nodo, apreciaremos un retardo muy pequeño para el enlace rápido y un retardo elevado para el enlace lento.

Si te interesa generar una animación (ver como fluyen los paquetes en la red por ejemplo), puedes usar la opción "Record Simulation For Subnet" que se encuentra también en el menú de simulación DES. Esto debe hacerse antes de proceder a simular. De esta manera, una vez finalizada la simulación, accediendo al menú "DES", podrás ver la animación seleccionando "Play Animation". Esta es la manera más sencilla, aunque también se pueden realizar simulaciones desde el editor de pruebas.
Otro aspecto importante y de gran utilidad es el log de la simulación. Puedes acceder a este, una vez terminada la simulación, pinchando con el botón derecho del ratón sobre el espacio de trabajo. El " Open DES Log" te dará información de posibles errores o sucesos ocurridos durante la simulación y en ocasiones te mostrará posibles soluciones.

En definitiva, los pasos fundamentales que seguiremos a la hora de realizar una simulación de eventos discretos con OPNET son:

1. Abre un proyecto nuevo. Si la configuración a realizar es una red que se compone de elementos ya configurados, procedemos a su diseño con la ayuda de las paletas de objetos.

2. Si no disponemos de los elementos de red necesarios, será necesario recurrir al editor de nodos y de procesos. En el editor de procesos, se crearán procedimientos lógicos mediante máquina de estado finito y programación en lenguaje C++. Una vez creado y compilado el proceso, iremos al editor de nodos e incluiremos una caja negra que hará referencia al nodo creado. Existen numerosos procesos ya implementados así que no será siempre necesario recurrir al editor de procesos.

3. Cuando ya se dispone de todo lo necesario, se vuelve al paso 1, esto es, los nodos creados, deberán incluirse en un proyecto para proceder a su simulación. Estos nodos no estarán disponibles en ninguna paleta de las existentes, así que deberemos crear una propia con todos los elementos que necesitemos. Para hacer esto debemos partir de una paleta cualquiera. Podemos observar en la parte superior izquierda un botón que dice "Configure Palette …". Pinchamos en él, y pinchando posteriormente en el botón "Clear" conseguimos partir de una paleta limpia. Ahora pinchando en "Link Models", "Node Models", … podemos incluir los modelos que consideremos convenientes. Al finalizar salvaremos la paleta. De esta manera ya disponemos de una paleta propia para nuestro proyecto, y podemos disponer los nodos y enlaces necesarios en el espacio de trabajo.

4. Con el diseño creado, sólo nos falta elegir las estadísticas que vamos a estudiar. Lo podemos hacer desde el menú desplegable del espacio de trabajo, seleccionando "Choose Individual Statistics".

5. Es el momento de editar los parámetros de la simulación. Para ello en el menú "DES" elegimos "Configure Discrete Event Simulation" o pinchamos en el botón que representa un corredor. Una vez finalizada la edición, pinchamos en "Run" y la simulación comenzará.

6. Cuando finaliza la simulación, podemos observar los resultados, accediendo a través del menú desplegable y pinchando en "View Results". Elegimos las estadísticas a estudiar de entre todas las que se simularon y analizamos los resultados.

Este es un procedimiento básico a seguir. Conforme se vayan realizando las prácticas, el alumno aprenderá otros métodos de simulación más complejos, aunque necesarios en determinadas situaciones.