UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

I.T. Telecomunicación (Sist. de Telecomunicación)

"Estudio comparativo de la implementación de los protocolos RTP/RTCP en los simuladores Opnet Modeler y Network Simulator 2"

TRABAJO FINAL DE CARRERA

Autor/es:
Rafael Pellicer Corbacho

Director/es:
D. Fernando Boronat Seguí

GANDIA, 2010

A mi abuela Amparo

Agradecimientos

A mis padres, familia, amigos
y a todos los que me han apoyado

ÍNDICE

1. INTRODUCCIÓN 1

1.1. Introducción 1
1.2. Objetivos 1
1.3. Estructura del proyecto 1

2. HERRAMIENTAS Y PROTOCOLOS 3

2.1. RTP/RTCP 3
2.1.1. RTP 3
2.1.2. RTCP 4

2.2. SIMULADORES DE REDES 6

2.2.1. Kiva 6
2.2.2. OMNET ++ 9
2.2.3. NCTUns 11
2.2.4. COMNET III 14
2.2.5. OPNET MODELER 17
2.2.6. NS2 (NETWORK SIMULATOR) 20

2.3. SOFTWARE ADICIONAL 22

2.3.1. La herramienta NAM 23
2.3.2. La herramienta XGraph 26

3. IMPLEMENTACIÓN DEL PROTOCOLO RTP/RTCP 29

3.1. RTP en Opnet Modeler 29
3.2. Implementación protocolos RTP/RTCP en NS2 30
3.2.1. Imperfecciones encontradas 30
3.2.2. Detalles de la implementación 31
3.2.3. Estado del arte 32

4. ESCENARIO Y PRUEBAS 35

4.1. El simulador OPNET 35
4.1.1. Instalación 35

4.1.2. Creación de la red 38
4.1.2.1. Abrir el simulador OPNET MODELER 38
4.1.2.2. Elección del tipo de escenario 39
4.1.2.3. Paleta de objetos 39
4.1.2.4. Selección de elementos 40
4.1.2.5. Definir background 42
4.1.2.6. Definir Aplicación 44
4.1.2.7. Definir Perfil 47
4.1.2.8. Configuración de los objetos 48
4.1.2.8.1. Variables del servidor multimedia 48
4.1.2.8.2. Variables de los destinos 49
4.1.2.9. Elección de estadísticas 50
4.1.2.10. Selección de variables globales 51
4.1.2.11. Choose Statistics (Advanced) 52
4.1.2.12. Configurar la simulación 52
4.1.2.13. Arrancar la simulación 53
4.1.2.14. Analizar los resultados 54

4.2. El simulador NETWORK SIMULATOR 2 56

4.2.1. Instalación 56
4.2.1.1. Entorno Windows 56
4.2.1.2. Entorno Linux 58
4.2.2. Ejemplo básico de configuración de un escenario 61
4.2.3. Principales comandos Tcl para el módulo RTP/RTCP nativo 67
4.2.4. Descripción del Módulo VOIP 69
4.2.5. Interfaz de Configuración del Módulo VoIP 72
4.2.6. Script de simulación 74
4.2.7. Proceso de simulación 84

5. RESULTADOS DE SIMULACIÓN 89

6. CONCLUSIONES 93

7. BIBLIOGRAFÍA 95

ÍNDICE DE FIGURAS

Figura 1. Formato de la cabecera de RTP 4
Figura 2. Interfaz gráfica de Kiva 8
Figura 3. Ejecución de una simulación en una interfaz gráfica de OMNET ++ 11
Figura 4. GUI de NCTUns 13
Figura 5. Interfaz Gráfica del Usuario 16
Figura 6. Esquema de simulación en NS2 21
Figura 7. Salida al ejecutar File.tcl en NS2 23
Figura 8. Primera ventana del NAM 24
Figura 9. Pantalla principal del NAM 24
Figura 10. Estructura del fichero de traza 25
Figura 11. Primera ventana Tracegraph 27
Figura 12. Análisis de traza en Tracegraph 27
Figura 13. Información sobre simulación en Tracegraph 28
Figura 14. Estructura de directorios del simulador NS2 32
Figura 15. Acceso al simulador 38
Figura 16. Asignación de nombre al proyecto y al escenario 39
Figura 17. Escenario vacío en Opnet Modeler 40
Figura 18. Botón de acceso a la paleta de objetos 40
Figura 19. Configuración delay en los enlaces 10BaseT 41
Figura 20. Escenario en Opnet Modeler 42
Figura 21. Se edita el background 43
Figura 22. Indica los atributos entre los routers gukumatz_0 y np8-gandia 44
Figura 23. Configuración de Aplicaciones VoIP 45
Figura 24. Valores por defecto Voice 45
Figura 25. Distribución de la duración del período de silencio 46
Figura 26. Distribución de la duración del período de ráfaga 46
Figura 27. Parámetros de configuración de perfiles 47
Figura 28. Parámetros del servidor multimedia 48
Figura 29. Parámetros del destino 49
Figura 30. Parámetros aplicaciones de los destinos 50
Figura 31. Elección de estadísticas 51
Figura 32. Ventana para configurar la simulación 52
Figura 33. Proceso de simulación 53
Figura 34. Ventana para visualizar resultados 54
Figura 35. Ejemplo de gráfica en el Opnet Modeler 55
Figura 36. Introducción del path en Windows XP 57
Figura 37. Comandos Tcl básicos en la configuración de una Sesión RTP en NS-2 68

Figura 38. Módulos para la simulación de aplicaciones VoIP en NS2 70
Figura 39. Representación de la topología de red 85
Figura 40. Representación de la topología de red con tráfico de fondo CBR 85
Figura 41. Inicio de la transmisión VoIP 86
Figura 42. Pérdida de paquete CBR con una carga de 1,98Mbps 87
Figura 43. Pérdida de paquete VoIP con una carga de 1,98Mbps 87
Figura 44. Transmisión VoIP OFF 88
Figura 45. Transmisión VoIP ON 88
Figura 46. Retardo extremo a extremo 89
Figura 47. Throughput extremo a extremo 90
Figura 48. Jitter 91

1. INTRODUCCIÓN

1.1. Introducción

La simulación es la manera más eficaz de evaluar las soluciones que se proponen para una red de comunicaciones. La simulación ofrece importantes ventajas como la posibilidad de implementar grandes redes de comunicación, la experimentación de redes con tecnologías aun no disponibles o la rápida implementación de los escenarios.

1.2. Objetivos

El objetivo principal de este proyecto es realizar una comparación entre dos simuladores de redes bien conocidos, como son el Opnet Modeler y el Network Simulator 2 mediante la aplicación Voz sobre IP (VoIP), extraídos del protocolo de transporte en tiempo real, RTP.

Como objetivos secundarios del proyecto:

- Observar los diferentes simuladores que permitan simular los protocolos RTP/RTCP.
- Aprender el manejo del software de simulación Opnet Modeler y Network Simulator 2 (NS2).
- Aprender el manejo de software adicional como el Network Animator (NAM) y el tracegraph.
- Aprender el lenguaje de programación OTcl.
- Estudio de los módulos de las aplicaciones VoIP desarrollados en NS2 y OPNET.
- Desarrollo de un escenario de prueba en ambos simuladores y estudiar los parámetros a ser evaluados.

1.3. Estructura del proyecto

En el **segundo capítulo** introducimos la tecnología RTP/RTCP, ya que es el protocolo objeto de estudio en este proyecto. En este mismo capítulo introducimos los diferentes simuladores de redes más comunes que nos podemos encontrar y el por qué de la elección de tanto el Network Simulator 2 como el Opnet Modeler.

En el **tercer capítulo** se muestra cómo está implementado el protocolo RTP/RTCP en los dos simuladores de redes escogidos en el capítulo anterior como también los diferentes módulos utilizados por grupos de investigadores en otras universidades.

En el **cuarto capítulo** se introducen todos los pasos que hay que seguir para una correcta instalación de los simuladores y también todos los pasos seguidos para la creación de un escenario de simulación.

En el **quinto capítulo** se muestran los resultados obtenidos y una comparación entre ellos.

En el **sexto capítulo** se exponen las conclusiones del presente proyecto.

2. HERRAMIENTAS Y PROTOCOLOS

2.1. RTP/RTCP

El **protocolo RTP** (*Real-time Transport Protocol*), que en español es **Protocolo de Transporte en tiempo real** surgió con la idea de crear un **protocolo específico** para la gran **demanda de recursos en tiempo real** por parte de los usuarios.

Le acompaña o complementa el **protocolo RTCP** (*RTP Control Protocol*), es decir, Protocolo de Control RTP, cuya función principal es **proporcionar mecanismos de realimentación** para informar sobre la calidad en la distribución de los datos.

En 1996 se publicó la **RFC 1889** (Request For Comments 1889) y en julio del 2003 se publicó una actualización, la **RFC 3550** (Request For Comments 3550) el estándar del protocolo **RTP**.

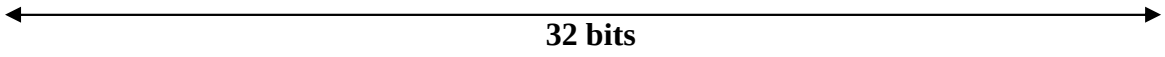
2.1.1. RTP

Con el avance de Internet, cada día se ha hecho más necesaria la transmisión de voz, vídeo e información en tiempo real. TCP no fue diseñado para este fin por lo que no cumple con las expectativas y necesidades de las nuevas aplicaciones. Así surgió el protocolo de transporte de tiempo real, el título oficial de esta recomendación es: “RTP: A Transport Protocol for Real-Time Applications” que describe los dos protocolos RTP y RTCP.

El protocolo RTP se creó específicamente para la transmisión de audio y vídeo, gracias a que incluye en su cabecera informaciones que sincronizan imagen y sonido, al tiempo que es capaz de determinar si se han perdido paquetes y si éstos han llegado en el orden correcto.

Por otra parte, las cabeceras del RTP también especifican el tipo de emisión realizada, por lo que permite diferentes tipos de compresión de datos. Cuando se realiza una conexión a través de RTP, se definen dos direcciones diferentes que son controladas desde dos puertos distintos, de forma que audio y vídeo viajan por separado controlados por RTCP. Además, tiene como objetivo añadir información feedback desde el cliente hacia el servidor para garantizar una calidad de servicio. Por contra, tenemos que RTP no asegura ni la entrega continua de información, ni la de todos los paquetes y ni siquiera puede evitar la entrega desordenada de los mismos.

En la siguiente imagen se muestra el formato de paquete RTP:



V=2	P	X	CSRC count	M	PT	Número de secuencia
Marca de Tiempo						
Identificación de la fuente de sincronización (SSRC)						
Identificación de la fuente de contribución (CSRC)						

Figura 1. Formato de la cabecera de RTP

A continuación se muestra la leyenda de la figura 1.

- Versión (V: Version, 2 bits). Corresponde a la versión de RTP en uso.

- Relleno (P: Padding, 1 bit). Este campo indica que el paquete contiene uno o más bytes de relleno que no son parte de la carga útil, bytes que deben ser ignorados.

- Extensión (X: Extension, 1 bit). Este campo es fijado a ‘1’ para indicar cuando la cabecera es fija, o a ‘0’ cuando la cabecera tiene alguna extensión.

- CSRC Count (CC, 4 bits). Contiene el número de identificadores de CSRC (definida más abajo) que siguen a la cabecera fija.

- Marcador: (M: Marker, 1 bit). Este campo se fija de acuerdo a la carga útil que es transportada. La RFC 3550 establece que las aplicaciones que no envíen paquetes durante periodos de silencio deben ajustar este campo a ‘1’ en el primer paquete enviado después de un período de silencio. Las aplicaciones que no soporten supresión de silencio deben ajustar este bit a ‘0’.

- Tipo de carga útil (PT: Payload Type, 7 bits). Es un campo que identifica el tipo de información contenida.

- Número de secuencia (Sequence Number, 16 bits). Este campo es ajustado por el transmisor a un valor aleatorio al principio de una sesión, y se incrementa en uno por cada paquete RTP que se envíe, esto le permite al receptor identificar cuando los paquetes llegan mal, o no llegan. Sirve para detectar pérdidas de paquetes.

- Marca de tiempo (Timestamp, 32 bits). Este campo indica el instante preciso en que fue generada la muestra de voz de la carga útil. Se utiliza para la sincronización y el cálculo del jitter.

- Fuente de sincronización (SSRC: Synchronization Source, 32 bits). Este campo indica el valor de sincronización fuente, que es la entidad responsable de ajustar los valores del número de secuencia y la marca de tiempo (timestamp).

- Fuente contribuyente: (CSRC: Contributing Source, 32 bits). Este campo contiene un valor de SSRC que contribuye a una sesión. Este campo es usado cuando el flujo RTP proviene de un mezclador, y es empleado para identificar la fuente de información multimedia al otro lado del mezclador.

2.1.2. RTCP

El protocolo de control de RTP (RTCP, RTP Control Protocol) permite el intercambio periódico de información de control entre participantes de la sesión, con la meta principal de proveer realimentación para informar sobre la calidad en la distribución de los datos. Esta información se usa, por ejemplo, para diagnosticar fallos en la distribución.

RTCP define cinco tipos diferentes de paquetes:

- *Información del emisor (SR, Sender Report)*. Es usado por los participantes de una sesión activa para intercambiar estadísticas de transmisión y recepción.

- *Información del receptor (RR, Receiver Report).* Es usado para enviar estadísticas por aquellos participantes que reciben y no envían información multimedia.

- *Descripción de la fuente (SDES: Source Description).* Contiene una o más descripciones relacionadas con la fuente, debe de contener un nombre canónico (CNAME, canonical name), que se usa para identificar participantes de la sesión.

- *Despedida (BYE).* Paquete utilizado para indicar la finalización de participación en una sesión de un emisor.

- *Aplicación (APP).* Este paquete se utiliza para transportar información específica de las diferentes aplicaciones que pueden utilizar el protocolo RTP.

2.2. SIMULADORES DE REDES

Existen muchas herramientas de software de simulación, las cuales han evolucionado permitiendo facilitar la implementación y el análisis de sistemas de comunicación cada vez más complejos.

Antes de la realización de este proyecto, realicé un estudio de algunas de las principales herramientas que se utilizan actualmente para la simulación de modelos y aplicaciones de red, con el fin de ver la posibilidad de utilizar los protocolos RTP/RTCP mediante la aplicación VoIP, como también ver las ventajas y desventajas de cada uno de ellos.

2.2.1. Kiva

Es un simulador de redes basado en Java que permite especificar diferentes esquemas de redes de datos y simular el encaminamiento de paquetes a través de dichas redes.

Características generales

Kiva es una herramienta software orientada principalmente a simular el comportamiento del protocolo IP, y especialmente para el estudio del tratamiento de los datagramas y el encaminamiento de los mismos por una red. También al utilizarlo, se puede estudiar el funcionamiento de los protocolos auxiliares ARP e ICMP y emular el funcionamiento básico de tecnologías de enlace como Ethernet. Con esta herramienta, se puede diseñar una topología de red con la interfaz gráfica, configurar el direccionamiento y las tablas de encaminamiento para los dispositivos y simular el envío de paquetes de un equipo a otro.

La principal aplicación del programa es en la enseñanza de los fundamentos sobre el funcionamiento de redes de datos; pero este entorno, también puede ser muy útil para el diseño y comprobación del encaminamiento en redes de datos a nivel comercial.

El objetivo principal de este programa, es ayudar a diseñar y comprender el funcionamiento de redes de datos y en especial el encaminamiento de paquetes en la arquitectura TCP/IP, sin necesidad de una infraestructura real y de herramientas de análisis de tráfico; éste programa, también es capaz de simular distintos tipos de errores en el funcionamiento de las redes, como la pérdida de paquetes o fallos en tablas de encaminamiento.

El programa es multiplataforma, dado que todo su entorno fue desarrollado con el programa de simulación Java, además Kiva ofrece un API que permite usar las funciones de simulación desde otras aplicaciones de Java.

Requerimientos del sistema

Para instalar el simulador de redes KIVA es necesario tener un sistema operativo Windows o Linux. Para la interfaz gráfica se requieren los archivos ejecutables, V 1.0 con API de simulación actualizado. Este archivo, incluye el paquete JAR con el último API de simulación, también se requiere tener instalada la biblioteca runtime de Java (J2SE JRE 1.4.2), que se puede descargar del web de Sun, para ejecutar la aplicación, primero hay que descomprimir los archivos en una carpeta, y después ejecutar el archivo ej.bat.

Interfaz gráfica de usuario

En la versión actual, la interfaz de usuario está implementada con un conjunto de clases que se deben descargar y ejecutar en el equipo del usuario, cada vez que se desee trabajar con el programa.

Kiva se compone de dos partes, totalmente implementadas con el lenguaje de programación Java. La primera es un API (Aplication Programing Interface, Interface de Programación de Aplicaciones), que ofrece un motor de simulación de redes a otras aplicaciones; este API está formado por cuatro bloques: el primero de estos es el bloque de gestión de eventos discretos, el segundo es el de los objetos que representan las redes de datos, el tercer bloque es el de los objetos que representan los equipos finales o de interconexión y finalmente, aparece un cuarto bloque con la pila de comunicaciones.

Los APIs son modulares y extensibles, de forma que se puedan ir incorporando fácilmente a éstos, nuevos tipos de redes y de equipos.

La segunda es propiamente la interfaz gráfica, la cual, también hace uso del API de simulación. La interfaz gráfica permite especificar las topologías de las redes de datos, mediante un editor gráfico; además permite la configuración del direccionamiento de los equipos de la red, el encaminamiento de la información y el acceso a las características que ofrece el API de simulación de una forma sencilla, sin necesidad de programar.

A continuación, se muestra una topología de red modelada con el programa de simulación Kiva.

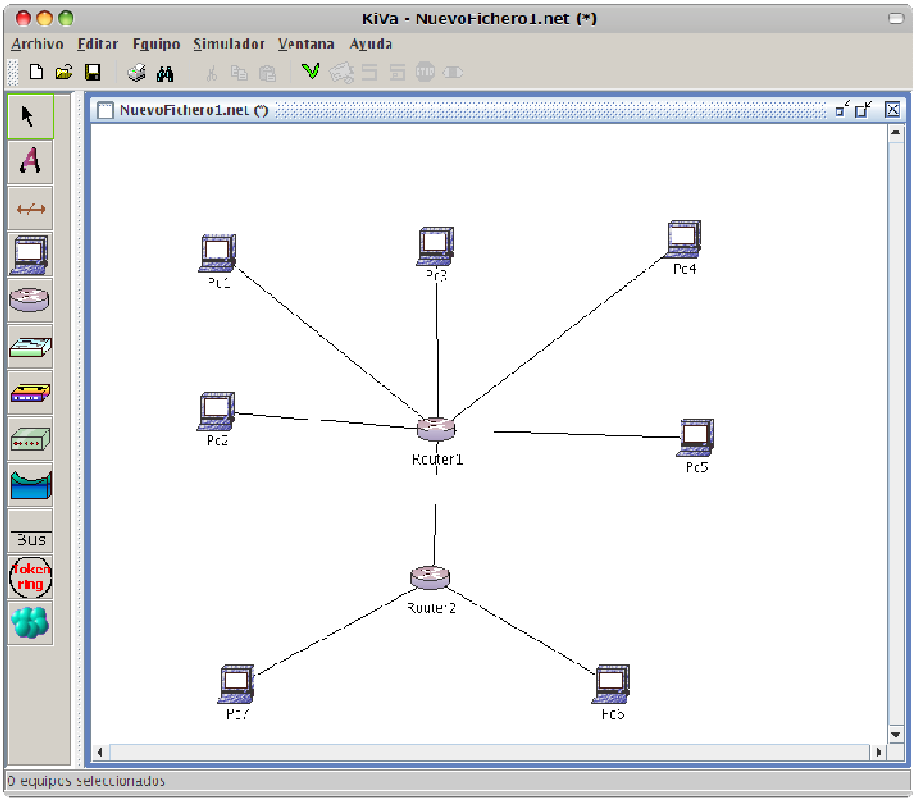


Figura 2. Interfaz gráfica de Kiva

Ventajas e inconvenientes del sistema

Kiva es uno de los programas más completos, para la simulación de redes de comunicaciones, sin embargo no tienen la misma orientación de la mayoría de simuladores que se desarrollaron para evaluar los parámetros de carga y rendimiento en las redes, Kiva se orienta al estudio del protocolo IP y las arquitecturas TCP/IP. En la siguiente tabla, se resumen las principales ventajas y desventajas del simulador KIVA.

Ventajas
• El programa se distribuye con software libre y además es multiplataforma. • Permite el estudio de las redes IP y especialmente el seguimiento y análisis del funcionamiento, el envío, el tratamiento y la recepción de los datagramas a través de arquitecturas TCP/IP. • Su orientación académica, hacen que sirva de ayuda para el diseño y comprensión del funcionamiento de redes de datos. • Sirve como complemento de los Fundamentos teóricos sobre arquitecturas por niveles, protocolos de enlace y arquitecturas TCP/IP.
Inconvenientes
• En la versión actual, la interfaz de usuario está implementada con un conjunto de clases, las cuales deben ejecutarse en el equipo del usuario, cada vez que se desee trabajar con éste programa. • Se deben descargar varios archivos para poder instalar el programa;

además se debe tener especial cuidado en descargar las versiones que se especifican ya que otras versiones de dichos paquetes, no permitirán que se complete la instalación.

- Para el diseño y comprobación del encaminamiento en redes de datos a nivel comercial o para fines de investigación y desarrollo; se debe hacer programación en Java.

2.2.2. OMNET ++

Es un programa orientado a simular objetos y a modular eventos discretos en redes de comunicaciones, posee una gran cantidad de herramientas y una interfaz que puede ser manejada en plataformas Windows y en distribuciones tipo Unix; haciendo uso de varios compiladores de C++.

OMNET ++ es una versión libre, para fines académicos, de la versión comercial OMNEST desarrollado por Omnest Global, Inc. OMNET++, así como las interfaces y las herramientas, se pueden ejecutar perfectamente sobre sistemas operativos Windows y sobre algunas versiones de UNIX y Linux, usando varios compiladores de C++.

Características generales

OMNET++ es una herramienta eficiente, enfocada al área académica y desarrollada para modelar y simular eventos discretos en redes de comunicaciones; básicamente este simulador de redes recrea dichos eventos discretos por medio de módulos orientados a objetos; puede ser utilizado para modelar el tráfico de información sobre las redes, los protocolos de red, las redes de colas, multiprocesadores y otros sistemas de hardware distribuido; además para validar arquitecturas de hardware y evaluar el rendimiento de sistemas complejos.

Este simulador, utiliza el lenguaje de programación NED, que se basa en el lenguaje C++; como herramienta para modelar topologías de red; este lenguaje facilita la descripción modular de una red, es decir, un modelo en OMNET ++ se construye con módulos jerárquicos mediante el lenguaje NED, dichos módulos pueden contener estructuras complejas de datos y tienen sus propios parámetros usados para personalizar el envío de paquetes a los destinos a través de rutas, compuertas y conexiones. Los módulos de más bajo nivel son llamados “simple modules” y son programados en C++ usando la librería de simulación.

Básicamente, con el lenguaje NED se definen tres módulos: módulos simples, módulos compuestos y de redes; dentro de los cuales se encuentran los componentes y especificaciones de la descripción de una red de comunicaciones.

Con el fin de facilitar el diseño de redes y la simulación de eventos sobre las mismas, OMNET ++, permite al usuario trabajar gráficamente empleando el editor del lenguaje NED (GNED). Este editor es la interfaz gráfica que permite crear, programar, configurar y simular redes de comunicaciones, sin necesidad de hacerlo utilizando la codificación del lenguaje NED; ya que automáticamente, GNED se encarga de generar el código del lenguaje, de acuerdo al diseño y configuración que realiza el usuario en forma gráfica. Además GNED, permite acceder fácilmente a dicho código.

Interfaz de usuario

Las simulaciones en OMNET++ pueden utilizar varias interfaces de usuario, dependiendo del propósito. La interfaz más avanzada permite visualizar el modelo, controlar la ejecución de la simulación y cambiar variables/objetos del modelo. Esto facilita la demostración del funcionamiento de un modelo. Para la interfaz de usuario, se pueden generar dos tipos de archivos ejecutables:

Interfaz de usuario gráfico, útil para depurar y comprender, los procesos y configuraciones que se aplican a las redes. A esta interfaz gráfica se accede con el editor GNED. GNED es la herramienta que simplifica el desarrollo de las simulaciones con OMNET ++, ya que permite trabajar sin necesidad de programar.

Interfaz de consola, más eficaz para realizar las simulaciones por lotes. OMNET++ contiene unas clases programadas en C++, diseñadas para recoger y exhibir datos estadísticos.

Pasos para realizar una simulación con OMNET++

Realizar una simulación con este software, involucra todo un proceso, a continuación se resumirán los pasos básicos para llevar a cabo una simulación:

1. Definir la estructura de la red la cual se hace mediante el Lenguaje NED en forma de código o utilizando el editor GNED, para hacerlo de forma gráfica.
2. Completar el comportamiento, es decir, configurar los parámetros y características de la red mediante el lenguaje de programación C++.
3. Configurar la simulación en el archivo .ini

En la siguiente figura, se muestra un diagrama con los pasos para crear y ejecutar una simulación con el programa OMNET ++.

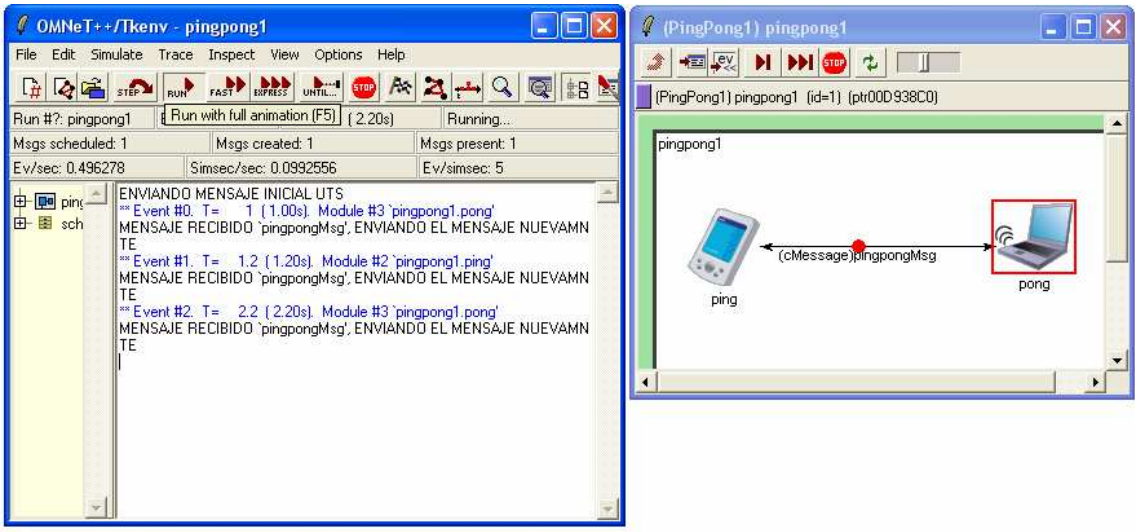


Figura 3. Ejecución de una simulación en una interfaz gráfica de OMNET ++

Ventajas e inconvenientes del programa

OMNET++ provee un amplio panorama para la realización de simulaciones. En la siguiente tabla, se resumen a algunas de las principales ventajas y desventajas que presenta OMNET++.

Ventajas
• OMNeT++ es gratuito solamente para propósitos académicos, lo que facilita su utilización en universidades y grupos de investigación. • Es multiplataforma. • Gracias a la programación por módulos, es posible simular procesos paralelos y distribuidos, los cuales pueden utilizar varios mecanismos para comunicarse entre sí.
Inconvenientes
• Para fines de investigación y desarrollo, es necesario saber programar en lenguaje NED, ya que el trabajo con el editor gráfico, es un poco más rígido. • Por ser un software de aplicación en áreas comerciales y para efectos de investigación y desarrollo, tiene un alto grado de complejidad en su manejo.

2.2.3. NCTUns

NCTUns (National Chiao Tung University, Network Simulator) es un simulador y emulador de redes y sistemas de telecomunicaciones avanzado. NCTUns es software libre y se ejecuta sobre Linux; además utiliza una metodología de simulación que entra y modifica el Kernel de Linux, lo cual hace que el programa tenga ventajas únicas en comparación con otros simuladores y emuladores de redes de comunicaciones.

Este simulador permite desarrollar, evaluar y diagnosticar el desempeño de protocolos y aplicaciones en diferentes tipos de redes (LAN, MAN, WAN). Las simulaciones hechas con esta herramienta, cuentan con características muy especiales, ya que NCTUns simula en tiempo real y con una interfaz similar a la de los sistemas reales, lo cual permite familiarizar más al usuario con el manejo del diseño, configuración e implementación de aplicaciones en redes de comunicaciones.

Características generales

NCTUns utiliza una sintaxis sencilla pero muy efectiva para describir la topología, los parámetros y la configuración de una simulación, esta descripción se genera a partir de la interfaz gráfica del usuario.

NCTUns fue desarrollado basado en el simulador NS, de ahí su nombre, solo que incluye una interfaz más amigable para la implementación de los modelos de red que se simulan. Este programa permite la simulación de arquitecturas de redes sencillas, sin embargo, su mayor potencial está en la simulación de redes tan complejas como las redes GPRS, satelitales y ópticas.
El NCTUns también puede ser utilizado como emulador, especialmente para redes móviles e inalámbricas; para dichas aplicaciones provee recursos para manejo y estudio de sistemas de radiofrecuencia y permite obtener mediciones para establecer niveles de calidad de servicio (QoS) de las señales irradiadas.
Adicionalmente, permite simular redes ópticas y como si fuera poco, puede usarse fácilmente como un emulador, cuando se desee desarrollar funciones de desempeño de un host real y ver cómo se comportaría bajo diferentes tipos de condiciones de red sin modificar su protocolo interno. Esto quiere decir que NCTUns tiene la posibilidad de emular un dispositivo de red del mundo real en su entorno gráfico e interconectarlo con dispositivos simulados o virtuales, para intercambiar paquetes.

Requerimientos del sistema

Para instalar el programa y realizar simulaciones sin ningún contratiempo, es necesario que se cumplan con unos requerimientos mínimos a nivel software y hardware.

Funciona en los sistemas operativos: Red Hat, Fedora Core 2.0 y core 3.0. A nivel hardware es necesario utilizar como mínimo un procesador Pentium de 1 GHz, 256 MB de memoria RAM y 200 MB de espacio libre en el disco duro.

Además, a nivel software; es importante tener instalado un compilador gcc, el sistema Xwindows Gnome o Kde, que el usuario tenga los privilegios del administrador o root para el manejo del programa y que se asigne el shell Bash /tcsh al usuario para el manejo por comandos en el modo consola.

Interfaz gráfica de usuario (GUI)

NCTUns provee una GUI (Interfaz Gráfica de Usuario) profesional y de alta integración, en la cual el usuario diseña y edita la topología de la red, configura los módulos de protocolos que manejará cada nodo de la red, asigna valores y define parámetros específicos de cada dispositivo.

Desde la interfaz de usuario, se programa y se visualiza la animación de la transferencia de paquetes durante el proceso de simulación, el desplazamiento; de los terminales móviles y la presentación de resultados mediante gráficos estadísticos. Además el usuario puede consultar el desarrollo de los procesos que se está ejecutando en determinado dispositivo durante la simulación, sin necesidad de pararla o cancelarla.

Las simulaciones remotas se realizan a partir del modelo de la red por medio de la GUI, la cual genera la simulación utilizando los sockets de Internet y TCP/IP, para comunicarse con los otros componentes y poder ejecutar simulaciones con máquinas remotas. Cuando la simulación se termina, los resultados y los archivos generados se transfieren nuevamente a la GUI. En la siguiente figura, se muestra la GUI del NCTUns.

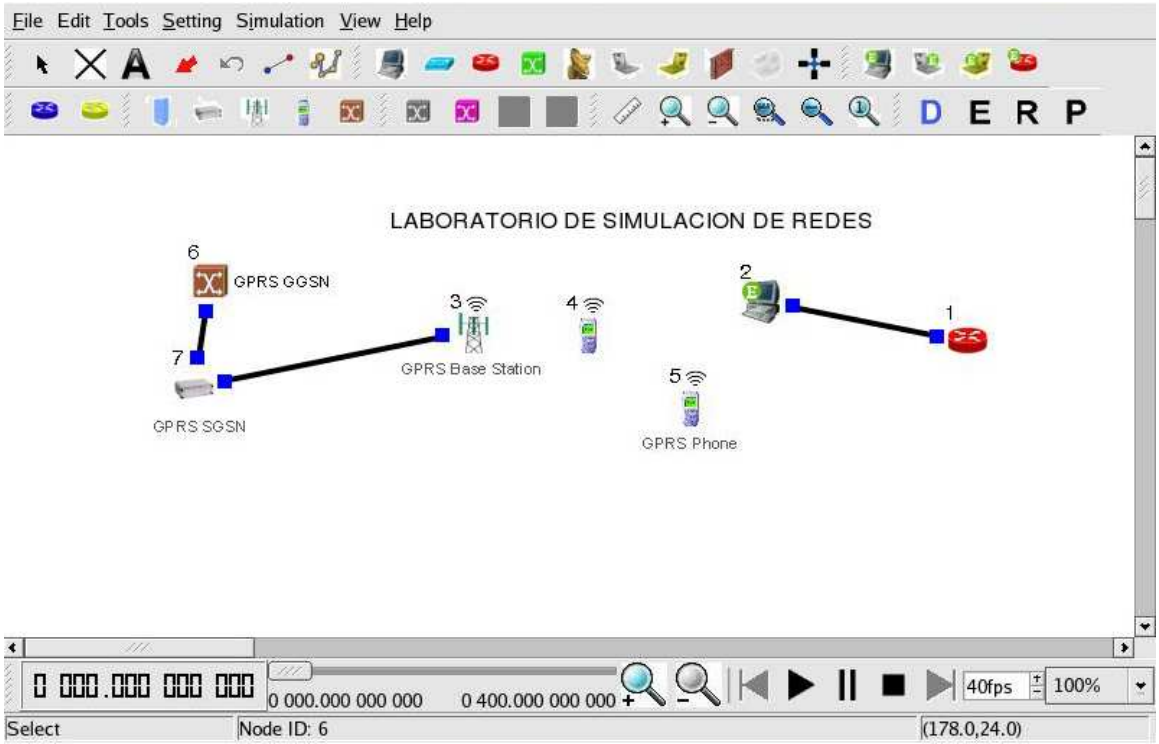


Figura 4. GUI de NCTUns

Ventajas e inconvenientes del programa

En la siguiente tabla se detallarán algunas ventajas y desventajas de la utilización del NCTUns.

Ventajas
• Es un software libre, con distribución de código abierto. • Utiliza directamente el conjunto de protocolos TCP/IP de Linux, por consiguiente se generan resultados de simulación de alta fidelidad y permite que la configuración y el uso de una red simulada, sea exactamente igual a los usados en redes IP del mundo real. • Puede ser utilizado como emulador. Esto permite que un host externo conectado a una red del mundo real, pueda intercambiar paquetes con nodos (por ejemplo: host, enrutadores o estaciones móviles celulares) en una red simulada en NCTUns. • Puede utilizar cualquier aplicación de UNIX existente en la vida real, como un generador de tráfico, además, puede utilizar las herramientas de configuración y monitoreo de UNIX. • Puede simular redes fijas, inalámbricas, redes celulares, redes GPRS y redes ópticas. • Puede simular una gran variedad de dispositivos de red, como: hubs, switches, enrutadores, estaciones móviles, puntos de acceso de WLANs, teléfonos GPRS, etc, así como obstáculos para las señales inalámbricas, además ofrece alta velocidad de simulación. • Simula varios protocolos de redes como: IEEE 802.3, IEEE 802.11, IP, IP Mobile, Diffserv, RIP, OSPF, UDP, TCP, RTP/RTCP, SDP, FTP, etc.
Inconvenientes
• Solamente funciona en sistemas Fedora core 3, para otras distribuciones de Linux es necesario hacer pruebas y configuraciones adicionales. • Existe muy poca información sobre el funcionamiento y configuración del software. • El anterior punto lleva a que sea mayor el tiempo de aprendizaje del simulador. • El servicio de soporte proporcionado por los autores del proyecto NCTUns es deficiente y en algunas ocasiones no funciona.

2.2.4. COMNET III

COMNET III es una herramienta comercial orientada al diseño, configuración y estudio de las redes de comunicaciones, desarrollado por CACI Products Inc; haciendo uso del lenguaje de programación MODSIM II. Por medio de este programa es posible crear topologías de redes complejas, configurar varias tecnologías, protocolos y dispositivos de red, para hacer un análisis detallado del funcionamiento y del rendimiento de redes tipo LAN, MAN y WAN, utilizando una interfaz gráfica en un ambiente de ventanas.

Características generales

Este software gráfico permite analizar y predecir el funcionamiento de redes informáticas, desde topologías básicas de interconexión hasta esquemas mucho más complejos de simulación con múltiples redes interconectadas con diversos protocolos y tecnologías como Ethernet, ATM, Satelitales, Frame Relay, X25, etc. Dentro del área de trabajo del programa, se hace la descripción gráfica del modelo de red, se asocian las fuentes generadoras de tráfico en la red, se configuran los parámetros y las características de los dispositivos de acuerdo a la aplicación que se desea implementar; luego se pone en marcha la simulación y finalmente, se analizan los resultados estadísticos sobre el desempeño de la red, los cuales son programados antes de iniciar la simulación y que se generan automáticamente cuando se concluye la simulación. Algunos de los parámetros que se pueden incluir dentro de los informes de la red esta: la ocupación de enlaces o nodos, la cantidad de mensajes generados, las colisiones, entre otros.

Este programa contiene una gran variedad de dispositivos de red como: hosts, hubs, switches, routers, access points, satélites, entre otros; los cuales pueden ser interconectados con enlaces y tecnologías como: Ethernet, FDDI, punto a punto, Frame relay, Aloha, PVC, CSMA, entre otros; a la vez que permite implementar gran variedad de protocolos; es decir COMNET III presenta características muy completas e interesantes, en cuanto a las interfaces que soporta para su uso, sin embargo cabe mencionar que el máximo desempeño de este simulador se alcanza al utilizar las librerías para los diferentes tipos de dispositivos de redes con sus diferentes parámetros. COMNET III es un software muy poderoso, sin embargo en la edición universitaria, presenta algunas limitaciones ya que no se pueden realizar las simulaciones que involucren más de 20 nodos.

Interfaz gráfica de usuario

COMNET III utiliza un ambiente gráfico de ventanas, el cual tiene una serie de menús y barras de herramientas que permiten crear el modelo de la red que se va a simular. Esta característica hace de COMNET una herramienta ideal para la universidad, ya que los tiempos de aprendizaje y de implementación de una simulación son cortos, si se tiene en cuenta las tecnologías y protocolos que soporta. En la siguiente figura se puede visualizar la interfaz gráfica del usuario que aparece al iniciar el programa.

Figura 5. Interfaz Gráfica del Usuario

En la barra de herramientas, se encuentran los iconos que se utilizan en el modelado de las redes, los iconos de la barra de herramientas 3D, permiten ver la red modelada en tres dimensiones. En la parte inferior se encuentra la paleta de colores con la cual se puede variar el color de los enlaces, del texto y del fondo del área de trabajo. Por último, está la barra de estado, en la que se visualizan los comentarios de los procesos que se están ejecutando dentro de la simulación, así como los errores de diseño y/o configuración de los elementos de la red.

Ventajas e inconvenientes del sistema

COMNET III es una de las herramientas más eficientes para el estudio de redes de comunicaciones, sin embargo se pueden presentar algunas dificultades a la hora de trabajar con este programa. Se resumen las principales ventajas y desventajas del software COMNET III en la siguiente tabla.

Ventajas
• El programa ofrece la posibilidad de simular una gran cantidad de protocolos y tecnologías de red, y ofrece la posibilidad de crear protocolos a medida que se van necesitando. • Permite configurar y observar una gran cantidad de parámetros durante la simulación como: colisiones, capacidad de los buffers de entrada y salida de los dispositivos, utilización del canal, anchos de banda, etc. • Ofrece la posibilidad de ver el intercambio de mensajes entre los nodos de la red de manera gráfica, según avanza la simulación. • Permite obtener gráficos y/o archivos de texto con las estadísticas de la simulación.

• Se pueden diseñar, configurar y simular redes complejas, que incluyan planes de contingencia, seguridad e implementación de tecnologías de superposición como LAN Emulation.
Inconvenientes
• Es un software propietario. • Por ser una de las herramientas de simulación más completas del mercado, la programación de los parámetros de los dispositivos y enlaces de la red tiende a ser compleja. • Además de los conocimientos sobre el manejo y el diseño de redes de comunicaciones, se requieren conocimientos en otras áreas como por ejemplo la estadística. • La versión universitaria del software, solo permite la implementación de redes con un máximo de 20 nodos. • Por ser un simulador de lenguaje específico, es un poco rígido para fines de investigación y desarrollo.

2.2.5. OPNET MODELER

Modeler es un simulador basado en eventos orientado a la simulación de redes de telecomunicaciones creado por OPNET® (Optimized Network Engineering Tools). Para ser más explícitos lo podríamos definir como un simulador dinámico porque la simulación evoluciona con el tiempo y discreto porque el comportamiento de los sistemas representados cambia únicamente en instantes concretos.

La herramienta Modeler es uno de los simuladores más avanzados en el campo de las redes de telecomunicaciones. Quizás, la característica más relevante es que es un simulador orientado a objetos, lo que permite interactuar al usuario sin problemas y ofrece una gran facilidad de interpretación y creación de escenarios aparte de tener en cada objeto una serie de atributos configurables.

Dispone de multitud de librerías, lo que permite simular gran diversidad de redes donde intervenga un amplio número de protocolos y variables específicas que el usuario podrá modificar y estudiar. Número de paquetes perdidos, throughput, jitter, caída de enlaces y potencia de transmisión son algunos de los parámetros que se pueden controlar.

Cabe destacar que OPNET permite entre otras cosas dotar de movilidad a los nodos de la red, modificar el código fuente de las librerías de los nodos para alterar su comportamiento ante diversas acciones, definir tipos de tráfico además de carga de la red debido a tipos de servicios, como por ejemplo HTTP, correo, **VoIP**, streaming, etc.

Características generales

Originalmente fue desarrollado por MIT e introducido al mercado en 1987 como el primer simulador comercial. Esta herramienta se utiliza para el modelado y simulación; está basada en la teoría de redes de colas e incorpora las librerías para facilitar el modelado de las topologías de red. El desarrollo de los modelos se realiza mediante la conexión de diferentes tipos de nodos, utilizando diferentes tipos de enlaces. Mediante OPNET MODELER, se deben especificar tres tipos de modelos, como se muestra en la siguiente tabla.

MODELO DE RED	Redes y subredes
MODELO DE NODOS	Nodos y estaciones
MODELO DE PROCESOS	Especifica la funcionalidad de cada nodo.

El modelo de la red, involucra la creación de nodos, los cuales internamente están constituidos por distintos tipos de módulos y conexiones; finalmente se define la función que desempeñará cada módulo o nodo durante la simulación, a través de los modelos de proceso.

Requerimientos del sistema

Este programa es multiplataforma y requiere las siguientes especificaciones para su correcta instalación y posterior uso. Ver la siguiente tabla.

Sistema operativo	**Requerimientos mínimos del sistema**
Windows NT, 2000, XP y UNIX	Procesador Intel Pentium III de 500 MHz, 64 MB en RAM y 100 MB disponibles en disco duro. Monitor SVGA, 8 MB en memoria de video y tarjeta de sonido

Interfaz gráfica de usuario

OPNET MODELER está basado en una serie de editores jerárquicamente organizados, los cuales permiten diseñar y configurar los modelos de red, de nodos y de procesos en las topologías de red que se van a simular con este programa. Los editores trabajan en forma directa y paralela la estructura real de la red, los equipos y los protocolos.

Editor de proyecto. Mediante este editor se representa gráficamente la topología de una red de comunicaciones, haciendo uso de un panel de herramientas o importándolo de las librerías de OPNET MODELER; además, esta herramienta provee un contexto geográfico configurable, un menú de configuración rápida de protocolos y especificaciones de vista, para aplicarse a la red que se va a simular.

Editor de Nodos. Este editor captura la arquitectura de una red, un dispositivo o un sistema, describiendo el fuljo de los datos entre elementos funcionales, a los cuales se les conoce como "módulos". Cada módulo, puede generar, enviar o recibir paquetes a los demás módulos de la red, de acuerdo a la función que representa dicho nodo. Generalmente, los módulos representan aplicaciones, protocolos, algoritmos o recursos como: buffers, puertos y buses, entre otros; es decir que todo nodo es asignado a un proceso o evento dentro de la simulación, y esto se logra, mediante el editor de procesos.

Editor de procesos. Dentro del editor de procesos, se encuentran unas máquinas, llamadas FSM; estas herramientas soportan las especificaciones, detalles, protocolos, recursos y aplicaciones que se desean configurar en la red modelada gráficamente con los editores de proyecto y de nodos respectivamente. En la figura de a continuación se muestra la interfaz gráfica de MODELER, la cual se compone de los editores de proyecto o de red, de nodos y de procesos.

Los estados y transiciones generadas como respuesta al desarrollo de cada evento ejecutado, contienen un código en lenguaje C/C++, y están soportados por una amplia librería de funciones designadas por la programación de los protocolos. Cada FSM puede definir sus propias variables de estado y pueden hacer llamados al código según las librerías que provea el usuario; es decir las FSMs dinámicamente se pueden generar, durante la simulación, en respuesta a un evento específico. Dentro de este editor, se puede acceder al código fuente en lenguaje C/ C++, que describe la red diseñada gráficamente.

Ventajas e inconvenientes del sistema

Mediante el uso de MODELER es posible acceder a una amplia gama de beneficios, sin embargo presenta algunas desventajas, tal y como se resume en la siguiente tabla.

Ventajas
• El programa incluye las librerías para acceder a un extenso grupo de aplicaciones y protocolos como: HTTP, TCP, IP, OSPF, BGP, EIGRP, RIP, RSVP, Frame Relay, FDDI, Ethernet, ATM, LANs 802.11 (Wireless), aplicaciones de voz, MPLS, PNNI, DOCSIS, UMTS, IP Multicast, Circuit Switch, MANET, IP Móvil; entre otras. • Tiene interfaces para la visualización del modelo en 3D. • Los APIs de simulación permiten acceder libremente al código fuente, lo cual facilita la programación de nuevos protocolos de red. • Las librerías de modelos de red estándar, incluyen dispositivos de red comerciales y genéricos. • Modelos de red jerárquicos. • Maneja topologías de red complejas con subredes anidadas ilimitadas. • Permite mostrar el tráfico por la red a través de una animación, durante y

después de la simulación. Los resultados se exhiben mediante gráficos estadísticos.
Inconvenientes
• Es un software propietario, lo cual lo hace costoso para ambientes universitarios. • Es necesario obtener la licencia para poder utilizar el software, ya que no existen versiones académicas o de prueba. • Complicada determinación de los intervalos de confianza. • El tiempo de aprendizaje es muy elevado.

2.2.6. NS2 (NETWORK SIMULATOR)

Network Simulator 2 (NS2) es un simulador pensado para el desarrollo de redes con un gran nivel de detalle de código abierto enfocado a la investigación y desarrollado en C++ y OTcl (una versión orientada a objetos del lenguaje Tcl).

La idea original de este proyecto surgió a finales de los años 80, a partir de REAL network simulator, y a mediados de los 90, recibió el apoyo de DARPA a través del proyecto VINT (Virtual InterNetwork Testbed) con SAMAN (Simulation Augmented by Measurement and Analysis for Networks), el proyecto NSF con CONSER (Collaborative Simulation for Education and Research). Actualmente hay que añadir el apoyo puntual de numerosos investigadores que implementan nuevas funcionalidades para el simulador.

NS2 es capaz de simular gran cantidad de protocolos de cualquier capa del modelo OSI (802.3, 802.11, protocolos de enrutamiento, **RTP**, TCP, UDP, http, telnet, ftp, cbr, CSMA/CA...), implementando todo tipo de redes (satélites, inalámbricas, cableadas...).

Junto a NS2, existen otras aplicaciones que aportan nuevas funcionalidades, como es el caso de Netwok Animator (nam), que reproduce gráficamente las simulaciones programadas, o Tracegraph, que muestra valores estadísticos y gráficas de parámetros estándar de las simulaciones (delay, paquetes recibidos, ancho de banda, etc.).

Aunque la herramienta se encuentra implementada en C++, se usa un segundo lenguaje interpretado llamado Tcl. Realmente lo que se usa es OTcl, que es la versión orientada a objetos de Tcl. Esto hace que deba existir una interacción entre ambos lenguajes dentro de la propia herramienta.

Sin embargo, podemos optar por usar únicamente Tcl cuando debamos implementar mediante scripts (programa que simula el funcionamiento de la topología de la red) la configuración de las topologías y usar C++ cuando queramos modificar la propia herramienta.

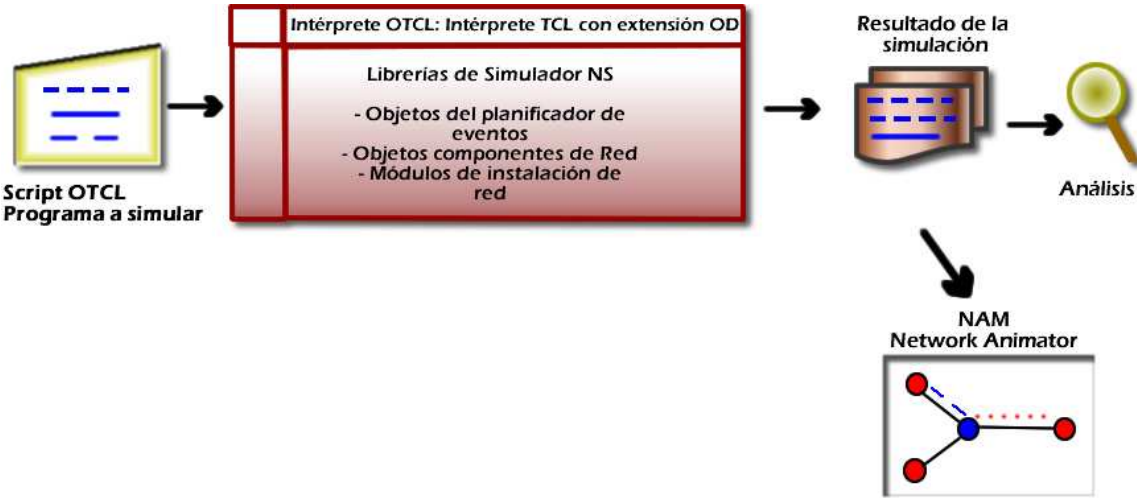


Figura 6. Esquema de simulación en NS2

Como se observa en la figura, NS2 es un intérprete de scripts del lenguaje TCL orientado a objetos, el cual tiene un planificador de eventos de simulación y librerías de objetos de componentes de red y librerías de módulos de instalación de red. Esto quiere decir que la simulación se debe programar en el lenguaje de scripts OTCL.

Requerimientos del sistema

NS2 es un paquete compuesto por un conjunto de componentes requeridos y otros tantos opcionales, este paquete contiene un script de instalación para configurar, compilar e instalar estos componentes. Para instalar este software se requiere cumplir con las especificaciones en la siguiente tabla.

Sistema Operativo
• Plataformas Unix (Free BSD, Linux, SunOS, Solaris) • Plataformas Windows desde la versión 95
Requerimientos mínimos Hardware
• Procesador Pentium II de 200 MHz o equivalente, 32MB de memoria RAM y mínimo 320 MB de espacio libre en el disco
Requerimientos Software
• Para plataformas tipo UNIX Tcl release 8.4.5, Tk release 8.4.5, Otcl release 1.9, TclCL release 1.16, Ns release 2.28, otros componentes opcionales: Nam release 1.11, Xgraph version 12, CWeb version 3.4g, SGB version 1.0 • En sistemas Windows es necesario MS Visual C++ 5.0 (o superior). • Otra forma de instalarlo, es a través de un programa de emulación de Linux, tal como Cygwin.

Interfaz de usuario

NS2 tiene un editor de topología por código, con el cual se diseña y se configuran las redes, los protocolos y las aplicaciones de red que se desean simular. También cuenta con una herramienta llamada Simulador de red automatizado (Automated Network Simulation), este asistente automáticamente carga las tareas que se ejecutan más frecuentemente en los dispositivos de la red.

Ventajas e inconvenientes del sistema

Se muestran las ventajas e inconvenientes del NS2 en la siguiente tabla.

Ventajas
• Este programa contiene módulos que cubren un extenso grupo de aplicaciones, protocolos de ruteo, transporte, diferentes tipos de enlaces, estrategias y mecanismos de ruteo; entre otros. Algunos de estos son: http, TcpApp, telnet, CBR (Constat Bit Rate), TCP, **RTP**, algoritmos de ruteo, enrutamiento jerárquico y enrutamiento manual. • Por ser uno de las más antiguas herramientas de simulación, el NS se ha convertido en un estándar de su área, esto ha llevado a que sea ampliamente utilizado y a que se encuentren en Internet un gran número de ayudas y proyectos realizados sobre NS2.
Inconvenientes
• La configuración de las simulaciones a través de código, hace que sea mayor el tiempo de desarrollo. Además también se incrementa el tiempo necesario para el aprendizaje del software. • NS2 requiere varios componentes adicionales instalados para su correcto funcionamiento.

2.3. SOFTWARE ADICIONAL

NS2 proporciona unas herramientas que facilitan la interpretación de los resultados de una manera gráfica. Estas herramientas son NAM, Network Animator y XGraph que crean una representación a partir de los ficheros de traza generados por NS2. Los ficheros de traza reflejan todos los eventos que han sucedido durante la simulación de la red. Cada una de las trazas incluidas en estos ficheros contiene información sobre el instante de tiempo en que ha ocurrido, sobre los nodos implicados, información sobre la trama MAC, sobre el paquete IP, e información adicional que depende del evento que se ha producido o del tipo de paquete del que se trata.

A continuación se muestra el proceso de transformación del script TCL en los ficheros de traza, es decir, los ficheros que permiten interpretar los resultados de la simulación:

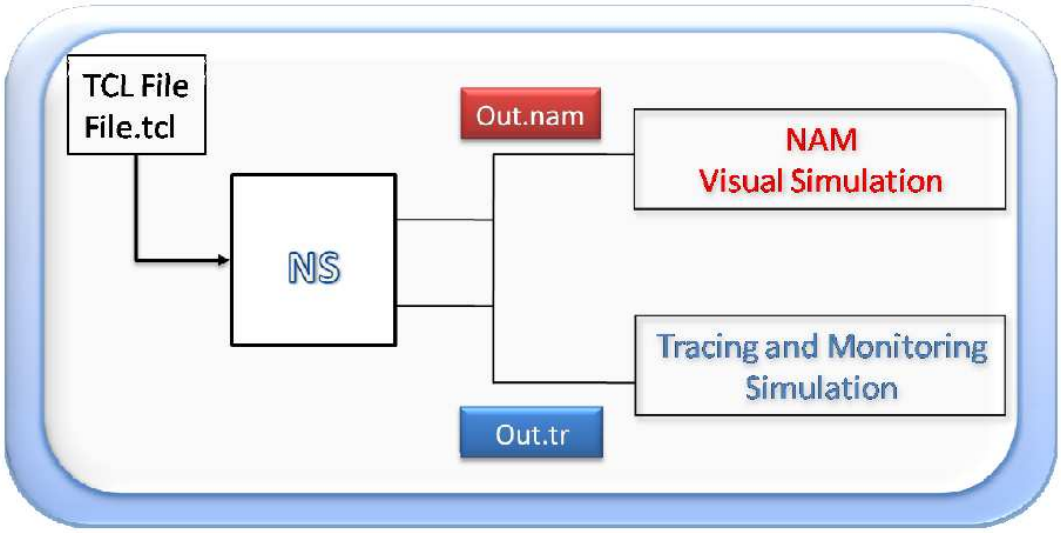


Figura 7. Salida al ejecutar File.tcl en NS-2

Otro de los software adicionales utilizados es el Matlab, ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Entre sus prestaciones básicas se halla la representación de datos y funciones que será la utilizada en este proyecto.

2.3.1. La herramienta NAM

Es una herramienta que facilita la interpretación de los resultados de la simulación en NS-2, permitiendo visualizarlos dinámicamente. Permite por ejemplo la visualización del comportamiento de los terminales de la red, la carga de tráfico entre dispositivos y la pérdida de paquetes.

NAM comenzó a desarrollarse en los noventa por Steven McCanne, miembro del "Network Research Group" del Lawrence Berkeley National Laboratory. Posteriormente fue mejorada por Marylou Orayami bajo la supervisión del proyecto VINT. Actualmente sigue desarrollándose en los proyectos SAMAN y CONSER.

Es una herramienta basada en el lenguaje Tcl/Tk y representa gráficamente los ficheros de trazas obtenidos en el simulador NS-2.

Para invocar la herramienta NAM basta con escribir la siguiente línea:

nam fichero_traza.nam

A continuación se muestra la primera ventana que nos aparece del NAM:

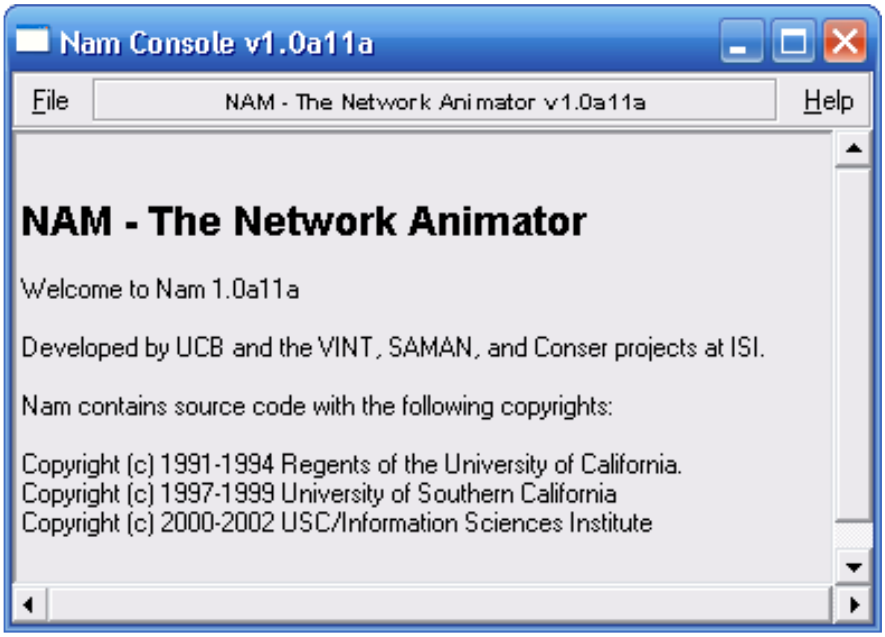


Figura 8. Primera ventana del NAM

En la siguiente imagen se puede ver la captura de un ejemplo de simulación realizada con la herramienta NAM:

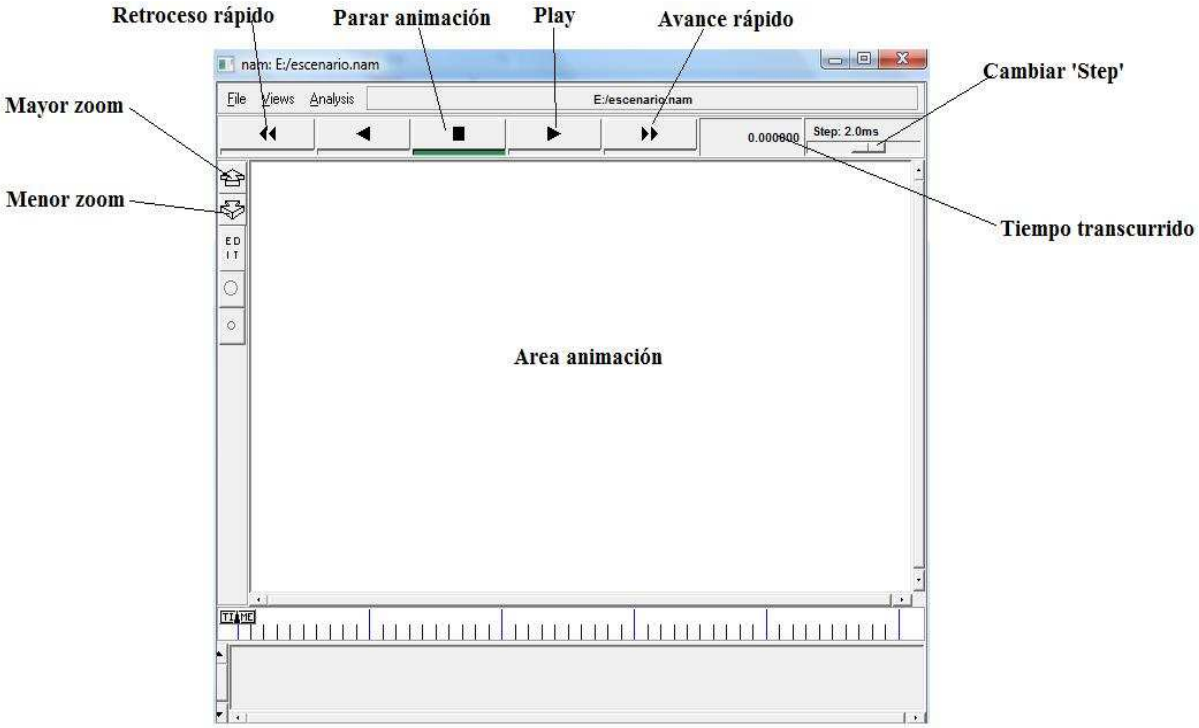


Figura 9. Pantalla principal del NAM

Las principales funciones de la herramienta NAM son:

- **Área animación**: en esta zona visualizaremos la topología de red que se ha especificado en la simulación.

- **Mayor/menor Zoom:** sirven para aumentar o disminuir el área de simulación.

- **Stop/Play Animación**: sirven para poner en marcha o parar la simulación.

- **Tiempo transcurrido:** es el tiempo desde que hemos puesto en marcha la simulación. El tiempo máximo será el que hayamos especificado en dicha simulación.

- **Step:** indica la velocidad con la que transcurre la simulación. Este valor esta en milisegundos y se puede variar mediante el slider que hay debajo del tiempo.

- **File:** contiene diferentes opciones como grabar la simulación, imprimir el área de simulación, etc.

- **Retroceso rápido:** la simulación se va retrocediendo multiplicando el paso del tiempo por 25.

- **Retroceso normal:** la simulación se retrocede según el paso del tiempo.

- **Avance rápido:** la simulación se va avanzando multiplicando el paso del tiempo por 25.

La estructura del archivo de traza NAM generado por el simulador sigue un formato común y es el siguiente:

evento	tiempo	de nodo	a nodo	Tipo paquete	Tamaño paquete	flags	fid	Dirección fuente	Dirección destino	Nº secuencia	Id paquete

Figura 10. Estructura del fichero de traza

Donde los campos se refieren a:

- **evento:** se refiere al evento y puede tomar los valores r recibido, d eliminado, + colocado en la cola, - sacado de la cola.
- **Tiempo**
- **de node:** nodo de donde viene el paquete, no necesariamente el de origen.
- **hacia node:** Siguiente nodo al que va el paquete.
- **Tipo de paquete.**
- Ta**maño del paquete.**

- **flags:** Banderas.
- **fid:** Id del flujo.
- D**irección fuente:** Nodo de origen del paquete. El formato es nodo.puerto
- **Dirección de destino:** Nodo destino del paquete. El formato es nodo.puerto
- **Número de secuencia del paquete**.
- **Id del paquete.**

Otras opciones que se pueden realizar en el script para que aparezcan en el NAM:

- **Color de los nodos:** por ejemplo si se quiere que el n0 aparezca en rojo, se escribir:

$ns n0 color red

- **Forma de los nodos**: por defecto, los nodos son redondos, pero pueden aparecer con una forma diferente. Por ejemplo tecleando:

$n1 shape box (o en lugar de "box" se puede usar "hexagon" o "circle")

- **Color de los enlaces**: se teclea por ejemplo:

$ns duplex-link-op $n0 $n2 color green

2.3.2. La herramienta XGraph

Otra herramienta que proporciona NS2 para el análisis de resultados es XGraph que permite crear representaciones gráficas de la salida de la simulación. Se trata de una herramienta sencilla utilizada para generar gráficos. Permite representar en una misma gráfica datos de un número indefinido de ficheros. La herramienta permite renombrar los títulos, numerar y etiquetar los ejes o leyendas, hacer zooms sobre la gráfica obtenida y grabar los gráficos como ficheros postscript.

Para invocar la herramienta XGraph basta con escribir en el script la siguiente línea:

exec xgraph xxxx.ns

Existe la misma herramienta que utiliza librerías de Matlab, se llama Tracegraph. En nuestro caso utilizaremos Tracegraph ya que nos permite obtener muchos más parámetros de la simulación y de una manera más sencilla. Se puede descargar de la página http://www.tracegraph.com/download.html eligiendo la

plataforma donde va a ser utilizada, ya sea Linux o Windows. Se accede en dicha web y se rellena un formulario con nombre, dirección de correo electrónico y la ciudad; y a continuación se accede a la descarga del software y las librerías de Matlab necesarias para su utilización.

A continuación se muestra la primera ventana del tracegraph donde se puede acceder a la traza que se genera anteriormente mediante el botón File.

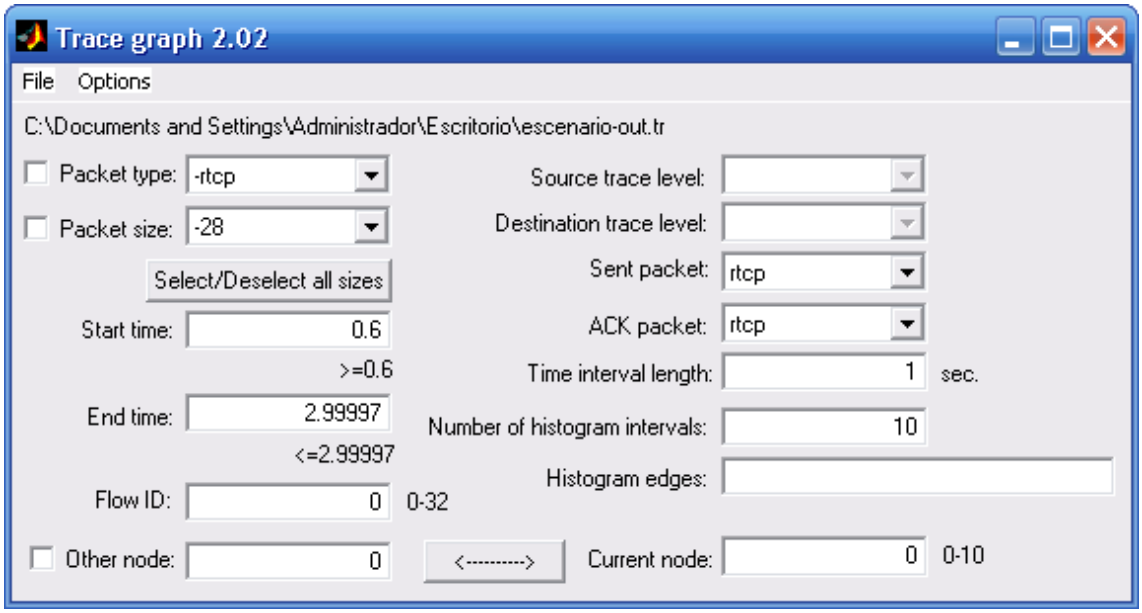


Figura 11. Primera ventana Tracegraph

A continuación se muestra una gráfica de una traza generada.

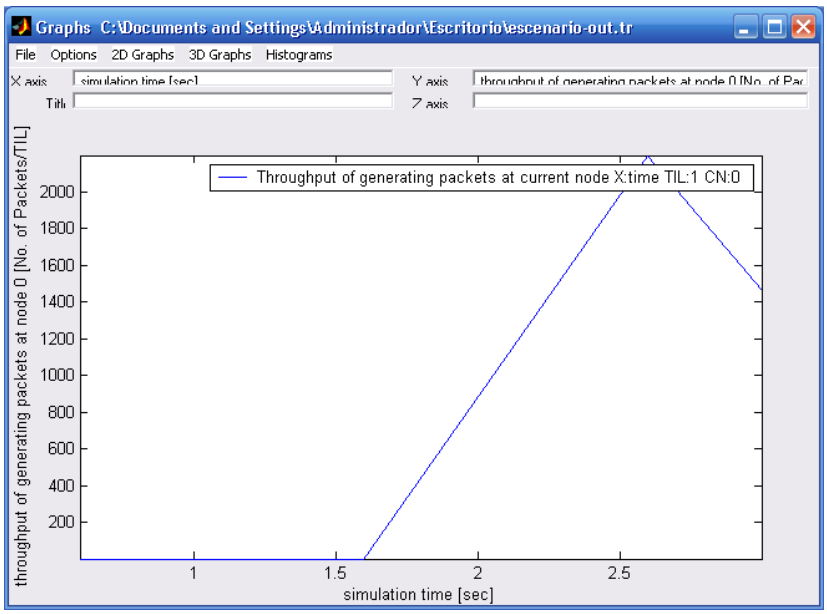


Figura 12. Análisis de traza en Tracegraph

Se puede ver las gráficas de la traza que se crea en la ventana Trace Graph en el menú 2D Graphs → y a continuación se muestran en el menú desplegable diferentes gráficas: throughput, jitter, packet size, etc.

Otras de las opciones que otorga el tracegraph es una ventana que muestra toda la información de simulación, el Network Information. Se muestra dicha figura a continuación.

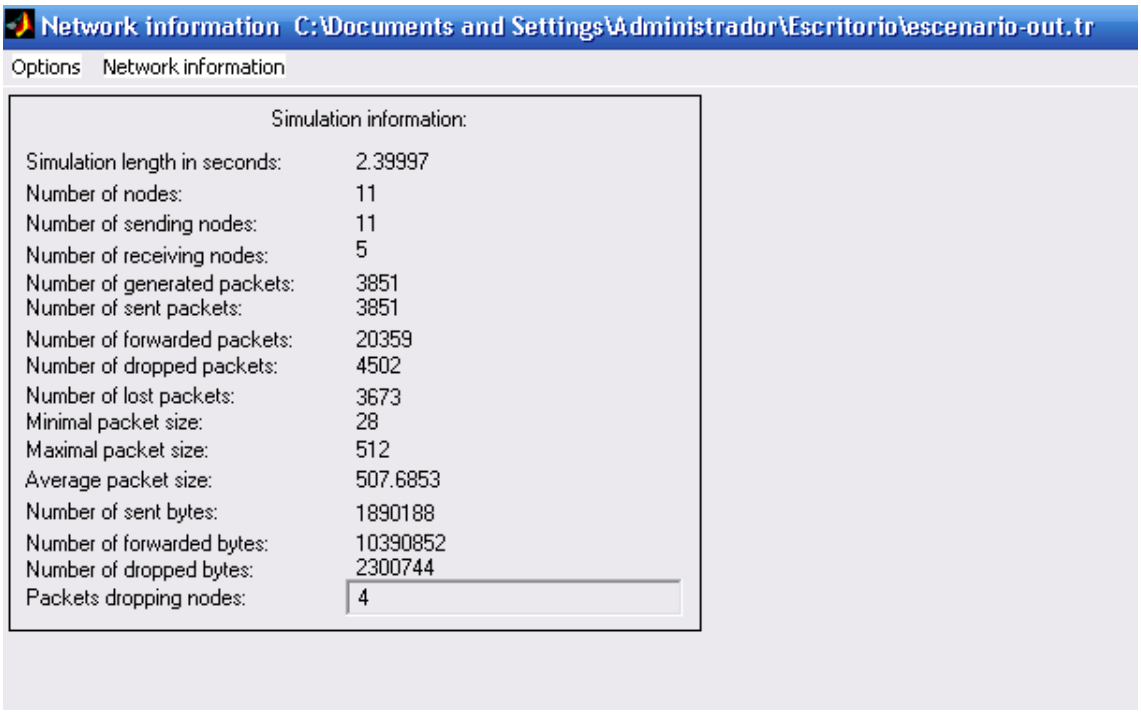


Figura 13. Información sobre simulación en Tracegraph

3. IMPLEMENTACIÓN DEL PROTOCOLO RTP/RTCP

3.1. RTP en Opnet Modeler

Para hacer uso del protocolo RTP en Opnet Modeler se dispone de la aplicación VoIP implementada en dicho simulador. El protocolo RTCP no está disponible en el Opnet Modeler (una desventaja frente al NS2).

El tráfico VoIP es la única aplicación en la herramienta Modeler que soporta el protocolo RTP por defecto. A continuación se especifica cómo utilizar dicha aplicación en el OPNET, que protocolos de transporte utiliza, que atributos se pueden modelar en la aplicación de voz y cómo se transportan los paquetes de voz de una fuente a su destino.

Una forma de modelar el tráfico de VoIP en OPNET es utilizar la aplicación de voz predefinida. Una aplicación de voz permite a dos clientes establecer un canal virtual sobre el que comunicarse usando señales de voz codificadas. El protocolo de transporte utilizado por defecto es UDP para esta aplicación. Los datos de voz llegan en impulsos seguidos de un periodo de silencio. También pueden especificarse esquemas de codificación para la conversión de voz a paquetes de datos.

Algunos atributos para modelar la aplicación de voz son:

- **Silence Length** (segundos): duración del silencio para las llamadas entrantes y salientes, junto con la distribución asociada (tiempo de OFF).
- **Talk Spurt Length** (segundos): duración de la longitud de las llamadas entrantes y salientes junto con las distribuciones asociadas (tiempo de ON).
- **Symbolic Destination Name**: nombre de destino simbólico del cliente.
- **Encoder Scheme**: codificación en el cliente.
- **Voice Frames per Packet:** número de tramas de voz que pueden ser enviadas en un solo paquete.
- **Type of Service:** parámetro de calidad de servicio para la asignación de prioridad del tráfico.
- **RSVP Parameters:** los parámetros de RSVP para hacer reservas de ancho de banda.

Cada paquete VoIP es enviado en una trama. Con cada paquete de 160 bytes de tamaño se le debe sumar las cabeceras de cada protocolo. Estas cabeceras son RTP + UDP + IP y Ethernet con el preámbulo de los tamaños de 12 + 8 + 20 + 26, respectivamente. Por lo tanto, un total de 226 bytes, o 1808 bits, debe ser transmitido 50 veces por segundo, o 90,4 Kbps, en una dirección. En ambos sentidos, el ancho de banda necesario para una sola llamada es de 100 pps o 180,8 Kbps para un flujo simétrico.

3.2. Implementación de los protocolos RTP/RTCP en NS2

La implementación de los protocolos RTP/RTCP nativo en el simulador NS2 es bastante genérica ya que sólo proporciona las principales funcionalidades de un protocolo de transporte común. Así pues, puede afirmarse que dicha implementación no se ajusta estrictamente a las especificaciones marcadas en la RFC 3550, en la cual se define el funcionamiento de dichos protocolos.

3.2.1. Imperfecciones encontradas

En cuanto a las incorrecciones o simplificaciones detectadas en la implementación de los protocolos RTP/RTCP en el simulador NS2, se puede remarcar:

- No se definen todos los tipos de paquetes del protocolo RTCP especificados en la RFC 3550, como son los mensajes *RTCP APP*, *RTCP SDES*, *RTCP BYE*, etc.

- Los tamaños de los paquetes y los formatos de los campos no se ajustan a lo especificado en la RFC. Por ejemplo:

    - Los números de secuencia de los paquetes RTP en NS2 son de 32 bits y la RFC apunta que esta longitud debe ser de 16 bits.
    - Los valores del tiempo global NTP (*Network Time Protocol*) incluidos en los paquetes RTCP SR se toman del reloj del planificador de eventos (*event scheduler*) del simulador, con lo que no se corresponden al formato temporal especificado en el protocolo NTP (expresado en año, mes, día, hora, minutos, segundos y fracciones de segundos).
    - Las marcas de tiempo de los relojes locales – *timestamp* – incluidas en los paquetes tienen una longitud de 64 bits y también se toman del reloj del planificador de eventos. En este caso la RFC especifica que su longitud sea de 32 bits y que su valor se tome de un temporizador cuyo valor inicial sea aleatorio y que se incremente de forma monotónica y lineal, con una precisión acorde con los parámetros de calidad de servicio requeridos por la aplicación utilizada.
    - Se incluye un campo denominado *bye* de 8 bits de longitud en los paquetes *RTCP RR* utilizado para gestionar el abandono de una sesión RTP. Este campo no se estipula en la RFC ya que dicha tarea se debería cometer mediante el envío, por parte del participante que desea abandonar la sesión, de un paquete *RTCP BYE* definido específicamente para este fin. Además, estos 8 bits

afectan a la longitud efectiva de los paquetes RTCP y, por tanto, a la sobrecarga (*overload*) introducido en la red.
    - Los dos primeros octetos de los paquetes RTP y RTCP se definen como un único campo entero llamado *flags*, obviando los campos de *Versión*, *Padding*, *eXtensión* y *Count of CSRC* definidos en la RFC.
    - El campo referente a la lista de fuentes contribuyentes – CSRC – no se tiene en cuenta.
    - Etc.

- Se simplifican muchos procedimientos. Por ejemplo: los números de secuencia inicial de los paquetes no son aleatorios, los valores de los temporizadores de los relojes locales de todas las sesiones se toman del reloj del planificador de eventos del simulador, con lo que todos los participantes estamparán en los paquetes enviados unos valores temporales definidos en un eje de tiempos común para todas ellas, etc.

- Cuando un Agente RTP recibe paquetes RTP procedentes de una fuente que pertenece al grupo al que está unido, se los pasa a la sesión a la que pertenece para su tratamiento, ésta calcula el *jitter* y, a continuación, los libera de la memoria, sin realizar ningún procesado más para estos (como por ejemplo el cálculo de su *playout delay*, su almacenaje en un buffer o su posterior reproducción).

3.2.2. Detalles de la implementación

Generalmente, los protocolos específicos se implementan en el simulador NS2 como Agentes (*Agent*), los cuales se predefinen en las *clases* particulares. En este caso, los protocolos RTP y RTCP son implementados como las clases C++ *RTPAgent* y *RTCPAgent*, ambas implementadas en los ficheros *rtp.cc* y *rtcp.cc* (remarcados en naranja en la Figura 14), respectivamente. Estos agentes se ocupan principalmente de la generación, el envío y la recepción de los paquetes RTP y RTCP.

Por otro lado, la clase *RTPSession*, definida en el fichero *session-rtp.cc*, es la encargada de gestionar todos los procedimientos de mantenimiento de la sesión RTP, como por ejemplo, el cálculo del intervalo de envío de paquetes RTCP, su confección, el mantenimiento de las tablas de datos de los participantes, etc.

Los ficheros C++ citados anteriormente utilizarán como fichero de cabeceras el archivo *rtp.h*, que se encuentra en el directorio *ns-2.XX/apps*, al igual que el archivo *rtp.cc* (ambos remarcados en naranja en la parte derecha de la Figura 14). Nótese que con ns-2.XX se pretende hacer referencia a la versión de NS2 utilizada. En el caso que nos ocupa XX debería sustituirse por 33 (*ns-*

2.33). Se ha preferido esta notación ya que la estructura de directorios del simulador es casi análoga independientemente de la versión de éste con la que se trate.

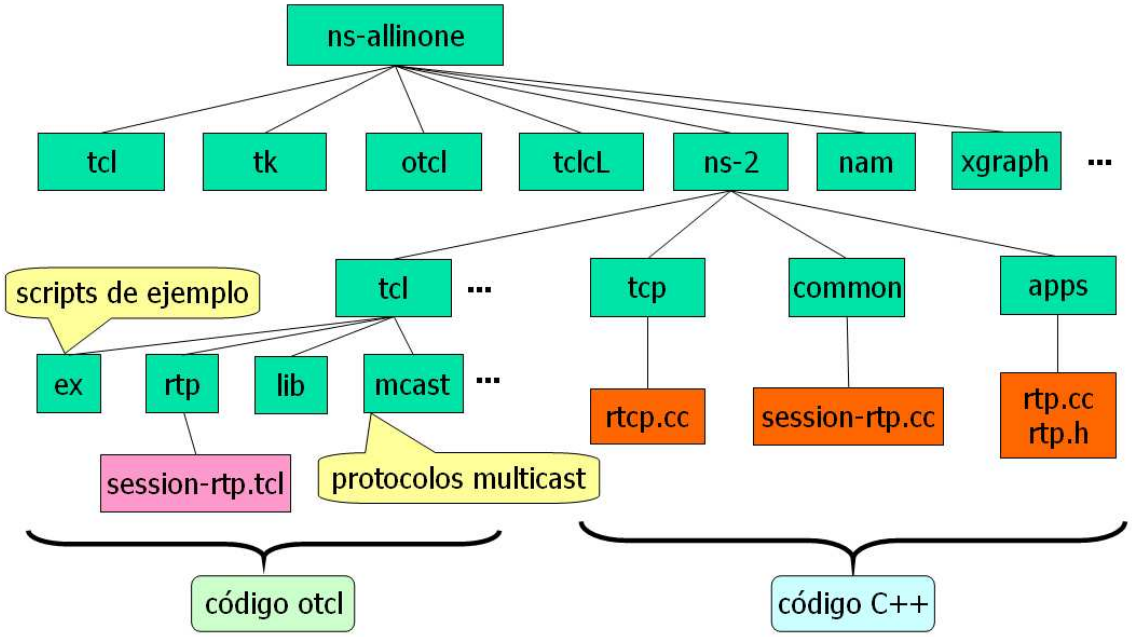


Figura 14. Estructura de directorios del simulador NS2 (se ha omitido el número de versión vinculado al nombre de los directorios debido a que su estructura es análoga independientemente de ésta).

3.2.3. Estado del arte

El simulador de redes NS2 representa una de las herramientas de simulación más ampliamente utilizadas por la comunidad investigadora en el ámbito de las redes de comunicaciones. Además, esta herramienta continuamente está en proceso de mejora y diversificación, gracias a la contribución por parte de la comunidad de investigadores aportando su propio código y modelos de simulación, puesto que se trata de una herramienta de código abierto (open source). Actualmente, dicha herramienta incorpora soporte para una amplia variedad de protocolos de red, agentes de transporte, aplicaciones, algoritmos, pensados tanto para redes cableadas como para redes inalámbricas.

Además de la implementación nativa de los protocolos RTP/RTCP en el simulador NS2, existen otras implementaciones con el objetivo de adaptar la implementación de estos protocolos a sus respectivos campos de trabajo, completando de esta manera su implementación e incorporando nuevas funcionalidades útiles para sus propósitos.

En primer lugar, Víctor Carrascal, del Grupo de Servicios Telemáticos de la Universidad Politécnica de Cataluña (UPC), definió unos nuevos Agentes RTP y RTCP con nuevas funcionalidades encaminadas al control de las pérdidas y del

jitter (variación del retardo en la red) en la transmisión (*streaming*) de trazas MPEG-2 en entornos inalámbricos con QoS (802.11e). El código fuente y las instrucciones para su instalación se pueden encontrar en su página personal [13].

Por otra parte, existe otra implementación más completa de los protocolos RTP/RTCP, que se ajusta en mayor medida a la RFC 3550, desarrollada por Academia de Investigación del Instituto de Tecnologías de la Computación perteneciente a la Universidad de Patras (Grecia). Esta implementación tiene como objetivo dotar de un comportamiento *TCP Friendly* en cuanto a la adaptación de la tasa de transmisión por parte de la fuente, en función de los parámetros recibidos en los informes de realimentación (*feedback*) RTCP enviados por los receptores, con el objetivo de ajustar el control de congestión en la red [14].

A fecha de octubre de 2009, esta implementación se encuentra pendiente de una nueva revisión que incorporará mayores funcionalidades y corregirá algunas imperfecciones detectadas en su código fuente.

Por otro lado, existe la implementación de un modelo flexible para la simulación de aplicaciones VoIP, así como para la evaluación de prestaciones de este tipo de aplicaciones en diversos escenarios de red. Debido a ello, un grupo de investigadores de la Universidad de Pisa (Italia), desarrolló un nuevo módulo para el simulador NS2, posibilitando la simulación de aplicaciones VoIP con un mejor grado de realismo respecto al código nativo del simulador [15].

El módulo VoIP es utilizado en este proyecto para ser comparado con el simulador Opnet Modeler, ya que este último solo utiliza el protocolo RTP utilizando el tráfico VoIP como se ha explicado anteriormente [16].

4. ESCENARIO Y PRUEBAS

En este capítulo se muestra la instalación del Opnet Modeler y del Network Simulator 2, junto con el módulo VoIP implementado por los investigadores italianos.

Se va a mostrar de manera detallada la creación y simulación de un escenario real con la utilización de la aplicación VoIP en los dos simuladores.

4.1. El simulador OPNET

4.1.1. Instalación

1. Se decide descargar OPNET 14.5 entre muchas versiones disponibles en el siguiente enlace:

https://enterprise1.opnet.com/support/product/md/145PL8

Para la correcta descarga es preciso tener una cuenta, por lo que antes de nada, hay que registrarse.

2. En caso de tener una cuenta creada se pasará directamente al paso 4, para seguir con la instalación.

3. Si no se tiene una cuenta, se accederá al siguiente enlace:

https://enterprise1.opnet.com/support/ts_password_req.html

Aparece un formulario a rellenar para enviarlo. Tras enviar el formulario se recibe una cuenta y una contraseña a través de la dirección de correo electrónico.

4. Si se tiene una cuenta, se accederá a la parte derecha del siguiente enlace:

https://enterprise1.opnet.com/support/product/md/

5. En la cuenta se introducirá el nombre de usuario y la contraseña que se obtiene en el correo electrónico. Se leerá cuidadosamente los requisitos del sistema y se hará clic en "Estoy de acuerdo...". Se instalará el software paso a paso. Tras esto, se empezará a descargar y se esperará a que acabe dicha instalación.

6. Una vez instalado, se ejecutará el OPNET en el menú Inicio.

7. En el menú Inicio, se hará clic en “Administración de licencias”.

8. Al entrar en “Administración de licencias”, hay que registrarse en el sistema utilizando el nombre de usuario y contraseña existentes.

9. Para instalar OPNET, se necesita un compilador de C++, se va a utilizar el Visual Studio 2005 y se tiene que tener en cuenta que los PATH sean los correctos y también hay que copiar unas librerías del OPNET en otro directorio [17]. Si no obtendrás errores como:

1) Si no están los PATH correctamente especificados:

comp_msvc: Unable to execute compiler (Win32 error code: 2)
Check that Visual C++ has been installed correctly, and that
its BIN directory is included in the Path environment variable

2) Si no se han copiado los archivos DLL aparece un error que se indica en la FAQ ID 1685 R6034: Microsoft Visual C++ Runtime Library “An application has made an attempt to load the C runtime library incorrectly.”

SOLUCIÓN:

1) Añadir en los PATH:

   Inicio → Panel de Control (Vista Clásica) → Sistema: Pestaña Opciones Avanzadas - Botón “Variables de Entorno”, y en variables del sistema modificar o añadir las que no existan ya.

   Path=
   C:\Archivos de Programa\Microsoft Visual Studio 8\Common7\IDE;
   C:\Archivos de Programa\Microsoft Visual Studio 8\VC\BIN;
   C:\Archivos de Programa\Microsoft Visual Studio 8\Common7\Tools;
   C:\Archivos de Programa\Microsoft Visual Studio8\Common7\Tools\bin;
   C:\Archivos de Programa\Microsoft Visual Studio 8\VC\PlatformSDK\bin;
   C:\Archivos de Programa\Microsoft Visual Studio 8\SDK\v2.0\bin;
   C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727;
   C:\Archivos de Programa\Microsoft Visual Studio 8\VC\VCPackages;
   C:\Archivos de Programa\Microsoft Visual Studio 8\SDK\v2.0\Bin;
   C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727;
   C:\Archivos de Programa\Microsoft Visual Studio 8\VC\bin;
   C:\Archivos de Programa\Microsoft Visual Studio 8\Common7\IDE;
   C:\Archivos de Programa\Microsoft Visual Studio 8\VC\vcpackages;

   INCLUDE=
   C:\Archivos de Programa\Microsoft Visual Studio 8\VC\ATLMFC\INCLUDE;
   C:\Archivos de Programa\Microsoft Visual Studio 8\VC\INCLUDE;
   C:\Archivos de Programa\Microsoft Visual Studio 8\VC\PlatformSDK\include;

C:\Archivos de Programa\Microsoft Visual Studio 8\SDK\v2.0\include;

LIB=
C:\Archivos de Programa\Microsoft Visual Studio 8\VC\ATLMFC\LIB;
C:\Archivos de Programa\Microsoft Visual Studio 8\VC\LIB;
C:\Archivos de Programa\Microsoft Visual Studio 8\VC\PlatformSDK\lib;
C:\Archivos de Programa\Microsoft Visual Studio 8\SDK\v2.0\lib;

LIBPATH=
C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727;
C:\Archivos de Programa\Microsoft Visual Studio 8\VC\ATLMFC\LIB;

NetSamplePath=C:\Archivos de Programa\Microsoft Visual Studio 8\SDK\v2.0;

DevEnvDir=C:\Archivos de Programa\Microsoft Visual Studio 8\Common7\IDE;

FrameworkDir=C:\WINDOWS\Microsoft.NET\Framework;

FrameworkSDKDir=C:\Archivos de Programa\Microsoft Visual Studio 8\SDK\v2.0;

FrameworkVersion=v2.0.50727;

VCBUILD_DEFAULT_CFG=Debug^|Win32;

VCBUILD_DEFAULT_OPTIONS=/useenv;

VCINSTALLDIR=C:\Archivos de Programa\Microsoft Visual Studio 8\VC;

VSINSTALLDIR=C:\Archivos de Programa\Microsoft Visual Studio 8;

2) Copiar en otro directorio:

Tras configurar las variables de entorno, se abrirá el explorador de Windows y se copiará los ficheros que se encuentran en el directorio <opnet_dir>\sys\pc_intel_win32\bin\manifest al otro directorio <opnet_dir>\sys\pc_intel_win32\bin.

Si se usa .NET 2005 en una máquina de 64-bit machine, se necesitará copiar desde el directorio <opnet_dir>\sys\pc_amd_win64\bin\manifest al otro directorio <opnet_dir>\sys\pc_amd_win64\bin.

4.1.2. Creación de la red

A continuación se explicará detalladamente cómo realizar una simulación en OPNET Modeler de una red VoIP con transmisión mediante el protocolo RTP donde el escenario estará compuesto por cinco nodos, uno de ellos será el servidor VoIP, otro será el nodo destino y los otros routers intermedios.

Se define el tráfico que correrá por la red y se mostrará como recoger y analizar resultados.

4.1.2.1. Abrir el simulador OPNET MODELER

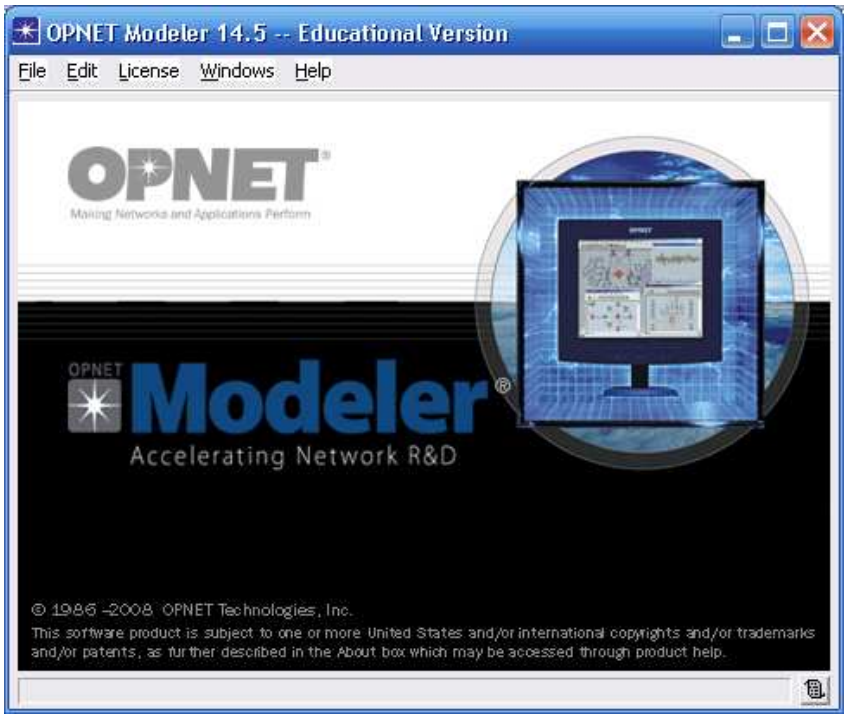


Figura 15. Acceso al simulador

Antes de empezar con la simulación se comprueban las preferencias del Opnet Modeler, para ello se selecciona Preferences desde el menú Edit.

Se debe comprobar que el atributo license_server corresponde con el nombre del host desde el que se obtuvo la licencia. Si Opnet Modeler obtuvo directamente la licencia del ordenador en el que fue instalado, este valor debería ser localhost.

También se debe modificar el atributo license_server_standalone a TRUE, ya que este atributo especifica si el programa actúa como su propio servidor de licencias.

Crear un nuevo proyecto File → New → Project

Se crea un nuevo proyecto al entrar en el botón File, a continuación aparece un menú desplegable donde al cliquear sobre New y a continuación Project, se crea un nuevo proyecto.

Aparece una nueva pantalla donde se introducen dos datos. En la primera casilla se asigna un nombre al proyecto. El nombre del escenario puede ser distinto al del proyecto, ya que en un mismo proyecto puede haber varios escenarios para comparar diferentes casos. La siguiente casilla es aconsejable que esté activada ya que así nos ayudará a crear el escenario.

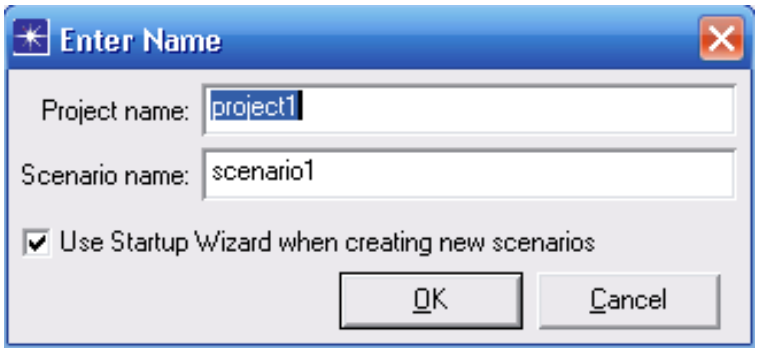


Figura 16. Asignación de nombre al proyecto y al escenario

4.1.2.2. Elección del tipo de escenario

A continuación se especifica el tipo de escenario. Se escoge la primera opción, para crear un escenario vacío (create empty scenario) y se cliquea a siguiente.

A continuación se escoge la opción "Campus" ya que la red se quiere simular en un espacio abierto. Seguidamente se aceptan los valores por defecto, cambiando únicamente las dimensiones del espacio, 10 km x 10 km. Las siguientes opciones se aceptan tal y como vienen por defecto haciendo click sobre el botón de "finish".

El escenario deberá quedar de la siguiente manera:

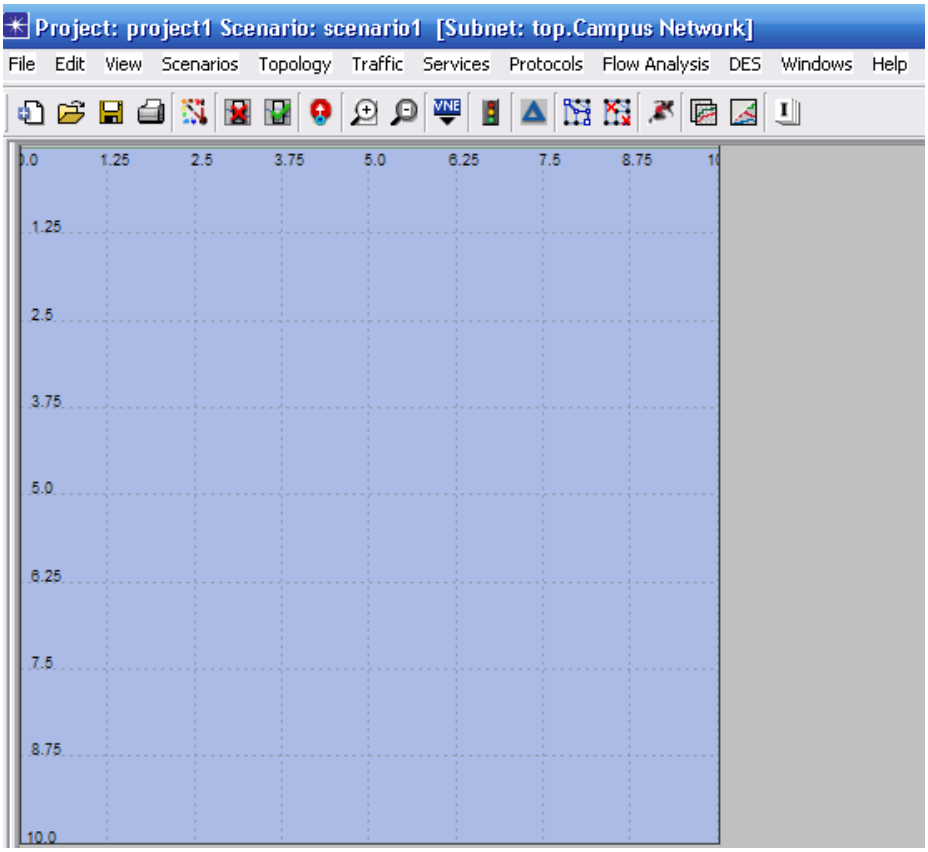


Figura 17. Escenario vacío en Opnet Modeler

4.1.2.3. Paleta de objetos

El escenario representado como una cuadrícula aparece vacío. En este escenario se debe colocar los diferentes objetos que se encuentran en la "paleta de objetos" ordenada por familia de protocolos.

Para acceder a la paleta de objetos directamente se accede mediante el botón que se encuentra en la parte superior izquierda.

Figura 18. Botón de acceso a la paleta de objetos

4.1.2.4. Selección de elementos

Desde la paleta de objetos, se escoge el objeto ethernet_server y se arrastra al escenario. Es utilizado como el servidor de VoIP de nuestra topología.

Como routers intermedios se utilizan el router Cisco 7200, router Cisco 2621 y el router Nortel passport 8600. Se Cambia el nombre de los objetos para poderlos distinguir mejor. Para ello, se hace clic con el botón derecho del ratón en cada objeto y se selecciona la opción "Set name". En nuestro caso hemos

llamado kisin al primer router Cisco 7200, gukumatz al segundo router Cisco 2621 y np8-gandia al router Nortel passport 8600.

Se utiliza como objeto para el ordenador destino, el ethernet_wrkstn de la paleta de objetos, también se cambian el nombre como anteriormente a por ejemplo destino1.

Para la interconexión de los objetos se utilizan los diferentes tipos de enlaces que se pueden elegir en la paleta de objetos, en nuestro caso utilizamos una conexión 10BaseT entre el servidor multimedia y el router al que se une, llamado kisin y también la unión de enlaces 10BaseT para la unión del router np8-gandia con el destino 1. Para las demás interconexiones se utiliza el enlace PPP_E1 de 2Mbps. Una vez realizada toda la interconexión se seleccionan todos los enlaces y se hace clic con el botón derecho sobre uno de ellos, y se indica edit attributes. Se indica la opción "Apply to selected objects" para que los cambios que se realizan se guarden en todos los objetos seleccionados. A su vez se hace clic sobre "Advanced" y aparece la opción "delay", donde se indica el retardo que introducen los enlaces en la simulación. En nuestro caso se va a utilizar un delay de 20 milisegundos. A continuación se indica la figura con los cambios realizados.

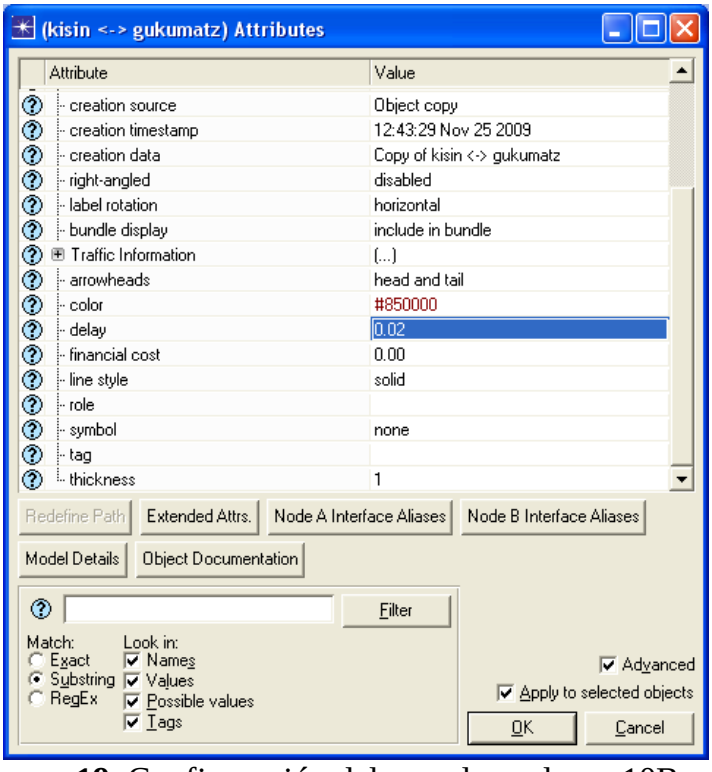


Figura 19. Configuración delay en los enlaces 10BaseT

Seguidamente se guardan los cambios en File → Save. Esta opción se realiza cada cierto tiempo para evitar problemas.

El escenario de simulación debe quedar de la siguiente manera:

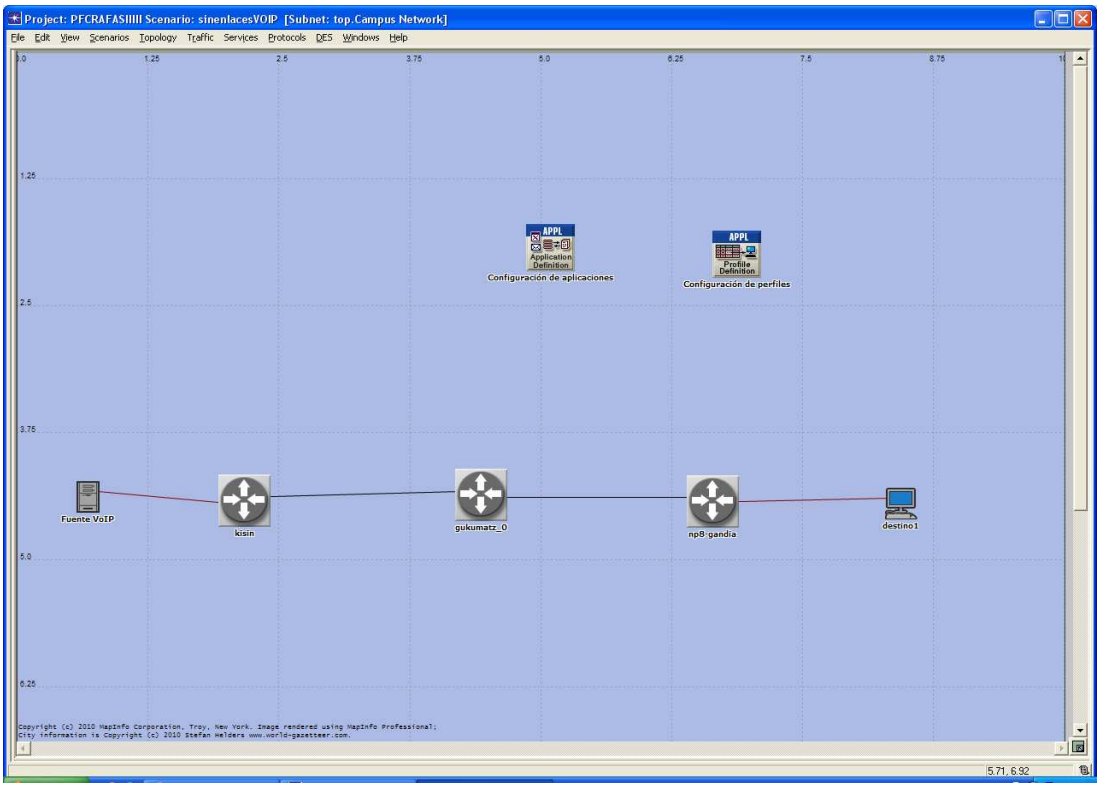


Figura 20. Escenario en Opnet Modeler

4.1.2.5. Definir background

El background es la carga de fondo introducida en una simulación, tráfico que se introduce de forma indeseada en un enlace para producir o forzar en la simulación pérdida de paquetes, servirá para analizar diferentes tiempos y situaciones en nuestro escenario.

Se utiliza un background entre todos los routers con una carga de definida de la siguiente manera, del inicio hasta los 120 segundos con una carga de 1.000.000 bps, desde los 120 segundos a los 180 segundos con una carga de 1.900.000 bps, desde los 180 segundos a los 240 segundos con una carga de 1.980.000 bps para forzar pérdidas y desde los 240 segundos hasta el final de la simulación sin carga de fondo, es decir 0 bps.

Para definir el background se hace clic con el botón derecho del ratón sobre el enlace y se indica edit Attributes, en la figura que aparece se despliega en la opción Traffic Information una nueva Row 0, donde se indica la carga (Traffic Load (bps)), ésta carga se puede seleccionar entre las disponibles en el Opnet Modeler o editar como es el caso.

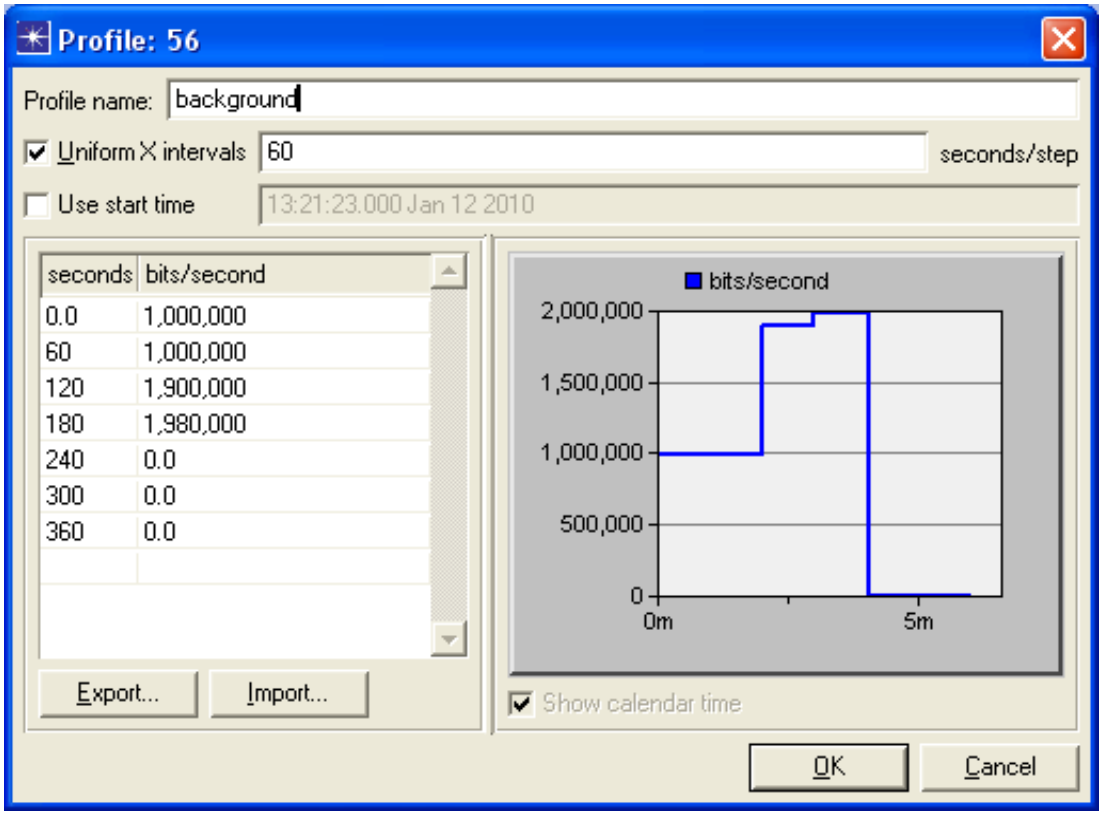


Figura 21. Se edita el background

Una vez realizada la edición del background aparecen los cambios como se indica en la siguiente figura. Se debe indicar en las dos direcciones y en los dos enlaces intermedios. Los demás valores se dejan por defecto y se guardan los cambios realizados.

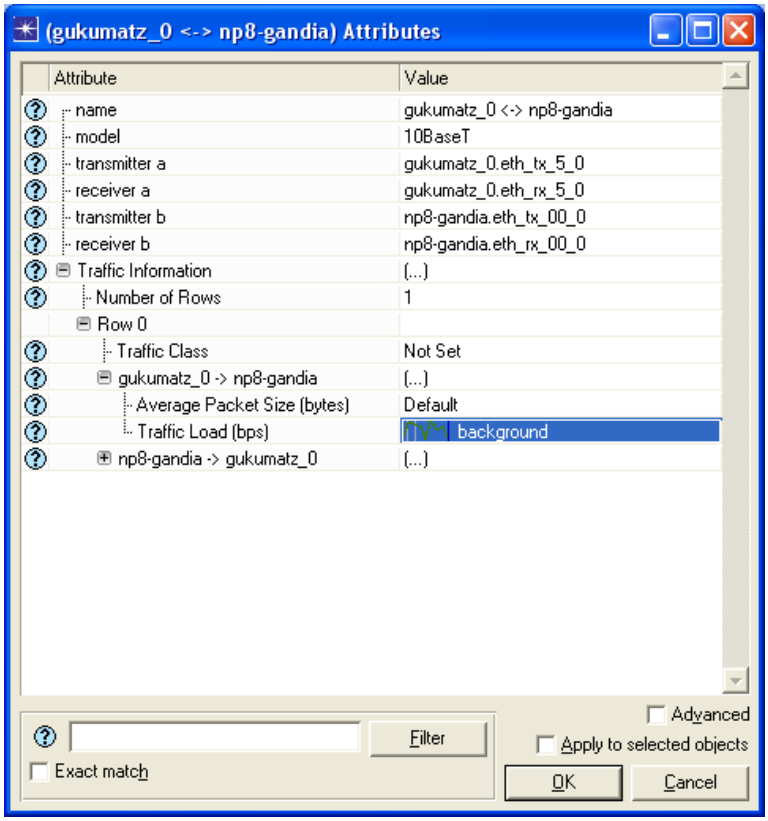


Figura 22. Indica los atributos entre los routers gukumatz_0 y np8-gandia

4.1.2.6. Definir Aplicación

Con el botón derecho del ratón se hace clic en el objeto Application Config y se selecciona la opción Advanced Edit Attributes. En esta opción se configuran todas las aplicaciones soportadas en la red. Se define un servicio VoIP, donde el Servidor multimedia ejercerá la función de Servidor VoIP.

Se expande la opción Application Definitions. Se hace clic en la derecha de rows y se selecciona el número 1. De esta manera se configura una única aplicación en este escenario de simulación. Aparece una nueva línea (row0). Se expande y e asigna un nombre, en nuestro caso “VoIP”.

Se expande la opción Description. Aquí se elige el tipo de aplicación a definir. Haciendo clic a la derecha de Voice y seleccionando la opción Edit. Se definen los parámetros de nuestro servicio VoIP. En nuestro caso se elige la opción “Edit”.

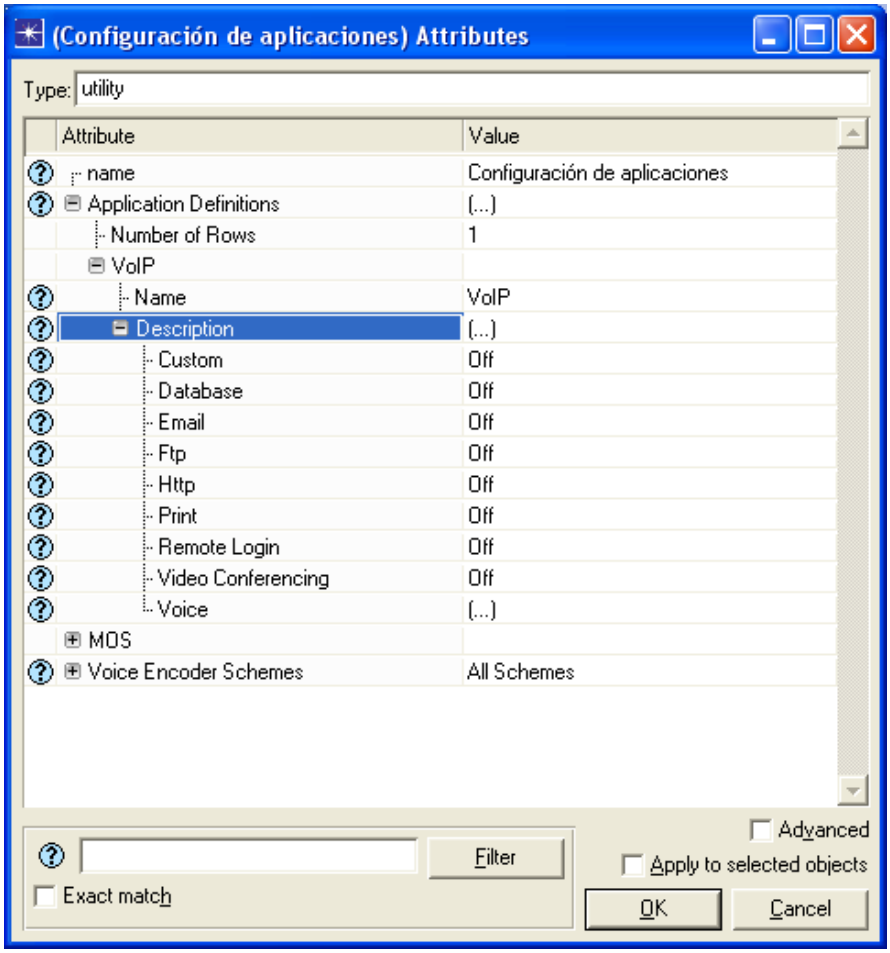


Figura 23. Configuración de Aplicaciones VoIP

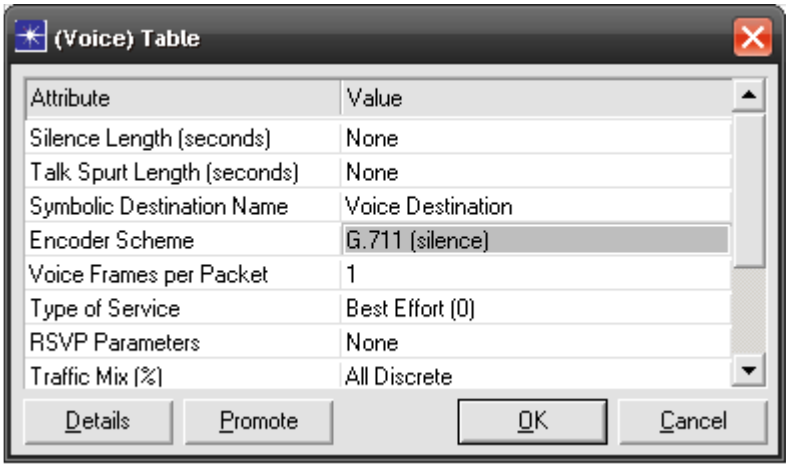


Figura 24. Valores por defecto Voice

En la figura 24 se puede observar la opción Encoder Scheme, esta opción es utilizada para cambiar la codificación de VoIP a utilizar en el escenario, la opción que se utiliza es “G.711”.

G.711 es un estándar de la ITU-T para la compresión de audio. G.711 es un estándar para representar señales de audio con frecuencias de la voz humana, mediante muestras comprimidas de una señal de audio digital con una tasa de

muestreo de 8000 muestras por segundo. El codificador G.711 proporciona un flujo de datos de 64 kbits por segundo.

Otra de las opciones a modificar en la voz utilizada son los parámetros “Silence Length (seconds)” y “Talk Spurt Length (seconds)”, se modula el tiempo que se alterna entre señal y silencio (ON/OFF) en una conversación. Se ha demostrado que la duración de los periodos se ajusta de forma más fidedigna con una distribución Gamma (media 401.1ms) para los ON y con una Weibull (media de 577.5 ms) para los OFF [12]. Dado que el modelo exponencial es ampliamente aceptado como aproximación y por tratamiento matemático, este es el usado.

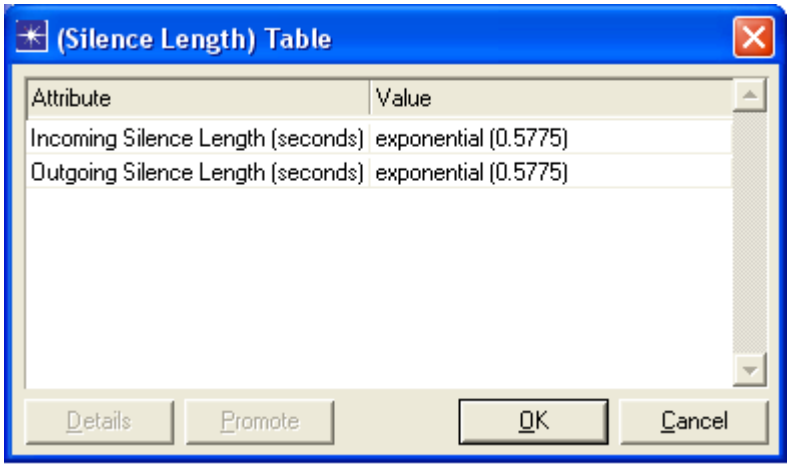


Figura 25. Distribución de la duración del período de silencio (OFF)

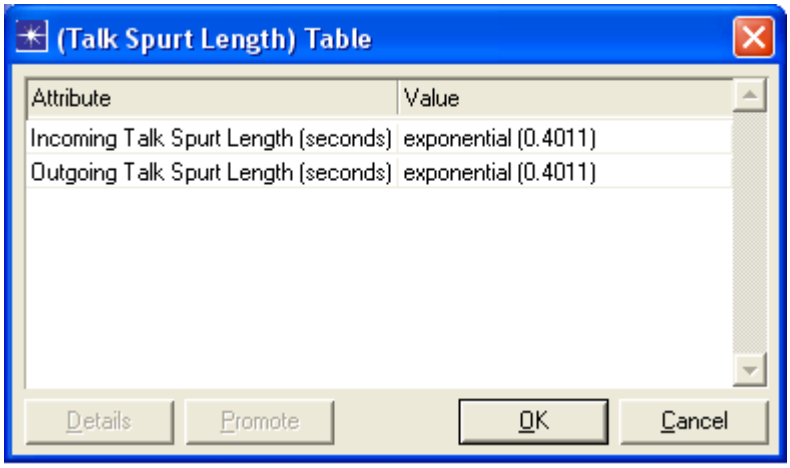


Figura 26. Distribución de la duración del período de ráfaga (ON)

4.1.2.7. Definir Perfil

Con el botón derecho del ratón se hace clic en el objeto Profile Config y selecciona la opción Advanced Edit Attributes. En esta opción se puede configurar el perfil del tráfico VoIP y también el inicio de la simulación.

Para modificar el inicio de la simulación desde el principio, se debe modificar "Start Time (seconds)", indicando "constant (0)", indica desde el segundo 0.

Se expanden la opción Profile Configuration. Se hace clic en la derecha de rows y se selecciona el número 1. De esta manera se indica que en el escenario solamente se configura un perfil de tráfico. Aparece una nueva línea (row0). Se expande y se asigna un nombre en Profile Name. En nuestro caso hemos puesto "Test VoIP".

El inicio de la aplicación VoIP se inicia al minuto 1, para que el intercambio inicial de información entre nodos no afecte a los resultados a analizar. Se debe modificar la pestaña "Start Time (seconds)", para éste caso se indica "constant (60)".

Los demás valores se dejan por defecto y se aceptan los cambios. Debería quedar tal como aparece en la siguiente figura.

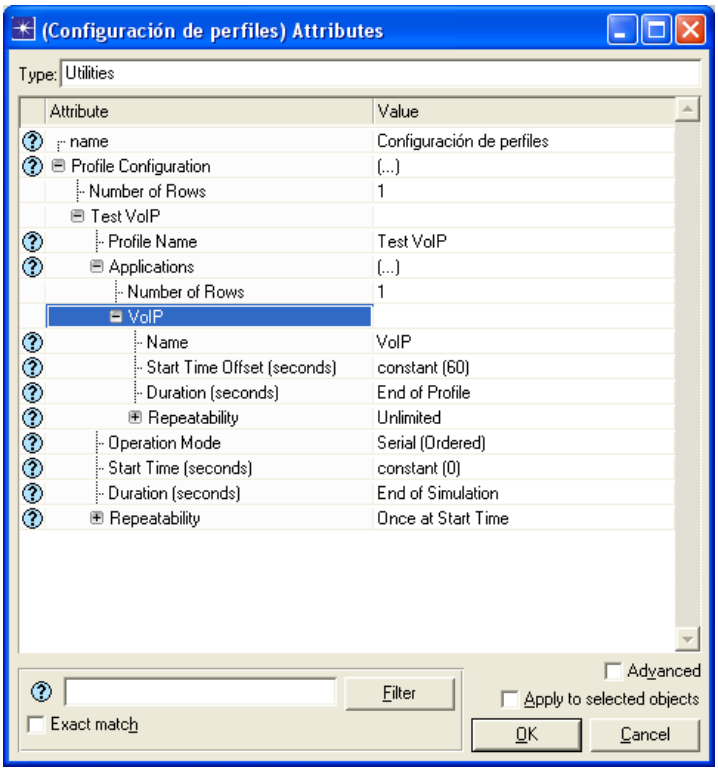


Figura 27. Parámetros de configuración de perfiles

4.1.2.8. Configuración de los objetos

4.1.2.8.1. Variables del servidor multimedia

Con el botón derecho del ratón se hace clic en el objeto servidor multimedia y se selecciona la opción Advanced Edit Attributes. En esta opción se configuran todas las variables del funcionamiento interno del nodo.

Para asignar la aplicación al servidor multimedia se debe expandir la opción Applications. A continuación se edita la opción Application: Supported Services. Se indicará que se soportan todos los servicios, indicando "All". Se aceptan los cambios.

Todos los demás parámetros son dejados por defecto tal y como se muestra en la siguiente figura:

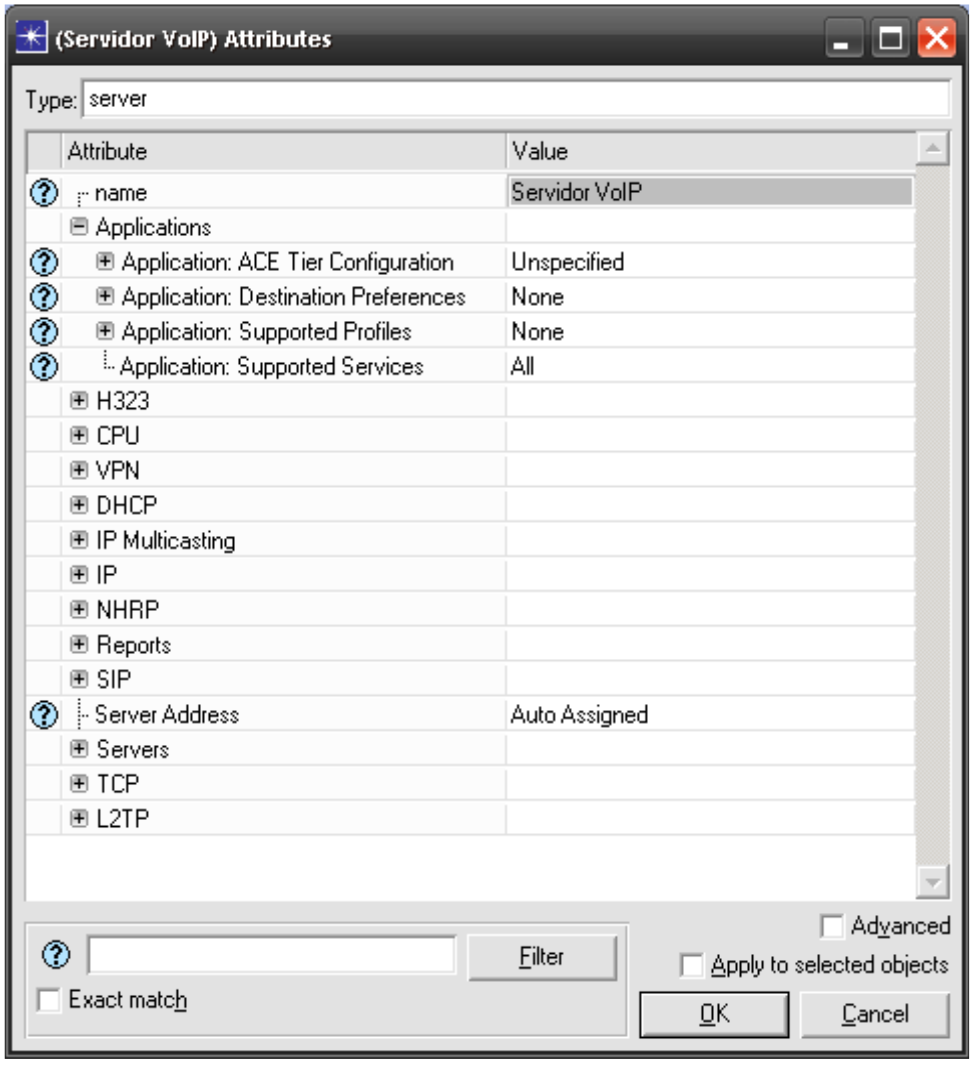


Figura 28. Parámetros del servidor multimedia

4.1.2.8.2. Variables de los destinos

Con el botón derecho del ratón se hace clic en uno de los destinos y se elige la opción Select Similar Nodes para seleccionar a todos los destinos a la vez y realizar los cambios todos a una. Se selecciona la opción Advanced Edit Attributes. Antes de nada se marca la casilla "Apply changes to selected objects" para que se guarden los cambios en todos los nodos seleccionados.

Para definir el uso de aplicaciones se debe expandir el menú de Applications. A continuación se edita la variable Application: Destination Preferences. Se insertamos una línea (1 row) en la ventana que aparece. En la casilla de Application se escoge la aplicación definida, en nuestro caso VoIP. En la casilla de Symbolic Name se escoge "VOZ" en nuestro caso. Seguidamente se hace clic en Actual Name y en Name se escoge en nuestro caso el nombre del servidor al cuál se conectarán los destinos, en nuestro caso "Servidor VoIP".

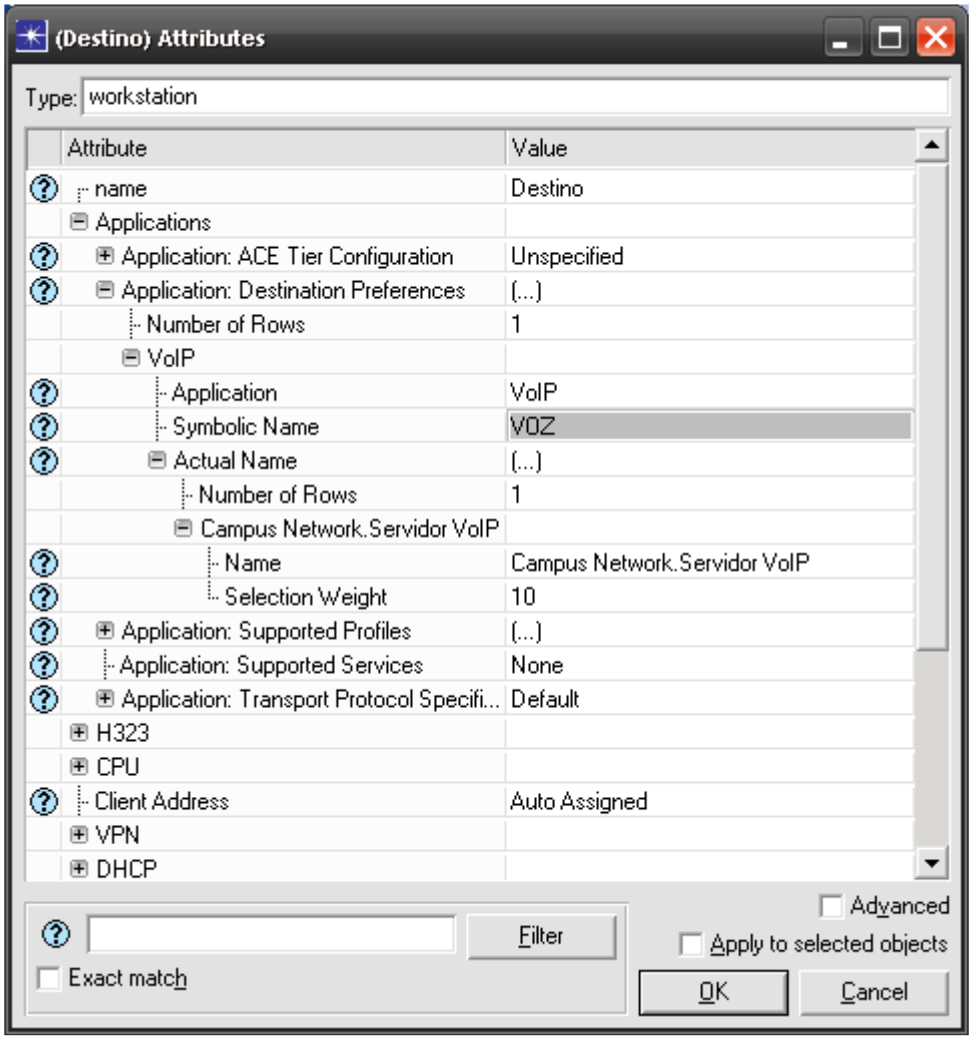


Figura 29. Parámetros del destino

A continuación, se definen los perfiles soportados por el cliente. Para ello, dentro del menú Applications se expande el submenú Application: Supported

Profiles. Aquí se añade una fila. Se expande la row 0 y en Profile Name escogemos el perfil definido. En nuestro caso es Test VoIP. Los cambios deben quedar tal y como aparece en la siguiente figura.

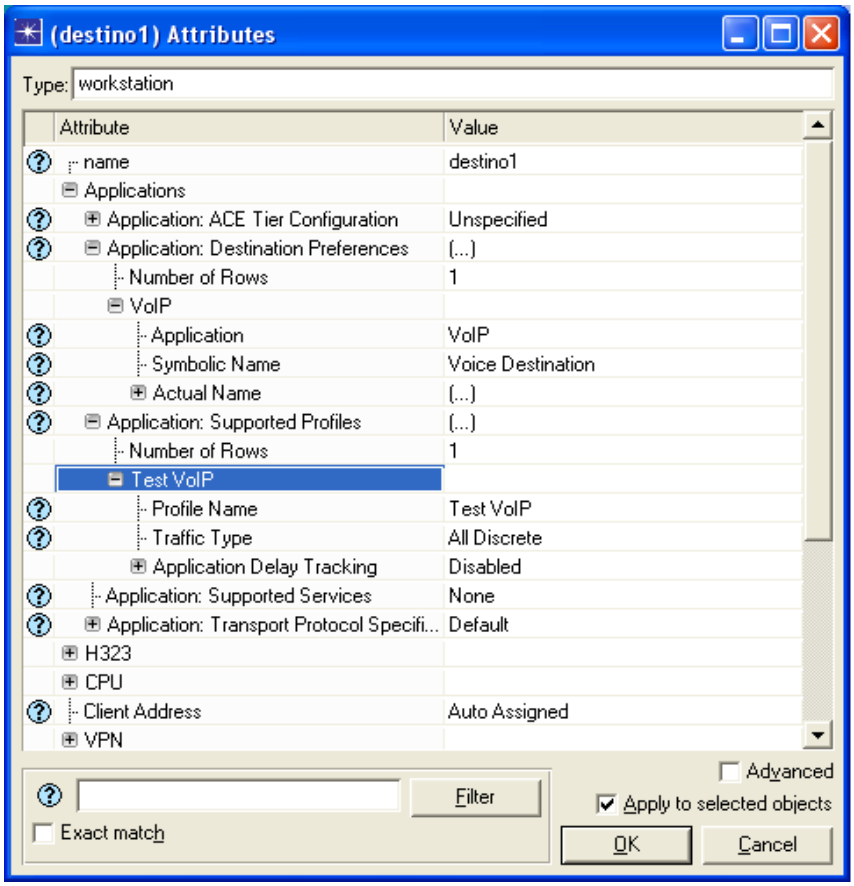


Figura 30. Parámetros aplicaciones de los destinos

A continuación pasamos a escoger las variables a analizar durante la simulación.

4.1.2.9. Elección de estadísticas

La selección de las estadísticas se puede hacer de diferentes maneras. Se pueden obtener resultados globales del escenario, resultados específicos de cada nodo, resultados de enlaces y resultados de demanda de tráfico.

Hay que tener en cuenta que una misma variable se puede analizar a partir de diferentes tipos de estadísticas, pero no se pueden repetir, es decir si por ejemplo se escoge la variable de Tráfico enviado en estadísticas globales no se podrá seleccionar también en estadísticas de nodo. En caso contrario puede generar un error en la simulación y no mostrar ninguna de las dos variables.

4.1.2.10. Selección de variables globales

Se hace clic con el botón derecho en cualquier sitio del escenario que no contenga ningún objeto y se escoge la opción Choose Individual DES Statistics. Se observa que aparecen tres menús que representan los tipos de variables. Se entra en Global Statistics y se seleccionan todas las de IP, RTP y Voice. De esta manera se analizan los tráficos globales de todo el escenario, es decir, la suma de los resultados de cada nodo.

Ahora se pasa a escoger las variables globales de cada nodo. Para ello se entra en Node Statistics y se escogen las estadísticas RTP, UDP y Voice y se aceptan los cambios. Estas estadísticas las tomará para cada nodo de la red.

Para ver qué significa cada variable se puede seleccionar la variable y presionar en la opción View Description.

En la figura 31 se puede observar la muestra de la elección de las estadísticas RTP.

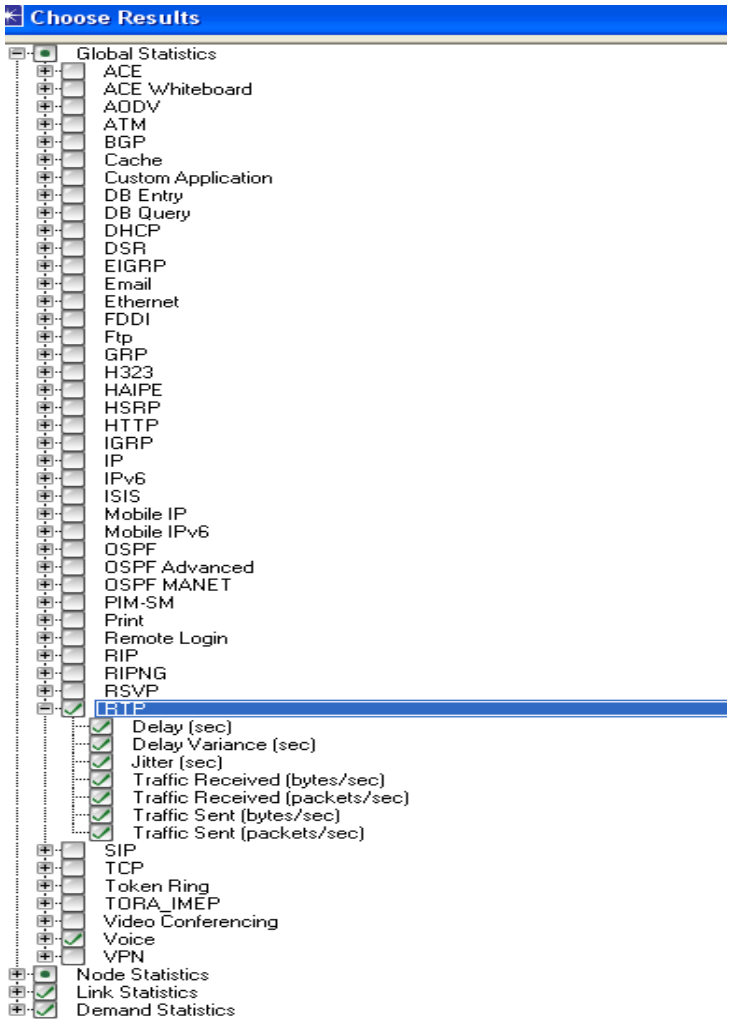


Figura 31. Elección de estadísticas

4.1.2.11. Choose Statistics (Advanced)

Una vez escogidas las estadísticas, para asegurar que se ha escogido todas las variables a estudiar se puede utilizar la herramienta Choose Statistics (Advanced) del menú DES de la barra de herramientas.

4.1.2.12. Configurar la simulación

Ya se puede proceder a ejecutar la simulación. Para ello se accede a partir de la opción Configure/Run Discrete Event Simulation o a través del siguiente acceso directo:

Aparecerá la siguiente ventana:

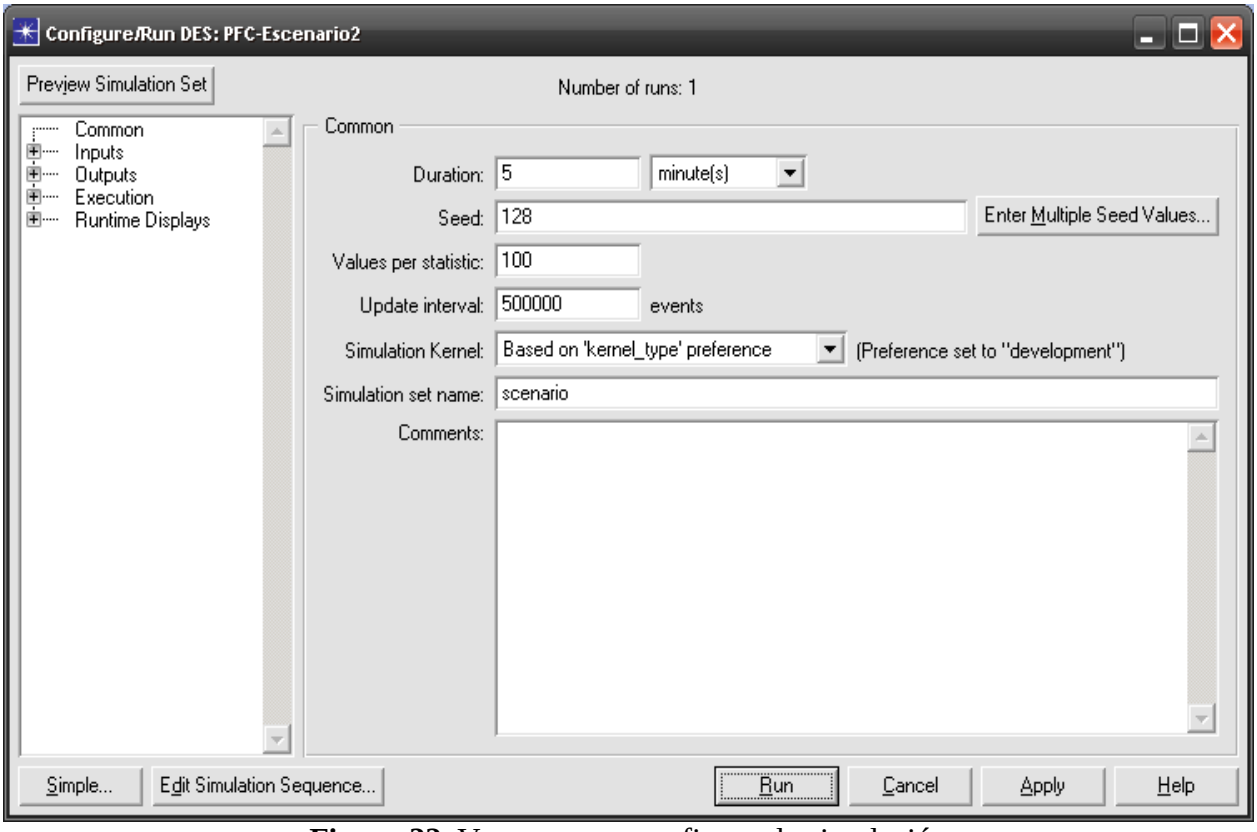


Figura 32. Ventana para configurar la simulación

En la ventana que aparece, en el menú Common se especifica las características generales de la simulación como por ejemplo la duración de la simulación (no es tiempo real), y el número de eventos.

Lo único que cambiaremos en nuestro caso será la duración. Se pone un tiempo de simulación de 5 minutos ya que con esta duración los resultados ya muestran estadísticas fiables.

El número de eventos es 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.

Es, 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 sean estadísticamente fiables.

4.1.2.13. Arrancar la simulación

Los demás valores se dejan por defecto y se hace clic al botón Run. Cuando acabe la simulación se cierra la ventana.

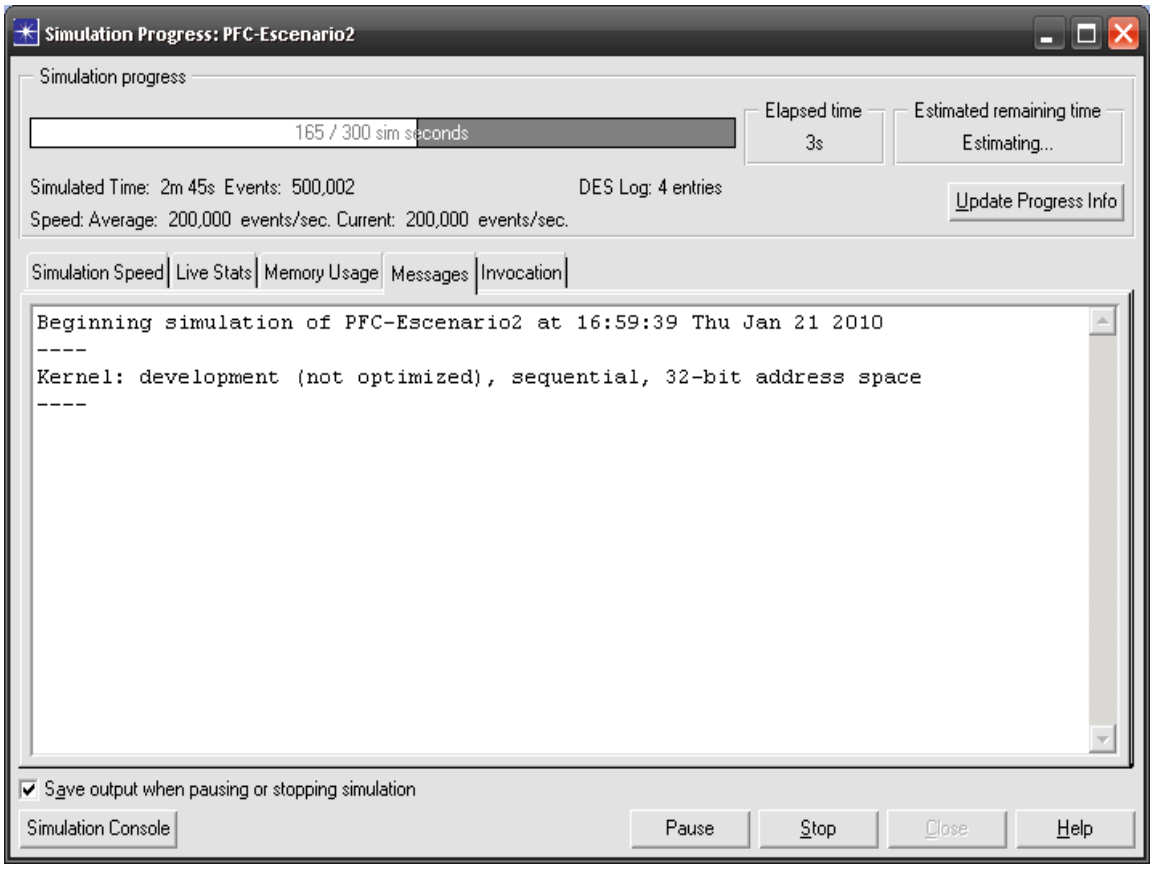


Figura 33. Proceso de simulación

4.1.2.14. Analizar los resultados

Para analizar los resultados se va a la opción DES → Results → View Results o mediante el siguiente acceso directo:

Aparece una ventana tal como se muestra a continuación, donde se puede visualizar los resultados obtenidos.

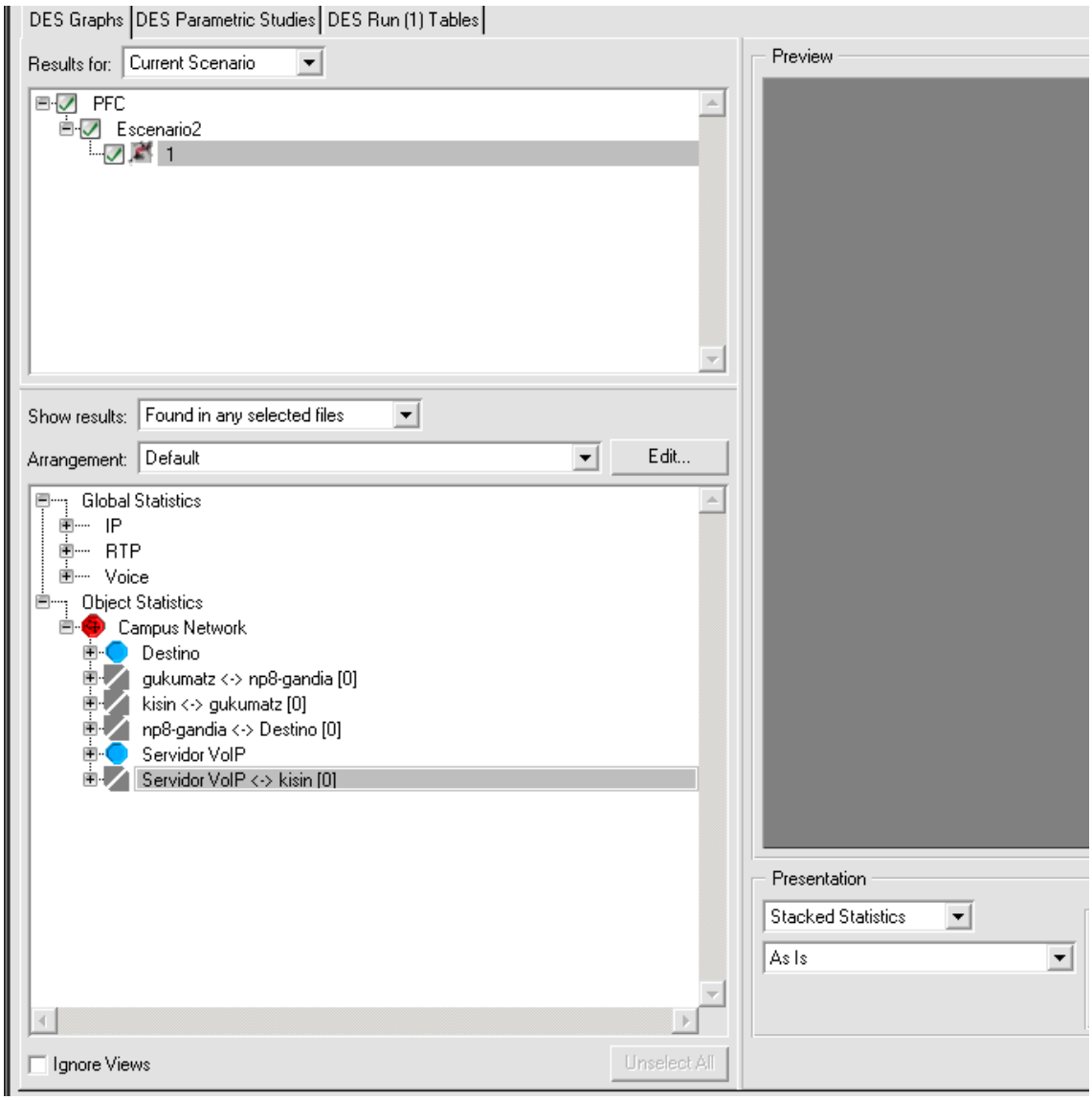


Figura 34. Ventana para visualizar resultados

Entre todos los resultados posibles a analizar en esta red de simulación, se pueden mostrar las gráficas del delay (retardo), jitter, tráfico recibido, tráfico

enviado y throughput (rendimiento) en el enlace que une el último router con el destino. En el capítulo se observarán algunos resultados obtenidos.

Para analizar los resultados, se debe ir a la parte de “Object Statistics” y dentro a su vez en el apartado “Campus Network” donde aparecen los resultados en la parte “RTP”.

El retardo es la diferencia que existe entre el momento en que una señal es trasmitida y el momento que una señal llega a su destino. Se mide en segundos.

El throughput (rendimiento) es el volumen de datos por unidad de tiempo que fluye por una red, se mide en bytes por segundo. El resultado se obtiene a nivel Ethernet, a un enlace entre dos nodos.

El Jitter es la variación en el retardo, en términos simples la diferencia entre el tiempo en que llega un paquete y el tiempo que se cree que llegará el paquete.

El background es la carga de fondo introducida en una simulación, tráfico que se introduce de forma indeseada en un enlace para producir o forzar en la simulación pérdida de paquetes.

Todas las gráficas que se deseen mostrar se encuentran en la parte izquierda, haciendo clic en el recuadro que se desea. Al marcar cada casilla aparece la gráfica, se marca el botón show, y aparece la siguiente imagen. En la figura 35 se muestra un ejemplo donde se aprecia el tráfico recibido el destino 1 por el servidor VoIP.

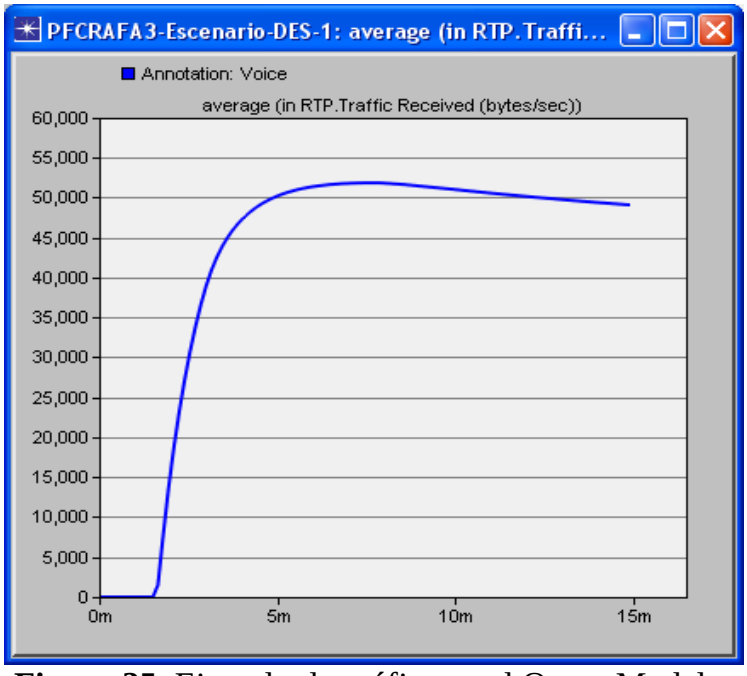


Figura 35. Ejemplo de gráfica en el Opnet Modeler

Los datos de las gráficas se pueden exportar para poder analizarlos desde una hoja de cálculo. Se hace clic sobre la imagen con el botón derecho del ratón y se elige la opción “Export Graph Data to spreadsheet”.

Mediante la hoja de cálculo se hará uso del potencial de Matlab para la representación de los resultados obtenidos con el simulador y, a su vez, poder compararlos con los resultados obtenidos en el otro simulador, el NS2.

4.2. El simulador NETWORK SIMULATOR 2

Para la realización de un escenario de simulación equivalente tanto en Opnet Modeler como en el Network Simulator 2 (NS2) se hace uso del módulo VoIP desarrollado por un grupo de investigadores italianos [15]. A continuación se muestra tanto la instalación del software NS2 como el módulo VoIP, así como su descripción y configuración.

4.2.1. Instalación

El simulador NS2 se programó originalmente para funcionar sobre sistemas Unix/Linux. Sin embargo, existe una adaptación del mismo para Windows basada en la emulación de una consola Unix.

A continuación se detalla el proceso de instalación en sistemas Linux y Windows, así como también la instalación del “parche” del módulo VoIP.

4.2.1.1. Entorno Windows

NS2 puede ejecutarse en sistemas Windows 9X/2000/XP gracias al software Cygwin, que permite emular una consola de comandos Unix/Linux en un sistema Windows.

A continuación se detallan los siguientes pasos para la instalación del entorno Cygwin:

1.- Instalar Active TCL versión 8.4.7, disponible en: http://downloads.activestate.com/ActiveTcl/Windows/8.4.7/ActiveTcl8.4.7.0-win32-ix86-108887.exe

2.- Reiniciar el ordenador.

3.- Crear un directorio en el que guardar los ejecutables de NS, por ejemplo en C:\ns

4.- Descomprimir el fichero ns-allinone-2.31-cygwin-binaries.zip en el directorio creado en el paso anterior. El fichero está disponible en

http://www.isi.edu/nsnam/dist/binary/ns-allinone-2.31-cygwin-binaries.zip

5.- Copiar el fichero cygwin1.dll en el directorio c:\ns\usr\bin.

El fichero está disponible en:

http://www.it.uc3m.es/rcalzada/ns2/cygwin1.dll.

6.- Copiar el fichero nam.exe en el directorio c:\ns\usr\bin

El fichero está disponible en http://www.it.uc3m.es/rcalzada/ns2/nam.exe (En este paso se sobrescribe el fichero original).

7.- Incluir en el PATH el directorio c:\ns\usr\bin

C:\> PATH=%PATH%;c:\ns\usr\bin

Se introduce el PATH de la siguiente manera. En sistemas Windows XP, este paso se realiza acudiendo a Panel de Control → Sistema → Opciones avanzadas → Variables de Entorno. En Variables de usuario seleccionamos Nueva e introducimos el path.

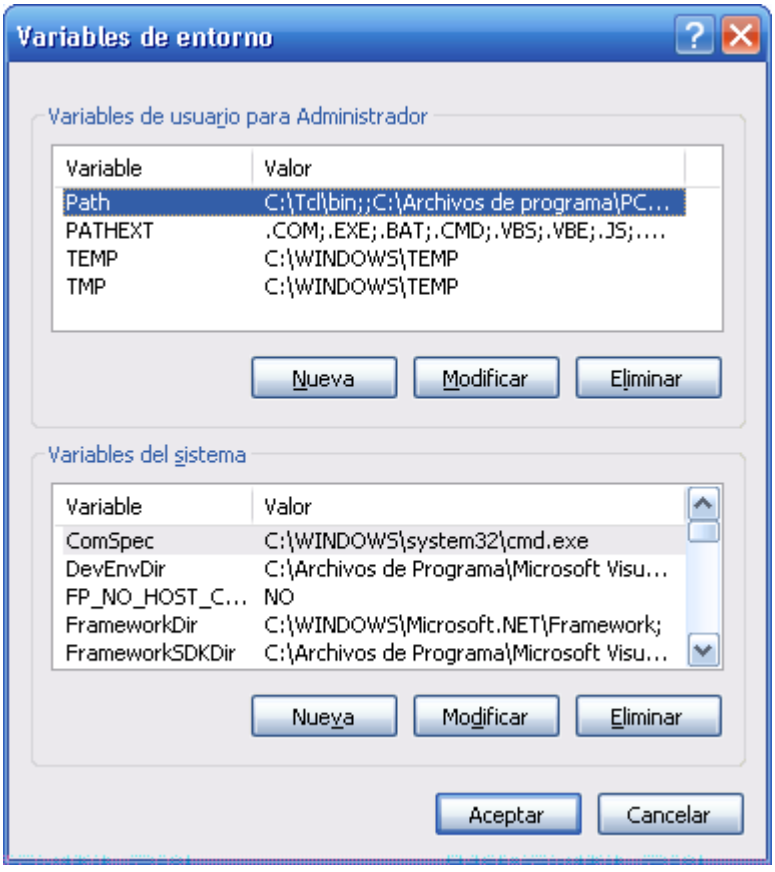


Figura 36. Introducción del path en Windows XP

Realizando esta instalación trabajaremos en una ventana de NS2 (símbolo de sistema), pero para nuestro propósito es suficiente.

4.2.1.2. Entorno Linux

Para instalar NS2 en sistemas Linux podemos compilar el código fuente en C++ en nuestro ordenador, o bien realizar una instalación más sencilla pero específica para cada distribución.

En nuestro caso, vamos a explicar la instalación en sistemas Ubuntu [1] por la importancia de esta distribución y por la necesidad de instalar el módulo VoIP a dicho software. Esta se realiza desde una ventana de terminal.

Primero se ha realizado la partición en el disco duro para instalar el sistema operativo y posteriormente se ha instalado NS2 disponible en la página oficial del simulador. La versión de NS2 que se ha instalado es la 2.31 y se puede instalar de dos maneras:

1. Descargando paquete a paquete guardándolos en disco y posteriormente compilarlos
   o
2. Instalar un solo archivo comprimido (ns-allinone-2.31) que contiene todos los paquetes, como ha sido nuestro caso.

Este archivo ns-allinone-2.31 (“*todo en uno”)* contiene los paquetes básicos siguientes:

- Tcl release 8.4.14 (componente necesario).
- Tk release 8.4.14 (componente necesario).
- Otcl release 1.13 (componente necesario).
- TclC release 1.19 (componente necesario).
- Ns release 2.31 (componente necesario).
- Nam release 1.13 (componente opcional).
- Xgraph versión 12 (componente opcional).
- CWeb versión 3.4g (componente opcional).
- SGB versión 1.0 (componente opcional).
- Gt-im gt-itm y sgb2ns 1.1 (componente opcional).
- Zlib versión 1.2.3 (componente opcional).

Los pasos realizados para la instalación del ns-allinone-2.31 se detallan a continuación:

1.- Descarga del paquete ns-allinone e instalación:

$ wget http://nchc.dl.sourceforge.net/sourceforge/nsnam/ns-allinone-2.31.tar.gz
$ tar -xzvf ns-allinone-2.31.tar.gz
$ cd ns-allinone-2.31

2.- Se descargan las librerías necesarias para el correcto funcionamiento ejecutando en la consola el comando *apt-get install* seguido de la librería que queramos descargar.

$ sudo apt-get install build-essential autoconf automake libxmu-dev
$ ./install

(En caso de error, reiniciar el equipo y probar el siguiente paso antes de ./install:)

$ sudo apt-get install -f build-essential libxt-dev libxt6 libsm-dev libsm6 libice-dev libice6 libxmu-dev

3.- Establecimiento de variables de entorno

$ gedit ~/.bashrc

Añadir las siguientes líneas al final, reemplazando "/your/path" por la ruta adecuada. Por ejemplo, “/home/rapelcor"

 # LD_LIBRARY_PATH
OTCL_LIB=/your/path/ns-allinone-2.31/otcl-1.13
NS2_LIB=/your/path/ns-allinone-2.31/lib
X11_LIB=/usr/X11R6/lib
USR_LOCAL_LIB=/usr/local/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB:$X11_LIB:$USR_LOCAL_LIB

 # TCL_LIBRARY
TCL_LIB=/your/path/ns-allinone-2.31/tcl8.4.14/library
USR_LIB=/usr/lib
export TCL_LIBRARY=$TCL_LIB:$USR_LIB

 # PATH
XGRAPH=/your/path/ns-allinone-2.31/bin:/your/path/ns-allinone-2.31/tcl8.4.14/unix:/your/path/ns-allinone-2.31/tk8.4.14/unix
NS=/your/path/ns-allinone-2.31/ns-2.31/
NAM=/your/path/ns-allinone-2.31/nam-1.13/

PATH=$PATH:$XGRAPH:$NS:$NAM

4.- Una vez hemos hecho todos estos pasos se debe reiniciar la consola para que el *bash.bashrc* se actualice.

$ source ~/.bashrc

5.- Para verificar que no ha habido fallos en la instalación se ejecuta el archivo *validate* que también se encuentra dentro del fichero.

$ cd ns-2.31
$ ./validate

Este paso no es necesario para el funcionamiento del programa, pero siempre es útil comprobar los posibles fallos en la instalación.

Para comprobar que la instalación se ha producido satisfactoriamente ejecutar el comando ns por consola y comprobar que aparece el símbolo %.

6.- Por último ejecutamos algún ejemplo entrando en el directorio:

$ home/ns-allinone-2.31/ns-2.31

Y ejecutamos el ejemplo con el comando ns:

$ ns ns-tutorial/examples/example2.tcl

Proceso de instalación del módulo VoIP

El proceso de instalación del citado módulo es muy sencillo. En la página Web http://cng1.iet.unipi.it/wiki/index.php/Ns2voip se dispone del patch de instalación (uno para la versión ns-2.31 y otro para ns-2.33, en función de la versión de simulador que se utilice).

En primer lugar, deberá descargarse este patch a un directorio local de la máquina en la que esté instalado el simulador y situarse a través de la consola en el directorio en el que esté instalado:

cd /directorio_instalacion_simulador/ns-allinone-2.31/ns-2.31

Tras ello, se aplicará el patch de instalación en este directorio:

gzip –dc /directorio_descarga_patch/ns2voip-2.31-070906.patch.gz | patch -Np1

A continuación, se compilará de nuevo el simulador mediante el siguiente comando:

/directorio_instalacion_simulador/ns-allinone-2.31/install

Y, finalmente, se podrá ejecutar el script OTcl de ejemplo localizado en el directorio voip junto con el código fuente del módulo completo.

/directorio_instalacion_simulador/ns-allinone-2.31/ns-2.31/voip/voip.tcl

4.2.2. Ejemplo básico de configuración de un escenario

NS2 no es un software gráfico, por lo que para poder realizar simulaciones se ha de programar scripts con código OTcl en los que se crean escenarios de red, se indican los protocolos a utilizar y el tipo de tráfico existente.

A continuación se muestran los pasos e instrucciones sencillas para una configuración de escenarios para un script escrito en lenguaje OTcl, utilizado como herramienta de diseño y configuración para la simulación en NS2 (las líneas iniciadas con '#' indican comentarios del autor que no serán ejecutados por el intérprete Tcl).

Se define una nueva simulación en un script en lenguaje OTcl siempre mediante una nueva instancia de la clase Simulator.

```
set ns [new Simulator]
```

Posteriormente se abre un archivo para escritura (w) out.nam, para enviar todos los datos obtenidos por el simulador hacia el objeto $nf.

```
set nf [open out.nam w]
$ns namtrace-all $nf
```

El trazado anterior es poco legible, pero debe ser creado para que el programa NAM lo interprete.

Sin embargo, hay otro tipo de trazado que es más legible, se crea el archivo out.tr para poder ser interpretado por el software Xgraph o tracegraph. Para ello es necesario añadir las siguientes líneas, muy parecidas a las de arriba

```
set f [open out.tr w]
$ns trace-all $f
```

Para la creación de los nodos en el escenario se introduce el comando "[$ns node]", como se muestra en la siguiente instrucción:

set <nombre nodo>[$ns node]

Un ejemplo de la instrucción anterior con la creación de un nodo con nombre n0 es la siguiente:

ns n0[$ns node]

En la creación de enlaces entre nodos se define con la siguiente instrucción:

$ns (simplex/duplex)-link $nodo1 $nodo2 [BW] [retardo] [tipo_de_cola]

Se observa en la instrucción anterior la opción entre la configuración simplex donde sólo se permite la transmisión en un solo sentido y la configuración dúplex donde se permite simultáneamente la transmisión tanto en un sentido como en el otro.

En el apartado [BW] se indica el ancho de banda que se configura en el enlace. En el apartado [retardo] se indica el retardo que introduce el enlace en el escenario.

En el apartado [tipo_de_cola] se indica el tipo de cola, **por defecto** se utiliza el tipo de cola DropTail pero también se puede indicar Queue y PriQueue. El modelo de colas DropTail consiste en una cola simple FIFO (el primer paquete en entrar también es el primer paquete en salir) en la que se descartan los paquetes que sobrepasen la capacidad del tamaño del buffer de la cola. La clase PriQueue significa que se está dando prioridad a los paquetes que se han enviado utilizando protocolos de enrutamiento.

En la siguiente instrucción se muestra un ejemplo de un enlace dúplex desde un nodo llamado n0 a otro nodo llamado n1 con un ancho de banda de 1 Megabit, con un retardo introducido de 10 milisegundos y utilizando un tipo de cola DropTail.

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

En una topología con transmisión multicast, se requiere mejoras en los nodos y enlaces de dicha topología. Por lo tanto, se debe especificar unos requisitos para que el simulador reconozca la transmisión multicast antes de crear el escenario. Se define de la siguiente manera:

set ns [new Simulator -multicast on]

o

set ns [new Simulator]
$ns multicast

El simulador NS2 admite tres estrategias multicast para el cálculo de la ruta: centralizada, modo denso (DM) o el modo de árbol compartido (ST).

Los siguientes son ejemplos válidos de las invocaciones de enrutamiento multicast en NS2:

set cmc [$ns mrtproto CtrMcast];
#especifica multicast centralizado para todos los nodos
#cmc es el protocolo de multicast para manipular objetos

$ns mrtproto DM;
#especifica el modo de multicast de todos los nodos

$ns mrtproto ST;
#especifica el modo de árbol compartido para ejecutar en todos los nodos

En NS2 se denomina agentes a los objetos encargados de generar tráfico de un nodo hacia otro. Se crea el agente y se asocia a un nodo ($ns attach-agent). A continuación, se conectan los agentes para indicar el flujo o camino que sigue el tráfico de un nodo a otro ($ns connect). Entre los agentes soportados en el NS2 más interesantes destacan TCP, UDP, RTP, RTCP.

En el siguiente ejemplo se ilustra la creación y modificación de un agente en OTcl, para su fácil explicación:

set <nombre del agente 1> [new Agent/ <tipo de agente>]
set <nombre del agente 2>[new Agent/ <tipo de agente>]
$ns attach-agent $<nodo emisor> $<nombre del agente 1>
$ns attach-agent $<nodo receptor>$<nombre del agente 2>
$ns connect $<nombre agente 1> $<nombre agente 2>

A su vez, a cada agente se le debe asociar un tipo de tráfico, entre el tráfico soportado son el FTP y telnet, y el tráfico que se puede crear es de tipo Exponential, CBR, Pareto y Trace.

El tráfico Exponential genera el tráfico de acuerdo a un aumento exponencial de encendido/apagado de distribución. Los paquetes se envían a tasa fija durante los períodos de encendido, y no se envían los paquetes durante los períodos de apagado. Dentro y fuera de los períodos se han tomado de un aumento exponencial de distribución. Los paquetes son de tamaño constante. Las variables que se pueden modificar en el tráfico Exponential son las siguientes:

packetSize_ → tamaño constante del paquete generado
burst_time_ → promedio de encendido
idle_time_ → promedio de apagado
rate_ → tasa de envío

A continuación se muestra un ejemplo de configuración del tráfico Exponential:

```
set <nombre tráfico> [new Application/Traffic/Exponential]
$e set packetSize_ 210
$e set burst_time_ 500ms
$e set idle_time_ 500ms
$e set rate_ 100k
```

El tráfico Pareto utiliza el mismo método que el tráfico Exponential pero utilizando la distribución Pareto. Las variables que se pueden modificar en el tráfico Pareto son las siguientes:

packetSize_ → tamaño constante del paquete generado
burst_time_ → promedio de encendido
idle_time_ → promedio de apagado
rate_ → tasa de envío
shape_ → parámetro utilizado para la distribución Pareto

A continuación se muestra un ejemplo de configuración del tráfico Pareto:

```
set <nombre trafico> [new Application/Traffic/Pareto]
$p set packetSize_ 210
$p set burst_time_ 500ms
$p set idle_time_ 500ms
$p set rate_ 200k
$p set shape_ 1.5
```

El tráfico CBR genera un tráfico de paquetes de tamaño constante. Las variables que se pueden modificar en el tráfico CBR son las siguientes:

rate_ → tasa de envío
interval_ → intervalo entre paquetes (opcional)
packetSize_→ tamaño constante del paquete generado
random_ → introducción de ruido en la transmisión (apagado por defecto)
maxpkts_ → el número máximo de paquetes a enviar (por defecto es 2^8)

A continuación se muestra un ejemplo de configuración del tráfico CBR:

```
set <nombre tráfico> [new Application/Traffic/CBR]
$e set packetSize_ 48
$e set rate_ 64Kb
$e set random_ 1
```

El tráfico Trace genera un tráfico de acuerdo con un archivo de traza generado. A continuación se muestra un ejemplo de configuración del tráfico Trace:

```
set tfile [new Tracefile]
$tfile filename example-trace
set t1 [new Application/Traffic/Trace]
$t1 attach-tracefile $tfile
set t2 [new Application/Traffic/Trace]
$t2 attach-tracefile $tfile
```

Para la comprensión de la creación de agentes y creación de tráfico se va a ilustrar a continuación un ejemplo con los pasos básicos para configurar una fuente de tráfico exponencial en un agente UDP, el tráfico fluye del nodo s1 al nodo k1.

```
set agente1 [new Agent/UDP]
set agente2 [new Agent/UDP]
$ns_ attach-agent $node_(s1) $agente1
$ns_ attach-agent $node_(k1) $agente2
$ns_ connect $agente1 $agente2

set e [new Application/Traffic/Exponential]
$e attach-agent $agente1
$e set packetSize_ 210
```

```
$e set burst_time_ 500ms
$e set idle_time_ 500ms
$e set rate_ 100k
```

Una vez se generan los agentes y el tráfico se debe indicar al agente creado el tiempo en el cual debe generar tráfico mediante el comando start.

```
$ns_ at <segundos> "$<nombre tráfico> start"
```

En el ejemplo anterior es:

```
$ns_ at 0.0 "$e start"
# el tráfico empieza a generarse a los 0 segundos
```

Se utiliza el comando stop para dejar de generar tráfico. Se muestra un ejemplo a continuación:

```
$ns_ at 4.5 "$e stop"
# el tráfico deja de generarse a los 4.5 segundos
```

En el programa se introduce un procedimiento "finish", el cual cerrará el fichero de valores de trazado (out.nam) y pondrá en marcha a NAM, este deberá tener la siguiente estructura:

```
proc finish{} {
global ns nf
$ns flush-trace
close $nf
exec nam out.nam &
exit=
}
```

La siguiente línea que escribiremos será el tiempo que vamos a simular la red

```
$ns at <tiempo><elemento>
```

Donde <tiempo> será el valor en segundos y <elemento> será en que procedimiento se cierra la simulación, por ejemplo:

```
$ns at 5.0 "finish"
```

Así se le dice a NS que ejecute la simulación durante 5.0 segundos y después ejecute el procedimiento "finish"

En la última línea del programa será para que arranque la simulación de la siguiente forma:

```
$ns run
```

4.2.3. Principales comandos Tcl para el módulo RTP/RTCP nativo

Los procedimientos OTcl (*Tool Command Language* orientado a objetos) relativos a la configuración de los protocolos RTP/RTCP se definen en el fichero *session-rtp.tcl*. En este fichero se especifican métodos para la inicialización de sesión, gestionar su unión y abandono a determinados grupos multicast, asignarles un identificador de flujo específico con tal de diferenciarlos en los ficheros traza de salida generados, indicar la tasa de envío de la fuente RTP, asignar tamaños de paquetes RTP específicos, calcular el periodo de envío de informes RTCP, detener la transmisión RTP, liberar la sesión, etc.

Tal y como puede observarse en la Figura 36, cuando se instancia una nueva sesión, mediante el comando **[new Session/RTP]**, se crean 4 nuevos objetos asociados a ella: un agente RTP (*RTPAgent*), un agente RTCP (*RTCPAgent*), un agente fuente (*RTPSource*) y un temporizador (*RTCPTimer)*. Cuando esta sesión se une a un grupo multicast a través de la invocación del método OTcl **join-group**, los agentes RTP y RTCP se unen a los grupos multicast separados. Esta es la solución adecuada cuando se utiliza el mismo grupo multicast (misma dirección IP) pero con diferente número de puerto (en general se reservan puertos de transporte consecutivos para una misma sesión). El procedimiento **start** inicializa el agente RTCP, mientras que el procedimiento **transmit** lanza el agente RTP. Con éste último procedimiento se arrancará un nuevo *timer* vinculado al Agente RTP, definido a través de la clase C++ *RTPTimer*, que se encargará de gestionar el envío de paquetes RTP a una tasa determinada. El periodo de envío de los paquetes RTP será en función del tamaño especificado para los paquetes RTP y de la tasa binaria especificada como parámetro en el método **transmit**, tal y como se puede observar en la siguiente relación:

$$tasa\_envío(segundos) = \frac{packetSize \times 8(bits)}{tasa\_binaria\left(\frac{bits}{segundo}\right)}$$

En la siguiente figura se pueden apreciar los pasos descritos anteriormente y la sintaxis de los comandos OTcl necesaria para su correcta ejecución.

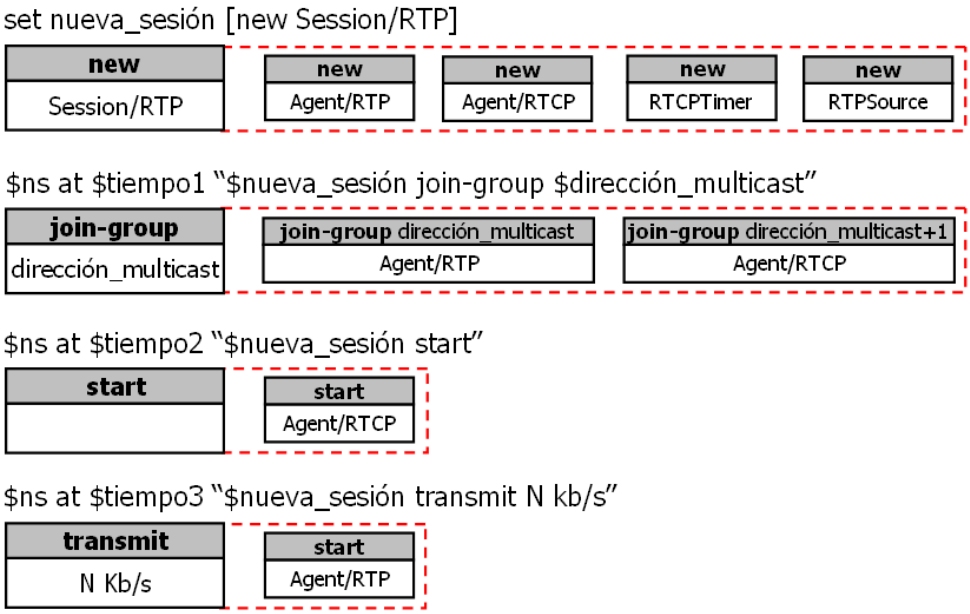


Figura 37. Comandos Tcl básicos en la configuración de una Sesión RTP en NS-2.

A continuación se muestra parte del código de un sencillo script escrito en lenguaje Tcl, utilizado como herramienta de diseño y configuración de escenarios para la simulación en NS2, en el que se definen sesiones RTP (las líneas iniciadas con '#' indican comentarios del autor que no serán ejecutados por el intérprete Tcl):

```
#Se define una nueva sesión que será la fuente RTP
set s0 [new Session/RTP]

# Se define otras sesiones para la recepción de los flujos
RTP de la sesión transmisora
set s1 [new Session/RTP]
set s2 [new Session/RTP]

...

#Se configura el protocolo multicast (mproto) a utilizar
#Existen distintas alternativas: CtrMcast, DM, ST, BST
set mproto DM

#Se agregan agentes multicast a todos los nodos
set mrthandle [$ns mrtproto $mproto {}]

#Se crea un nuevo grupo multicast
```

```
set group [Node allocaddr]

#Se unen las sesiones al grupo multicast mediante el
método 'join-group'
#Se inicializan los Agentes RTCP de las sesiones mediante
el método 'start'

$ns at 0.1 "$s0 join-group $group"
$ns at 0.1 "$s0 start"

$ns at 0.1 "$s1 join-group $group"
$ns at 0.1 "$s1 start"

$ns at 0.1 "$s2 join-group $group"
$ns at 0.1 "$s2 start"

#La fuente RTP inicia la transmisión de paquetes RTP a
una tasa de 400 Kbps

$ns at 0.5 "$s0 transmit 400kb/s"

...

#La session RTP abandona el grupo multicast
$ns at 20.0 "$s1 leave-group"

...

#La fuente transmisora detiene el flujo RTP
$ns at 30.0 "$s0 stop"
```

4.2.4. Descripción del Módulo VOIP

En esta sección se describe el modelo de simulación de una aplicación VoIP y su implementación en el simulador NS2. En este módulo se ha diseñado la parte emisora y receptora en bloques diferenciados. De este modo, el bloque emisor incluye:

- Un bloque de codificación de tramas de voz.
- Un multiplexor que puede agregar varias tramas de voz en un contenedor de tramas (*Payload*).

El bloque receptor incluye:

- Un algoritmo genérico de almacenamiento de tramas en el buffer receptor y su posterior reproducción.
- Un demultiplexor, en el caso que las tramas vengan agregadas.

La implementación en NS2 de los bloques descritos incluye varios sub-módulos inter-operables, tal y como puede apreciarse en la siguiente figura, que se describirán a continuación.

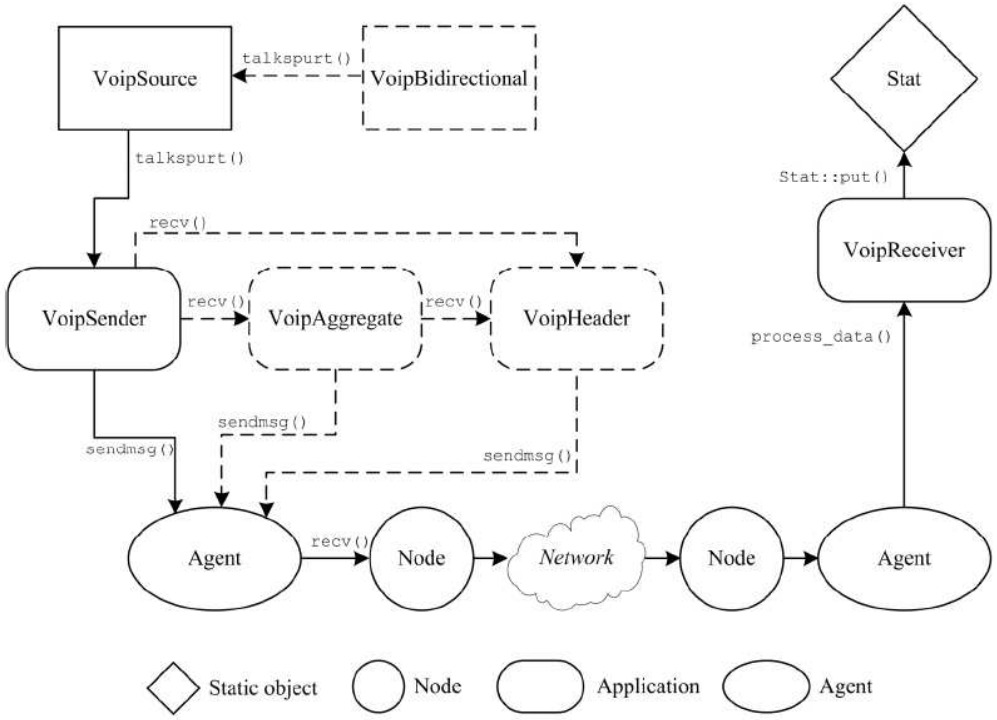


Figura 38. Módulos para la simulación de aplicaciones VoIP en NS-2

En primer lugar, se implementó una estructura de dato específica, denominada *VoipFrame*, con el objetivo de definir las tramas de voz a generar y transmitir, con los siguientes campos:

- *talkspurt*: identificador del número de ráfaga.
- *nframes*: número de tramas en la ráfaga.
- *frame*: identificador del número de trama en cada ráfaga.
- *timestamp*: marca de tiempo en la que se generó la trama.
- *size_*: tamaño de la trama en bytes.

Las tramas VoIP se empaquetan en un contenedor de tramas denominado *VoipPayload*, que simplemente es una lista de objetos *VoipFrame*.

La actividad del habla del usuario se modela por medio de periodos alternados de ráfagas y silencios (*talkspurt/silence*), generados por el módulo *VoipSource*. En este módulo puede configurarse la duración de estos periodos

por medio de una distribución exponencial o Weibull. Al inicio de cada ráfaga, el modulo *VoipSource* llama al método *talkspurt()* del objeto *VoipSender*, especificando la duración de dicho periodo. Cuando el objeto *VoipSource* lanza una nueva ráfaga de tramas de voz, el objeto *VoipSender* empieza la generación de una serie de tramas *VoipFrame*, cuyo tamaño y tasa de generación depende del codificador de voz especificado en el proceso de simulación. El objeto *VoipSender* soporta la codificación G.711.

El contenido de las tramas VoIP generadas será distinto en función de si los bloques de multiplexación de tramas y/o inclusión de cabecera están habilitadas o no. Si se habilita la multiplexación, se crea un objeto *VoipAggregate y* se enlaza al objeto *VoipSender*, el cual llamará a la función de recepción *recv()* por cada trama generada para pasársela al objeto multiplexor. Este objeto *VoipAggregate* espera hasta la recepción de un determinado número de tramas, antes de empaquetarlas en un objeto *VoipPayload* y enviarlas hacia el objeto *VoipHeader* (si está habilitado), o bien, hacia los agentes de transporte para que lo envien a través de la red simulada.

Si no se habilita la multiplexación, el objeto *VoipSender* genera directamente un objeto *VoipPayload* por cada trama VoIP generada, que será enviado al objeto *VoipHeader*, si se habilita este último. Si el objeto *VoipHeader* no se habilita, el contenedor *VoipPayload* se envía de forma directa al agente de transporte. El objeto *VoipHeader* recibe un objeto *VoipPayload* por parte del objeto *VoipSender* o del objeto *VoipAggregate*, en función de si la multiplexación de tramas se habilita o no. El objeto *VoipHeader* se encarga de añadir las cabeceras RTP/UDP/IP al objeto *VoipPayload* y soporta compresión de cabecera. Tras este paso, el objeto *VoipPayload* es enviado al agente de transporte, el cual se encargará de su transmisión por la red simulada, experimentado posiblemente pérdidas, retrasos o re-ordenamientos en la recepción de los paquetes de voz. Esto explica por qué los citados objetos se implementas en NS2 como aplicaciones.

Por otra parte, los paquetes de voz que no se pierden en la trayectoria a través del escenario simulado, llegan al agente de transporte receptor, el cual se encargará de transferir los datos encapsulados a la aplicación *VoipReceiver*. Ésta última se encarga de recuperar las tramas *VoipFrame* encapsuladas en un contenedor *VoipPayload* e implementa un buffer de reproducción.

En este módulo VoIP se implementa tres políticas de *buffering* distintas: sin *buffering (*alternativa únicamente válida para pruebas*)*, *buffering* estático y *buffering* optimizado. Estas dos últimas alternativas tratan de reproducir las tramas VoIP a una tasa constante.

4.2.5. Interfaz de Configuración del Módulo VoIP

El código fuente del módulo VoIP está bastante documentado en la referencia citada y, además, puede convertirse en manual electrónico a través de la herramienta Doxigen. Adicionalmente, se provee un sencillo script OTcl de configuración de aplicaciones VoIP que puede servir como ejemplo para los usuarios interesados en la simulación de este tipo de aplicaciones.

VoipSender

Las siguientes sentencias OTcl crean un objeto VoipSource, VoipSender, un Agente UDP y los conecta:

```
set src [new VoipSource]

set snd [new Application/VoipSender]

set agt [new Agent/UDP]

$src encoder $snd

$snd attach-agent $agt
```

Los comandos necesarios para conectar los objetos siguen la sintaxis estándar utilizada en el simulador NS2 y son explicados en su manual. Los siguientes comandos se encargan de la configuración de los objetos del bloque emisor:

```
$voip_source start/stop
```

Lanza y detiene la generación de ráfagas de muestras de voz.

```
$voip_source model

    exponential $on $off |

    one-to-one |

    one-to-many |

    many-to-one |
```

many-to-many

Habilita un modelo específico para la generación de periodos de ráfagas y silencios. El modelo exponencial requiere como parámetro la media de la duración de los periodos de ráfaga y silencios, mientras que los demás opciones se basan en distribuciones Weibull.

VoipSource

$voip_snd codec G.711

Selecciona el codificador a utilizar para las tramas VoIP.

VoipAggregate

$voip_aggr nframes $n

Configura el objeto agregador con un número máximo de n tramas.

VoipReceiver

$voip_rcv id $ID

Asocia un identificador de flujo al receptor VoIP con tal de diferenciarlo entre los demás en el fichero de estadísticas generados

$voip_rcv cell-id $cell

Se utiliza para asociar el decodificador VoIP a una celda, cuando se pretenda agrupar las estadísticas por celdas.

$voip_rcv emodel $ie $bpl $a $ro $th

Configura los parámetros para el cómputo del E-model. En el artículo se detalla este método con mayor profundidad.

$voip_rcv emodel G.711

Selecciona el decodificador a utilizar, que influirá en el cómputo de los parámetros E-model.

$voip_rcv buffer-size $n

En el caso de *buffering* estático, este comando especifica el valor máximo de tramas VoIP que puede ser almacenado en el buffer.

$voip_rcv initial-delay $delay

Especifica el retardo introducido en el buffer a la primera trama VoIP de una ráfaga.

$voip_rcv playout-rate $time

Especifica el tiempo, en milisegundos, entre la generación de dos tramas consecutivas por parte del transmisor. Este valor depende del codec utilizado, y servirá al buffer receptor para reproducir las muestras a la misma tasa a la que se generaron.

VoipHeader

$voip_compression $size | nocompression

Especifica el tamaño de la cabecera RTP/UDP/IP. Si no se especifica, el valor por defecto es *nocompression*, lo que resulta en un tamaño de cabecera de 40 bytes (20 RTP + 8 UDP + 12 IP). En caso contrario, el tamaño de la cabecera comprimida será de *size* bytes.

4.2.6. Script de simulación

El escenario de simulación debe ser semejante o con las mismas prestaciones que el utilizado en el software Opnet Modeler.

El escenario de simulación consta de una fuente VoIP (VoIP Source), de tres receptores VoIP (VoIP Receiver) y de routers intermedios.

La conexión entre la fuente VoIP y el primer router utiliza un enlace de 10Mbits al igual que el último router con los destinos VoIP. Los demás enlaces son de 2Mbits. Se utiliza en todos los enlaces un delay de 0.02 segundos.

Para la configuración de tráfico de fondo (background) se hace uso de los routers 9 y 10, con tráfico CBR y ajustando en cada tiempo la carga que se considera oportuna, al igual que en el Opnet Modeler. A continuación, se muestra los tiempos utilizados en el background junto a su carga.

```
$ns at 0.0 "$cbr0 start"

#La fuente CBR empieza a los 0 segundos

$ns at 60.0 "$source($i) start"

#La fuente VoIP empieza transmitir a los 60 segundos

$ns at 120.0 "$cbr0 set rate_ 1.9mb"

#La fuente CBR transmite a 1.9Mbits a los 120 segundos

$ns at 180.0 "$cbr0 set rate_ 1.98mb"

#La fuente CBR transmite a 1.98Mbits a los 180 segundos

$ns at 240.0 "$cbr0 stop"

#La fuente CBR deja de transmitir a los 240 segundos

$ns at 300.0 "$source($i) stop"

#La fuente VoIP deja de transmitir a los 300 segundos
```

El tráfico VoIP es utilizado entre el VoIP Source y el VoIP Receiver 1, se observaran paquetes de color rojo.

La codificación utilizada en el tráfico VoIP es G.711, se observa en el script con la siguiente sentencia:

```
set opt(codec)   "G.711"
```

Para la configuración de la distribución exponencial y la modulación de la voz ON y OFF, se observa en el script con la siguiente sentencia:

```
$source($i) model exponential 0.4011 0.5775
```

A continuación se muestra el script utilizado en este proyecto. Se utiliza otro color para destacar las partes modificadas para nuestro escenario.

```
#########################################################
#                    CONFIGURATION OF PARAMETERS
#########################################################

#########################################################
# Simulation environment
#########################################################

set opt(run)        0         ;# replic ID
set opt(duration)   300.0     ;# run duration, in seconds
set opt(warm)       36.0      ;# run duration, in seconds
set opt(out)        "out"     ;# statistics output file
set opt(debug)      ""        ;# debug configuration file, "" = no
debug
set opt(startdebug) 300.0     ;# start time of debug output, in
seconds

set opt(aggregate) 1
set opt(tagrand)   "constant"
set opt(tagmean)    0.10
set opt(tagvar)     0.01
set opt(tagper)     0.00

set opt(codec)   "G.711" ;# G.711, G.723.1, G.729A, GSM.EFR,
GSM.AMR
set opt(initialDelay) 0.060

#########################################################
#                    DEFINITION OF PROCEDURES
#########################################################

#########################################################
# parse command-line options and store values into the $opt(.)
#########################################################

proc getopt {argc argv} {
        global opt

        for {set i 0} {$i < $argc} {incr i} {
                set arg [lindex $argv $i]
                if {[string range $arg 0 0] != "-"} continue
```

```
                set name [string range $arg 1 end]
                set opt($name) [lindex $argv [expr $i+1]]
        }
}

#########################################################
# print out options
#########################################################

proc printopt { } {
        global opt

        foreach x [lsort [array names opt]] {
                puts "$x = $opt($x)"
        }
}

#########################################################
# die function
#########################################################

proc die { x } {
        puts $x
        exit 1
}

#########################################################
# alive function
#########################################################

proc alive { } {
        global ns opt

        if { [$ns now] != 0 } {
                puts -nonewline \
                [format "elapsed %.0f s (remaining %.0f s) completed
%.f%%" \
                 [$ns now] \
                 [expr $opt(duration) - [$ns now]] \
                 [expr 100 * [$ns now] / $opt(duration)]]
                if { [$ns now] >= $opt(warm) } {
                        puts " stat collection ON"
                } else {
                        puts ""
```

```
            }
        }
        $ns at [expr [$ns now] + $opt(duration) / 10.0] "alive"
}

#########################################################
# Proceso de finalización
#########################################################

proc finish {} {
        global ns simtime

        # print statistics to output file
        $ns stat print

        # print out the simulation time
        set simtime [expr [clock seconds] - $simtime]
        puts "run duration: $simtime s"

        puts "running nam..."
        exec /usr/local/ns-allinone-2.31/nam-1.13/nam voip.nam &

        exit 0
}

#########################################################
# Inicio de simulación
#########################################################

proc init {} {
        global opt defaultRNG ns simtime

        # create the simulator instance
        set ns [new Simulator]  ;# create a new simulator instance
        $defaultRNG seed 1

        # initialize statistics collection
        $ns run-identifier $opt(run)
        $ns stat file "$opt(out)"
        $ns at $opt(warm) "$ns stat on"
        $ns at $opt(duration) "finish"

        $ns stat add voip_frame_delay avg discrete
        $ns stat add voip_frame_sent avg counter
        $ns stat add voip_frame_rcvd avg counter
```

```
        $ns stat add voip_mos_talkspurt avg discrete
        $ns stat add voip_none_frame_sent avg counter
        $ns stat add voip_none_frame_rcvd avg counter
        $ns stat add voip_none_mos_talkspurt avg discrete
        $ns stat add voip_end_mos avg discrete
        $ns stat add voip_end_per avg discrete
        $ns stat add voip_end_cell_outage avg discrete
        $ns stat add voip_buffer_overflow_drop avg counter
        $ns stat add voip_buffer_out_of_time_drop avg counter
        $ns stat add voip_%_of_bad_talkspurts avg discrete

        #$ns stat add Weibull_ON  avg discrete
        #$ns stat add Weibull_OFF avg discrete

        # open trace files
        set opt(trace) [open "/dev/null" w]

        set simtime [clock seconds]

        $ns trace-all $opt(trace)

                set f [open voip.tr w]
        $ns trace-all $f
        $ns namtrace-all [open voip.nam w]
}

#########################################################
#Configuración del Escenario
#########################################################

proc scenario {} {
      global ns opt
#Se crean 4 nodos, el nodo 0 es la fuente VoIP y los nodos 1,2 y
#3 son los receptores
     set n0 [$ns node]
     $n0 shape "box"
#Indica la forma cuadrada en la representación del nodo
        $n0 color red
#Indica el color del nodo en su representación
          set n1 [$ns node]
        $n1 shape "box"
        $n1 color red
        set n2 [$ns node]
        $n2 shape "box"
        $n2 color red
```

```
        set n3 [$ns node]
        $n3 shape "box"
        $n3 color red
#Se muestran las etiquetas referentes a cada nodo en la
#representación
        $n0 label "VoIP Source"
        $n1 label "VoIP Receiver 1"
        $n2 label "VoIP Receiver 2"
        $n3 label "VoIP Receiver 2"

#Se crean 5 routers intermedios entre la fuente VoIP y los
#destinos VoIP
        set router1 [$ns node]
        set router2 [$ns node]
        set router3 [$ns node]
        set router4 [$ns node]
        set router5 [$ns node]

        #Se crea el nodo CBR
        set n_cbr_src [$ns node]
        $n_cbr_src color blue
        $n_cbr_src shape "hexagon"
        $n_cbr_src label "CBR Source"
        set n_cbr_sink [$ns node]
        $n_cbr_sink color blue
        $n_cbr_sink shape "hexagon"
        $n_cbr_sink label "CBR Sink"

#########################################################
#Definición de los enlaces
#########################################################

#DropTail significa que las colas asociadas con FIFO

        $ns duplex-link $n0 $router1 2Mb 20ms DropTail
        $ns duplex-link-op $n0 $router1 orient right
        #Esto es opcional, es para dar la posición de los nodos en
        el escenario

        $ns duplex-link $n_cbr_src $router1 2Mb 10ms DropTail
        $ns duplex-link-op $n_cbr_src $router1 orient up

        $ns duplex-link $router1 $router4 2Mb 20ms DropTail
        $ns duplex-link-op $router1 $router4 orient right-up
```

```
$ns duplex-link $router1 $router5 2Mb 20ms DropTail
$ns duplex-link-op $router1 $router5 orient right-down

$ns duplex-link $router1 $router2 2Mb 20ms DropTail
$ns duplex-link-op $router1 $router2 orient right

$ns duplex-link $router4 $router3 2Mb 20ms DropTail
$ns duplex-link-op $router4 $router3 orient right-down

$ns duplex-link $router5 $router3 2Mb 20ms DropTail
$ns duplex-link-op $router5 $router3 orient right-up

$ns duplex-link $router2 $router3 2Mb 20ms DropTail
$ns duplex-link-op $router2 $router3 orient right

$ns duplex-link $router3 $n1 2Mb 20ms DropTail
$ns duplex-link-op $router3 $n1 orient right-up

$ns duplex-link $router3 $n2 2Mb 20ms DropTail
$ns duplex-link-op $router3 $n2 orient right

$ns duplex-link $router3 $n3 2Mb 20ms DropTail
$ns duplex-link-op $router3 $n3 orient right-down

$ns duplex-link $router3 $n_cbr_sink 2Mb 10ms DropTail
$ns duplex-link-op $router3 $n_cbr_sink orient down

#Se asocia los nombres de colores con números enteros
$ns color 1 red
$ns color 2 blue

#########################################################
#Definición de los Agentes y tráfico asociado
#########################################################

set udp0 [new Agent/UDP]
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 1000
$cbr0 set rate_ 1mb
$cbr0 set random_ false
#$cbr0 set interval_ 0.005

$cbr0 attach-agent $udp0
$ns attach-agent $n_cbr_src $udp0
$udp0 set fid_ 2
```

```
        set null [new Agent/Null]
        $ns attach-agent $n_cbr_sink $null
        $ns connect $udp0 $null

      for { set i 0 } { $i < 1 } { incr i } {
        set source($i) [new VoipSource]
        $source($i) model exponential 0.4011 0.5775
        #Parámetro ON/OFF de la voz
        #$source($i) model one-to-many

#########################################################
#Tiempos de simulación
#########################################################

                $ns at 0.0 "$cbr0 start"

                $ns at 60.0 "$source($i) start"

                $ns at 120.0 "$cbr0 set rate_ 1.9mb"

                $ns at 180.0 "$cbr0 set rate_ 1.98mb"

                 $ns at 240.0 "$cbr0 stop"

                 $ns at 300.0 "$source($i) stop"

        set encoder($i) [new Application/VoipEncoder]
        $encoder($i) codec $opt(codec)

        $source($i) encoder $encoder($i)

        set decoder($i) [new Application/VoipDecoder]
        set decoder($i) [new Application/VoipDecoderStatic]
        set decoder($i) [new Application/VoipDecoderOptimal]
        $decoder($i) id $i
        $decoder($i) cell-id 0
        $decoder($i) emodel $opt(codec)

        #Only for static decoder
        #$decoder($i) buffer-size 20
# in number of VoIP frames
        #$decoder($i) initial-delay $opt(initialDelay) ;# in ms

        set agtsrc($i) [new Agent/UDP]
```

```
#set the sender trace file name (sd)
          #$agtsrc($i) set_filename fuente1
         set agtdst($i) [new Agent/UDP]
          #$agtdst($i) set_filename_rec destino1
         $agtsrc($i) set fid_ 1
         $ns attach-agent $n0 $agtsrc($i)
         $ns attach-agent $n1 $agtdst($i)


         $ns connect $agtsrc($i) $agtdst($i)

$encoder($i) attach-agent $agtsrc($i)
$decoder($i) attach-agent $agtdst($i)

         set aggregate($i) [new Application/VoipAggregate]
         #$aggregate($i) size 200
         $aggregate($i) nframes $opt(aggregate)
         $aggregate($i) attach-agent $agtsrc($i)
         #$encoder($i) aggregate $aggregate($i)

         #set header($i) [new Application/VoipHeader]
         #$header($i) nocompression
         #$header($i) attach-agent $agtsrc($i)
         #$encoder($i) header $header($i)
         #$aggregate($i) header $header($i)

         $ns at $opt(startdebug) "$source($i) debug"
         $ns at $opt(startdebug) "$encoder($i) debug"
         $ns at $opt(startdebug) "$decoder($i) debug"
         $ns at $opt(startdebug) "$aggregate($i) debug"

         # end-to-end modules statistics collection
         set tag [new e2et]
         set mon [new e2em]

         if { $opt(tagrand) == "uniform" } {
            set tag_ranvar [new RandomVariable/Uniform]
            $tag_ranvar set min_ 0
            $tag_ranvar set max_ [expr $opt(tagmean) / 2]
         } elseif { $opt(tagrand) == "exponential" } {
            set tag_ranvar [new RandomVariable/Exponential]
            $tag_ranvar set avg_ $opt(tagmean)
         } elseif { $opt(tagrand) == "normal" } {
            set tag_ranvar [new RandomVariable/Normal]
            $tag_ranvar set avg_ $opt(tagmean)
```

```
            $tag_ranvar set std_ [expr $opt(tagvar) / 2]
          } elseif { $opt(tagrand) == "weibull" } {
            set tag_ranvar [new RandomVariable/Weibull]
            $tag_ranvar set shape_ 2
            $tag_ranvar set scale_ [expr $opt(tagmean) / 0.88623]
          } elseif { $opt(tagrand) == "constant" } {
             set tag_ranvar [new RandomVariable/Constant]
            $tag_ranvar set val_ $opt(tagmean)
          } else {
            puts "Unknown distribution '%s'"
            exit 0
          }

          if { $opt(tagrand) != "none" } {
            $tag ranvar $tag_ranvar
          }

          $tag per $opt(tagper)
          $agtsrc($i) attach-e2et $tag
          $agtdst($i) attach-e2em $mon
          $mon index $i
          $mon start-log

          $ns stat trace voip_frame_delay $i delay.$i
        }
}
```

4.2.7. Proceso de simulación

Se obtiene una representación gráfica de la simulación descrita en el script gracias al software Network Animator (NAM). Al ejecutar el script mediante “ns <nombre del script>.ns” en el terminal debe aparecer a continuación la ejecución del NAM automáticamente, ya que se construyen tanto el escenario.nam como el escenario-out.tr, para la simulación hecha en este proyecto. Se puede ejecutar por separado el NAM escribiendo en el terminal “nam <nombre del archivo nam>.nam.

A continuación se muestran diferentes capturas de pantallas de la simulación realizada en este proyecto con las diferentes explicaciones que se producen a lo largo de la simulación.

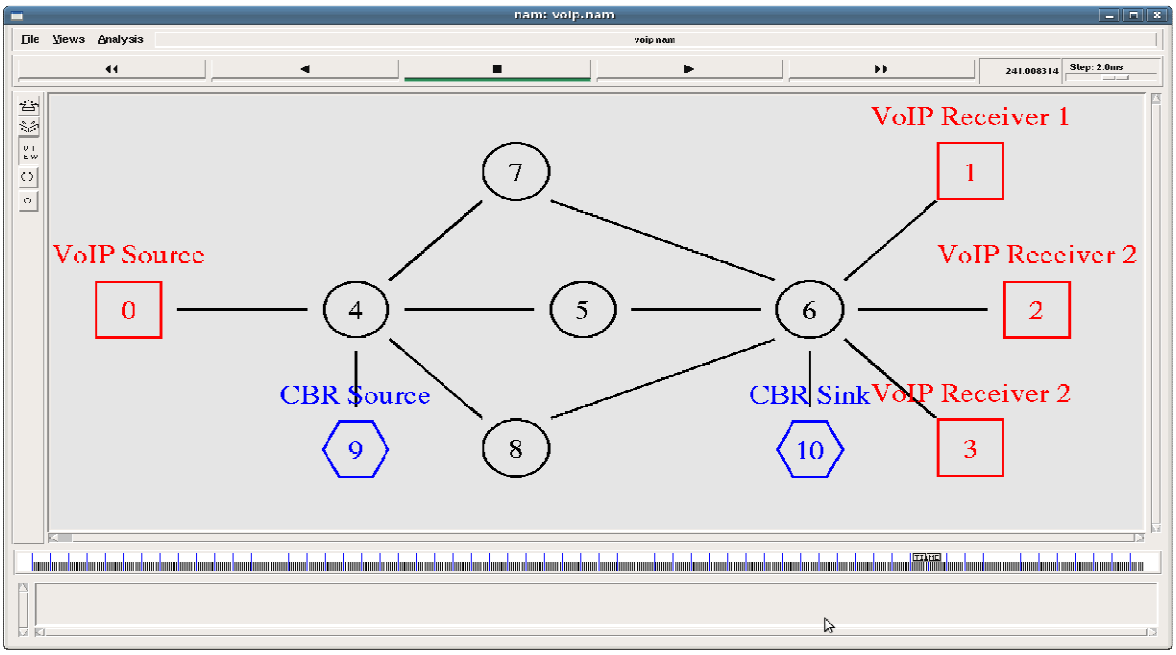


Figura 39. Representación de la topología de red.

En la figura 39 se muestra la topología que se ha utilizado en el proyecto, el nodo 0 recuadrado de color rojo indica la fuente VoIP, los nodos 1, 2 y 3 recuadrados de color rojo indican los destinos VoIP. Los nodos 9 y 10 de color azul, con forma hexagonal indican la fuente CBR que se modela con diferentes cargas y dependiendo del tiempo, como se ha explicado anteriormente.

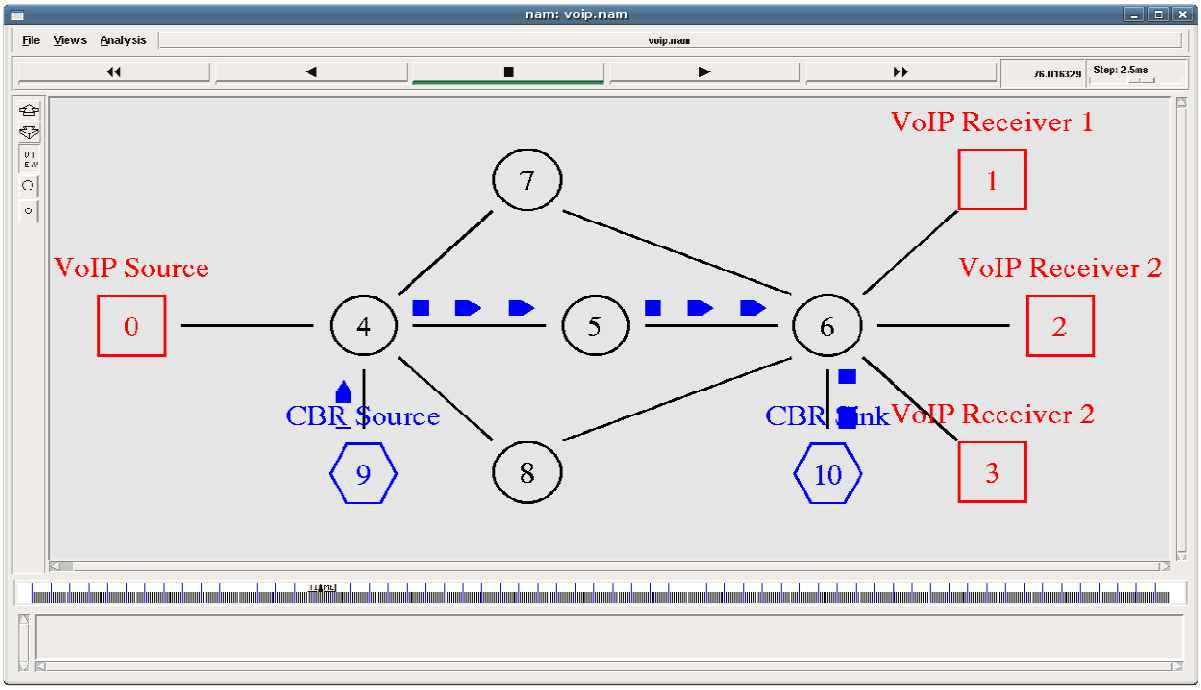


Figura 40. Representación de la topología de red con tráfico de fondo CBR.

En la figura 40 se observa la transmisión CBR de carga 1 Mbps desde el inicio hasta los 120 segundos. La carga CBR se puede apreciar en los paquetes de color azul que son transmitidos desde el nodo 9 (CBR Source) al nodo 10 (CBR Sink).

En la figura 41 se observa la transmisión VoIP desde los 60 segundos del nodo 0 al nodo 1, se visualiza mediante paquetitos rojos, también decir que continua la transmisión CBR con carga 1 Mbps.

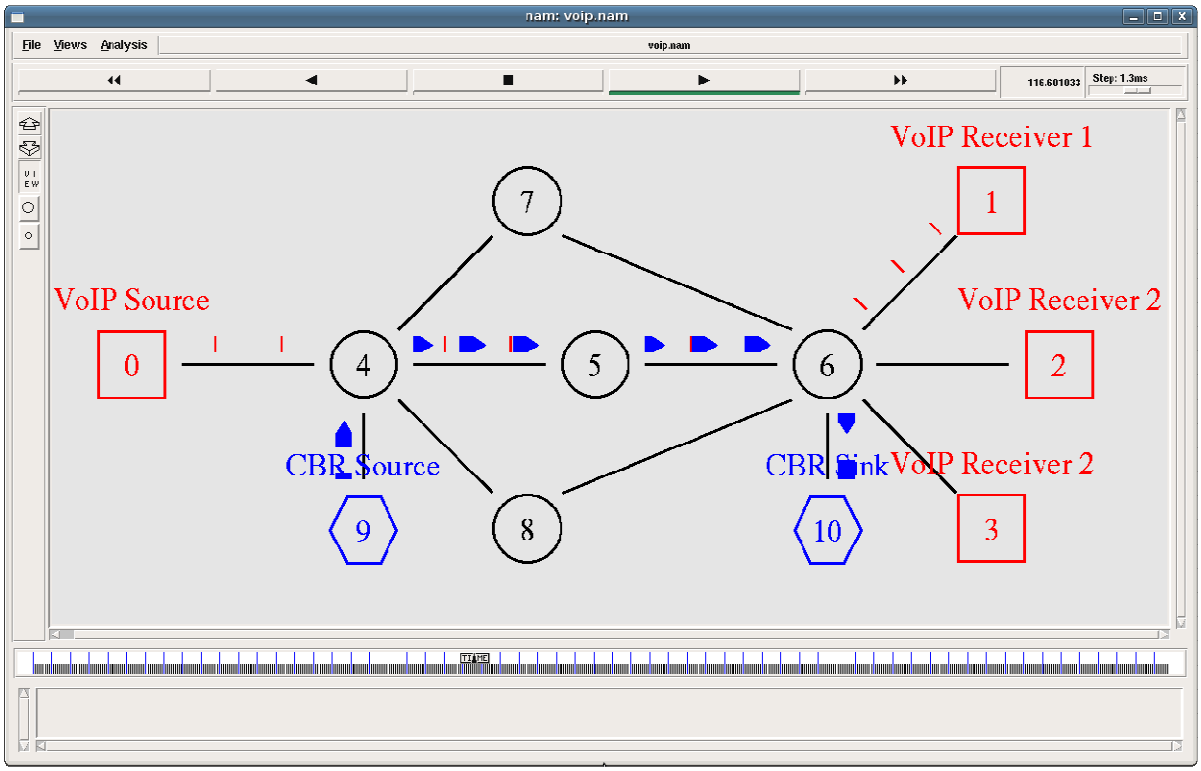


Figura 41. Inicio de la transmisión VoIP.

En las figuras 42 y 43 se observa la pérdida de paquetes CBR y VoIP. En la figura 42 se observa cómo cae del router 4 un paquete CBR (cuadrado de color azul) y en la figura 43 se observa cómo cae del router 4 un paquete VoIP (rombo de color rojo). Esto ocurre debido a que a partir de los 180 segundos hasta los 240 segundos de simulación, la carga CBR se aumenta hasta los 1,98Mbps en enlaces de 2Mbits, lo cual hace que se produzcan pérdidas.

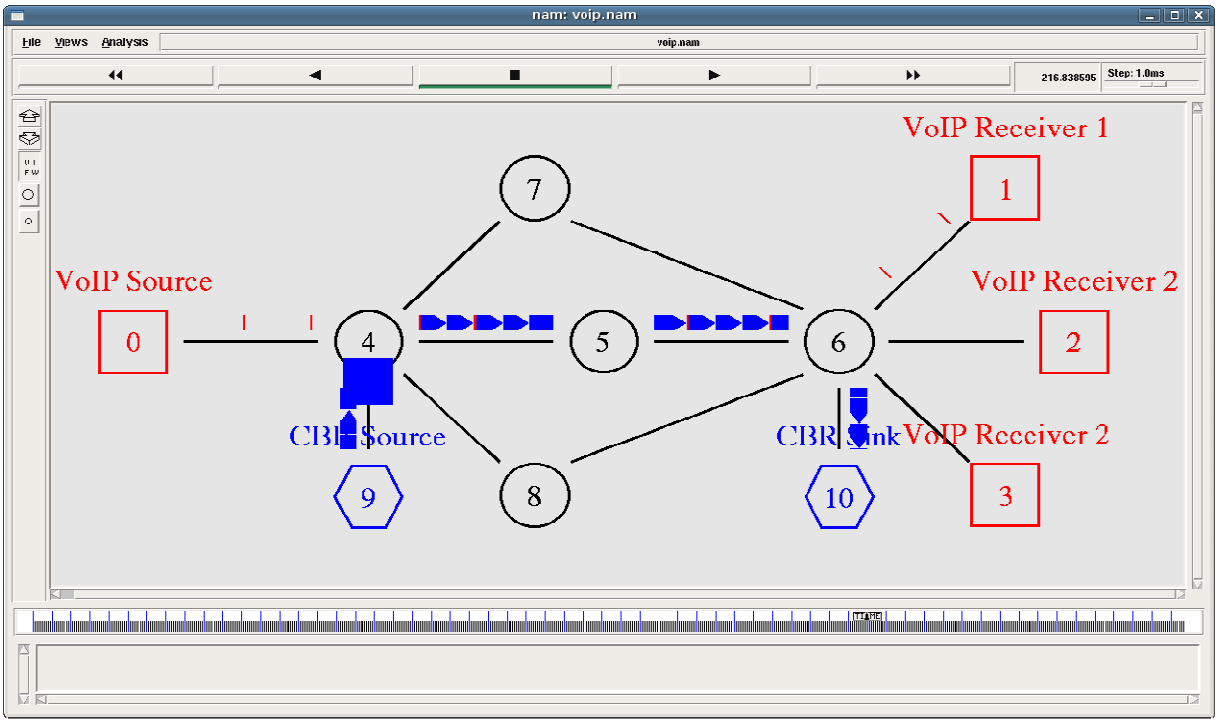


Figura 42. Pérdida de paquete CBR con una carga de 1,98Mbps.

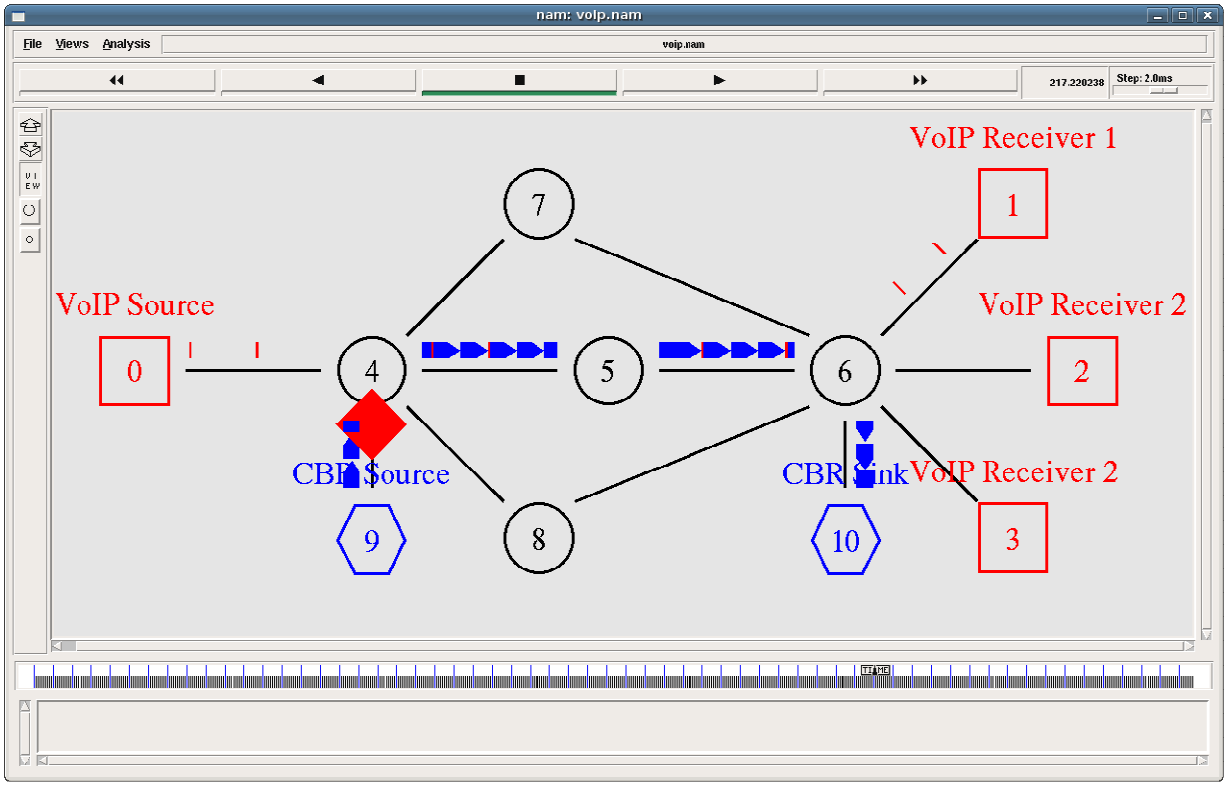


Figura 43. Pérdida de paquete VoIP con una carga de 1,98Mbps

A partir de los 240 segundos hasta los 300 segundos de simulación se deja de transmitir la carga CBR en la simulación, dejando, a su vez, sólo la transmisión VoIP. En las siguientes dos figuras se observará la transmisión VoIP en ON y en OFF.

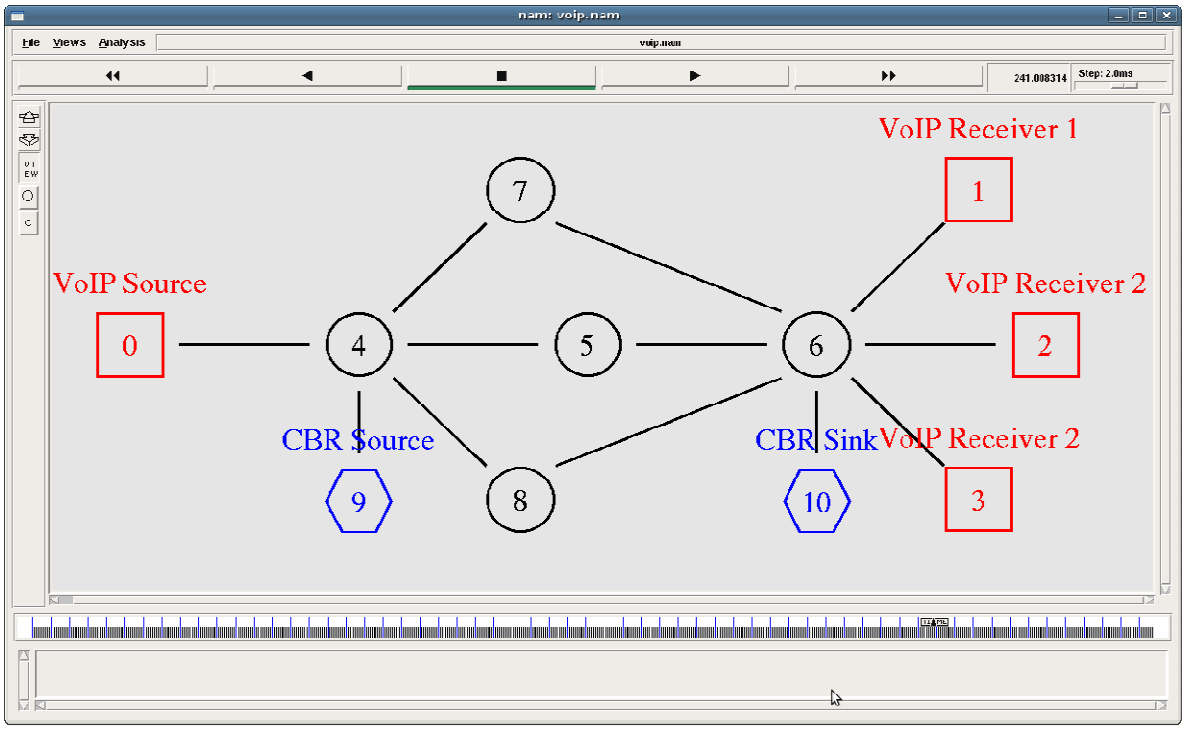


Figura 44. Transmisión VoIP OFF

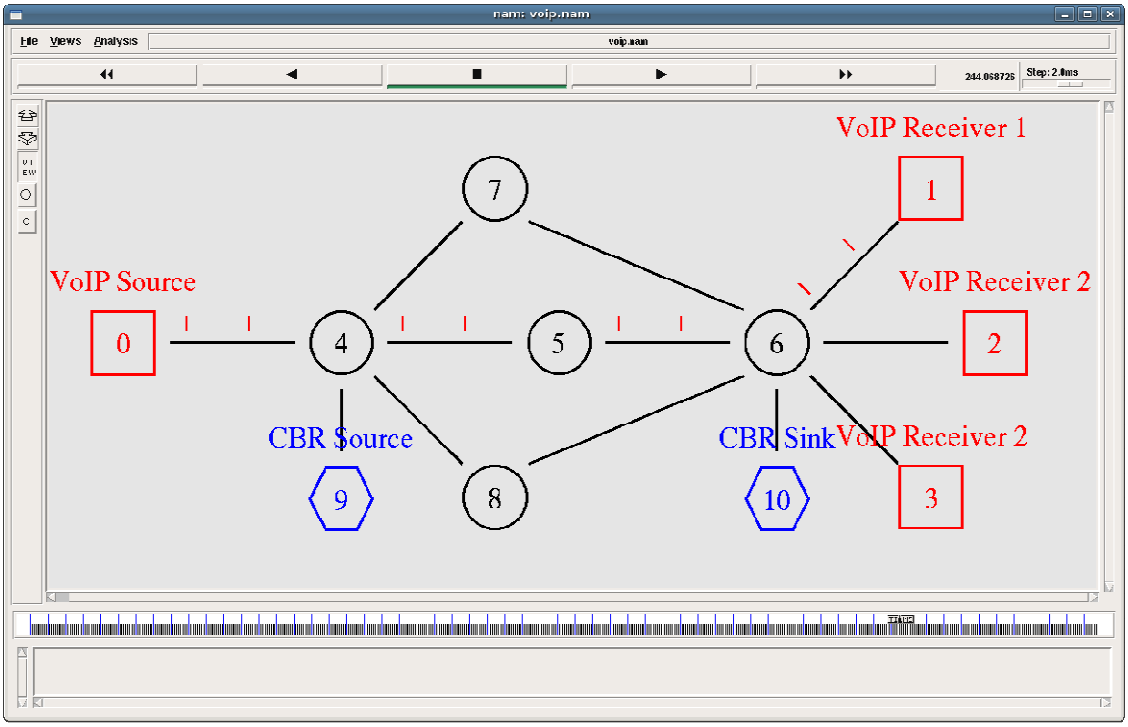


Figura 45. Transmisión VoIP ON

Con ésta simulación se ha pretendido analizar una red semejante a la simulada en el simulador Opnet. Se observaran resultados en diferentes situaciones: sin carga (0Mpbs), con un poco de carga (1Mbps), subiendo la carga (1,9Mbps) y, por último, llevando al límite la carga para provocar pérdidas (1,98Mbps) .

5. RESULTADOS DE SIMULACIÓN

Gracias a la utilización del potente software Matlab se va a mostrar a continuación las gráficas que comparan algunos aspectos del escenario de simulación entre los dos simuladores utilizados, Opnet Modeler y NS2.

Se van a mostrar los resultados más relevantes obtenidos por los dos simuladores, el retardo (delay) extremo a extremo, el jitter y el throughput.

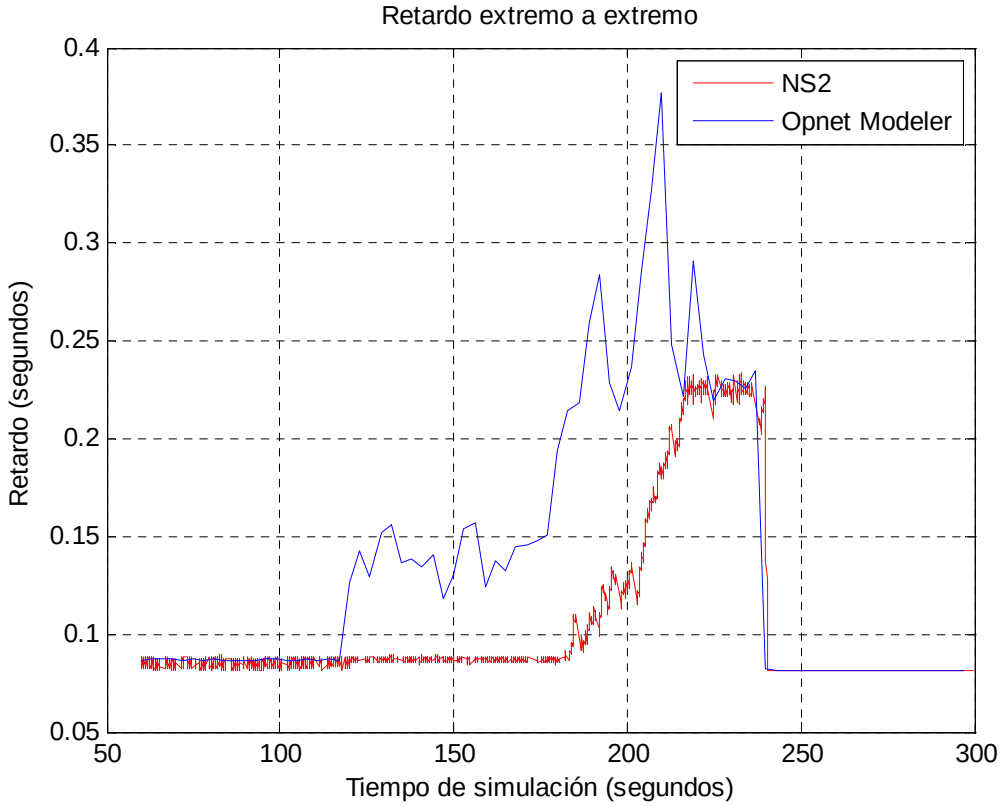


Figura 46. Retardo extremo a extremo

En la figura 46 se observa el retardo en la simulación tanto en NS-2 (en rojo) como en Opnet (en azul). Desde los 60 hasta los 120 segundos de la simulación, se observa el mismo delay en las dos simulaciones, que va fluctuando alrededor de 0.08 segundos ya que solo afecta el retardo de los enlaces, que están configurados con 0.02 segundos de retardo cada uno. Desde los 120 hasta los 180 segundos de la simulación, se observa una subida del retardo, ya que el tráfico de fondo está configurado en 1.9Mbits. Se observa que en el escenario del simulador Opnet Modeler se nota más la subida de dicho tráfico de fondo ya que el retardo se incrementa más que en el escenario del simulador NS2. Desde los 180 a los 240 segundos de la simulación se observa una subida del retardo en ambas simulacions, con una carga de tráfico de fondo de 1.98Mbits (enlaces muy congestionados).

A partir de los 240 segundos hasta el final de la simulación el retardo en ambas simulacions es exactamente igual, 0.08 segundos (suma del retardo de los enlaces), ya que el tráfico de fondo en ese período de la simulación es de 0Mbits.

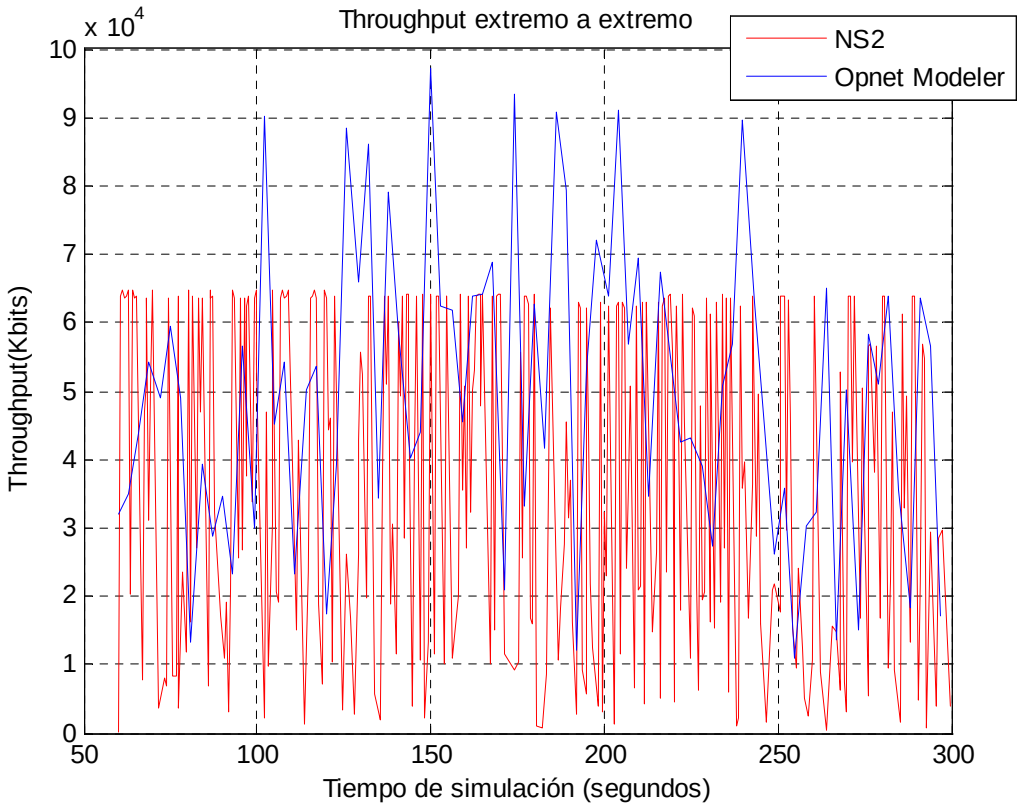


Figura 47. Throughput extremo a extremo

En la figura 47 se observa el throughput, la gráfica del simulador NS2 es global, de extremo a extremo sobre UDP. En cambio, los datos relativos a la simulación en el Opnet Modeler corresponden al throughput del enlace entre el último router y el destino.

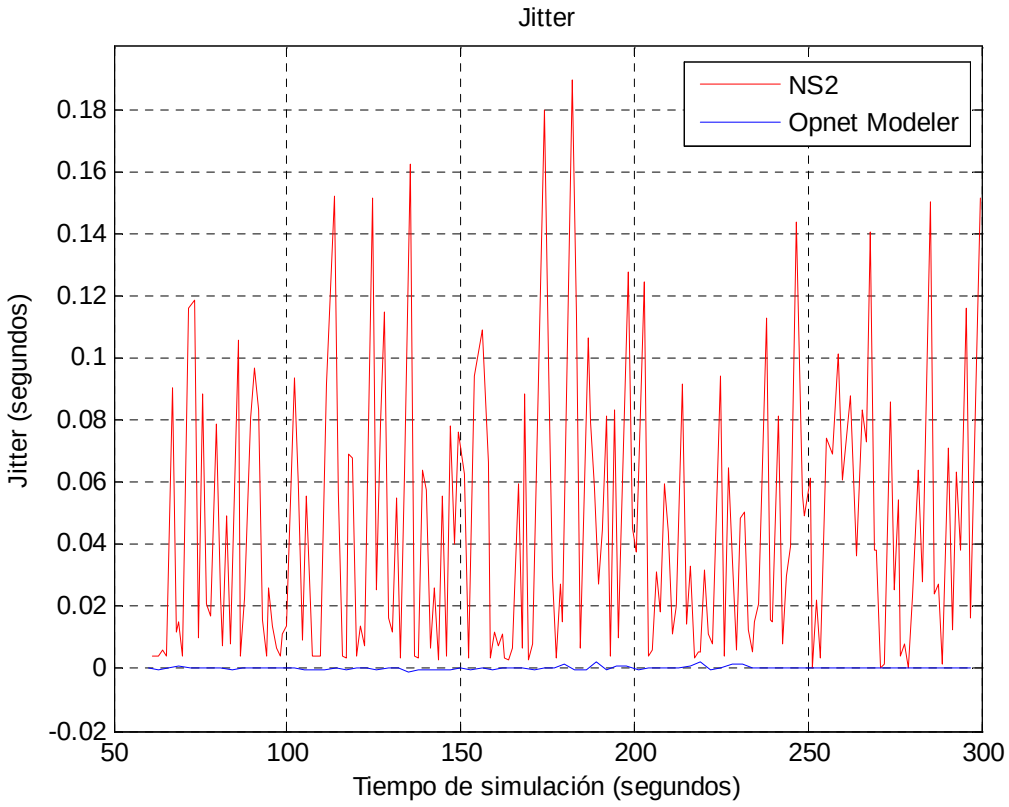


Figura 48. Jitter

En la figura 48 se observa el Jitter. Tal y como anteriormente se ha explicado, consiste en la variación en el tiempo en la llegada de los paquetes, causada por congestión de red, pérdida de sincronización o por los retardos sufridos en las diferentes rutas seguidas por los paquetes para llegar al destino.

Se observa que el valor obtenido en la simulación con Opnet Modeler es inferior a 100 ms, como se recomienda en las comunicaciones VoIP. En cambio, en la simulación en NS2 aparecen picos que superan éste límite. Estas diferencias entre la simulación en Opnet Modeler y el módulo VoIP de NS2 utilizado en las simulaciones se puede deber a las disciplinas de prioridad utilizadas en las colas y en la reserva de ancho de banda por defecto utilizados en uno y otro simulador. En cuanto a los resultados se puede decir que la comunicación VoIP en el simulador Opnet Modeler utiliza mecanismos de QoS (calidad de servicio) más fiables que el módulo VoIP del NS2, dado que el Jitter siempre es inferior a 100 ms.

Para mejorar la QoS se opta por la utilización de un Buffer de compensación del Jitter. Consiste básicamente en asignar una cola para ir recibiendo paquetes y sirviéndolos con un pequeño retraso. En el momento en que se necesite reproducir el contenido de algún paquete, si aún no está en el buffer (porque se perdió o no ha llegado todavía), se descarta. Normalmente en los teléfonos IP (hardware y software) se pueden modificar los buffers. Un

aumento del tamaño del buffer implica menos pérdida de paquetes pero mayor retraso, ya que se tarda más tiempo en reproducir el contenido del buffer. Una disminución implica menor retardo pero más pérdida de paquetes, ya que contiene menos paquetes almacenados y todo su contenido se reproduce más rápido.

6. CONCLUSIONES

La realización de éste proyecto ha sido muy satisfactoria para mí ya que en primer lugar se me ha dado la oportunidad de conocer dos programas muy potentes de simulación de redes, como son el Opnet Modeler y el Network Simulator 2, así como, también a su vez, comprender el funcionamiento de los protocolos de transferencia en tiempo real RTP/RTCP muy utilizados hoy en día en aplicaciones multimedia, como VoIP.

A primera vista, la evaluación de los diferentes simuladores parece ser un procedimiento sencillo. Sin embargo, no es el caso. La curva de aprendizaje para cada uno de los simuladores era diferente. Para crear en NS2 un escenario ha implicado el aprendizaje del lenguaje de programación OTcl, para la creación del script donde se definía el escenario de simulación.

Opnet Modeler es un software comercial de simulación muy completo en el campo de las redes de telecomunicaciones. Es un software que permite al usuario interactuar sin problemas y ofrece una gran facilidad de interpretación y creación de escenarios, aparte de tener en cada objeto una serie de atributos configurables con un entorno de simulación gráfico y muy atractivo. Dispone de múltiples librerías, lo que permite simular gran diversidad de redes donde intervienen numerosos protocolos, entre ellos el que se necesitaba en este proyecto, el RTP.

El aprendizaje de este software con tal nivel para poder realizar las simulaciones deseadas contrajo un largo costo de tiempo invertido en aprender a manejar todas las posibilidades que otorga.

Uno de los inconvenientes que me encontré con Opnet Modeler es que hay que adquirir licencias de pago. Para nuestro entorno universitario obtuvimos una licencia gratuita limitada, pero si se requiere soporte técnico o algún módulo adicional en especial para la simulación, había que adquirir una licencia de pago, en nuestro caso utilizamos la versión OPNET MODELER cedida a la Universidad a través del University Program de OPNET.

Por otro lado, la naturaleza gratuita de NS2 lo hace atractivo para los investigadores. También por la posibilidad de ampliar el NS2 con diferentes módulos, como es el caso del módulo VoIP realizado por investigadores italianos.

Opnet Modeler tiene el inconveniente de no tener implementado el módulo RTCP y por tanto carece de estadísticas relacionadas en los resultados de simulación obtenidos.

7. BIBLIOGRAFÍA

[1] Software Opnet Modeler 14.5. Disponible en:
< https://enterprise1.opnet.com/support/product/md/145PL8>

[2] Software Network Simulator 2. Disponible en: < http://isi.edu/nsnam/ns/ >

[3] Página oficial Ubuntu. Web: < http://www.ubuntu.com/ >

[4] Software Network Animator NAM. Disponible en:
< http://www.isi.edu/nsnam/nam/ >

[5] Software Tracegraph 2.02. Disponible en: < http://www.tracegraph.com/ >

[6] Manual NS2. Disponible en:
< http://www.isi.edu/nsnam/ns/ns-documentation.html >

[7] Tutorial de NS2 de Marc Greis. Disponible en:
< http://www.isi.edu/nsnam/ns/tutorial/ >

[8] Instalación NS2 en Ubuntu. Disponible en:
< http://blog.pucp.edu.pe/item/37506 >

[9] RFC 3550. RTP: A Transport Protocol for Real-Time Applications.
Disponible en: < http://www.rfc-editor.org/rfc/rfc3550.txt >

[10] Prácticas OPNET MODELER DEL CURSO de doctorado “Introducción A la simulación de redes con OPNET”, del profesor Fernando Boronat Seguí.

[11] Manual lenguaje OTcl. Disponible en:
< http://otcl-tclcl.sourceforge.net/otcl/ >

[12] B. Bellalta, Oliver M.; D. Rincón. “Performance of the GPRS RLC/MAC protocols with VoIP traffic”. Personal Indoor an Mobile Radio Communications, 2002. IEE international symposium on, Vol: 1, 15-18. Sept 2002.

[13] Página web personal de Víctor Carrascal, donde se encuentra el código fuente e instrucciones para la instalación de agentes RTP y RTCP en NS2.
http://gridnet.upc.es/~vcarrascal/ns2/

[14] Código fuente, documentación y guía de instalación en NS2 de los protocolos RTP/RTCP de la Academia de Investigación del Instituto de Tecnologías de la Computación perteneciente a la Universidad de Patras (Grecia): http://ru6.cti.gr/ru6/ns_rtp_extensions.php

[15] La documentación detallada del módulo VoIP desarrollado por un grupo de investigadores de la Universidad de Pisa (Italia). Se puede encontrar en la siguiente referencia:

User-level Performance Evaluation of VoIP Using ns-2
Andrea Bacioccola, Claudio Cicconetti, Giovanni Stea
Workshop on Network Simulation Tools (NSTools) 2007
Nantes, France, October 22, 2007

[16] Módulo VoIP desarrollado y proceso detallado de la instalación en el NS2 se encuentra en el enlace: http://cng1.iet.unipi.it/wiki/index.php/Ns2voip

[17] Correcta instalación de Opnet Modeler 14.5 y Visual Studio 2005 se encuentra en el enlace: http://www.jesusllor.es/?p=322