

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior
INGENIERÍA TÉCNICA DE TELECOMUNICACIÓN
ESPECIALIDAD EN TELEMÁTICA



TRABAJO FIN DE CARRERA

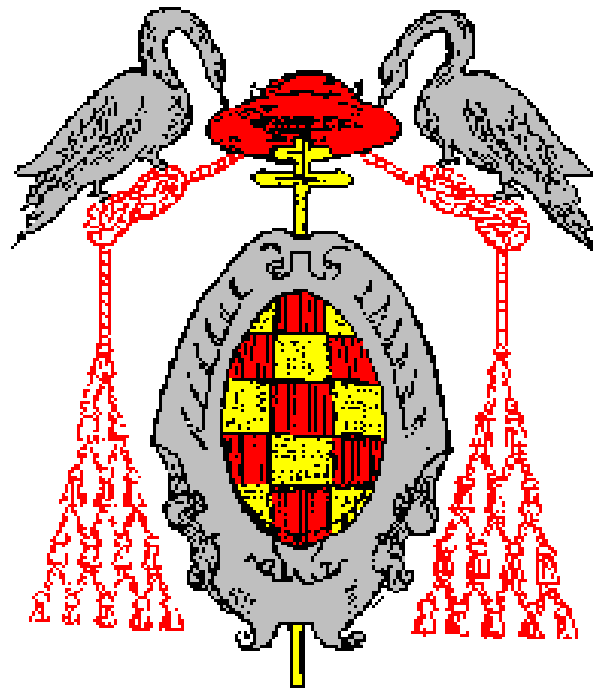
SIMULACIÓN DE UNA RED DE TELEFONÍA
MÓVIL GSM MEDIANTE EL ENTORNO DE
SIMULACIÓN OMNET++

DAVID NÚÑEZ CLEMENTE

MAYO 2006

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

INGENIERÍA TÉCNICA DE TELECOMUNICACIÓN
ESPECIALIDAD EN TELEMÁTICA.



Trabajo Fin de Carrera

**Simulación de una Red de Telefonía Móvil GSM mediante el Entorno
de Simulación OMNET++.**

DAVID NÚÑEZ CLEMENTE

Mayo 2006.

*A todos aquellos que me
han ayudado durante
estos años, a mis amigos,
a Pilar García por su
ayuda en este proyecto y
en especial a mis padres
y a mi hermano, que me
han apoyado siempre.
Gracias.*

ÍNDICE DE CONTENIDOS

1	RESUMEN	13
2	INTRODUCCIÓN.....	15
2.1	DEFINICIONES.....	18
3	ESTADO DEL ARTE	21
4	OBJETIVOS DEL PROYECTO	23
5	BASE TEÓRICA. DESCRIPCIÓN DE UNA RED GSM.....	25
5.1	INTRODUCCIÓN	25
5.2	RED TELEFÓNICA PÚBLICA MÓVIL.....	26
5.2.1	<i>Historia</i>	27
5.2.2	<i>Servicios</i>	28
5.2.3	<i>Jerarquía.....</i>	28
5.2.4	<i>Numeración.....</i>	29
5.2.5	<i>Arquitectura gsm.....</i>	29
5.2.6	<i>Señalización</i>	37
5.3	CAPA FÍSICA	38
5.3.1	<i>Banda de frecuencias</i>	38
5.3.2	<i>TDMA.....</i>	40
5.3.3	<i>Características del canal radio móvil.....</i>	40
5.3.4	<i>Ráfagas.....</i>	40
5.3.5	<i>Canales.....</i>	40
5.4	PLANIFICACIÓN RADIO.....	43
5.4.1	<i>Introducción</i>	43
5.4.2	<i>Teoría celular.....</i>	43
5.4.3	<i>Diseño de la red celular.....</i>	44

5.4.4	Métodos de predicción de pérdidas.....	48
5.4.5	Balances de enlace	50
5.5	SUBSISTEMA BSS	52
5.5.1	Introducción.....	52
5.5.2	Interfaz ABIS.....	53
5.5.3	Conexión	54
5.5.4	Interfaz A.....	57
5.6	SUBSISTEMA SSS.....	57
5.6.1	Introducción.....	57
5.6.2	Funciones de red.....	58
5.6.3	Componentes del subsistema de conmutación	59
5.7	SEÑALIZACIÓN.....	59
5.7.1	Protocolo LAPD.....	60
5.7.2	Protocolo BSSAP	60
5.7.3	Protocolo MAP.....	61
6	OMNET++	63
6.1	INTRODUCCIÓN.....	63
6.2	CARACTERÍSTICAS PRINCIPALES DE OMNET++.....	64
6.2.1	Los Módulos en OMNET++	69
6.2.2	Programación de algoritmos	71
6.2.3	Cómo usar OMNET++	71
6.3	EL LENGUAJE NED	73
6.3.1	Vista rápida del lenguaje NED	73
6.3.2	Directivas de importación.....	73
6.3.3	Descripción de canales	73
6.3.4	Módulos simples.....	74
6.3.5	Módulos compuestos	75
6.3.6	Definición de una red.....	77
6.4	LOS MÓDULOS SIMPLES.....	77
6.4.1	Creación de un módulo simple.....	78
6.4.2	Añadir funcionalidad a un módulo.....	78
6.4.3	Enviar y recibir mensajes.....	79
6.5	LOS MENSAJES	80
6.5.1	Definición de mensajes.....	82
6.6	LA LIBRERÍA DE SIMULACIÓN	83
6.6.1	Las funciones estadísticas.....	83
6.6.2	Imprimir en pantalla	84
6.6.3	Tiempo de simulación.....	84
6.6.4	Grabar resultados.....	85
6.7	CONSTRUIR, CONFIGURAR Y EJECUTAR LA SIMULACIÓN	86
6.7.1	Configuración de simulaciones	89
6.8	ANIMACIÓN Y VISUALIZACIÓN.....	93
6.8.1	Asignación de características de visualización en el fichero ned	93
6.8.2	Función bubble.....	96
6.9	ANÁLISIS DE LOS RESULTADOS	97
7	OPENGL	105
7.1	GESTIÓN DE VENTANAS.....	106
7.2	GESTIÓN DE VISUALIZACIÓN.....	106
7.3	CREAR UNA APLICACIÓN.....	107
7.3.1	Función de eventos de GLUT.....	107
7.3.2	Área de proyección de la imagen.....	108
7.3.3	Eventos de teclado.....	110

7.3.4	Eventos de ratón.....	110
7.3.5	Primitivas de dibujo	110
7.3.6	Color	112
7.3.7	Plantilla para la creación de programas con opengl.....	112
8	DESARROLLO	115
8.1	PROGRAMACIÓN DE LA APLICACIÓN.....	115
8.1.1	Estructura de la red	116
8.1.2	Movimiento de las estaciones móviles.....	116
8.1.3	Inicio de funcionamiento de MS.....	119
8.1.4	Establecimiento de una llamada	122
8.1.5	Duración de una llamada.....	125
8.1.6	Actualización de posición.....	125
8.1.7	Cálculo de potencias.....	126
8.1.8	Elección de BTS.....	127
8.1.9	Establecimiento de canales.....	127
8.1.10	Liberación de canales.....	128
8.1.11	Procedimientos MAP.....	128
8.1.12	Realización de traspasos.....	130
8.2	FICHEROS.....	132
8.2.1	El fichero “omnetpp.ini”.....	132
8.2.2	El fichero “gsmsim.ned”.....	136
8.2.3	El fichero “gsmsim.h”.....	145
8.2.4	El fichero “ms.cpp”	148
8.2.5	El fichero “air.cpp”	152
8.2.6	El fichero “bts.cpp”	154
8.2.7	El fichero “bsc.cpp”	158
8.2.8	El fichero “msc.cpp”	163
8.2.9	El fichero “vlr.cpp”	165
8.3	MENSAJES INCLUIDOS EN LA APLICACIÓN.....	167
8.3.1	Mensajes propios de la aplicación.....	168
8.3.2	Mensajes interfaz Um.....	170
8.3.3	Mensajes interfaz Abis	176
8.3.4	Mensajes interfaz A.....	182
8.3.5	Mensajes interfaz B.....	186
8.4	APLICACIÓN DE APOYO: SIMOFFLINE.EXE.....	191
8.4.1	Objetivo de la aplicación.....	191
8.4.2	Programación	191
9	HERRAMIENTAS	193
9.1	INSTALACIÓN DE VISUAL STUDIO 6.0	194
9.2	INSTALACIÓN DE OMNET++	195
9.3	INSTALACIÓN DE OPENGL GLUT	196
10	RESULTADOS.....	197
10.1	EJEMPLO 1	197
10.2	EJEMPLO 2	206
10.3	EJEMPLO 3	208
11	MANUAL DE USUARIO.....	215
11.1	FUNCIONAMIENTO GSMSIM.EXE.....	215
11.1.1	Herramientas del simulador.....	216
11.1.2	Ventanas de detalles	219
11.1.3	Proceso de ejecución.....	229

11.2	FUNCIONAMIENTO SIMOFFLINE.EXE	237
11.2.1	Objetivo de la aplicación.....	237
11.2.2	Funcionamiento de la aplicación	238
11.3	FUNCIONAMIENTO OMNET++ SCALARS Y OMNET++ PLOVE.	244
11.3.1	OMNeT++ Scalars.....	245
11.3.2	OMNeT++ Plove.....	248
12	PLIEGO DE CONDICIONES	253
13	COSTES Y PRESUPUESTO	255
13.1	MANO DE OBRA	255
13.2	MATERIAL	256
13.3	COSTES TOTALES	257
13.4	PRESUPUESTO	257
14	CONCLUSIONES Y TRABAJOS FUTUROS	259
14.1	TRABAJOS FUTUROS.....	260
14.1.1	Trayectorias. Módulos de movilidad.....	260
14.1.2	Modelos de propagación y obstáculos	261
14.1.3	Mensajes y módulos.....	264
14.1.4	Programa de inicialización.....	264
14.1.5	Programa de visualización de posiciones	264
14.1.6	Realización de un simulador de UMTS.	266
15	BIBLIOGRAFÍA	267
15.1	OMNET ++:.....	267
15.2	RED GSM	268
15.3	ESTÁNDARES GSM DE ETSI:.....	268
15.4	PROGRAMACIÓN C++:	269
15.5	PÁGINAS WEB:	269
APÉNDICE A.	CÓDIGO FUENTE SIMOFFLINE.EXE	271
APÉNDICE B.	CÓDIGO FUENTE DE GSMSIM.EXE	283
APÉNDICE C.	CONTENIDO DEL CD.....	403

ÍNDICE DE FIGURAS

FIGURA 1 TABLA DE USUARIOS DE GSM EN EL MUNDO DE 2002 A 2005.	26
FIGURA 2 JERARQUÍA EN GSM.	28
FIGURA 3 EQUIPOS EN UNA RED GSM.	34
FIGURA 4 ESQUEMA DE LOS NIVELES DE UNA RED GSM.	38
FIGURA 5 FORMA CELULAR	45
FIGURA 6 ANTENAS SECTORIZADAS.	47
FIGURA 7 DENSIDAD DE TRÁFICO EN GSM.	48
FIGURA 8 POSICIÓN DEL EQUIPO TRAU.	53
FIGURA 9 TOPOLOGÍA DE CONEXIONES ENTRE BTS Y BSC.	56
FIGURA 10 SIMULACIÓN DE UNA RED LAN PERTENECIENTE AL INET FRAMEWORK DEMO FOR OMNET/OMNET++, RED.	66
FIGURA 11 SIMULACIÓN DE UNA RED LAN PERTENECIENTE AL INET FRAMEWORK DEMO FOR OMNET/OMNET++, MÓDULO COMPUESTO HUB.	67
FIGURA 12 EJEMPLO DE CONEXIÓN ENTRE DOS MÓDULOS SIMPLES.	68
FIGURA 13 EJEMPLO DE CONEXIÓN DE DOS MÓDULOS SIMPLES PERTENECIENTES A UN MÓDULO COMPUESTO AL EXTERIOR.	68
FIGURA 14 SISTEMA CON VARIOS MÓDULOS.	70
FIGURA 15 EJEMPLO DE CONFIGURACIÓN DE MÓDULOS.	75
FIGURA 16 PASOS EN LA CONSTRUCCIÓN, CONFIGURACIÓN Y EJECUCIÓN DE UNA SIMULACIÓN EN OMNET++. ...	87
FIGURA 17 AJUSTES DEL PROYECTO EN MICROSOFT VISUAL C++ 6.0.	88
FIGURA 18 AJUSTES DEL PROYECTO EN MICROSOFT VISUAL C++ 6.0. VENTANA DE PROJECT SETTINGS.	89
FIGURA 19 POSICIÓN DE LOS TELÉFONOS MÓVILES EN MATRIZ.	95
FIGURA 20 POSICIÓN DE LOS MÓDULOS EN ANILLO.	96
FIGURA 21 RESULTADO DE LA FUNCIÓN BUBBLE EN LA SIMULACIÓN.	97

FIGURA 22 ICONOS PARA EL ACCESO A LOS PROGRAMAS DE VISUALIZACIÓN DE GRÁFICAS Y PARA EL EDITOR DE LENGUAJE NED.	98
FIGURA 23 VISUALIZACIÓN DE RESULTADOS DE SIMULACIÓN OMNET++ SCALARS	99
FIGURA 24 FILTROS DE SCALARS.....	99
FIGURA 25 ICONOS EN SCALARS.....	100
FIGURA 26 TIEMPO DE LLAMADA.....	101
FIGURA 27 APLICACIÓN DE LA HERRAMIENTA DE ZOOM A LA GRÁFICA.....	102
FIGURA 28 PROGRAMA PLOVE PARA VISUALIZAR RESULTADOS VECTORIALES.....	103
FIGURA 29 HERRAMIENTAS DEL PROGRAMA PLOVE.....	103
FIGURA 30 GRÁFICA DE POTENCIAS DE LA ESTACIÓN MÓVIL 0.....	104
FIGURA 31 COORDENADAS ESPACIALES EN OPENGL.....	109
FIGURA 32 REPRESENTACIÓN DE LA IMAGEN DESDE EL PUNTO DE OBSERVACIÓN.....	109
FIGURA 33 TIPOS DE PRIMITIVAS Y RESULTADO.....	111
FIGURA 34 VELOCIDAD DE UN EQUIPO MS.....	117
FIGURA 35 MOVIMIENTO LINEAL DE MS.....	118
FIGURA 36 MOVIMIENTO ALEATORIO DE MS.....	119
FIGURA 37 INICIO DE UN MÓDULO MS.....	121
FIGURA 38 FUNCIÓN DE ESTABLECIMIENTO DE LLAMADA.....	123
FIGURA 39 ESTABLECIMIENTO DE LLAMADA.....	124
FIGURA 40 CÁLCULO DE LA DISTANCIA ENTRE MS Y BTS.....	127
FIGURA 41 LIBERACIÓN DE CANAL.....	128
FIGURA 42 PROCEDIMIENTOS MAP PARA LA ACTUALIZACIÓN DE POSICIÓN.....	129
FIGURA 43 PROCEDIMIENTOS MAP PARA EL ESTABLECIMIENTO DE LLAMADA ENTRANTE (ARRIBA) O SALIENTE (ABAJO).....	130
FIGURA 44 REALIZACIÓN DE TRASPASOS.....	131
FIGURA 45 PLANO ESPACIAL DE POSICIÓN DE LOS EQUIPOS. EJEMPLO 1.....	135
FIGURA 46 PLANO ESPACIAL DE POSICIÓN DE LOS EQUIPOS. EJEMPLO 2.....	136
FIGURA 47 EJEMPLO DE SIMULACIÓN CON GSMSIM.....	139
FIGURA 48 CREAR UN PROYECTO OMNET++.....	195
FIGURA 49 OMNET++ APPWIZARD.....	196
FIGURA 50 TIEMPO DE LLAMADA.....	198
FIGURA 51 PORCENTAJE DE LLAMADAS.....	198
FIGURA 52 PORCENTAJE DE LLAMADAS.....	199
FIGURA 53 LLAMADAS CON ÉXITO Y BLOQUEADAS.....	200
FIGURA 54 ASIGNACIONES.....	200
FIGURA 55 LLAMADAS CON ÉXITO.....	201
FIGURA 56 POTENCIAS EN MS(7).....	201
FIGURA 57 POTENCIAS EN MS(11).....	202
FIGURA 58 POTENCIA EN MS(5).....	202
FIGURA 59 POTENCIA EN MS(1).....	203
FIGURA 60 POTENCIA EN MS(9).....	203
FIGURA 61 POTENCIA RECIBIDA EN MS (19).....	204
FIGURA 62 CANALES USADOS EN LA BTS 0 DURANTE LA SIMULACIÓN.....	204
FIGURA 63 LLAMADAS TOTALES Y CON ÉXITO EN LA BTS(0).....	205
FIGURA 64 USO DE CANALES TCH.....	205
FIGURA 65 POTENCIA EN MS(65).....	206
FIGURA 66 NÚMERO DE LLAMADAS CURSADAS.....	207
FIGURA 67 CANALES OCUPADOS EN LA BTS(0).....	207
FIGURA 68 OCUPACIÓN DE CANALES EN BTS(1).....	208
FIGURA 69 LLAMADAS EN EL EJEMPLO 3.....	211
FIGURA 70 PORCENTAJE DE LLAMADAS CON ÉXITO EN EL EJEMPLO 3.....	211
FIGURA 71 LLAMADAS EN EL EJEMPLO 3.....	212
FIGURA 72 CANALES TCH OCUPADOS.....	212
FIGURA 73 CANALES TCH OCUPADOS.....	213

FIGURA 74 CANALES TCH OCUPADOS	214
FIGURA 75 VENTANA PRINCIPAL DEL SIMULADOR.....	216
FIGURA 76 HERRAMIENTAS DEL SIMULADOR.....	217
FIGURA 77 VENTANA DE REPRESENTACIÓN DE LA RED.....	218
FIGURA 78 VENTANA DE MENSAJES.....	218
FIGURA 79 VENTANA DE VISUALIZACIÓN DE MSC-VLR.....	220
FIGURA 80 VENTANA DE CONEXIÓN MSC HACIA VLR.....	220
FIGURA 81 VENTANA DE CONEXIÓN VLR HACIA MSC.....	221
FIGURA 82 VENTANA DE CONEXIÓN BSC HACIA MSC.....	221
FIGURA 83 VENTANA DE CONEXIÓN MSC HACIA BSC.....	222
FIGURA 84 ICONO DE VISUALIZACIÓN DE EVENTOS.....	222
FIGURA 85 VENTANA DE EVENTOS.....	223
FIGURA 86 VENTANA DE DETALLES DEL MÓDULO BSC.....	224
FIGURA 87 VENTANA DE DETALLES DEL MÓDULO BSC. INFORMACIÓN DE PUERTAS.....	224
FIGURA 88 VENTANA DE DETALLES DEL MÓDULO BSC. INFORMACIÓN DE PARÁMETROS.....	225
FIGURA 89 VENTANAS DE PARÁMETROS DE UN MÓDULO BTS.....	225
FIGURA 90 VENTANA DE INFORMACIÓN DE RED. PARÁMETROS.....	226
FIGURA 91 VENTANA DE INFORMACIÓN DE RED. MÓDULOS.....	227
FIGURA 92 VENTANA DE DETALLES DE MENSAJE.....	228
FIGURA 93 GRÁFICAS DE VARIACIÓN DE POTENCIA RECIBIDA EN MS(17).....	229
FIGURA 94 INICIO DE LA EJECUCIÓN. MENSAJE INIT.....	230
FIGURA 95 VENTANA DE STOP.....	231
FIGURA 96 INFORME DE POTENCIAS EN PANTALLA DE EVENTOS.....	231
FIGURA 97 INFORMACIÓN DE ASIGNACIÓN DE CANALES EN LA PANTALLA DE EVENTOS.....	232
FIGURA 98 MENSAJES EN LA PANTALLA DE VISUALIZACIÓN DE RED.....	233
FIGURA 99 MENSAJES EN LA VENTANA DE VISUALIZACIÓN DEL MÓDULO MSC_VLR.....	233
FIGURA 100 REPRESENTACIÓN DE LA FUNCIÓN BUBBLE().....	234
FIGURA 101 ACCESO A LOS PARÁMETROS DE UN MÓDULO DESDE EL ÁRBOL DE LA RED.....	235
FIGURA 102 ACCESO A LOS MENSAJES DE UN MÓDULO A TRAVÉS DEL ÁRBOL DE LA RED.....	236
FIGURA 103 SIMOffLine Y ARCHIVOS DE POSICIÓN.....	238
FIGURA 104 PANTALLA DE SIMOffLine.EXE.....	239
FIGURA 105 MENÚ DE SIMOffLine.EXE.....	240
FIGURA 106 SELECCIÓN DE MS EN SIMOffLine.EXE.....	241
FIGURA 107 FINALIZACIÓN DE SELECCIÓN DE MS EN SIMOffLine.EXE.....	242
FIGURA 108 FIN DE SELECCIÓN DE MS EN SIMOffLine.EXE.....	243
FIGURA 109 TRAYECTORIA DE UN MÓVIL.....	244
FIGURA 110 PANTALLA DE DATOS DE SCALARS.....	245
FIGURA 111 CUADROS DE SELECCIÓN DE FILTRADO.....	246
FIGURA 112 MENÚ DESPLEGABLE DE FILTRADO.....	246
FIGURA 113 CUADRO DE DIÁLOGO. GENERACIÓN DE GRÁFICAS.....	247
FIGURA 114 MENÚ DESPLEGABLE PARA OPCIONES DE GRÁFICA.....	247
FIGURA 115 ICONOS PARA GUARDAR.....	248
FIGURA 116 PANTALLA OMNeT++ PLOVE.....	249
FIGURA 117 SELECCIÓN DE DATOS.....	250
FIGURA 118 HERRAMIENTAS PLOVE.....	250
FIGURA 119 PRESENTACIÓN DE DATOS CON PLOVE.....	251
FIGURA 120 MODELO DE MOVIMIENTO MANHATTAN.....	261
FIGURA 121 COBERTURA DE ESTACIONES BASE, DE LA TEORÍA A LA PRÁCTICA.....	262
FIGURA 122 COBERTURA DE MICROCELAS EN UNA CIUDAD.....	263
FIGURA 123 LOS EDIFICIOS ACTÚAN COMO OBSTÁCULOS.....	263
FIGURA 124 POTENCIA TRANSMITIDA POR UNA ANTENA EN 3D.....	265

ÍNDICE DE TABLAS

TABLA 1 VELOCIDADES DE TRANSMISIÓN E INFORMACIÓN EN GSM.	43
TABLA 2 POTENCIA DE TRANSMISIÓN DE UNA MS	50
TABLA 3 POTENCIA DE TRANSMISIÓN DE UNA ANTENA EN GSM900	51
TABLA 4 POTENCIA DE TRANSMISIÓN DE UNA ANTENA EN GSM1800	51
TABLA 5 POTENCIA EMITIDA POR UNA MICROCELDA	51
TABLA 6 PARÁMETROS GENERALES DEL FICHERO OMNETPP.INI.	91
TABLA 7 PARÁMETROS MÓDULO MOBILE STATION.	92
TABLA 8 TIPOS DE POSICIONES DE LOS MÓDULOS.	95
TABLA 9 VECTORES EN BSC.	124
TABLA 10 PARÁMETROS DE MS	140
TABLA 11 PARÁMETROS DE AIR	141
TABLA 12 PARÁMETROS DE BTS	141
TABLA 13 PARÁMETROS DE BSC	141
TABLA 14 PARÁMETROS DE MSC	142
TABLA 15 PARÁMETROS DE VLR	142
TABLA 16 PARÁMETROS DE LA RED	143
TABLA 17 PUERTAS DE LA MS	143
TABLA 18 PUERTAS DE AIR	143
TABLA 19 PUERTAS DE BTS	144
TABLA 20 PUERTAS DE LA BSC	144
TABLA 21 PUERTAS DE LA MSC	145
TABLA 22 PUERTAS DEL VLR	145
TABLA 23 CONEXIONES DE LA RED	145
TABLA 24 MENSAJES PROPIOS DE LA SIMULACIÓN	170
TABLA 25 MENSAJES INTERFAZ UM	176

TABLA 26 MENSAJES INTERFAZ ABIS	182
TABLA 27 INTERFAZ A	185
TABLA 28 INTERFAZ B.....	191
TABLA 29 INSTALACIÓN DE GLUT.	196
TABLA 30 COSTE DE MANO DE OBRA.	255
TABLA 31 COSTE DE MATERIAL	256
TABLA 32 COSTES TOTALES	257
TABLA 33 PRESUPUESTO	257

ÍNDICE DE ECUACIONES

ECUACIÓN 1 NÚMERO DE RADIOCANALES	39
ECUACIÓN 2 PARÁMETRO J.....	46
ECUACIÓN 3 PÉRDIDAS EN EL ESPACIO LIBRE	48
ECUACIÓN 4 ECUACIÓN DE PÉRDIDAS DE HATA BÁSICA	49
ECUACIÓN 5 CORRECCIÓN POR ALTURA PARA UNA CIUDAD PEQUEÑA.....	49
ECUACIÓN 6 CORRECCIÓN POR ALTURA PARA UNA CIUDAD GRANDE EN FRECUENCIAS < 200MHz.....	49
ECUACIÓN 7 CORRECCIÓN POR ALTURA PARA UNA CIUDAD GRANDE EN FRECUENCIAS > 400MHz.....	49
ECUACIÓN 8 PÉRDIDA BÁSICA EN UN ENTORNO SUBURBANO.....	49
ECUACIÓN 9 PÉRDIDA BÁSICA EN UN ENTORNO RURAL	49
ECUACIÓN 10 CÁLCULO DE NUEVA POSICIÓN DE UNA MS.....	117
ECUACIÓN 11 PÉRDIDAS EN EL ESPACIO LIBRE	126

1 RESUMEN

El proyecto se centra en la realización de un simulador para telefonía móvil que se pueda implantar en laboratorios del departamento y así servir como ayuda a los docentes a la hora de impartir asignaturas relacionadas con la telefonía móvil. La interfaz que se presenta es una interfaz sencilla que permite al usuario o al alumno actuar de una forma cómoda e intuitiva. Con cada simulación, el programa obtiene unos datos de salida que pueden ser llevados posteriormente a gráficas y visualizarse mediante herramientas de fácil manejo. Todo el desarrollo se ha realizado con unas librerías de programación basadas en OMNET++ y que se apoyan en el lenguaje C++, un lenguaje ampliamente extendido y bien conocido.

ABSTRACT

This Project is focused on the development of a Public Land Mobile Telephonic Network simulator which could be installed in the department laboratories, and in that way, serve as a helpful tool in the teaching of subjects related to mobile telephone technology. The interface is very simple, and it allows to the user or the learner to manage it in an intuitive and comfortable fashion. In each simulation, the program obtains output values which can be translated into graphics and be displayed with user-friendly tools. All developments have been

made with the programming libraries called OMNET++ which are based on the C++ programming language, a widely-used and well-known language.

2 INTRODUCCIÓN

OMNET ++ es un entorno de simulación de eventos discretos, basado en módulos, de código y arquitectura abierta, con un núcleo de simulación cuya principal área de aplicación es el desarrollo de redes de comunicaciones. Gracias a su flexible arquitectura y adaptabilidad ha sido aplicado con satisfacción en numerosos sistemas de telecomunicaciones como arquitecturas de colas, arquitecturas hardware, etc. Poco a poco OMNET ++ se está convirtiendo en una plataforma de simulación muy popular dentro de la comunidad científica así como para las empresas.

La gran ventaja que aporta para los centros educativos es que se trata de un entorno cuya licencia es gratis para uso académico y para uso sin ánimo de lucro. Su versión comercial, de pago y destinada exclusivamente a empresas, se denomina OMNEST, y se basa en los mismos elementos que la versión educativa, y se diferencia únicamente en la necesidad de comprar la licencia para uso comercial. La principal característica que aporta es la de ser una herramienta muy útil para la realización de simulaciones de redes y de visualización de resultados como gráficas y estadísticas basadas en parámetros que se incorporan a la simulación, y sobre todo, el entorno gráfico basado en ventanas y en aplicaciones visuales

permiten de una forma muy cómoda para el usuario poder visualizar el funcionamiento de una red.

Centrando la introducción en este entorno, OMNET ++ aporta una arquitectura basada en modelos. Estos modelos son programados individualmente en C++ a partir de las librerías y las clases ya existentes en el entorno, y a continuación embebidas en un módulo más grande o en una red usando un lenguaje de alto nivel denominado NED. El entorno de desarrollo incluye una librería de simulación basada en C++, un compilador para lenguaje NED denominado nedtool, un editor gráfico para archivos NED (GNED), una interfaz gráfica de usuario para la ejecución de la simulación y otra basada en línea de órdenes denominadas Tkenv y Cmdenv respectivamente, además de dos herramientas de visualización de resultados denominadas Plove y Scalars. Por último incluye un compilador para generar código C++ a partir de un fichero .msg. Los ficheros .msg son ficheros que contienen la descripción de los mensajes en el caso de que para la aplicación que se desarrolla se quieran realizar mensajes específicos y no utilizar los mensajes de la clase cMessage, sino nuevos tipos derivados de éstos.

Todos los ficheros escritos en NED se compilan mediante el compilador nedtool que genera un código C++. C++ es un lenguaje de alto nivel ampliamente usado desde su creación. Existen numerosos entornos de programación para poder desarrollar los proyectos basados en C, así como compiladores tanto de entorno visual como basados en comandos. Actualmente está muy extendido el uso del entorno de desarrollo Visual .NET de Microsoft, el cual se puede usar perfectamente para el desarrollo de este tipo de proyectos basados en OMNET ++, o incluso se pueden seguir utilizando versiones anteriores mucho más estables en el desarrollo de proyectos OMNET++ como el Visual C++ 6.0.

Una vez vistas las características que aporta OMNET ++, hay que hablar de las posibilidades que aporta poniendo como ejemplo algunas de las aplicaciones que se han realizado. Hasta la fecha se han publicado numerosos modelos de simulación de código abierto implementados con esta herramienta, tales como simulación de redes IP, IPv6, MPLS, Wi-fi, Ethernet, Token ring, FDDI. Muchos de estos ejemplos han sido publicados en el sitio web de la Comunidad de OMNET++¹ los cuales dan a conocer en algunos casos descripciones de los desarrollos y en

¹ <http://www.omnetpp.org>

otros permiten incluso descargar el ejecutable o los códigos fuente. Debido a su estructura puede ser aplicable a todo tipo de redes de pequeño o gran tamaño. Hay que añadir que el continuo desarrollo de las librerías y las sucesivas mejoras de las mismas junto con los compiladores nedtool y opp_msg están haciendo que el entorno sea cada vez más potente, más flexible y más sencillo de integrar con los compiladores existentes en la actualidad.

El principal propósito de este proyecto es la aplicación de este entorno a una red de telefonía móvil GSM², pudiendo crear así un entorno agradable para la actividad didáctica así como una interfaz cómoda para el usuario. Esta simulación se podría aplicar en la enseñanza de asignaturas del departamento en cuyo temario se incluya el estudio de redes de telefonía móvil. Además este proyecto podría ampliarse y mejorarse mediante nuevos proyectos para dar respuesta a nuevas necesidades, o para nuevas aplicaciones. Redes como las redes de telefonía móvil GSM, son redes muy complejas en las cuales hay un gran nivel de jerarquía, con un número muy grande de equipos y diferentes interfaces de comunicación entre los mismos. Para dar cuenta de su complejidad, además hay que añadir el número de servicios que debe proporcionar una red GSM.

Las redes GSM se estructuran de forma similar a la estructura de niveles OSI, con diferentes niveles. Estos niveles incluso dentro del mismo equipo pueden usar diferentes protocolos dependiendo del interfaz en el que actúen. La innovación que presenta este proyecto es que se permite la visualización de la estructura de la red así como el paso de mensajes mediante una interfaz visual muy sencilla además de presentar los mensajes de comunicación entre equipos, lo que permite a la hora de poder ser usado en un laboratorio o por un estudiante, un aprendizaje más cómodo, y un entorno de simulación más ligero e intuitivo que un analizador de protocolos.

Otra de las ventajas que se podría destacar de este entorno es su flexibilidad y escalabilidad. Al ser librerías basadas en OMNET++ y gratuitas para el entorno educativo, el profesor de una asignatura o el propio alumno o cualquier usuario del mismo, si tiene acceso al código

² GSM son las siglas de Global System for Mobile communications, nombre con el que se conoce a la telefonía móvil digital que sustituyó a la telefonía analógica en países como los europeos. Representa alrededor del 77% (datos de GSM World <http://www.gsmworld.com>) del mercado mundial de telefonía celular. Actualmente se está procediendo al paso a UMTS, siglas de Universal Mobile Transfer System, también conocido como telefonía móvil de tercera generación (3G).

fuelle del simulador GSM, podría modificarlo añadiéndole o quitándole elementos, mejorando partes o realizando ajustes a su gusto y antojo para poder dar lugar a una simulación ajustada a las necesidades específicas que se puedan dar en el laboratorio o a sus propias necesidades, y partiendo de una base inicial que sería este proyecto.

A lo largo del presente libro se pretende dar unas pequeñas nociones de programación con las librerías de este entorno. Sin embargo para obtener un mayor conocimiento del mismo conviene que se acuda directamente a los manuales que se publican en el sitio oficial y que son actualizados junto con las nuevas versiones de las librerías. El libro se estructura principalmente en dos partes, un manual de desarrollador, y un manual de usuario. En caso de querer usar la aplicación sin llevar a cabo modificaciones, y simplemente trabajar con ella se recomienda que se pase directamente a la parte del libro que se dedica al manual de usuario. Si por el contrario lo que se pretende como objetivo es utilizar la aplicación y además poder modificar el programa realizado, lo aconsejable es que se introduzca en las páginas del manual de desarrollo, donde se encontrará toda la información sobre el desarrollo de la aplicación. Si se carece de conocimientos de OMNET++ y su entorno, lo mejor es que empiece jugando con la aplicación siguiendo los consejos del manual de usuario y después se adentre en los capítulos referentes a la descripción del desarrollo.

2.1 Definiciones

A continuación se mencionan una serie de términos que van a ser usados a lo largo de la memoria. Son siglas correspondientes a términos técnicos con las que tal vez no se este familiarizado, por lo que aquí se pretende presentar el significado de estas siglas.

AGCH	Access Grant Channel. Tipo de canal.
AuC	Authentication Centre. Equipo de autenticación.
BCCH	Broadcast Control Channel. Canal de control de difusión.
BER	Bit Error Rate. Tasa de error de bit.
Bm	Full-rate traffic channel. Canal de tráfico full-rate.
BN	Bit Number. Número de bit.
BSC	Base Station Controller. Controlador de estación base.
BSIC	Base transceiver Station Identity Code. Número de identificación de BTS.
BSS	Base Station System. Sistema de estación base.
BSSAP	Base Station System Application Part. Parte de aplicación de BSS.

BSSMAP	Base Station System Management Application Part. Parte de aplicación de gestión de BSS.
BSSOMAP	Base Station System Operation and Maintenance Application Part. Parte de operación y mantenimiento de BSS.
BTS	Base Transceiver Station. Estación base transceptora.
DCCH	Dedicated Control CHannel. Canal de control dedicado.
DCS1800	Digital Cellular System at 1800MHz. Sistema digital celular en la banda de 1800MHz.
DL	Downlink. Enlace descendente entre BTS y MS.
DTAP	Direct Transfer Application Part. Parte de aplicación de transferencia directa.
ETSI	European Telecommunications Standards Institute.
FACCH	Fast Associated Control Channel. Canal de control asociado rápido.
FACCH/F	Fast Associated Control Channel/Full rate. Canal de control asociado rápido full-rate o velocidad completa.
FACCH/H	Fast Associated Control Channel/Half rate. Canal de control asociado rápido half-rate o mitad de velocidad.
FCCH	Frequency Correction Channel. Canal de corrección de frecuencia.
GMSC	Gateway Mobile-services Switching Centre. Puente MSC.
GSM	Global System for Mobile communications.
GSM MS	GSM Mobile Station. Estación móvil de GSM.
GSM PLMN	GSM Public Land Mobile Network. Red telefónica móvil pública terrestre GSM.
HLR	Home Location Register.
IMEI	International Mobile station Equipment Identity
IMSI	International Mobile Subscriber Identity
LA	Location Area
LAC	Location Area Code
LAI	Location Area Identity
LAN	Local Area Network
LAPDm	Link Access Protocol on the Dm channel
MAP	Mobile Application Part
MS	Mobile Station. Estación móvil.
MSC	Mobile-services Switching Centre, Mobile Switching Centre
O&M	Operations & Maintenance
PCH	Paging CHannel
QOS	Quality Of Service
RACH	Random Access CHannel
RAND	RANDom number (used for authentication)
REJ	REJect(ion)
REL	RELease
REQ	REQuest
SACCH	Slow Associated Control CHannel
SACCH/C4	Slow Associated Control CHannel/SDCCH/4
SACCH/C8	Slow Associated Control CHannel/SDCCH/8

SACCH/T	Slow Associated Control CHannel/Traffic channel
SACCH/TF	Slow Associated Control CHannel/Traffic channel Full rate
SACCH/TH	Slow Associated Control CHannel/Traffic channel Half rate
SCCP	Signalling Connection Control Part
SCH	Synchronization CHannel
SDCCH	Stand-alone Dedicated Control CHannel
SS7	Signalling System No. 7
TCH	Traffic Channel. Canal de tráfico entre MS y BTS.
TCH/F	A full rate TCH. Canal TCH de velocidad maxima.
TCH/H	A half rate TCH. Canal TCH de velocidad mitad.
TDMA	Time Division Multiple Access.
TE	Terminal Equipment
TMSI	Temporary Mobile Subscriber Identity
TRX	Transceiver. Equipo transceptor en la BTS.
TS	Time Slot
UL	Uplink. Enlace descendente entre MS y BTS.
VLR	Visitor Location Register

3 ESTADO DEL ARTE

En la actualidad existen diferentes entornos en los que se pueden implementar simulaciones de redes, y que poseen características similares a OMNET++. Entre ellos podemos destacar ns2, un simulador bien establecido en la realización de simuladores de redes TCP-IP y Wireless, OPNET Modeler, una plataforma comercial muy parecida a OMNET++ utilizada para varios tipos de redes, J-Sim, un simulador basado en Java, y por último GloboSim, unas librerías usadas para diferentes propósitos.

Si comparamos OMNET++ con los anteriores, además de contar con la ventaja de que es totalmente gratuito para uso académico como GloboSim, es quizás el más flexible. Permite el desarrollo de simulaciones con modelos modulares, añadiendo la posibilidad de que los módulos compuestos pueden tener cualquier nivel de jerarquía. En cualquier caso, las clases bases pertenecientes al kernel del simulador pueden ser modificadas o incluso derivadas cómodamente para permitir al desarrollador crear un entorno a su total comodidad.

Otra ventaja que se puede destacar es que añade junto con su entorno, un conjunto de librerías que permiten escribir datos sobre la simulación. Estos datos pueden ser después tratados y a continuación visualizados en pantalla mediante las herramientas estadísticas que añade, o

incluso pasados a imagen o añadidos a documentos de texto. Esta característica le añade además la propiedad de poder obtener gráficos muy cómodamente para que puedan ser añadidos con posterioridad a memorias de trabajo.

La última de las características que se quiere mencionar es el lenguaje de programación que se emplea. Todo el código generado para cada uno de los módulos incluidos en la simulación se debe desarrollar en lenguaje C++, un lenguaje muy extendido, y conocido por todos los profesionales del sector además de ser conocido por los alumnos a los que puede ir destinado este proyecto. Esto permite, especialmente en el caso de los alumnos, que no sea necesario el aprendizaje de un lenguaje nuevo y complicado para aplicarlo exclusivamente a la simulación de redes, sino centrarse en la realización del simulador.

En cuanto al estado del arte en simuladores GSM, apenas se encuentran en la actualidad simuladores que permitan al usuario flexibilidad, comodidad, sencillez, y a la vez un amplio abanico de posibilidades de simulación. Cabe mencionar que en el mercado sí que se encuentran simuladores destinados a entornos empresariales. Sin embargo en el área en el que se puede emplear este proyecto, que es el área académica, las posibilidades son muy reducidas. Dentro de los proyectos basados en OMNET++ y publicados en su página web, se han desarrollado aplicaciones para diversos tipos de redes, incluso de arquitecturas hardware, o sensores, sin que se haya realizado un simulador para GSM con unas características adecuadas para su uso en un entorno académico. La única mención que se realiza sobre telefonía móvil para este entorno, es un simulador realizado por el Instituto Tecnológico de Budapest en el año 1999. Aparte de ser un simulador muy poco potente con muy pocas posibilidades³ y muy poca flexibilidad, actualmente no se encuentra soportado. Hay que añadir que debido al año en que fue realizado, se encuentra anticuado con respecto a las librerías⁴ actuales de OMNET++, ya que emplea características de éstas que han sido totalmente modificadas en la actualidad⁵.

³ Este simulador con unas características muy limitadas, simplemente permite la introducción de un número de canales y la obtención de estadísticas sobre número de llamadas.

⁴ En el momento de escribir esta memoria la última versión que se distribuye es la 3.2

⁵ Una de las características que hacen que el simulador desarrollado en Budapest no pueda seguir siendo soportado es que basa el tratamiento en la función `activity()` en vez de en la función `handleMessage()`. Aunque no se han mencionado hasta ahora, se verán más

4 OBJETIVOS DEL PROYECTO

El objetivo principal del proyecto es el desarrollo de un pequeño entorno de simulación para una red GSM, que sea sencillo para el usuario, y que además aporte las características de fácil modificación y ampliación. Este proyecto podría ser a su vez una primera parte de un proyecto mucho mayor que sería una red GSM completa incluyendo desde los equipos del nivel jerárquico más bajo hasta los del nivel más alto, incluyendo la simulación de los diferentes niveles de cada equipo. La razón de que la realización de un simulador completo para GSM se deba llevar a cabo en diferentes fases es que este tipo de red es muy compleja, con un número muy grande de equipos. Además, con este proyecto se cubre la parte de la red correspondiente a la antena y la estación base de control, conocido como subsistema base,

adelante. La primera, se sigue añadiendo en las librerías simplemente por compatibilidad con las primeras versiones. En el manual se insta al desarrollador a no usarla salvo que no sea estrictamente necesario debido a los problemas que causa y a la gran memoria que necesita para ser empleada. Ambas funciones, pese a tener la misma funcionalidad, no son en ningún caso compatibles. La consecuencia es que si se quiere pasar una aplicación desarrollada con `activity()` a `handleMessage()` será necesario reescribir toda la aplicación.

junto con el equipo móvil (MS), e incluso se introduce una MSC sencilla que pertenece al subsistema de señalización y conmutación, asociada a un VLR o base de datos local.

Como se ha mencionado antes, otro de los objetivos es que sea fácilmente modificable y ampliable. Esto se consigue gracias a las características que aportan las librerías de OMNET++. ¿Cuáles son las ventajas? Pues la principal que puede tener con respecto a otros simuladores del mercado es que al ser una aplicación sencilla, basada en código libre, un profesor o un profesional con conocimientos sobre estas librerías puede, partiendo de la base que aporta este proyecto, adaptar éste para objetivos específicos, ya sea el de orientarlo a una determinada actividad didáctica, o el de suprimir ciertas partes que no le sean necesarias para hacer una aplicación más rápida, etc.

Debido a su sencillez, y también a la gran cantidad de cómputo que puede suponer la simulación de una red muy grande, esta versión del proyecto no sería aplicable a un entorno empresarial. Hay que recordar que los ordenadores sobre los que se ejecuta esta aplicación son ordenadores de tipo PC, los cuales poseen unas características muy limitadas para grandes tareas de cómputo. Este proyecto sí que puede ser modificado y mejorado para aplicarse a entornos empresariales. Sin embargo cabe recordar que la finalidad no es la aplicación a empresas sino que el campo de aplicación son los entornos educativos y académicos.

El método que se va a emplear para el desarrollo de la red es realizar un módulo para cada equipo, que son el Terminal móvil (MS), la estación base (BTS), y la estación base controladora (BSC). Todos estos módulos desarrollados, están integrados formando una pequeña red. En ésta, los equipos se comunican mediante mensajes a partir de la clase cMessage aportada por el entorno OMNET. Cada mensaje, está identificado con un número, que permite distinguirlo, y así, realizar para cada módulo funciones específicas para cada mensaje. La razón de asignar una función a cada mensaje es la de permitir una estructura de programación por módulos, haciendo más sencillo para otros desarrolladores su modificación.

5 BASE TEÓRICA. DESCRIPCIÓN DE UNA RED GSM

5.1 Introducción

Desde que se comenzó a implantar la telefonía, ha supuesto una auténtica revolución por el ser humano. Gracias al teléfono, las comunicaciones ganaron en velocidad y en calidad. Fue años después, en la década de 1960 cuando las radiocomunicaciones móviles comenzaron a desarrollarse con intensidad. Las comunicaciones por radio han aportado a la humanidad la movilidad, la posibilidad de desplazarse a la vez que pueden comunicarse. Hoy día se puede hablar con cualquier persona en cualquier parte del mundo.

Conforme ha ido implantándose esta tecnología el número de servicios ofrecidos ha ido en un crecimiento continuo, desde el establecimiento de llamadas hasta las ofertas de servicios que incluyen los móviles de última generación como videoconferencia, y transmisión de datos en banda ancha.

Las redes de telefonía móvil son redes tremendamente complejas. A la complejidad de establecer el canal radio, hay que añadir la movilidad de los usuarios, y también la movilidad

del entorno en que se encuentra el equipo móvil. Es necesario garantizar una buena calidad de cobertura, independientemente de la posición del usuario, de su velocidad de movimiento o de posibles obstáculos móviles que pueden ocultar la visión de una antena. Además hay que tener en cuenta la potencia de transmisión para evitar interferencias, y la batería limitada con que cuentan los equipos móviles. A las características del canal radio se suma la gestión de canales y servicios llevada a cabo por equipos superiores en la jerarquía de la red. Además se necesita la autenticación y comprobación de usuarios, por lo que se deben incluir registros o bases de datos en la que aparezca almacenado los datos de usuario y su clave de acceso a la red. También se debe proporcionar itinerancia con otras redes o roaming. Esto quiere decir que un usuario de España puede usar su móvil igualmente en Francia, Alemania, Austria, Suecia, etc.

En definitiva, las redes de telefonía móvil han sido una gran revolución para las comunicaciones desde que comenzaron a implantarse. Su crecimiento espectacular (ver Figura 1) y su bajo coste de los terminales para el usuario han permitido que un gran número de usuarios sea capaz de acceder a este servicio, llegando incluso a desbancar en número de líneas a la telefonía fija.

GSM regional statistics

Millions	2002				2003				2004				2005			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
World	676.1	714.6	758.0	809.3	851.5	894.7	945.4	1,012.0	1,070.9	1,131.6	1,207.2	1,296.0	1,378.2	1,467.1	1,560.7	-
Africa	27.8	30.3	33.0	36.1	39.6	43.1	47.0	52.3	57.5	62.2	68.9	78.4	88.3	97.5	111.9	-
Americas	3.5	4.0	6.1	9.5	11.9	15.4	20.0	29.0	35.6	43.7	53.1	66.7	87.7	104.0	117.4	-
Asia Pacific	246.4	268.6	290.8	314.0	334.1	352.2	370.3	394.6	419.9	443.4	470.8	494.4	524.5	556.3	584.0	-
Europe: Eastern	52.6	58.4	65.0	73.0	79.3	86.5	96.3	108.5	118.7	132.5	150.3	174.6	194.0	213.9	236.2	-
Europe: Western	291.5	295.2	300.9	310.0	314.6	319.9	328.2	337.0	342.1	346.1	352.6	361.7	366.9	370.3	375.7	-
Middle East	34.8	37.3	39.6	41.8	43.9	46.6	49.6	52.8	55.8	59.4	63.5	67.6	73.0	79.3	85.3	-
USA/Canada	19.5	20.8	22.5	24.9	28.1	30.9	34.1	37.8	41.2	44.3	48.0	52.6	43.7	45.7	50.2	-

Source: Wireless Intelligence

Figura 1 Tabla de usuarios de GSM en el mundo de 2002 a 2005.

El número de usuarios de GSM no ha dejado de crecer desde el comienzo de su implantación. Se puede en la tabla como se ha duplicado el número de usuarios desde el primer cuatrimestre de 2002 hasta el primer cuatrimestre del 2005. Fuente: GSM World <http://www.gsmworld.com>

5.2 Red telefónica pública móvil

Como se ha comentado anteriormente, una de las principales características de la red telefónica móvil pública es la capacidad para permitir movilidad al usuario sin que para ello

se vea impedido de hacer uso de la red. Para ello es necesario establecer una infraestructura de estaciones. Esta infraestructura deberá dar cobertura a todo un país. En este tipo de telefonía se entienden dos tipos de enlaces radio, el enlace ascendente, de móvil a antena, y descendente de antena a móvil. La red debe además conocer en todo momento la localización exacta de un equipo móvil para poder encaminar las llamadas. Gracias a la movilidad, los números de teléfono se asocian a una persona, liberándose de la asociación espacial.

Dentro de la PLMN existen diferentes equipos. El centro de conmutación recibe el nombre de MSC, similar a los centros de conmutación de las redes fijas, aunque con funciones mucho más amplias que éstos. Un MSC a su vez tiene bajo su control varias estaciones base y controla a los equipos móviles que se encuentren bajo su área de influencia.

5.2.1 Historia

Desde sus inicios, la telefonía móvil atrajo a un gran número de usuarios, lo que provocó que fuese necesario entorpecer mediante precios disuasorios para los potenciales clientes. La tecnología de las PLMN de primera generación fue analógica, lo cual limitaba enormemente la capacidad y el crecimiento, además de varios problemas en el empleo de espectros de frecuencias. Para solventar estos problemas se creó en 1982 el grupo denominado GSM (Groupe Spéciale Mobile) para crear un estándar europeo para las redes PLMN.

La norma estableció dos bandas de frecuencias para GSM, una para el enlace ascendente y otra para el descendente, en la banda de 890 – 915 y de 935 – 960 MHz. Además se decidió que la norma tendría diferentes fases, en las cuales se fuesen implantando nuevas prestaciones. Entre los objetivos que se buscaban fueron crear un estándar europeo, interoperabilidad de equipos, interfuncionamiento con redes fijas existentes, oferta de gran cantidad de servicios, elevadas prestaciones de cobertura y calidad de servicio y ser económicamente rentable y atractiva para operadores y usuarios. El nombre del grupo, dio el nombre a la tecnología que crearon, pasándose a llamar GSM o Global System for Mobile Communications.

Según los últimos análisis de consumo y extensión de la red telefónica pública móvil, esta tiene una mayor implantación que su hermana mayor la red fija, convirtiéndose en un sector muy demandado y muy competitivo, en el que crecen a gran velocidad el número de servicios implementados y sus características. Debido al crecimiento de esta tecnología fue necesario ampliar la banda de frecuencias, y se creó lo que se conoce como DCS, que es la telefonía

móvil GSM en la banda de los 1800MHz.

El estándar GSM ha sido implantado en más de un centenar de países alrededor del mundo. Actualmente existen más de mil millones de usuarios GSM, con un crecimiento espectacular. La siguiente generación de telefonía móvil se llama UMTS o Universal Mobile Telecommunications System. Toma de GSM la mayoría de sus características, aunque aporta una nueva tecnología de acceso radio que permite obtener un mayor ancho de banda.

5.2.2 Servicios

El estándar GSM define una red telefónica móvil digital, con estructura celular en el interfaz radio, con funciones de movilidad, y con seguridad para la protección de datos. Los servicios ofrecidos abarcan un amplio espectro, siendo los servicios básicos telefonía y datos. El servicio telefónico es similar al de las redes fijas. El servicio de datos permite la comunicación de datos a velocidades de datos comprendidas entre 300 y 9600 bit por segundo. También se incluye un servicio de mensajes cortos o SMS permite el envío de mensajes de hasta 160 caracteres entre usuarios de la red.

5.2.3 Jerarquía

Al igual que toda red de comunicación, GSM se basa en una jerarquía de varios niveles. La zona celular es la zona más baja de la jerarquía, en la cual se encuentran los usuarios con sus terminales móviles y una estación base. El nivel MSC/VLR abarca el control de un área de localización generalmente. Por encima se encuentra la zona de servicio de un operador de PLMN-GSM.

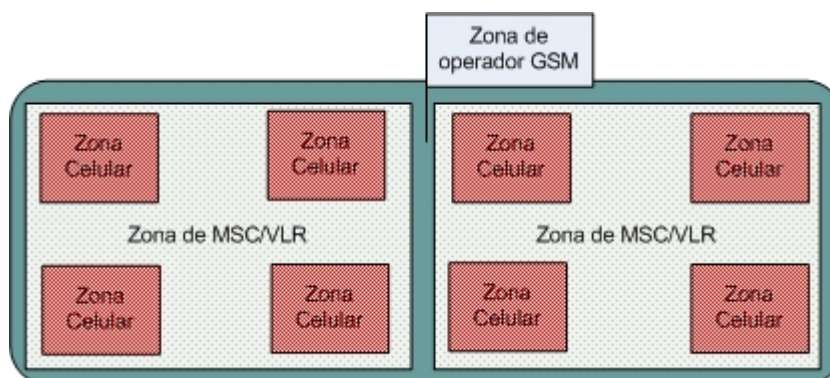


Figura 2 Jerarquía en GSM.

En la imagen se puede ver cada zona de la jerarquía GSM. El nivel más alto lo ocupa el operador de la red (en realidad el nivel más alto es la red GSM mundial, pero por simplificar se tiene en cuenta sólo la red de un operador). El más bajo es la zona celular en la que se encuentran las estaciones base y los equipos móviles.

5.2.4 Numeración

Como cualquier otra red, es necesario establecer números de identificación para los usuarios, y para permitir el encaminamiento de llamadas. Principalmente son los siguientes:

- Número de identificación de usuario
- Número asociado al equipo móvil
- Número telefónico del abonado

A cada abonado se le asigna un número de identificación llamado IMSI, un número interno de la red para el acceso a las bases de dato de usuario. Para evitar que este número sea captado se emplea un número temporal denominado TMSI. Cada equipo móvil tiene un número propio llamado IMEI, para validación del Terminal. Por último, cada abonado tiene asociado un número telefónico o MSISDN.

Las áreas de localización son áreas formadas por una o varias células. Un área de localización puede ocupar una zona MSC/VLR, o compartir con otra u otras áreas de localización (LA) la zona MSC/VLR. Las LA se identifican mediante un número denominado LA code en siglas LAC.

5.2.5 Arquitectura gsm

5.2.5.1 Subsistemas

La red GSM se divide en tres principales subsistemas:

- subsistema de estaciones base
- subsistema de conmutación y gestión
- subsistema de operación y mantenimiento

Cada subsistema está formado por uno o varios equipos que se comunican a través de varios interfaces (definidas por letras) mediante determinados protocolos. Las MS o estaciones

móviles no se consideran parte de la red, pero si que tienen un interfaz común con la red, denominado interfaz aire o interfaz Um. El interfaz entre el subsistema de estaciones base (BSS) y el subsistema de conmutación y gestión (SMSS) recibe el nombre de interfaz A.

El subsistema de estaciones base comprende las funciones de la capa física del nivel OSI para la conexión con los equipos móviles. La capa física se divide en canales físicos, que a su vez se dividen en canales lógicos para el intercambio de información. Los equipos principales son la BSC o Base Station Controller y la BTS o Base Transceiver Station cuya interfaz se denomina Abis.

El subsistema de conmutación y gestión se encarga de la gestión y control de todas las funciones para el manejo de protocolos de señalización, para el establecimiento de llamadas, y para la movilidad. Las funciones básicas son localización, registro de abonados, encaminamiento, gestión de recursos radio, tratamiento de llamadas y trasposos. Los equipos que lo constituyen son el MSC o Mobile Switching Center, el HLR o Home Location Register, y el VLR o Visitor Location Register.

El subsistema de operación y mantenimiento gestiona la red y la seguridad de equipos y usuarios. Para ello se disponen de tres equipos, el OMC u Operations and Maintenance Centre, el AuC o Authentication Centre, y el EIR o Equipment Identity Register.

5.2.5.2 Equipos

Ya se han mencionado por encima los nombres de los principales equipos que integran una red de telefonía GSM. Ahora se va a explicar cual es su cometido dentro de la red.

5.2.5.2.1 Estación Móvil

Con estación móvil se denomina al equipo físico del usuario que emplea el interfaz Um para acceder a la red GSM. Proporciona una interfaz de comunicaciones entre usuarios y vía radio, y establece la transmisión de información de usuario entre usuario y red. Además realiza la sintonización de frecuencias y el seguimiento de las estaciones base. También adapta velocidades de datos y procesa la voz de analógico a digital y de digital a analógico. La norma establece diferentes tipos de estaciones móviles en cuanto a sus características. Estas características difieren en cuanto a potencia de transmisión y sensibilidad de recepción, además de características de batería. Las estaciones móviles más limitadas son los teléfonos

móviles. Su peso y tamaño exigen una batería muy limitada en cuanto a peso, tamaño y por tanto capacidad.

5.2.5.2.2 BTS

La BTS son los equipos formados por los transmisores – receptores de radio, los elementos de conexión al sistema, las antenas, y las instalaciones accesorias como tomas de tierra, pararrayos, etc. El número de este tipo de equipos en una red es muy elevado, por lo que deben de ser sencillo, económicos y fiables. Además debido a la imposibilidad de tener un sistema de gestión destinado a cada BTS, es necesario que su gestión y mantenimiento se pueda realizar a distancia.

5.2.5.2.3 BSC

La BSC es el equipo que controla una o varias estaciones transceptoras BTS. Se compone de dos interfaces, el A y el Abis. Las principales funciones son la transmisión y la recepción, la localización de las estaciones base, el establecimiento de llamadas y liberación, el traspaso dentro de su área de influencia (dentro de la misma BSC) y mantenimiento de estaciones BTS.

5.2.5.2.4 MSC/VLR

En realidad son equipos independientes, sin embargo su funcionamiento está totalmente ligado por lo que en la mayoría de los casos están integrados. Son imprescindibles para la existencia de la red GSM. El primero es el Mobile Switching Center, es decir el nodo que contiene las funciones de conmutación y señalización básica, donde su principal misión consiste en la gestión completa de las llamadas desde y hacia usuarios GSM.

El VLR responde a las siglas de Visitor Location Register, y se trata de una base de datos en la que se guarda información temporal de cada cliente que se encuentra en el área de influencia del MSC al que está conectada. Las especificaciones GSM permiten que un VLR esté asociado a un único MSC o a varios.

El VLR intercambia información con el HLR, guardando información de IMSI o TMSI, datos de encaminamiento, datos de suscripción, servicios contratados, seguridad, tripletas de autenticación, etc.

Para evitar el intercambio excesivo de información entre VLR y MSC, normalmente se

fabrican integrados para evitar así la necesidad de señalizadores intermedios. Otra ventaja es que cada fabricante puede definir protocolos propietarios variantes de MAP. La MSC tiene una parte destinada exclusivamente a la realización de traspasos entre dos BSS conectadas y que recibe el nombre de HOCA (Handover Control Application). También se debe proporcionar el control de autenticación y de actualización de posición de los móviles, la prestación de servicios suplementarios, tarificación, etc.

La cobertura de acción de un MSC puede ser muy diversa y se denomina área MSC. Un área MSC puede estar formada por una o más áreas de localización, aunque determinadas por la cantidad de usuarios. A la hora de elegir entre MSC grandes o pequeñas deben sopesarse las ventajas económicas asociadas a las de mayor tamaño frente al peligro que supondría que uno de ellos quedara fuera de servicio y los clientes de su VLR no pudieran ni realizar ni recibir llamadas mientras durase la avería.

Gracias al almacenamiento de los datos en el VLR de área de localización, identidad de abonado, número telefónico y número de encaminamiento, la red puede establecer llamadas hacia un móvil destino. El equipo MSC comunica al equipo de nivel superior denominado GMSC que quiere establecer una llamada. Éste avisa al HLR para conocer dónde se encuentra el móvil destino, que le responde con el área de localización. Conocido el área de localización, la GMSC puede establecer una comunicación con el MSC al que pertenece el área de localización, e indicarle que tiene una llamada para un equipo presente bajo su área. Éste se comunica con su VLR y envía un mensaje de búsqueda hacia el móvil destino. Hay que tener en cuenta que el VLR tiene un número de abonados que puede gestionar limitado, por lo que este factor debe estudiarse a la hora de planificar una red.

5.2.5.2.5 GMSC

En las redes GSM existe un equipo denominado Gateway Mobile Switching Centre. Se encarga de la gestión de comunicaciones entre diferentes MSC y entre la red móvil y la red fija u otras redes. Hay dos tipos de configuraciones, una en la cual existe un único GMSC o un número muy pequeño que controla a todos los MSC, y otra configuración en la cual todos los equipos tienen la propiedad de actuar como GMSC. La diferencia principal entre un GMSC y un MSC es que son capaces de encaminar hacia otras redes, y además de hacer consultas directamente a un HLR.

5.2.5.2.6 HLR

Se trata de una base de datos inteligente en la que se guarda información estática relativa al servicio de todos los clientes de la red GSM perteneciente a un operador y también información dinámica de los mismos. Puede haber uno o varios, sin embargo lo más habitual es tener varios en una misma red y replicados debido a que tienen una limitada capacidad y a la necesidad de una alta fiabilidad de los mismos. En una red GSM establecida por un operador en un país con un número de habitantes como el de España sería necesario establecer varios equipos HLR, aunque si se tratase de Andorra, quizás con un equipo y otro de réplica sería suficiente. Generalmente se cifra el tamaño límite en unos 500000 usuarios por HLR.

5.2.5.2.7 AuC

Se conoce como authentication centre al equipo que se encarga de almacenar las claves de usuario. Debido a la seguridad que necesitan estos datos, se almacenan en un equipo diferente con unas características de seguridad adicionales al resto de la red.

5.2.5.2.8 EIR

Equipo que se encarga de almacenar la información acerca de un terminal. Los terminales se pueden encontrar en tres listas, una lista blanca, negra o gris. La lista blanca es para aquellos equipos que cumplen todas las características de la red y pueden conectarse sin ningún tipo de problema. Los equipos en la lista gris han tenido fallos, o dan problemas a la red, por lo que tienen un acceso restringido. Y por último, se encuentran los equipos en la lista negra, que tienen prohibido el acceso, ya sea por un mal funcionamiento, o por que se trate de un equipo robado.

5.2.5.2.9 OMC

Se encarga de supervisar la gestión de la red con tareas técnicas y administrativas. Pueden ser funciones de gestión de abonados, de seguridad de la red, de explotación y funcionamiento de la red, de control del sistema, o de mantenimiento.

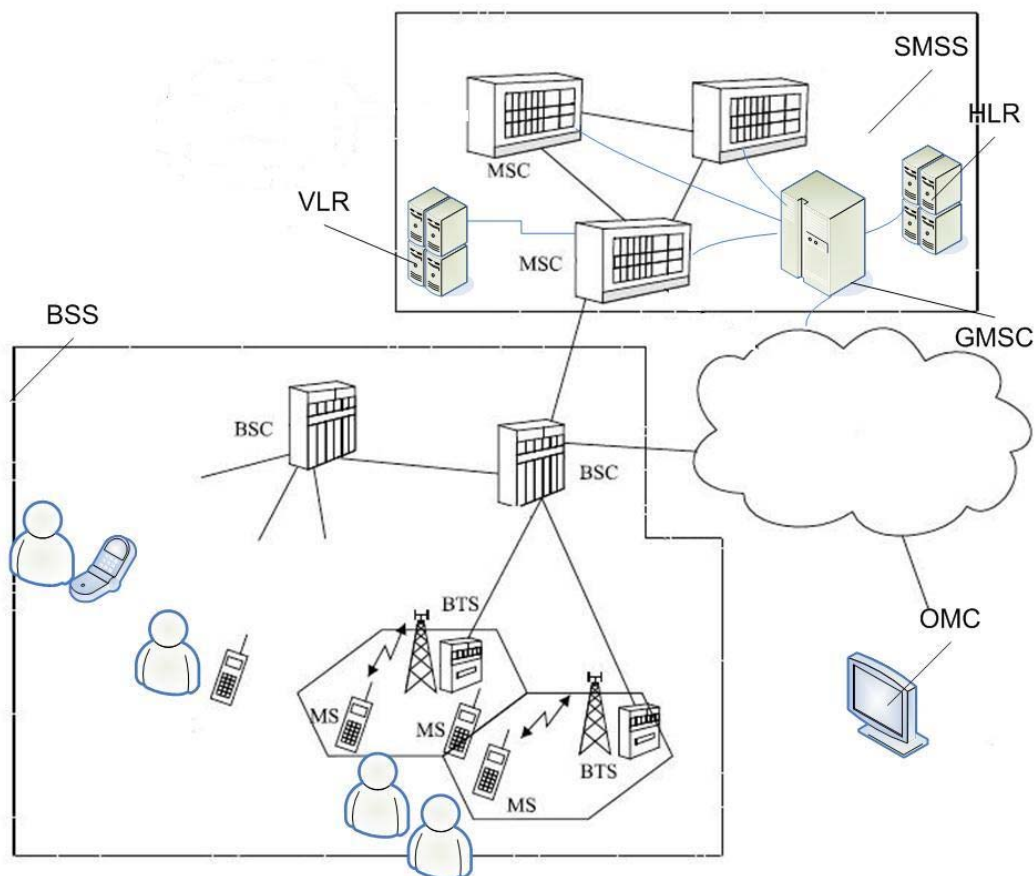


Figura 3 Equipos en una red GSM.

La red GSM se subdivide básicamente en tres subsistemas, el subsistema base (BSS), el subsistema de conmutación (SMSS) y el subsistema de operación y mantenimiento (OMC).

5.2.5.3 Interfaces del servicio GSM

En una red GSM existen diferentes tipos de interfaces entre equipos, y a su vez existen diferentes protocolos que actúan en cada nivel de la red de comunicación. La torre de protocolos de GSM sigue una estructura OSI, en la cual se tienen diferentes capas físicas, debido a los diferentes medios de transmisión que se emplean. Por ejemplo, en el interfaz entre MS y BTS, el medio empleado es el radio, mientras que entre BTS y BSC la decisión depende de la distancia entre éstas. Por ejemplo, si BTS está muy alejada de BSC y no se dispone de una infraestructura cableada, lo más rentable económicamente será establecer un enlace radio entre BTS y BSC, mientras que si está cerca o se tiene cableado se usará este sistema. A la hora del desarrollo de la simulación no se tiene en consideración la diferencia

entre estos dos tipos de enlaces, por lo que en la simulación no se distinguirá entre conexiones radio o cableadas en el interfaz Abis. Este aspecto no es de gran importancia para la simulación ya que sea de una forma o de otra, la implementación del sistema es muy parecida, la única diferencia a tener en cuenta es que la probabilidad de error de bit será mayor en el interfaz radio que en cableado, aunque no se tendrá en cuenta.

5.2.5.3.1 INTERFAZ A (MSC-BSC)

Intercambio de información sobre gestión de BSS, de llamadas y de movilidad de los móviles. A través de esta interfaz se negocian los circuitos a utilizar entre BSS y MSC. Se establecen 32 canales de comunicación a 64Kbit/s por lo que se necesita un enlace de 2Mbit/s. Generalmente el enlace se establece por una conexión cableada.

5.2.5.3.2 INTERFAZ Abis (BSC-BTS)

Es el interfaz establecido entre BTS y BSC. Como se ha dicho antes puede ser cableado o por conexión radio. La decisión se toma en función de la distancia entre ellos. Generalmente en zonas rurales se opta por enlaces radio, mientras que en las zonas urbanas es mucho más fácil el establecimiento de cables. Una de las funciones de este interfaz es el control de la antena a distancia, otra es el envío de información de la MSC a la estación móvil en algunos casos de forma transparente.

5.2.5.3.3 INTERFAZ B (MSC-VLR)

Unión entre la base de datos y el nodo de conmutación de la red GSM. Comúnmente se implementan integrados ya que son dos módulos que están directamente relacionados. Además esto asegura a las empresas poder realizar equipos con interconexiones específicas. Según las recomendaciones, los equipos MSC y VLR deben llevar unas características determinadas. Sin embargo en la realidad cada fabricante adopta unas características propias para sus equipos. En la simulación se pretende implementarlo en un módulo complejo con dos módulos simples MSC y VLR interconectados.

5.2.5.3.4 INTERFAZ C (HLR-GMSC)

La HLR es una base de datos que almacena los registros de los usuarios de la red. Este equipo es consultado por la GMSC cada vez que una MSC o la propia GMSC necesita información sobre el MSRN del móvil llamado para conocer su localización y así encaminar la llamada hacia el MSC destino. No se implementará en la simulación actual.

5.2.5.3.5 INTERFAZ D (HLR-VLR)

Se emplea para el intercambio de información entre las bases de datos de información para datos relativos a la posición del móvil e información de servicios contratados. Tampoco se implementará para esta simulación ya que el objetivo de ésta es desarrollar hasta la MSC.

5.2.5.3.6 INTERFAZ E (MSC-MSC)

Enlace destinado a la gestión de traspasos interMSC. Sirve para comunicar información de traspasos o encaminamiento de llamadas. En algunos casos todas las MSC se implementan como GMSC de forma que tienen la posibilidad de encaminar con otras redes y consultar a la HLR.

5.2.5.3.7 INTERFAZ F (MSC-EIR)

Es un enlace destinado a la conexión del equipo de conmutación y el registro de identificación de equipos. Sirve para evitar el acceso a la red de equipos a los que no se les está permitido el acceso. Los equipos se clasifican en tres listas, la blanca para equipos aptos, la gris para equipos con pequeños problemas, y negros para los que tienen denegado el acceso. No se implementará para esta versión del programa de simulación

5.2.5.3.8 INTERFAZ G (VLR-VLR)

En caso de que un móvil inicie la petición de actualización de posición en un nuevo VLR utilizando el TMSI se usará este canal. Siempre que le sea posible, el nuevo VLR obtiene el IMSI y las tripletas de autenticación a partir del antiguo VLR. En un principio no se va a introducir.

5.2.5.3.9 INTERFAZ H (HLR-AuC)

Es la interfaz utilizada por el HLR para solicitar las tripletas al AuC cuando no dispone de ellas. El objetivo de introducir este equipo en la red es el de aumentar la seguridad. Los datos de claves son almacenados por un equipo al que sólo se puede acceder desde el HLR, aumentando la seguridad.

5.2.5.3.10 INTERFAZ Um (BTS-MS)

Este es el interfaz que se ha implementado en el programa de simulación para la conexión entre la estación móvil y la antena. Es un interfaz radio, dividido en canales físicos y canales

lógicos. Los canales lógicos se dividen a su vez en canales de tráfico o de señalización. Cada portadora lleva 8 canales (cada TRX), de los cuales se reserva como mínimo uno para señalización y el resto para tráfico.

5.2.6 Señalización

En GSM se establecen dos sistemas de señalización. Para la subred de mantenimiento y la de conmutación se emplea el sistema de señalización SS7, mientras que para la MS y el subsistema BSS se emplea un sistema de señalización específico.

Se han definido diferentes partes de aplicación, el BSSAP para BSS, y DTAP, parte del MAP para SS7. El protocolo de señalización tiene relación con los niveles 1 a 3 del modelo OSI. Tiene una capa física, una capa de enlace que emplea los protocolos LAPD y LAPDm, y una capa de red que se divide a su vez en tres subcapas, la capa de gestión de llamadas CM, la capa de gestión de la movilidad MM, y la capa de gestión de recursos radio RR.

Debido a que se emplean diferentes tipos de equipos, diferentes interfaces, y diferentes sistemas de señalización, los protocolos varían de un equipo a otro (Figura 4). En la MSC, los mensajes destinados a la MS de ISUP/TUP se convierten a mensajes de CM. Los mensajes de MAP se pasan a mensajes de MM. El protocolo BSSAP se encarga de la transmisión de los mensajes CM y MM, y el control directo del BSS. Se divide en dos partes en DTAP y en BSSMAP, para la transferencia de mensajes de nivel 3 y para el tratamiento de los recursos radio respectivamente. Los protocolos RR, BTSM, LAPD se emplean entre BSC y BTS. Los mensajes de tipo RR pasan de forma transparente por la BTS hacia el MS, mientras que algunos mensajes de recursos radio, se envían con el protocolo BTSM y se transforman a protocolo RR' para ser recibidos por la MS en el nivel de red en la subcapa de RR. El nivel de enlace entre BSC y MS usa un protocolo idéntico, el protocolo LAP-D, aunque en el interfaz Um se emplea una variante llamada LAP-Dm. El cambio de nivel de enlace se ve entre MSC y BSC, donde cambia el sistema de señalización empleado.

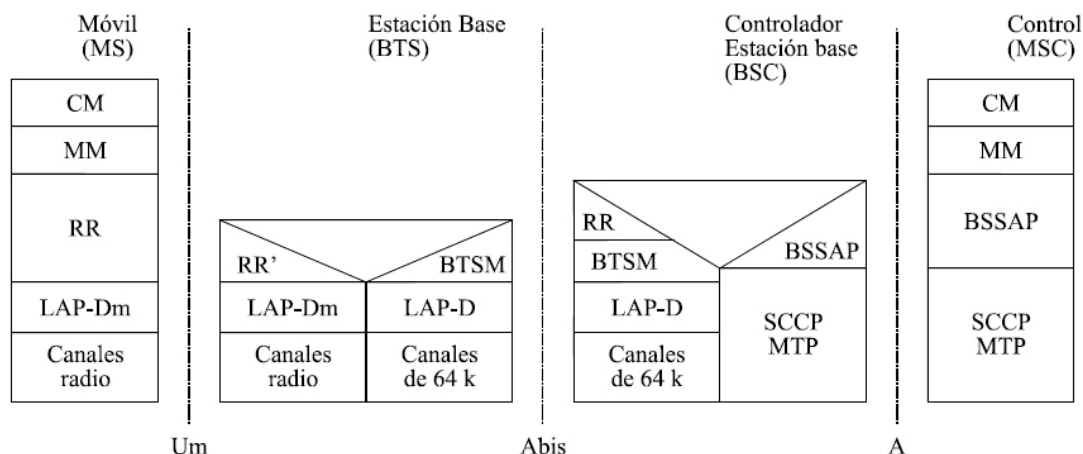


Figura 4 Esquema de los niveles de una red GSM.

Representación de los diferentes niveles e interfaces de una red GSM entre los equipos MSC y MS. El canal físico varía de un interfaz a otro, la igual que los protocolos de nivel de enlace y de nivel de red.

La interfaz radio debe permitir un acceso libre a los usuarios, posibilitar las llamadas, y asegurar la calidad de las mismas. Se usan dos portadoras y numerosos canales lógicos. Al conjunto de las dos portadoras se les suele llamar radiocanal. Los canales lógicos son estructuras de datos que realizan funciones de intercambio de información para seguimiento de móviles, establecimiento de llamadas y mantenimiento de la comunicación. Se dividen en dos, canales de comunicación y canales de control.

La capa RR debe encargarse de proporcionar enlaces entre la MS y la red. Con esto se refiere al establecimiento de llamadas, asignación de radiocanales, aviso a los móviles, traspasos y saltos de frecuencia. La capa MM se ocupa de dar las características de movilidad a los usuarios, gracias a la localización y el seguimiento. La capa CM se ocupa del establecimiento y control de llamadas además de los servicios suplementarios asociados.

5.3 Capa física

5.3.1 Banda de frecuencias

Para telefonía móvil es necesario establecer dos enlaces radio (o un enlace radio dúplex) para permitir la comunicación bidireccional. Para ello se emplean dos portadoras distintas, la portadora ascendente y la portadora descendente. A este par de frecuencias se le denomina

radiocanal. Este tipo de uso del canal recibe el nombre de multiplex por división en la frecuencia, y los radiocanales se asignan en una banda de frecuencias dividida en dos subbandas. Cada portadora perteneciera a una subbanda diferente, y estaría separada de la otra. Para reducir interferencias con otros servicios radio, los bordes de las bandas de frecuencia no se suelen utilizar.

Teniendo en cuenta estas consideraciones, el número de radiocanales que se pueden emplear es:

$$N = \frac{BW}{Af}$$

Ecuación 1 Número de radiocanales

- BW: Ancho de banda del radiocanal
- Af: separación entre radiocanales (en Europa = 200KHz)
- N: número de radiocanales

El enlace ascendente es el enlace más débil en la comunicación, debido a la limitación de potencia que imponen los equipos móviles. Para favorecer a este enlace, se reserva la parte baja del espectro al enlace ascendente o Uplink, ya que tiene unas menores pérdidas de propagación. De la Ecuación 1 se obtienen un total de 124 radiocanales. Debido a la limitación de este espectro se definió una nueva banda denominada E-GSM900, que actúa en las frecuencias circundantes a la banda primaria. Sin embargo, esta banda también se quedó pequeña, por lo que fue necesario volver a encontrar una banda, esta vez en los 1800MHz, que se denominó GSM1800. Esta nueva banda es mucho mayor que la GSM900, por lo que permite el aumento de operadores, además de que se mantiene la estructura de canales, obteniéndose un total de 372 radiocanales. Es necesario que los equipos se distingan en función de estas dos bandas. Existen tres tipos, para la banda GSM900, para la banda GSM1800 y los móviles duales que permiten conmutar de una banda a otra. La asignación de frecuencias es importante para evitar interferencias, y mucho más importante en la frontera entre países, donde se puede dar lugar a problemas de comunicación. Para ello en las fronteras se emplean canales preferentes repartidos según los Estados y operadores para actuar en las zonas fronterizas.

5.3.2 TDMA

Además de emplear el método de acceso por división en frecuencia, también se emplea el acceso por división en el tiempo, también llamado TDMA. El eje temporal se divide en tramas divididas cada una de ellas en 8 intervalos de tiempo o TS. Con esto se consigue dar ocho canales físicos sobre cada radiocanal. Para emplear evitar que el móvil tenga que emitir a la vez que recibir, el intervalo de trama TDMA en el enlace ascendente se retrasa 3 TS con respecto al descendente. Las tramas de TDMA se agrupan a su vez en multitramas, supertramas e hipertramas. Para que este sistema funcione, es necesario que se sincronicen móvil y estación base. Para ello, el móvil debe alinearse temporalmente con la estación base.

5.3.3 Características del canal radio móvil

Para GSM, en sus dos bandas es necesario aplicar los modos de propagación. No hay que tener en cuenta sólo el espacio libre, sino que son elementos de importancia los obstáculos, la vegetación, los edificios, los vehículos, la propagación en calles, o el efecto de propagación sobre el agua.

5.3.4 Ráfagas

Existen diferentes tipos de ráfagas completas de 147 bits de duración, estas ráfagas son de corrección de frecuencias, sincronización, relleno y ráfagas normales. Están constituidas por un núcleo de información, rodeados de bits de cola. El periodo entre dos ráfagas se denomina tiempo de guarda para minimizar las probabilidades de colisión.

Las ráfagas de acceso son empleadas por el móvil para acceder a una estación base cuando necesita un canal, por lo que sólo se transmiten en el canal ascendente. Las ráfagas de corrección de frecuencia se emplean en el canal descendente para ajustar la frecuencia de sintonización, y su campo de información son 148 bits a cero. La ráfaga de sincronización se emplean únicamente en el enlace descendente cuya finalidad es la de sincronizar el reloj de la estación móvil con el de la estación base. Las ráfagas de relleno se emplean cuando no hay ninguna información que transmitir. El canal BCCH debe transmitir continuamente información, mientras que los canales de tráfico no necesitan estar siempre ocupados. Por último las ráfagas normales llevan información de tráfico y canales de control.

5.3.5 Canales

Los canales se clasifican en dos tipos, canales comunes y canales dedicados.

- Canales comunes: los canales comunes son los que transmiten información de señalización común, es decir la información destinada a todos los móviles que se encuentran en una celda, y son canales de punto a multipunto. Los canales lógicos comunes son BCCH, FCCH, SCH, PCH, AGCH, NCH y RACH. Los canales AGCH y RACH se transmiten en los canales descendentes y ascendentes respectivamente. La portadora 0 de una celda es la que funciona como señal de referencia para los móviles, y usan su canal físico TS 0 para la transmisión de los canales comunes de señalización. A esta portadora se le denomina portadora BCCH. Los canales FCCH y SCH se pueden transmitir únicamente en el TS 0.
 - FCCH y SCH: son canales de adquisición de frecuencia y de sincronización temporal
 - BCCH, PCH, AGCH: son canales unidireccionales de difusión y de búsqueda de móviles y acceso de móviles. Utiliza ráfagas normales en el enlace descendente (DL) El BCCH lleva información de la celda, del área de localización, de la organización de los CCCH y de los canales para PCH. Los canales PCH se utilizan para las búsquedas de móviles. Los canales AGCH se emplean para iniciar llamadas. La configuración de este enlace puede variar en función de la cantidad de canales de búsqueda o acceso se quiera dar. Para dar una gran capacidad se asignan 36 tramas de las 51 de la multitrama, y el de menor capacidad lleva 12 de 51. Es posible definir más de un canal físico dentro de la portadora BCCH para transmitir canales lógicos, aumentando al capacidad.
 - NCH: canales de notificación
 - RACH: canales de acceso empleados por los móviles para acceder a la red. Es similar a ALOHA.
- Canales dedicados: los canales dedicados son aquellos que transmiten información correspondiente a una conexión establecida entre un equipo móvil y la red GSM. Son canales punto a punto y transmiten voz, datos o señalización.
 - TCH: los canales de tráfico transmiten la información de voz y de datos. Son bidireccionales, que utilizan el mismo intervalo en ambas portadoras. En una portadora pueden ocupar cualquier intervalo excepto el cero que va en el

BCCH. La velocidad de transmisión de voz es de 22,8 o de 11,4 si es un canal Full-rate o Half-rate respectivamente. Para la transmisión de datos las tasas son diferentes, y van de 2,4Kbit/s a los 9,6Kbit/s.

- SACCH: es el canal de señalización lento que se asocia a un canal de tráfico. Este canal lleva información asociada a la conexión necesaria para la movilidad y para los recursos radio. Lleva mensajes de información sobre las medidas de nivel y sobre calidad de los canales. Estos canales van intercalados en las tramas dedicadas a tráfico.
- FACCH: canal utilizado para transmisión de información urgente. Para ello roba bits de tráfico en las tramas. Se emplea para la realización de trasposos.
- SDCCH: Canal que lleva información de establecimiento de llamada, de encendido/apagado, actualización de posición, envío y recepción de imágenes. Usan un canal SACCH asociado.

Los canales que deben presentarse son al menos un único FCCH, un único SCH, y al menos un BCCH. Debido a la diferente necesidad de capacidad en función de la localización de la estación, la configuración de canales es diferente para cada caso. Las velocidades de transmisión de cada canal aparecen en la Tabla 1.

Velocidades de transmisión e información en los canales GSM	
Canal	Velocidad de transmisión/información
BCCH	Transmisión: 1,94Kbit/s Información: 781 bit/s
RACH	
CCCH	Transmisión: 1,94Kbit/s Información: 390bit/s
SDCCH	Transmisión: 1,94Kbit/s Información: 390bit/s
SACCH	Transmisión: 970bit/s

	Información: 390bit/s
TCH	Transmisión: 22,8Kbit/s Información: 13Kbit/s

Tabla 1 Velocidades de transmisión e información en GSM.

Se ven las diferentes velocidades de transmisión en función del tipo de canal. Datos extraídos del libro: Comunicaciones Móviles GSM, Hernando Rábanos, José María. Ver bibliografía.

5.4 Planificación radio

5.4.1 Introducción

La tarea más importante en el dimensionamiento de una red GSM es la planificación radio, ya que es la que determina la calidad de la red además de la cobertura y capacidad de la misma. Para ello hay que definir el número de estaciones base, y la ubicación de las mismas, añadiendo los parámetros radioeléctricos (antenas, potencias, frecuencias). Hay que tener especial cuidado con las características del emplazamiento, ya sea en túneles, aparcamientos, aeropuertos, centros comerciales, zonas urbanas o zonas rurales. La planificación es una tarea muy complicada por la gran cantidad de factores que se deben tener en cuenta.

5.4.2 Teoría celular

Debido a la necesidad de ofrecer servicio a un gran número de usuarios obligó a plantear las redes celulares. Con el uso de una única antena en una zona de cobertura, se necesitan un gran número de canales por cada antena para dar una baja probabilidad de bloqueo. La solución es la colocación de varias antenas para repartir la carga. Sin embargo la reutilización de las mismas frecuencias da lugar a interferencias. La idea que plantea la teoría celular es la aplicación de un número determinado de canales por celda, reutilizándose en zonas no directamente cercanas lo que permite reducir la interferencia. Esto quiere decir que si se crea una célula, las células adyacentes deben tener diferentes frecuencias de uso. La teoría celular se divide en dos postulados:

- Dividir la zona de cobertura en zonas más pequeñas denominadas celdas, con un tamaño variable que depende de la demanda de tráfico bajo esa celda.

- Reutilizar las frecuencias en celdas separadas por una distancia que permita que la interferencia cocanal sea pequeña y tolerable para la comunicación.

El tráfico ofrecido en una zona es proporcional a la superficie de la misma, si las celdas son pequeñas, el tráfico ofrecido también será pequeño. Gracias a la reutilización se aumenta la cantidad de frecuencias disponibles. Se conoce como índice de reutilización de un sistema celular al cociente entre el número de radiocanales que se ofrecen y el número de frecuencias disponibles. La distancia de reutilización es la distancia que se necesita separar las frecuencias cocanales para que la comunicación sea adecuada y no se vea entorpecida por las interferencias. Es necesario buscar un número óptimo para dar el tamaño a la red celular.

5.4.3 Diseño de la red celular

La planificación celular es un estudio básicamente teórico. Para el diseño celular hay que tener también en cuenta a parte de los factores técnicos los factores económicos que afectan al posicionamiento de las células. No en todos los casos se puede colocar una antena en la ubicación exacta, ya que depende de la legislación, del coste de alquiler, del suministro eléctrico. Los operadores deben en todo momento optimizar la relación calidad/coste. Las tareas a llevar a cabo son las siguientes:

- Desarrollo de un modelo de tráfico y movilidad de los clientes
- Elección de tamaño y tipo de células
- Diseño de la red celular con los tipos de células.
- Orientación de las antenas
- Estudio de zonas con cobertura difícil como túneles, interiores de edificios, etc.
- Reparto de frecuencias
- Evaluación de señal deseada con respecto a señales interferentes.
- Interconexión entre BTS, BSC y MSC.

5.4.3.1 Tipos de celdas

La forma de las celdas depende de la potencia emitida por cada antena. Existen dos tipos de

antenas, las omnidireccionales que emiten desde un punto y abarcan una cobertura de un círculo, y las sectorizadas, que abarcan tres áreas llamadas sectores. A la hora de visualizar una celda se elige un hexágono, ya que es la figura geométrica que permite cubrir toda la zona abarcando la mayor cantidad de superficie. También se han realizado estudios con triángulos y cuadrados, aunque estos cubren una superficie inferior a la que cubre un hexágono.

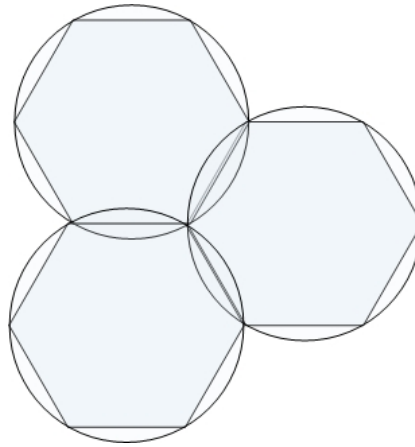


Figura 5 Forma celular

El hexágono es el polígono regular que mejor cumple los requisitos de superficie para la representación teórica de una red celular.

El tamaño de las celdas pueden clasificarse en función del tamaño de la célula. Difiere mucho la cobertura de una célula en función de la zona en la que se encuentren. Las células en zonas rurales tienen grandes radios de cobertura, mientras que las células en aeropuertos tienen pequeños radios. Esta longitud viene determinada por las necesidades de tráfico y por las pérdidas del medio. El límite lo impone la tecnología y es de 35Km. En algunos casos especiales se puede llegar a los 70Km pero modificando el reparto de TS. Las diferentes longitudes que se presentan son:

- Grandes o macroceldas: radio de 1,5 a 35Km, para zonas rurales.
- Pequeño o miniceldas: de 0,7 a 1,5Km, para zonas urbanas.
- Microceldas: para zonas con alta densidad de tráfico. De 0,3 a 0,7Km.
- Picoceldas: para lugares específicos como aeropuertos o centros comerciales. De 30 a 200m
- Celdas paraguas: para cubrir zonas de sombra dejadas por otras celdas.

El uso de celdas pequeñas también tiene sus inconvenientes. Un gran número de celdas pequeñas da lugar a un gran número de traspasos, lo cual puede congestionar la red. Hay que llegar a un compromiso entre número de celdas, calidad del servicio, tráfico ofrecido y carga aportada a la red.

El objetivo de la planificación celular es el de reducir el tamaño de la agrupación de células llamada J . Este valor se obtiene de la Ecuación 2.

$$J = i^2 + j^2 + ij$$

Ecuación 2 Parámetro J

J: tamaño de la agrupación
i: distancia a célula cocanal
j: distancia a célula cocanal

Para GSM se emplean valores de $i = 2$ y $j = 0$, dando un valor de $J = 4$, y valores de $i = 1$ y $j = 1$, con un valor de $J = 3$. Para estaciones sectorizadas este número viene dado por el valor $3J$, y se usa la notación $J/3J$. La utilización de antenas sectorizadas reduce la interferencia, ya que se tratan de antenas directivas. En este caso, en vez de calcular las interferencias con seis estaciones cocínales sólo sería necesario hacer el estudio con las tres más directivas hacia esta antena. Ver la Figura 6.

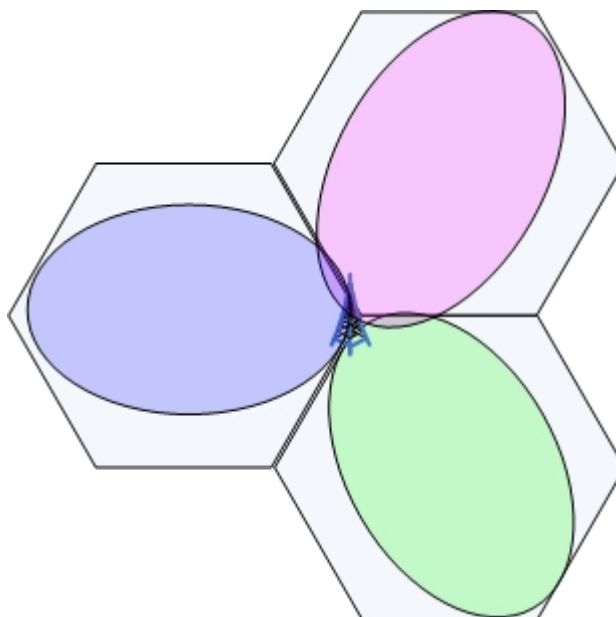


Figura 6 Antenas sectorizadas

Las antenas sectorizadas disminuyen la interferencia co-canal. Se basan en antenas tres antenas con 120° de giro una con respecto a la otras.

La interferencia co-canal impone la limitación del tamaño de las células. Si se toman seis estaciones base co-canales interferentes, alrededor de una estación base, la interferencia máxima debe estar 9dB por debajo de la señal de la célula para que la señal fuese detectada y demodulada. Para el caso del primer canal adyacente, la interferencia puede ser hasta 9 dB mayor, para el segundo canal adyacente 41dB mayor y para el tercer canal adyacente puede ser hasta 49dB mayor.

La densidad de tráfico se distribuye de diferente forma en función de la zona geográfica. Normalmente sigue una distribución exponencial, en la cual la mayor densidad de tráfico se encuentra presente cerca del centro de la ciudad, mientras que la menor densidad de tráfico se encuentra en las zonas rurales.

Para mejorar la cobertura en zonas rurales es aconsejable instalar las antenas en emplazamientos dominantes, intentando maximizar la zona cubierta con visión directa. Para zonas urbanas se deben tener en cuenta los altos niveles de interferencias, por lo que se utilizarán emplazamientos poco dominantes para no crear interferencias con otras antenas, y evitando obstáculos. Se suelen utilizar antenas sectoriales.

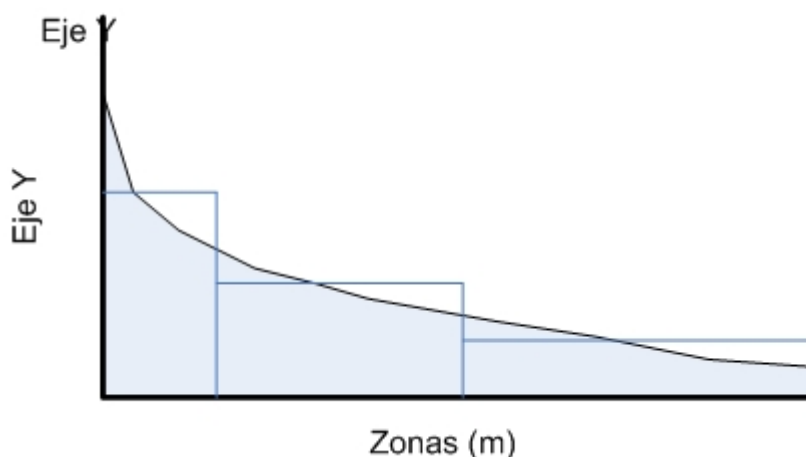


Figura 7 Densidad de tráfico en GSM.

En zonas cercanas al centro de una ciudad las densidades son mayores que en zonas más alejadas. El eje vertical indica la densidad y el X la distancia al centro de la ciudad. En la imagen se pueden ver tres zonas, una urbana de alta densidad (izquierda), otra de las afueras de una ciudad, y por último una zona rural.

5.4.4 Métodos de predicción de pérdidas

5.4.4.1 Método de pérdidas en el espacio libre

El método se basa en el cálculo de las pérdidas de potencia en el espacio libre, es decir en un entorno en el cual no hay obstáculos, y suponiendo visión directa entre emisor y receptor. Supone una dependencia lineal entre las pérdidas del trayecto en dB y el logaritmo de la distancia.

$$L = L_0 + 10n \log(d)$$

Ecuación 3 Pérdidas en el espacio libre⁶

L: atenuación

L_0 : atenuación a la distancia de un metro, para el espacio libre es $L_0 = 32,45 + 20 \log(f)$ con f frecuencia en GHz.

N: exponente de pérdidas

D: distancia entre transmisor y receptor en metros

⁶ Extraído de “Caracterización estadística de la propagación para comunicaciones móviles en el interior de fábricas” Rafael Herradón Díez, Florentino Pérez Muñoz, Departamento de Ingeniería Audiovisual y Comunicaciones. Universidad Politécnica de Madrid.

5.4.4.2 Método de Okumura – Hata

Se basa en unas medidas realizadas en Japón sobre valores de intensidad de campo para entornos urbanos, y con diferentes alturas de antenas. La altura de referencia para la antena de recepción es de 1,5m, y las curvas poseen unas correcciones para tener en cuenta las ondulaciones del terreno y los obstáculos. Hata desarrollo las expresiones que proporcionan el valor medio de la pérdida básica de propagación⁷.

$$L_b = 69,55 + 26,16 \log f - 13,82 \log h_b - a(h_m) + (44,9 - 6,55 \log h_b) \log d$$

Ecuación 4 Ecuación de pérdidas de Hata básica

$$a(h_m) = (1,1 \log f - 0,7)h_m - (1,56 \log f - 0,8)$$

Ecuación 5 Corrección por altura para una ciudad pequeña

$$a(h_m) = 8,29(\log 1,54h_m)^2 - 1,1$$

Ecuación 6 Corrección por altura para una ciudad grande en frecuencias < 200MHz

$$a(h_m) = 3,2(\log 11,75h_m)^2 - 4,97$$

Ecuación 7 Corrección por altura para una ciudad grande en frecuencias > 400MHz

$$L_{bs} = L_b - 2(\log(f / 28))^2 - 5,4$$

Ecuación 8 Pérdida básica en un entorno suburbano

$$L_{br} = L_b - 4,78(\log(f))^2 + 18,33 \log f - 40,94$$

Ecuación 9 Pérdida básica en un entorno rural

Donde:

f: frecuencia en MHz

h_b: altura efectiva de la antena de la estación base en m

h_m: altura de la antena de la estación móvil

d: distancia en Km.

⁷ Fórmulas extraídas de la norma GSM03.30.

$a(h_m)$: corrección por altura h_m

5.4.4.3 Método COST 231

Método empleado para caracterizar la propagación aplicada a ciudades similares a las europeas, con dos escenarios posibles, las células grandes y pequeñas y las microceldas. La atenuación calculada depende de la pérdida en el espacio libre, de las pérdidas por difracción y dispersión del tejado a la calle, y de las pérdidas por difracción multipantalla. Se puede emplear en distancias entre los 10 m y los 3 Km.

5.4.5 Balances de enlace ⁸

A continuación se presentan los valores de potencias de emisión y sensibilidad de una red GSM.

Clase de MS	GSM 900 Máxima potencia de salida nominal	DCS 1 800 Máxima potencia de salida nominal power	Tolerancia (dB) en condiciones	
			normal	extremo
1	-- ----	1 W (30 dBm)	± 2	± 2.5
2	8 W (39 dBm)	0.25 W (24 dBm)	± 2	± 2.5
3	5 W (37 dBm)	4 W (36 dBm)	± 2	± 2.5
4	2 W (33 dBm)		± 2	± 2.5
5	0.8 W (29 dBm)		± 2	± 2.5
NOTE: La mínima potencia nominal en GSM 900 MS es 5 dBm y para todo el resto de clases en DCS 1 800 MS es 0 dBm.				

Tabla 2 Potencia de transmisión de una MS

⁸ Tablas y datos extraídos del documento GSM05.05

TRX Clase de potencia	Máxima potencia de salida
1	320 - (<640) W
2	160 - (<320) W
3	80 - (<160) W
4	40 - (<80) W
5	20 - (<40) W
6	10 - (<20) W
7	5 - (<10) W
8	2.5 - (<5) W

Tabla 3 Potencia de transmisión de una antena en GSM900

TRX Clase de potencia	Maxima potencia de salida
1	20 - (<40) W
2	10 - (<20) W
3	5 - (<10) W
4	2.5 - (<5) W

Tabla 4 Potencia de transmisión de una antena en GSM1800

GSM 900 micro-BTS		DCS 1 800 micro-BTS	
TRX clase	Maxima potencia de salida	TRX clase	Maxima potencia de salida
M1	(>19) - 24 dBm	M1	(>27) - 32 dBm
M2	(>14) - 19 dBm	M2	(>22) - 27 dBm
M3	(>9) - 14 dBm	M3	(>17) - 22 dBm

Tabla 5 Potencia emitida por una microcelda

Sensibilidad de recepción de equipos MS y BTS

-	for DCS 1 800 class 1 or class 2 MS	:	-100 / -102 dBm *
-	for DCS 1 800 class 3 MS	:	-102 dBm
-	for GSM 900 small MS	:	-102 dBm
-	for other GSM 900 MS and normal BTS	:	-104 dBm
-	for GSM 900 micro BTS M1	:	-97 dBm
-	for GSM 900 micro BTS M2	:	-92 dBm
-	for GSM 900 micro BTS M3	:	-87 dBm
-	for DCS 1 800 micro BTS M1	:	-102 dBm
-	for DCS 1 800 micro BTS M2	:	-97 dBm
-	for DCS 1 800 micro BTS M3	:	-92 dBm

5.5 Subsistema BSS

5.5.1 Introducción

Se denomina subsistema BSS al conjunto que forman la BTS (Base Transceiver Station) y la BSC (Base Station Controller). Una de las funciones de este conjunto de equipos y además la más importante es la manejar la comunicación radio con los equipos móviles, gestionar los trasposos entre las celdas conectadas a una misma BSC y controlar el nivel de potencia transmitida por los móviles y la BTS. El subsistema BSS actúa sobre dos interfaces, el interfaz radio o Um, y el interfaz A hacia el subsistema SSS.

Además de estos dos equipos también hay que mencionar el equipo denominado TRAU (Transcoding and Rate Adaptation Unit) que adapta la interfaz A bis con una tasa de transferencia de 16 Kbits/s en full-rate a la A que es de 64 Kbit/s. La posición de la TRAU difiere dependiendo de las necesidades del operador, sin embargo su control y gestión se realiza desde la BSC. El posicionamiento entre la BTS y la BSC tiene un problema, y es que no tiene ahorros en el uso de medios de transmisión ya que se pasa de un canal a 16Kbit/s a otro de 64Kbit/s para que sea compatible con la red ISDN y la estructura de SS7. El otro caso, el que permite un mejor ahorro de medios es posicionar la TRAU cerca de la MSC y enviar cuatro canales de 16Kbit/s multiplexándolos en un canal de 64Kbit/s entre BSC y TRAU.

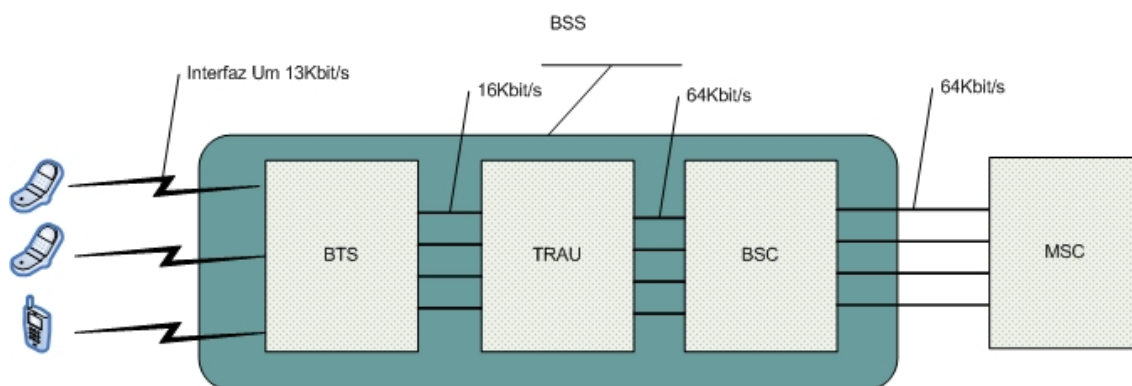


Figura 8 Posición del equipo TRAU.

El equipo de transcodificación y adaptación de velocidad puede colocarse en diferentes lugares. En esta configuración se encuentra entre la BTS y la BSC. Otra configuración es entre BSC y MSC. Sea cual sea su posición, la BSC siempre es la encargada de gestionar este equipo.

5.5.1.1 Funciones de la BSC

Principalmente se encarga de las funciones de control del subsistema BSS. Se encarga de los recursos radio, mientras que los subniveles MM y CM pasan de forma transparente a este equipo. Las funciones más importantes son:

- Control del interfaz radio. Realización de estadísticas de tráfico
- Control de BTS. Configuración de frecuencias.
- Conexión con MS. Establecimiento y liberación de canales, asignación de recursos radio, algoritmos de control de potencia.
- Traspasos cuando se realicen dentro de la misma BSS. Informa a la MSC (mensaje HANDOVER PERFORMED).
- Intercambio transparente de mensajes entre MS y MSC/VLR o HLR.
- Control del interfaz Abis (interfaz entre BTS y BSC).
- Distribución de mensajes de aviso (PAGING)

5.5.2 Interfaz ABIS

La interfaz existente entre el BSC y la BTS se denomina interfaz Abis. El nivel físico de esta

conexión está formado por enlaces de 2Mbit/s que forman 32 canales de 64Kbit/s. cada BTS controla una celda, y puede tener uno o más equipos transceptores TRX a su cargo. Los TRX son los equipos que soportan los 8 canales radio básicos de una trama TDMA, es decir de una portadora. Otro equipo perteneciente a la BTS es el BCF (Base Control Function) que se encarga de gestionar las funciones comunes dentro de la BTS. La configuración que puede soportar el interfaz Abis puede ser de tres tipos:

- Una única conexión para un único TRX.
- Una única conexión para varios TRX.
- Varias conexiones para varios TRX.

Se emplean dos tipos de canales, conexiones para canales de tráfico denominadas SDC (Speech and Data Channel) y canales de señalización llamados SCH (Signalling Channel). Por cada canal de 64Kbit/s se pueden transmitir hasta cuatro canales de tráfico TCH full rate y hasta ocho TCH half rate. En esta configuración sobran 12 Kbit/s que se emplean para sincronización entre BTS y TRAU como hemos mencionado antes.

5.5.3 Conexión

Todas las BTS pertenecientes a una BSS van unidas al equipo BSC. Las configuraciones de conexión son diferentes dependiendo de los objetivos que se pretendan alcanzar.

No siempre las conexiones entre BTS y BSC son directas. En algunos casos conviene conectar una BTS a otra y que ésta última se encargue de llevar ambos canales al BSC. Los motivos pueden ser de tipo económico o para aumentar la fiabilidad de la red. Las diferentes tipologías son las siguientes:

- Conexión en estrella: todas las BTS se conectan directamente a un BSC.
- Conexión en estrella remota: todas las BTS se conectan a una BTS que a su vez se conecta a la BSC.
- Conexión en cascada: se conectan una tras otra.
- Conexión en anillo: las BTS forman un anillo de conexión.

De las diferentes conexiones anteriores, las tres primeras son las más usadas, sin embargo la

decisión entre una u otra depende exclusivamente de la zona en la cual se necesite implementar la red, ya que por ejemplo para una zona rural, el establecimiento de una conexión tipo estrella por cable si BTS y BSC están muy distantes entre sí puede suponer un coste muy elevado. En este caso tal vez sea más rentable establecer un canal radio entre ambos equipos, o establecer una conexión cableada con una BTS intermedia. En estos casos hay que evaluar la fiabilidad frente al precio.

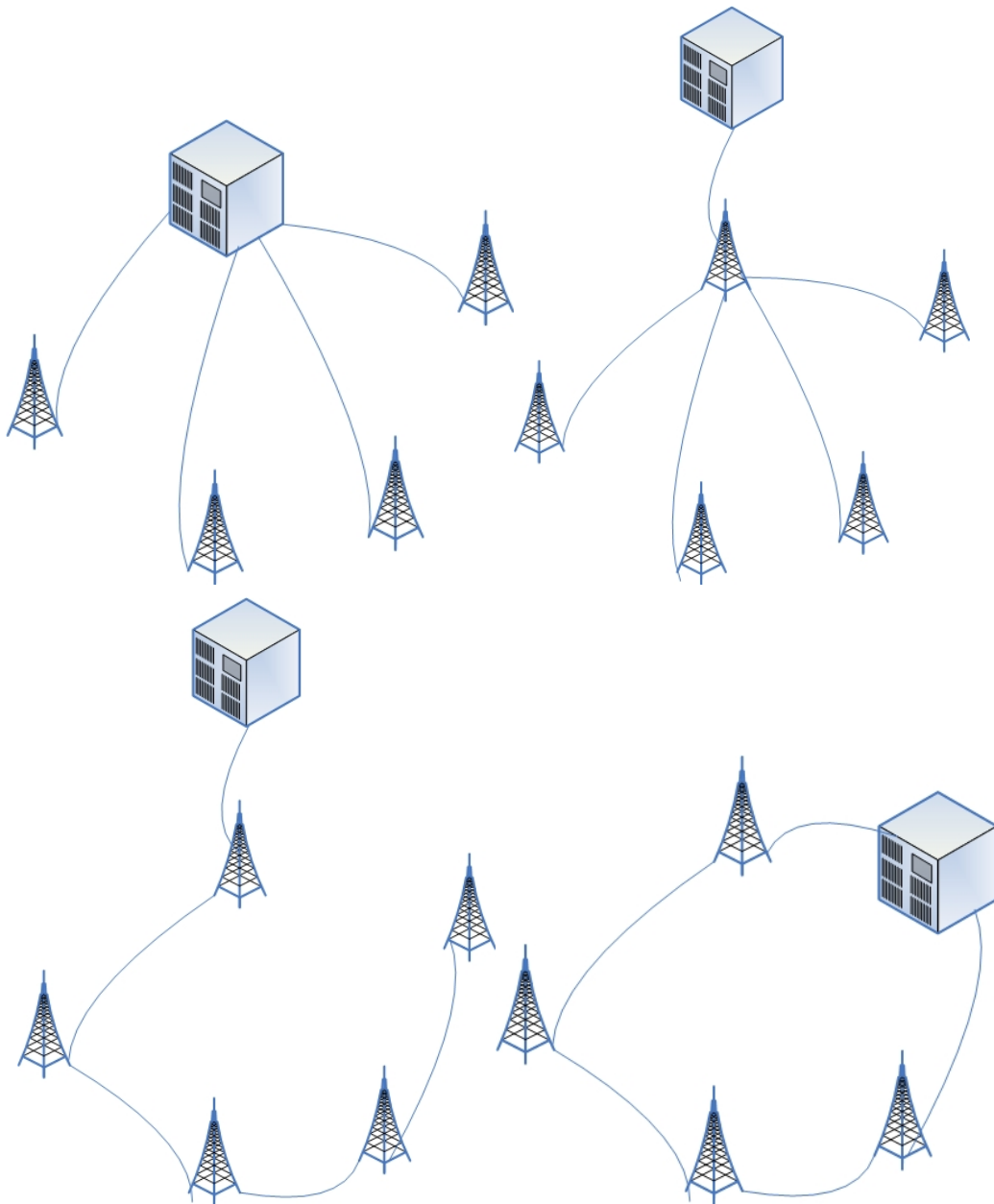


Figura 9 Topología de conexiones entre BTS y BSC.

En la imagen se pueden ver los cuatro tipos de conexiones que pueden establecerse. Las imágenes corresponden de izquierda a derecha y de arriba abajo a una topología en estrella, en estrella remota, en cascada y en anillo.

5.5.4 Interfaz A

Entre la BSS y la MSC se encuentra la interfaz A. Dependiendo de la localización del equipo TRAU, este interfaz tendrá una configuración u otra. Generalmente, se trata de una conexión de 2Mbit/s dividida en 32 canales de 64Kbit/s.

5.6 Subsistema SSS

5.6.1 Introducción

El nombre de subsistema SSS (también denominado con las siglas SMSS) viene del inglés Switching and Management Subsystem y denomina a la parte de la red GSM que incluye las funciones necesarias para la conmutación de llamadas, además de las bases de datos de usuario que permiten el establecimiento de los diferentes servicios. Esta parte de la red se considera el corazón de GSM. La MSC es el centro de conmutación. De este dependen una o varias subredes o subsistemas BSS. Directamente conectado a este equipo se encuentra la VLR o el registro de visitantes, en el cual se almacenan determinados datos de los móviles que se encuentran dentro del ámbito de la MSC.

El equipo HLR es uno o varios registros que se conectan directamente con los VLR y que almacenan la información de todos los equipos móviles de la red. A su vez se conectan con los AuC (Authentication Center) que contienen la información de usuario y de claves. La razón de que se implementen o incluyan varios HLR es por un lado la redundancia, es decir que la red sea tolerante a fallos, y el otro que el HLR tiene una limitación de número de registros, por lo que en una red grande (la mayoría de ellas, ya que un HLR puede almacenar apenas unas centenas de miles) serán necesarios varios HLR para tener todos los datos de usuario.

El GMSC es el Gateway MSC que se encarga de comunicar la red GSM con redes externas, ya sean redes de datos, o redes telefónicas como la PSTN (red telefónica pública fija). Además constituye la conexión con el equipo GM-SC que se encarga de la gestión y envío de los mensajes SMS.

A continuación se pasa a describir en un breve resumen las principales características de la red GSM que se mencionan en las especificaciones o libros.

5.6.2 Funciones de red

- Funciones para provisión del servicio básico
 - Debe ofrecer obligatoriamente :
 - Gestión de llamadas
 - Autenticación de la identidad de usuario
 - Llamadas de emergencia
 - Servicios suplementarios
 - Servicios de grupo de voz (especificaciones GSM2+)
 - Servicio de mensajes cortos SMS
 - Confidencialidad de los elementos de información de señalización
- Funciones para soportar la estructura celular
 - Registro de posición VLR y HLR
 - Traspasos (Handover)
 - Restablecimiento de llamada
- Funciones adicionales para la gestión de llamadas
 - Formación de colas de llamadas
 - OACSU
 - Servicios de seguridad
 - Recepción discontinua
 - Soportar DTMF
- Funciones para la gestión de red

5.6.3 Componentes del subsistema de conmutación

Los principales componentes de este subsistema son los siguientes:

- MSC
- VLR
- HLR
- GMSC

No se describirán puesto que ya han sido explicados en el apartado de arquitectura de GSM.

5.7 Señalización

La señalización en GSM es una tarea muy compleja debido al gran tamaño del sistema. Además hay que añadir la gestión de la movilidad, lo que aumenta la carga de señalización en la red. Las funciones de señalización se estructuran en niveles se forma parecida al modelo OSI. La capa de red se divide en tres subcapas que proporcionan funciones y servicios diferentes. El primer nivel corresponde al nivel físico que proporciona un medio de transmisión para el intercambio de información. El segundo nivel es el nivel de enlace, que emplea el protocolo LAPD y LAPDm para los interfaces Abis y Um respectivamente, y su función básica es proporcionar una comunicación fiable al nivel superior. El nivel tercero se denomina nivel de red. Como su nombre indica se encarga de funciones de gestión de red. En GSM se encuentra dividido en tres subniveles llamados RR, MM y CM. El subnivel RR se encarga de la gestión de los recursos radio. La subcapa MM lleva a cabo la gestión de la movilidad. Por último la CM se encarga de la gestión de las comunicaciones.

Para que se comuniquen los diferentes niveles se emplean primitivas de servicio. En realidad sólo existe una única conexión a nivel físico, aunque los niveles interactúan con las entidades de su mismo nivel con conexiones lógicas. Un mensaje enviado desde un extremo bajará los niveles hasta el nivel físico, se enviará y subirá hasta el nivel hasta el que va destinado. El protocolo para el nivel tres es el protocolo BSSAP que está formado a su vez por el BSSMAP para el subnivel RR y el protocolo DTAP para CM y MM. MAP el protocolo empleado para la comunicación entre los nodos de la red GSM que pertenecen al subsistema SSS.

5.7.1 Protocolo LAPD

Este protocolo se emplea en el interfaz Abis en el nivel 2 o nivel de enlace. Existe una variante denominada LAPDm que se usa en el interfaz Um. Proporciona conexiones de enlace entre BTS y BSC, numeración de tramas, detección y recuperación de errores, control de flujo, y alineación y delimitación de tramas. Los mensajes se estructuran en tramas con campos de ocho bits y que comienzan y finalizan con la secuencia '01111110'.

5.7.2 Protocolo BSSAP

BSSAP es la parte de aplicación de BSS, y es un protocolo que permite la comunicación entre MSC y BSS. Este protocolo se divide a su vez en dos protocolos denominados BSSMAP y DTAP. Los mensajes del protocolo BSSMAP van dirigidos al subsistema BSS, mientras que los mensajes del protocolo DTAP (Direct Transfer Application Part) pasan de forma transparente a la BSS y van destinados directamente a una MS. Algunos de los procedimientos BSSMAP son los siguientes:

- Asignación: realiza la asignación de canales. El equipo MSC envía un mensaje ASSIGNMENT REQUEST (ASSIGNMENT COMMAND entre BSC y MS) para iniciar el procedimiento. Si la asignación es correcta la MS devuelve un mensaje ASSIGNMENT COMPLETE hacia la MSC.
- Indicación de recursos: informa de los recursos radio disponibles en una celda.
- Liberación de recursos: después de la utilización de los recursos se debe proceder a la liberación de los mismos.
- Aviso: los mensajes de aviso se emplean para la localización de móviles dentro de un área de localización. Cuando el móvil recibe un mensaje de aviso (PAGING) responde solicitando un canal.
- Cifrado: procedimiento para iniciar el cifrado de los datos en la red GSM.
- Traspaso: traspaso realizado entre dos BSC pertenecientes a una misma MSC (intramsc) o a diferentes MSC (intermsc).
- Traspaso intracelular: traspaso entre células dentro de una misma BSC, informa a la MSC de la realización del traspaso con el mensaje HANDOVER PERFORMED.

- Traspaso intercelular: traspaso entre canales dentro de una misma celda. Envía el mensaje HANDOVER PERFORMED hacia la MSC.

5.7.3 Protocolo MAP

Como se ha dicho antes es un protocolo para la comunicación entre los equipos del subsistema SSS. Dentro de este protocolo se definen diferentes procedimientos.

- Procedimientos relativos a la movilidad: se emplean para controlar en todo momento la posición del móvil dentro de la red. Entre estos procedimientos se encuentra el procedimiento de actualización de posición. Se emplea cuando el móvil cambia de localización o cada cierta cantidad de tiempo establecido por el operador para actualizar la posición y el estado de un móvil en el equipo VLR, y HLR. Para ello emplea un mensaje denominado A_LU_REQUEST, que en el protocolo MAP es MAP_UPDATE_LOCATION_REQUEST. Cuando un VLR recibe este mensaje comprueba que la estación móvil se encuentra en la base de datos y actualiza su valor. En caso contrario comprueba si se encuentra registrado en otra VLR. El último paso consiste en actualizar los valores en el HLR. En caso correcto el HLR enviará hacia el VLR las claves de autenticación. En este procedimiento la autenticación y el chequeo del número IMEI son opcionales. Cuando se ha terminado la comunicación entre bases de datos, se devuelve al equipo móvil un mensaje de A_LU_CONFIRM para confirmar la actualización de localización. Otro procedimiento es la cancelación de posición. Este procedimiento se lleva a cabo a instancia del HLR cuando el móvil se ha registrado en otra VLR.
- Procedimientos de traspaso: los procedimientos de traspaso sólo actúan en el protocolo MAP si son mensajes intramsc o intermsc, para los trasposos intracelulares, intercelulares solamente se notifica a la MSC.
- Procedimientos de gestión de llamadas: son procedimientos para la obtención de información de encaminamiento, para servicios suplementarios, para establecimiento de llamadas originadas en el móvil o terminadas en el móvil.

6 OMNET++

6.1 Introducción

OMNET++ es un simulador de redes de eventos discretos, basado en módulos orientados a objetos. Las principales aplicaciones o campos de aplicación de este simulador son modelado de tráfico de redes de telecomunicaciones, modelado de protocolos, modelado de sistemas de colas, modelado de multiprocesadores y otros sistemas distribuidos hardware, validación de arquitecturas hardware, y otras muchas aplicaciones en el campo de las telecomunicaciones.

Un módulo OMNET++ consiste en una jerarquía de modelos. La profundidad de modelos es ilimitada, y la jerarquía de los mismos permite reflejar la estructura de una red que se comunica a través de paso de mensajes. Los mensajes pueden ser muy simples o llegar a una enorme complejidad, y pueden ser destinados directamente al destino o a través de un camino predefinido.

Los módulos son programados en C++ usando la librería de simulación de C++, son compilados con los compiladores habituales como Dev C++, Visual Studio 6.0, Visual Studio .NET, o en que caso de que se utilice Linux, cualquiera de los compiladores disponibles con

este sistema operativo. Lo único a tener en cuenta es enlazar las librerías que se pueden obtener en la página oficial de OMNET++.

Otra de las características de este entorno es que permite la ejecución distribuida y paralela de la simulación, como por ejemplo con MPI⁹. Los modelos desarrollados además no necesitan ningún elemento añadido para ser ejecutados en paralelo, ya que la diferencia se encuentra a nivel de configuración.

OMNEST es la versión comercial de OMNET++. Para obtener licencias se tiene que contactar con la compañía OMNEST Global Inc. OMNET++ es sólo gratis para uso académico y sin beneficio.

Este pequeño manual no pretende ser una completa guía para desarrollo de aplicaciones con OMNET++. El objetivo de las siguientes páginas es realizar un pequeño resumen de las principales características, funciones y elementos del entorno para que el lector pueda comprender las páginas que describen la implementación del proyecto. Para aquellos usuarios avanzados o con un alto conocimiento de las características del entorno, así como para aquellos que ya han trabajado con el entorno este capítulo quizás no vaya a añadir nada a sus conocimientos, sin embargo para aquellos usuarios con pocos conocimientos o ninguno se les recomienda que lean las siguientes páginas.

6.2 Características principales de OMNET++

Para empezar a trabajar con OMNET++, no se necesita nada más que tener un compilador de C++ adecuado para el entorno. Se soportan un gran número de compiladores, incluso de tipo GNU¹⁰, por lo que probablemente no se tendrá mucho problema en conseguir alguno. En cuanto a la obtención del entorno, simplemente tiene que dirigirse a la página oficial de OMNET++ que se menciona en la bibliografía [43]. Junto con las librerías para C++, en el fichero descargable del sitio web, se añaden las herramientas que permiten que la declaración

⁹ Siglas de Message Passing Interface, librería específica para C y FORTRAN que permite la ejecución de programas en paralelo. Se basa en el paso de mensajes entre procesadores para soportar el paralelismo en vez de en compartir variables como hace POSIX.

¹⁰ Siglas de General Public License, proyecto iniciado en 1984 para proporcionar software libre a usuarios de todo el mundo. El software libre permite el acceso al código fuente, poder modificarlo a gusto del usuario, y poder realizar copias libremente.

de los módulos, su desarrollo y la implementación de los mensajes sean sencillos para el usuario. Las librerías y programas se instalarán en un fichero en el ordenador, generalmente C:/Omnet++/. En caso de querer cambiar la ruta se recomienda suprimir los espacios en los nombres de las rutas, ya que el entorno encuentra problemas a la hora de trabajar con ficheros cuyos nombres contienen espacios. En el apartado de instalación se dedica un apartado al contenido de la instalación. En cuanto a los programas instalados, se tiene un editor visual de código NED denominado GNED, un lector de datos escalares denominado Omnet Scalars y otro destinado a datos vectoriales denominado Omnet Plove.

A la hora de realizar una aplicación para simular una red, OMNET++ permite una jerarquía de modelos, sin un límite de niveles de jerarquía. Para ello la simulación se compone de módulos simples, desarrollados en C++ que pueden ser integrados en módulos complejos, que a su vez componen la red. Por ejemplo para una red de tipo Ethernet, se pueden modelar los niveles físico, enlace y red en módulos simples mediante la derivación a partir de la clase `cSimpleModule`¹¹, y a continuación ser integrados en un módulo compuesto que podría denominarse ordenador. A su vez cada módulo ordenador puede añadirse a una red LAN. Dentro del inspector visual del editor GNED o del entorno visual de ejecución denominado Tkenv, tendremos la posibilidad de ver la red completa, un ordenador y los módulos que simulan cada una de las capas en diferentes ventanas. En la Figura 10 y en la Figura 11 se puede ver un ejemplo ya implementado de una red de características que se han comentado, en la que se pueden ver los diferentes módulos que la componen.

¹¹ Clase base incluida en las librerías de OMNET++ y que sirve para crear módulos simples o derivar nuevos tipos de módulos.

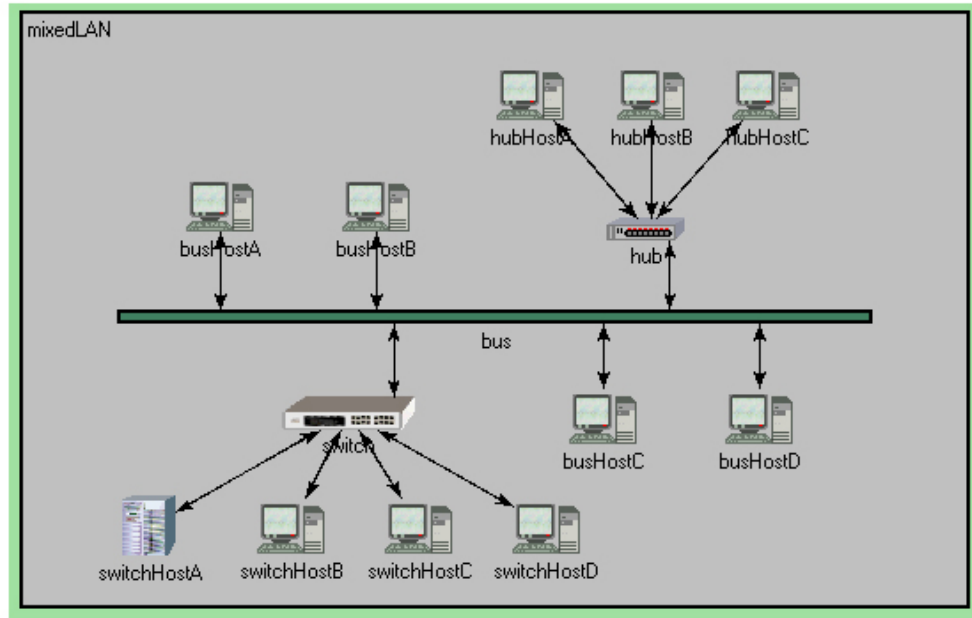


Figura 10 Simulación de una red LAN perteneciente al INET Framework Demo For OMNET/OMNET++, red.

INET Framework es un programa creado por varios desarrolladores cuyo objetivo es crear a partir de OMNET/OMNET++ un entorno para la simulación de redes y protocolos relacionados con TCP/IP e Internet. Esta figura pertenece en concreto al simulador denominado “LANs : Mixed Ethernet”. Para crear la simulación se han declarado varios módulos compuestos llamados host y switch, y dos módulos simples de interconexión denominados bus y hub.

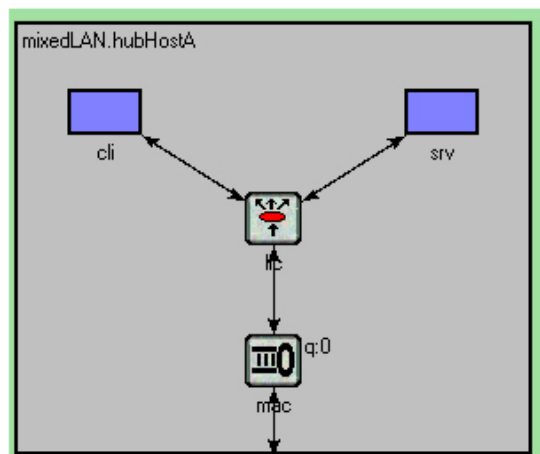


Figura 11 Simulación de una red LAN perteneciente al INET Framework Demo For OMNET/OMNET++, módulo compuesto hub.

Representación de un módulo Host. Esta figura pertenece a la representación de los módulos simples que componen el módulo compuesto de nombre host, y que se compone los módulos cli, srv, llc y mac.

Cada uno de los módulos se comunican entre sí mediante las conexiones que se definen en el fichero NED. NED es el lenguaje que se utiliza para crear las redes. Es un lenguaje de alto nivel orientado a objetos y muy sencillo de manejar. Mediante este código se crean las conexiones y los elementos de la red. Para poder ser compilado junto con los módulos, lo que se realiza es pasarlo a código C++ mediante un compilador proporcionado denominado “nedtool”. Gracias a las últimas revisiones del entorno de simulación, la última versión 3.0 y sucesivas incorpora un método que permite que los ficheros sean cargados dinámicamente desde el entorno. El compilador nedtool dará a su salida un fichero con la estructura “nombreficheroned_n.cc”. Si es usuario de Windows, conviene que cambie la extensión de *.cc a *.cpp. Este fichero se debe cargar en el proyecto o compilar con el resto de ficheros para obtener el ejecutable.

Los módulos se comunican entre sí mediante conexiones que se establecen en el código NED. Para facilitar el establecimiento de las conexiones entre módulos se emplea un editor visual muy sencillo de usar y muy útil denominado GNED. Es un editor de tipo visual que tiene dos opciones, pantalla del código, y pantalla gráfica de la red.

Las conexiones se pueden realizar entre módulos simples o entre módulos compuestos o incluso de módulo simple a módulo compuesto como se puede ver en las figuras Figura 12 y Figura 13 .

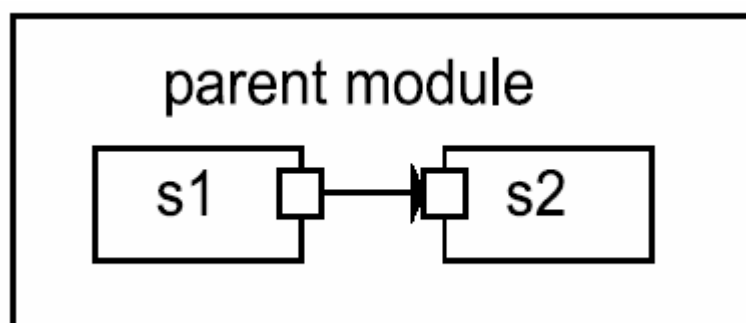


Figura 12 Ejemplo de conexión entre dos módulos simples¹².

El módulo simple denominado s1 tiene una puerta ("gate") de tipo salida ("out") mientras que el módulo s2 tiene una puerta de tipo entrada ("in"). Ambas puertas se comunican mediante una conexión ("connections") establecida entre ambas puertas.

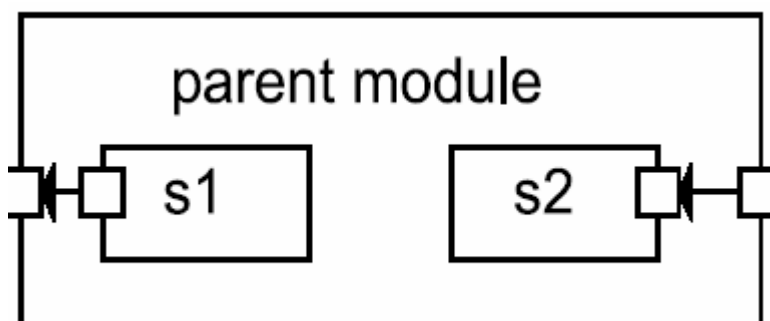


Figura 13 Ejemplo de conexión de dos módulos simples pertenecientes a un módulo compuesto al exterior¹³.

Los módulos s1 y s2 no se encuentran conectados entre sí, sin embargo si que se conectan hacia el módulo compuesto al que pertenecen. El módulo s1 tiene una puerta de salida y el s2 una de entrada. El módulo s1 podrá enviar un mensaje hacia el módulo compuesto al que pertenece y a continuación éste lo enviará hacia el exterior que será otro módulo perteneciente a la red que se simula.

¹² Imagen extraída del manual de OMNET de Andrés Varga.

¹³ Imagen extraída del manual de OMNET de Andrés Varga.

Cada uno de estos enlaces, se denominan en el fichero de topología de red *connection*. Además se permite que se puedan introducir parámetros en la comunicación como *propagation delay*, *bit error rate* y *data rate*.

Otros parámetros pueden ser introducidos en la simulación mediante el fichero de comunicación denominado *omnetpp.ini*. En este fichero se escriben principalmente los parámetros que corresponden a cada módulo. Esto se explicará más adelante con un ejemplo realizado.

Además el entorno introduce la clase *cOutVector* y funciones para la realización de ficheros de estadísticas sobre el funcionamiento de la red. Estos ficheros después pueden ser visualizados desde los programas de visualización de estadísticas que incluye el entorno.

Como se ha mencionado en líneas anteriores, OMNET++ es un simulador de eventos discretos. Esto quiere decir que es un simulador en el cual los estados cambian en instantes discretos en el tiempo, y los eventos toman un tiempo nulo en ser realizados. Nada ocurre entre dos eventos consecutivos, no hay ningún cambio de estado entre estos dos eventos, al contrario que en los sistemas continuos.

6.2.1 Los Módulos en OMNET++

Como se ha mencionado con anterioridad, los módulos que pueden ser creados con el entorno son de dos tipos, simples y compuestos. Esto permite al desarrollador un amplio número de opciones a la hora de crear una simulación. La diferencia entre ambos reside en que los compuestos contienen uno o varios submódulos, mientras que los simples son el nivel más bajo de la jerarquía de módulos. La funcionalidad de los mismos se añade mediante el código desarrollado en C++. En la Figura 14 se puede ver un ejemplo de estructuración de módulos.

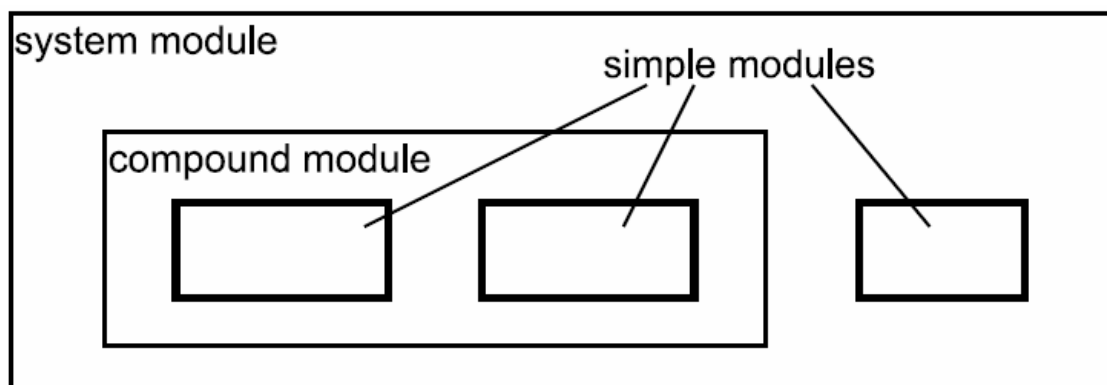


Figura 14 Sistema con varios módulos¹⁴.

Ejemplo de sistema que representa un módulo compuesto formado por dos módulos simples y un módulo simple independiente.

El método para comunicar estos módulos es el paso de mensajes entre ellos. Los mensajes pueden representar tramas o paquetes de una red real, pudiendo incluir diferentes estructuras de datos. Para enviar los mensajes se puede realizar de dos formas, mediante conexiones establecidas por un camino predefinido, o a través de puertas y conexiones. También se incluye la funcionalidad para que los mensajes se puedan enviar mensajes a sí mismos, mensajes que reciben el nombre de “self-messages” y que en la mayoría de los casos se usan para implementar temporizadores.

Las puertas son el método de conexión de cada módulo con el exterior, y son el punto por el que llegan los mensajes. Se pueden definir tantas puertas como se quiera, y son de dos tipos, de salida (output) o de entrada (input). A cada puerta le corresponde una conexión, en la que se enlaza la puerta de salida de un módulo con la de entrada de otro. Las conexiones pueden ser de módulo simple a simple, simple a compuesto, o compuesto a simple. Las conexiones establecidas entre módulos simples reciben el nombre de “rutas” y hay que destacar que los módulos compuestos actúan de forma transparente con respecto a los mensajes.

Los módulos además pueden tener parámetros asignables desde el fichero de configuración de la ejecución denominado omnetpp.ini. Esta opción permite al usuario una gran flexibilidad a la hora de simular redes con diferentes parámetros y diferentes resultados. Se puede simular por ejemplo una red con diferentes grados de congestión, observando los resultados a la

¹⁴ Imagen extraída del manual de OMNET de Andrés Varga.

salida.

6.2.1.1 Modelado de transmisiones de paquetes

Para facilitar la simulación de redes de paquetes, el entorno añade tres parámetros comúnmente utilizados, que son *propagation relay* (retraso de un mensaje en su paso a través de la red), *bit error rate* (tasa de error de bit), y *data rate* (tasa de datos), y que se pueden añadir de manera opcional en la conexión.

6.2.2 Programación de algoritmos

La forma de añadir funcionalidad a los módulos es mediante la programación de algoritmos en ellos. Estos algoritmos se deben programar mediante funciones en C++, que serán llamadas ante la llegada de un mensaje, y que después de manejar el mismo podrán devolver una respuesta o realizar un evento determinado por esta función. Tanto con los mensajes, como con módulos pueden aplicarse los conceptos de la programación orientada a objetos de polimorfismo, herencia, etc. Hay que tener en cuenta que si los módulos están implementados con `handleMessage`, es necesario que alguno (al menos uno) de los módulos genere un mensaje para que funcione la simulación. Si lo que se quiere es que genere un mensaje cada cierto tiempo lo mejor es implementar un auto-mensaje con un retardo. Esto se debe a que un módulo sólo actúa cuando recibe un mensaje, permaneciendo inactivo cuando no recibe nada.

6.2.3 Cómo usar OMNET++

Si se quiere comenzar a desarrollar una simulación con este entorno sólo hay que tener unas pequeñas nociones de cómo funciona, y de los ficheros necesarios para la implementación. En caso de no haber trabajado nunca con el entorno, un buen consejo sería comenzar leyendo el tutorial “Tic Toc Tutorial for OMNeT++” del sitio oficial de OMNET++. Este ejemplo se instala con el entorno en la ruta “`../omnetpp/samples/tictoc/`” lo que permite que sea fácilmente accesible. Además el tutorial explica detalladamente con ejemplos, y de una forma práctica cómo comenzar a implementar una aplicación de este tipo.

La creación de una simulación OMNET++ exige como mínimo un fichero `*.ned` y uno o más `*.cpp` que modelen el funcionamiento del módulo simple. Si lo que se quiere es crear mensajes propios para la simulación también se debe incluir un fichero `*.msg` o tantos como sean necesarios. Estos ficheros servirán para la creación de mensajes adaptados o destinados a esta simulación, que serán luego pasados a C++ por el compilador de mensajes. Como se verá más adelante, se tienen dos opciones, crear nuevos mensajes o derivarlos de la clase base

cMessage. Para pasar los ficheros de descripción de mensajes a código C++ se debe usar el programa **opp_msgc**.

Los ficheros *.ned son aquellos en los que se define la topología de la red, los módulos, puertas, conexiones, y posición en la visualización del entorno¹⁵. Se pueden escribir de dos formas, usando cualquier editor de texto, o usando el editor visual GNED. Para crear ficheros C++ se debe usar el programa **nedtool**.

Para los ficheros *.cpp que modelen un módulo, lo primero es crear una clase derivada de la clase base cSimpleModule. Para aportar la funcionalidad al módulo habrá que modificar las funciones que vienen incorporadas en la clase base, e introducir si se ve necesario nuevas funciones.

El entorno aporta dos elementos principales para la creación de simuladores. Uno es el kernel de simulación o núcleo de simulación. Es el código que controla la simulación y la librería de clases de simulación. Está escrita completamente en C++, y forma una librería con extensión *.a para Linux, o *.lib para sistemas Windows. El otro elemento importante a destacar son las interfaces de usuario para la ejecución de simulaciones. Están igualmente escritas en C++, y compiladas en un fichero de librería con las mismas extensiones que el anterior.

Para ejecutar el programa implementado, lo que se debe hacer es crear un fichero de configuración llamado **omnetpp.ini** que contenga los parámetros necesarios para la aplicación. Además, el fichero de configuración puede permitir tener varias opciones de configuración denominadas *run* que se pueden elegir al ejecutar. Por ejemplo se quiere realizar un simulador para un simulador del protocolo Aloha¹⁶, se pueden tener varias configuraciones. Éstas podrían ser una alta carga de equipos, media, y baja carga, lo que daría diferentes resultados a la salida.

¹⁵ Se puede especificar en qué lugar aparece un módulo o en el caso de varios módulos en array la presentación en diferentes estructuras, ya sea anillo, fila, matriz, aleatoria. Para más información ver el apartado dedicado a la animación y visualización gráfica de las redes al final de este capítulo.

¹⁶ Existe un ejemplo de Aloha en OMNET++ Demo Simulations que se distribuye junto con el entorno.

6.3 El lenguaje NED

6.3.1 Vista rápida del lenguaje NED

La topología de una simulación viene especificada por el lenguaje NED, que facilita enormemente el trabajo de descripción de una red. Una descripción de una simulación se compone de diferentes elementos denominados canales, módulos simples o compuestos, puertas, y parámetros. Estos deben ser escritos en un fichero con extensión *.ned.

6.3.1.1 Componentes de una descripción NED

Los componentes que pueden estar presentes en la declaración de una red o modelo pueden ser directivas de importación, definición de canales, definiciones de módulos simples y compuestos y definiciones de red.

6.3.1.2 Palabras reservadas

El usuario debe tener en cuenta no usar alguna de las palabras reservadas en este lenguaje para dar nombre a los módulos o redes que cree. Las palabras reservadas son las siguientes:

```
import channel endchannel simple endsimple module endmodule error delay datarate
const parameters gates submodules connections gatesizes if for do endfor network
endnetwork nocheck ref ancestor true false like input numeric string bool char xml
xmldoc
```

6.3.1.3 Identificadores

Con identificadores se quiere denominar a todos los nombres de módulos, canales, etc., que aparezcan en el fichero *.ned. Deben contener letras del alfabeto (a-z/A-Z) o números (0-9) o el carácter se barra baja “_”, sin embargo no pueden comenzar por un número.

6.3.2 Directivas de importación

Para importar declaraciones de otras redes lo único que es necesario es añadir la directiva import. Por ejemplo, suponiendo el caso de estar simulando una red WAN de ordenadores, en la que ya se tiene realizada la descripción de una red LAN, El paso a seguir es incorporar la red LAN en la descripción de la WAN, mediante la siguiente declaración: import “LAN”;

6.3.3 Descripción de canales

Esta sentencia se emplea para la especificación de un tipo de canal con unas características determinadas. Se pueden asignar tres tipos de valores, delay (retardo en segundos), error (tasa de error de bit) y datarate (tasa de transmisión de datos en bit/segundo). Se debe crear de la

forma siguiente:

```
channel canalEjemplo
  delay 0.001 // 1 ms
  error 1e-8
  datarate 9600 // 9600 bit/segundo
endchannel
```

6.3.4 Módulos simples

Los módulos simples son los que llevan toda la carga de la simulación. Para la definición de un módulo simple se debe seguir la siguiente estructura:

```
simple SimpleModuleName
parameters:
//...
gates:
//...
endsimple
```

Detrás de parámetros se deben escribir todos los parámetros de los módulos, y detrás de gates se deben escribir todas las conexiones de los módulos. Por ejemplo queremos crear un módulo denominado tarjetaRed que simulará la tarjeta de red ethernet de un ordenador, y otro módulo que representa el nivel de enlace.

```
simple tarjetaRed
parameters: //...
gates:
in: deRed, deNivelEnlace;
out: aRed, aNivelEnlace;
endsimple
```

```
simple nivelEnlace
parameters: //...
gates:
in: deTarjetaRed, deNivelRed;
out: aTarjetaRed, aNivelRed;
endsimple
```

En este caso tenemos un módulo que se comunica con el modelo que representa la capa superior (nivelEnlace) y dos conexiones en cada módulo que lo conectan en el caso de la tarjeta hacia la red y hacia el nivel de enlace, y en el caso del módulo de enlace hacia la tarjeta y hacia el nivel de red. In corresponde a la puerta de entrada y out corresponde a la de salida. Mediante los nombres de las puertas se pueden encaminar los mensajes con la función send.

Esta descripción de módulos se puede ver representada en la Figura 15.

Los parámetros de cada módulo pueden ser números, constantes, booleanos, xml, o incluso cadenas de caracteres (string). A cada parámetro se le asigna un valor en el momento de cargar la red con el archivo omnetpp.ini. También se pueden generar números aleatorios.

Las puertas se definen o de entrada (in) o de salida (out). Si en la declaración se quiere realizar un array de puertas, el nombre de la puerta irá seguida de dos corchetes [].

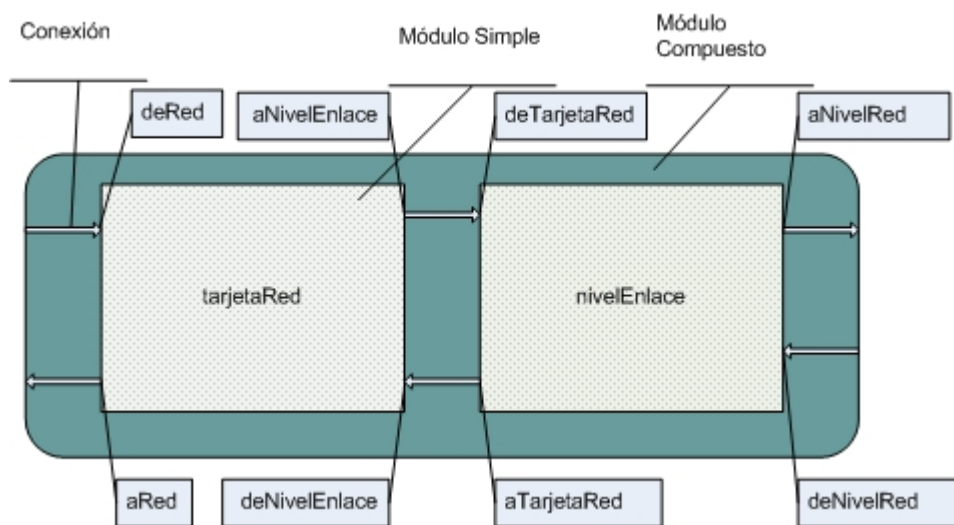


Figura 15 Ejemplo de configuración de módulos.

En la figura se puede ver un ejemplo de configuración de módulos para una simulación. Se han creado dos módulos simples denominados `tarjetaRed` y `nivelEnlace` integrados en un módulo compuesto que estará unido a otro módulo llamado `red` (mediante las puertas de la izquierda) y a otro llamado `NivelRed` (mediante las puertas de la derecha).

6.3.5 Módulos compuestos

Los módulos compuestos siguen una apariencia y declaración similar a la de los simples, y pueden estar compuestos por uno o varios módulos simples, que reciben el nombre de submódulos. Los módulos compuestos carecen de funcionalidad, la carga de la simulación la soportan los módulos simples que lo componen, por lo que no se necesita implementar el código C++ para ellos. Son casi como cajas transparentes de cara a los módulos simples que la componen. Es importante que la declaración de los módulos simples sea previa a los compuestos ya que si no es así se puede llegar a tener problemas a la hora de compilar.

La definición de parámetros es exactamente igual que para un módulo simple, al igual que la

descripción de una puerta, por lo que se saltará la descripción de la misma. Lo novedoso de este tipo de “cajas” con respecto a las simples son las declaraciones de los submódulos y la de las conexiones.

6.3.5.1 Submódulos

La sentencia submodules: da paso a la definición de uno o varios submódulos en un módulo compuesto. A grandes rasgos, la declaración posee las mismas características que la declaración de un módulo simple, con dos elementos, parameters y gatesizes. Es incluso posible la declaración de un array de módulos, por ejemplo un módulo compuesto por un array de 10 módulos simples, o por un valor introducido desde el fichero de configuración.

6.3.5.2 Conexiones

Las conexiones se deben de establecer entre dos puertas de diferentes módulos simples. La sentencia connections da paso a la declaración de estas conexiones. La conexión se establece la siguiente estructura¹⁷:

```
Modulo.Puerta-->[error 1e-9 delay 0.001 -->] Modulo.Puerta
```

```
Modulo.Puerta-->[tipo canal -->] Modulo.Puerta
```

Para establecer las conexiones también se pueden emplear bucles de tipo for, o incluso sentencias de tipo if. Si volvemos al ejemplo que estábamos utilizando de la tarjeta de red, la declaración del módulo quedaría así:

```
module Fisico_Enlace
parameters: //...
gates:
in: deRed, deNivelRed;
out: aRed, aNivelRed;

submodules:
  tarjeta: tarjetaRed
  //..
  enlace: nivelEnlace
  //..
connections:

tarjeta.aNivelEnlace-->enlace.deTarjetaRed ;
```

¹⁷ Los elementos entre corchetes [...] son opcionales.

```
tarjeta.deNivelEnlace<--enlace.deTarjetaRed
deNivelRed-->enlace.deNivelRed
aNivelRed<--enlace.aNivelRed
deRed-->tarjeta.deRed
aRed<--tarjeta.aRed
endmodule
```

6.3.6 Definición de una red

Para definir redes, que es el nivel más alto de la jerarquía de módulos, se debe incluir el siguiente código, donde `tipoRed` define a un módulo compuesto que compone a uno o varios módulos compuestos o simples:

```
network nombreRed: tipoRed
parameters: //...
endnetwork
```

Poniendo como ejemplo la red de telefonía móvil que se ha desarrollado en este proyecto, la red se llamaría `gsmsim`, y el módulo compuesto que integra al móvil, la antena y los demás equipos de gestión y control se le denomina `GSMSIM`. En el caso de que los valores se quieran leer de un archivo de configuración o de teclado durante la ejecución, se debe incluir la función `input()`. En el primero de los parámetros pide el número de estaciones dando 1 como valor por defecto.

```
network gsim : GSMSIM
parameters:
    number_bts = input(1,"Number of stations:"),
    number_ms = input(1,"Number of cars:"),
    xwidth = input(10000,"Width of the plane [m]:"),
    ydepth = input(10000,"Depth of the plane [m]:");
endnetwork
```

6.4 Los módulos simples

Los módulos simples son la unidad mínima de simulación. En ellos reside todo el peso de la simulación, por lo que son la parte más importante de la misma. Su funcionalidad está programada en C++, y se implementan mediante una clase derivada de la clase `cSimpleModule`. Esta clase contiene algunas funciones que se deben implementar como mínimo para poder funcionar. Las funciones mínimas necesarias son las siguientes:

- `void initialize()`
- `void handleMessage(cMessage *msg)`
- `void activity()`
- `void finish()`

La función `initialize()` es llamada cada vez que inicia la simulación, y en ésta deben estar incluidas las inicializaciones de los principales parámetros de información que la red necesita para funcionar, mientras que `finish()` hace el proceso contrario, borra todos los parámetros de la simulación.

Las funciones `handleMessage()` y `activity()` son funciones muy similares aunque con unas ciertas particularidades que las diferencia. Actualmente, con las últimas versiones se ha dejado de utilizar la segunda ya que da lugar a simulaciones más lentas y problemáticas. La tendencia es a utilizar la función `handleMessage`.

La principal aplicación que tiene es la de responder ante la entrada de mensajes en el módulo. Cada vez que un mensaje llega al módulo, esta función lo evalúa, lo trata, y actúa en consecuencia.

Para grabar datos de tiempo de simulación se debe emplear el tipo `simtime_t`, que es un tipo de `double`. Los mensajes son procesados en función de la llegada en tiempo de simulación, y en caso de llegar al mismo tiempo se ejecutan primero aquellos con menos valor de prioridad.

6.4.1 Creación de un módulo simple

La creación de un módulo simple en código C++ consiste básicamente en tres pasos, declarar una clase derivada de `cSimpleModule`, registrar este tipo de módulo mediante la macro `Define_Module(nombre de la clase)`, y por último implementar la clase.

En la declaración de la clase es importante destacar la inclusión de la macro `Module_Class_Members` (nombre de la clase, clase base, tamaño de la pila). Esta macro automatiza la creación de parte del código, en concreto la creación del constructor de la clase. El tamaño de la pila debe ser cero si se usa la función `handleMessage()`.

6.4.2 Añadir funcionalidad a un módulo

Para añadir funcionalidad a un módulo se debe modificar la función `handleMessage()`. Esta función actúa cada vez que un mensaje llega a una de las puertas de entrada del objeto de la clase. Hay que tener en cuenta que los mensajes con esta función no comienzan automáticamente, ya que sólo con `activity()` el kernel crea mensajes de inicio, por lo que esto quiere decir que se debe crear un auto-mensaje desde la función `initialize()` para que “despierte” el módulo y comience a trabajar.

Las ventajas de la función `handleMessage()` con respecto a `activity` son la utilización de menos memoria, además de una simulación mucho más rápida. En aquellos casos en los que crece el número de módulos a un número muy elevado, sencillamente los módulos con `activity` no van a funcionar o lo van a hacer en unas condiciones peores.

Hay que destacar que con la función `handleMessage`, no se pueden emplear las funciones como `receive()` y `wait()` propias de `activity`. Las únicas funciones que se pueden usar son las funciones `scheduleAt()`, `cancelEvent()`, y `send()`.

Las funciones `initialize()` y `finish()` son llamadas al principio y al final de una simulación. Son propias de cada módulo, y son muy importantes a la hora de inicializar parámetros o borrar su valor para una nueva simulación. Hay que tener en cuenta que no son lo mismo que un constructor y un destructor, y que deben implementarse en todos los casos. En ningún caso se debe colocar código relacionado con la simulación en el constructor o en el destructor, sino que se debe limitar en estos a implementar la creación y destrucción de elementos de esta clase. Normalmente en la función de finalización se incluye en código necesario para escribir estadísticas.

6.4.3 Enviar y recibir mensajes

Los mensajes en OMNET++ son objetos de la clase `cMessage`, de derivadas que hayan sido creadas, o de nuevos mensajes creados. Para crear un nuevo mensaje se debe crear un nuevo objeto o instancia de mensaje con el operador `new` de C++. Para borrarlo se debe emplear el operador `delete`.

Para enviar un mensaje hacia otro módulo es necesario que se emplee la función `send()`. Existen tres tipos de funciones `send()` con diferentes parámetros:

```
send (cMessage * msg, const char * gateName, int index = 0)
send (cMessage * msg, int gateId)
send(cMessage * msg, cGate * gate)
```

Las tres funciones reciben un puntero al mensaje que se quiere enviar, diferenciándose en la forma en que se referencia la puerta a la que va destinada el mensaje. La función usada para este proyecto ha sido la primera de las que se indica. El parámetro `char * gateName` lleva el nombre de la puerta a la que se quiere enviar el mensaje. Por ejemplo “`to_bts`”, “`to_ms`”. El tercer parámetro es interesante en el caso de que las puertas sean arrays. Con este número se consigue elegir a cual de todos se quiere enviar. En el caso de la aplicación desarrollada en

este proyecto, las estaciones móviles (MS) o teléfonos móviles son un array de elementos. Si se quiere que llegue un mensaje al MS[2], se debe poner en el parámetro int index un valor igual a 2. Por defecto el valor asignado es 0.

Se debe tener en cuenta un aspecto. En el envío de mensajes no se puede enviar dos veces el mismo mensaje. Esto quiere decir que si se ha aplicado un objeto mensaje a una función send, y se quiere a continuación enviar el mismo, se debe realizar una copia del original mediante el operador dup() de la clase cMessage antes de realizar el primer envío. Esto se puede aplicar para retransmisiones, o para el caso de mensajes a varios módulos (broadcast).

Existen otros dos tipos de funciones para enviar mensajes directamente o para enviar mensajes con retardo. La primera de ellas emplea la función sendDirect(). Se emplea para enviar un mensaje a un módulo sin que se tenga en cuenta la puerta ni la conexión. La segunda permite enviar mensajes con un retardo introducido como parámetro en la función. Esta función es sendDelayed(). A continuación se presentan las declaraciones de estas funciones.

```
sendDelayed (cMessage * msg, double delay, const char * gateName, int index = 0)
sendDelayed (cMessage * msg, double delay, int gateId)
sendDelayed (cMessage * msg, double delay, cGate * gate)
sendDirect (cMessage * msg, double delay, cModule * mod, int gateId)
sendDirect (cMessage * msg, double delay, cModule * mod,
const char * gateName, int index)
sendDirect (cMessage * msg, double delay, cGate * gate)
```

Por último se debe comentar el caso en el cual se quieran realizar auto-mensajes. Para ello se debe emplear la función scheduleAt(). Los auto-mensajes son mensajes que se envía un módulo a sí mismo. Se pueden utilizar para implementar temporizadores, o para despertar a un módulo al inicio de la simulación. En el caso de la aplicación que se presenta, los módulos que corresponden a teléfonos móviles se inician con un auto-mensaje denominado MOVE_MS. Gracias a este mensaje se puede actualizar la posición de cada móvil en un tiempo definido.

6.5 Los mensajes

Los mensajes pueden ser creados a partir de la clase cMessage. Cada mensaje tiene unos parámetros definidos por los atributos de la clase, y que son los siguientes:

- *name*: es un atributo de tipo cadena. Define el nombre del mensaje.

- *kind*: es el atributo de tipo de mensaje. Se pueden usar valores mayores o iguales a cero.
- *length*: se usa para definir el tamaño de un mensaje.
- *bit error flag*: se usa para simular un error en la comunicación, para ello se pone a un valor true según la probabilidad de error de bit asignada a la conexión o canal en el fichero .ned.
- *priority*: indica la prioridad del mensaje.

Para crear un mensaje se debe llamar al constructor de la clase `cMessage` con los parámetros del nombre de mensaje (una cadena de caracteres) y un número de identificador de tipo de mensaje (por ejemplo se asigna a los mensajes TCP el valor 0 y a los UDP el 1 para una comunicación con una red LAN):

```
cMessage *msg = new cMessage("MessageName", msgKind);
```

Para acceder a los atributos uno a uno, la clase `cMessage` implementa unos métodos para lectura y escritura de los mismos. Las funciones son las siguientes, las primeras son para dar valor a los atributos y las siguientes para acceder a ellos en forma de lectura.

```
msg->setKind( kind );  
msg->setLength( length );  
msg->setPriority( priority );  
msg->setBitError( err );  
msg->setTimestamp();  
msg->setTimestamp( simtime );  
  
int k = msg->kind();  
int p = msg->priority();  
int l = msg->length();  
bool b = msg->hasBitError();  
simtime_t t = msg->timestamp();
```

Otra posibilidad que ofrece el entorno de simulación es el de la creación de mensajes que sean enviados y recibidos por el mismo módulo. La creación y asignación de parámetros son iguales para el resto de mensajes, sin embargo la forma de enviarlos es diferente, mediante la función `schedule`:

```
scheduleAt()
```

La librería permite también la utilización de mensajes encapsulados. Para ello se debe llamar al método `encapsulate` de la clase `cMessage` y pasarle como atributo por valor el puntero del

mensaje que se quiere encapsular. Por ejemplo, se quiere mandar un mensaje entre dos módulos compuestos, uno simula un ordenador y otro un hub. Cada uno de ellos está compuesto de dos módulos, uno que simula el nivel de enlace y otro el de red. Se quiere enviar un mensaje del nivel de red de uno al de enlace de ese mismo y que el nivel de enlace lo encapsule en otro. El mensaje que encapsula se envía a continuación por la conexión y el módulo nivel de enlace del destino lo recoge lo desencapsula y lo envía al módulo que simula el nivel de red. El código fuente sería el siguiente:

Módulo nivel red:

```
cMessage * msgRed = new cMessage ("NIVEL RED");  
send (msgRed,"a_enlace");
```

Módulo nivel enlace

```
cMessage * msgEnlace = new cMessage ("NIVEL ENLACE");  
msgEnlace->encapsulate (msgRed);
```

Módulo nivel enlace de destino

```
cMessage * msgRed = msgEnlace->decapsulate();
```

Otra característica que permite el entorno es que la longitud de los mensajes se suma cuando se realiza una encapsulación. Si se tiene un mensaje de 20 bytes y se encapsula en otro de 4, el total sería 24 bytes.

La forma de añadir parámetros a un mensaje es mediante la clase cPar. El método addPar() de la clase cMessage, pasándole como atributo un puntero a un tipo cPar, añade un parámetro al mensaje. En la línea de código siguiente se añade un parámetro llamado "numms" al mensaje de nombre cl3i.

```
cl3i->addPar(* new cPar("numms") = iMS);
```

Para recoger el valor en el destino lo que se debe hacer es crear una variable que recoja el valor, y recogerlo con el método par de la clase cMessage.

```
int iMS = msg->par("numms");
```

6.5.1 Definición de mensajes

Otro camino para crear nuevos tipos de mensajes es mediante la definición de éstos. Para ello se debe crear un fichero con extensión *.msg y después pasarlo a C++ con el programa opp_msgc. En futuras versiones de OMNET++ está previsto que el programa nedtool

incorpore esta funcionalidad. El fichero deberá contener una declaración del tipo:

```
message MiMensaje
{
    fields:
        int parametro1;
        int parametro2;
        //...
};
```

6.6 La librería de simulación

Hasta ahora se ha hablado de la clase `cModule` y la `cMessage` que aporta la librería. Sin embargo, aparte de estas, también añade un gran número de clases y de funciones que son muy útiles para la simulación.

Para no extender excesivamente esta sección se va a explicar a continuación únicamente las funciones para variables estadísticas, las clases que permiten sacar resultados después de la simulación, y algún otro detalle de importancia. En el manual de OMNET++ se puede conseguir en el capítulo sexto “The Simulation Library” una completa referencia de las librerías, donde se explican las funciones estadísticas, la clase `cPar`, `cArray`, `cQueue`, `cTopology`, entre otras, y con más detalle que en este capítulo.

6.6.1 Las funciones estadísticas

Para las funciones estadísticas, OMNET++ incluye numerosas funciones que permiten trabajar con diferentes tipos de distribuciones de variables aleatorias. Existen principalmente dos tipos, las continuas y las discretas, y abarcan distribuciones uniformes, exponenciales, normales, etc. Son las siguientes:

Distribuciones continuas:

- uniform(a, b, rng=0)** distribución uniforme en el rango [a,b)
- exponential(mean, rng=0)** distribución exponencial
- normal(mean, stddev, rng=0)** distribución normal con media y desviación estándar
- truncnormal(mean, stddev, rng=0)** distribución normal truncada
- gamma_d(alpha, beta, rng=0)** distribución gamma con parámetros $\alpha > 0$, $\beta > 0$
- beta(alpha1, alpha2, rng=0)** distribución beta con parámetros $\alpha_1 > 0$, $\alpha_2 > 0$
- erlang_k(k, mean, rng=0)** distribución Erlang con $k > 0$
- chi_square(k, rng=0)** distribución chi-cuadrada con $k > 0$ grados de libertad
- student_t(i, rng=0)** distribución student-t con $i > 0$ grados de libertad
- cauchy(a, b, rng=0)** distribución de Cauchy con parámetros a,b donde $b > 0$

triang(a, b, c, rng=0) distribución triangular con parametros $a \leq b \leq c$, $a \neq c$
lognormal(m, s, rng=0) distribución lognormal con varianza $s > 0$
weibull(a, b, rng=0) distribución Weibull con $a > 0$, $b > 0$
pareto_shifted(a, b, c, rng=0) distribución de Pareto

Distribuciones discretas:

intuniform(a, b, rng=0) distribución uniforme from a..b
bernoulli(p, rng=0) Bernoulli con probabilidad $0 \leq p \leq 1$ (1 con probabilidad p y 0 con probabilidad (1-p))
binomial(n, p, rng=0) binomial con $n \geq 0$ y $0 \leq p \leq 1$
geometric(p, rng=0) geométrica con $0 \leq p \leq 1$
negbinomial(n, p, rng=0) binomial negativa con $n > 0$ and $0 \leq p \leq 1$
poisson(lambda, rng=0) distribución de Poisson

6.6.2 Imprimir en pantalla

La aplicación permite que se puedan imprimir datos en la ventana del programa. El método que se debe seguir es similar a las llamadas a cout empleadas en C++. En este caso se debe emplear el objeto ev de la clase cEnvir. A continuación se añade una muestra:

```
char c1 = 'a';  
char c2 = 'b';  
int num = 2;  
  
ev << " Caracteres a imprimir " << endl  
    << " Número: " << num << endl  
    << " Caracteres: " << c1 << c2 << endl;
```

6.6.3 Tiempo de simulación

En las aplicaciones de OMNET++ hay que diferenciar entre tiempo real y tiempo simulado. Cuando se refiere a tiempo real o a tiempo de cpu, se refiere al tiempo real que tarda un ordenador en ejecutar la aplicación. El tiempo simulado se refiere al tiempo que se simula en una ejecución. Para aclarar estos conceptos es mejor poner un ejemplo. La aplicación gsmSim tarda 40 segundos en realizar una simulación de 1 hora. Esto quiere decir que tarda 40 segundos reales en simular lo que sucedería en la red de GSM en una hora. Para establecer un límite a la simulación se debe introducir en el fichero de configuración ambos valores.

Cuando se pretende establecer eventos en determinados instantes de tiempo el camino es utilizar el tipo simtime_t. Este tipo, declarado como un tipo double en la librería se emplea para almacenar el valor de tiempo de simulación. Cuando se cree una variable de este tipo, se puede llamar a la función simTime() que devuelve el valor actual de tiempo de simulación en

un tipo `simtime_t`.

```
simtime_t tiempo = simTime();
```

6.6.4 Grabar resultados

Los resultados de la simulación se pueden grabar para luego ser analizados a posteriori. Los datos de salida principalmente pueden ser de dos tipos, escalares y vectoriales. En el primero de los casos se graba el valor, pero no el tiempo en el que se ha grabado, por lo que únicamente aparece el dato. En el caso de los vectoriales, se graba el valor y además la marca de tiempo. Los datos escalares son muy útiles para mostrar valores totales, o porcentajes. Los datos vectoriales son más útiles para ver gráficas de la evolución de la simulación en el tiempo. Para los dos tipos se generan unos ficheros de salida con extensión `*.sca` y `*.vec` y con un nombre definido en la configuración.

Los datos escalares se graban generalmente al final de la simulación (lo cual no implica que no se pueda grabar en cualquier otro momento de la misma). Lo habitual es que se graben desde la función `finish()` de cada módulo. Para ello se debe realizar una llamada a la función `recordScalar`.

```
recordScalar("Nombre", valor);
```

En el caso de los vectoriales, para la grabación de los resultados de la simulación se emplea la clase `cOutVector`, que guarda los datos de tipo vector. Cada objeto de esta clase puede ser un atributo de un módulo, o como un objeto independiente. Se debe tener en cuenta que si se crea un objeto local, este será destruido cuando salga de la función. Es decir si se crea un objeto de tipo `cOutVector` en la función de manipulación de mensajes, cuando la función finalice, también se destruirá el objeto, por lo que se perderán los datos. Si la intención es usar un objeto `cOutVector` para guardar datos de un único módulo, lo mejor es añadirlo a la declaración de ese módulo e iniciarlo en la función `initialize()`. Para darle un nombre se debe llamar al método `setName` de la clase `cOutVector` y pasarle una cadena de caracteres. Este es el nombre con el que se asociará en la visualización de resultados.

```
cOutVector nombreVector;
```

Para escribir los valores lo que se debe hacer es llamar a la función `record` de la clase `cOutVector` de la forma siguiente:

```
nombreVector.record(value);
```

La clase guarda automáticamente la marca de tiempo, por lo que no es necesario que sea el usuario el que la introduzca.

6.7 Construir, configurar y ejecutar la simulación

Los tres pasos que se deben seguir para construir cualquier tipo de simulación son la construcción, de la que ya se ha hablado previamente, la configuración de la simulación, y por último la ejecución de la misma. Se ha comentado ya cómo se debe desarrollar el código para aportar funcionalidad a los módulos y para definir la topología de la red. Sin embargo no se ha explicado cómo se organizan las librerías de OMNET++, asunto que se trata en este apartado.

Cuando se compila un fichero C++ que simula un módulo, el resultado que da es un fichero con la misma extensión que el fichero compilado y con la extensión *.obj o *.o dependiendo del sistema operativo empleado (Windows y Linux respectivamente). Estos ficheros de salida se deben enlazar con las librerías estáticas que incluye OMNET++. Se debe tener en cuenta que las extensiones también cambian en función del sistema operativo. Para Linux, se emplea la extensión *.a, y para Windows *.lib. En lo sucesivo se mencionarán únicamente las librerías para Windows, ya que son las empleadas en este proyecto. Estas librerías deben incluirse en los parámetros de link de la opción Settings del proyecto en el programa Microsoft Visual C++ 6.0, o en el caso de gcc se deben incluir en la línea de comandos empleada. Las principales son las siguientes:

- Librerías del kernel de simulación y de la librería de clases: libsim_std.lib.
- Librerías de las interfaces de usuario: tkenv.lib, cmdenv.lib, y enviro.lib. La librería enviro contiene algunas clases como la clase que permite la salida de caracteres por pantalla, y que se utiliza mediante el objeto ev. Tkenv es la librería empleada para la interfaz gráfica de la aplicación y cmdenv para la interfaz por línea de órdenes. En la Figura 16 se puede observar un diagrama de bloques de los pasos para crear una simulación con OMNET++.

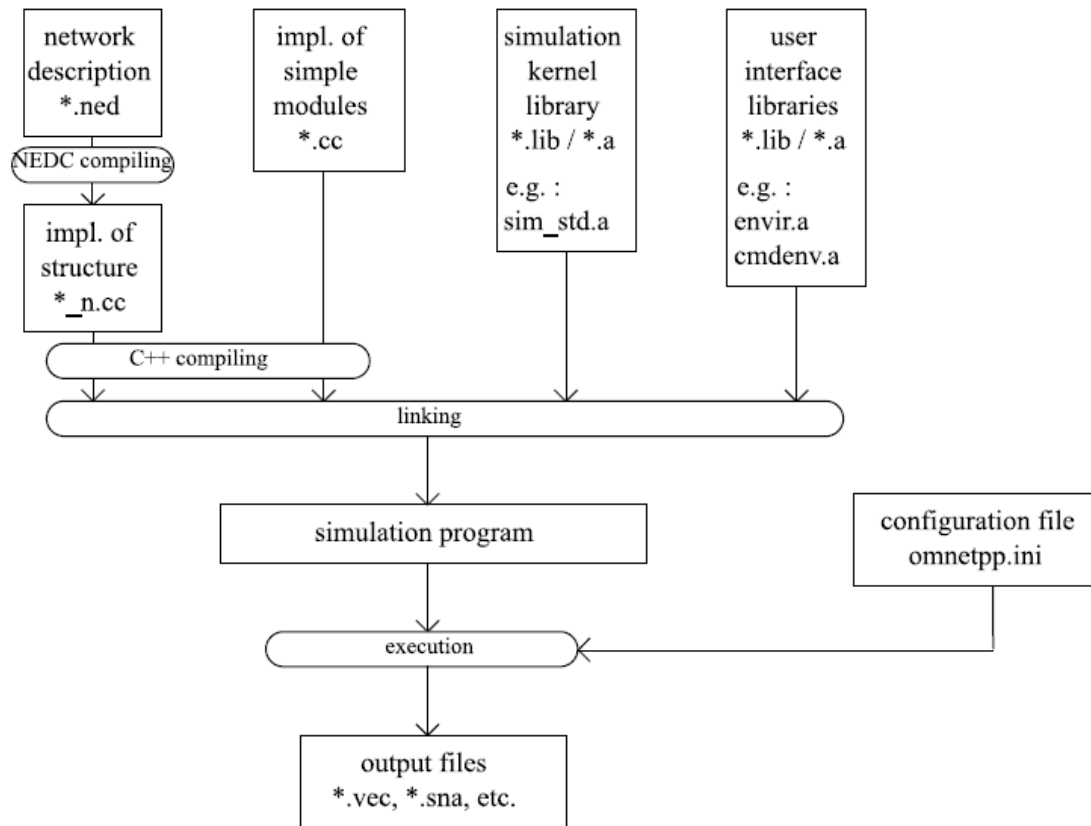


Figura 16 Pasos en la construcción, configuración y ejecución de una simulación en OMNET++¹⁸.

A grandes rasgos los pasos para obtener una simulación con OMNET++ son la construcción, la configuración y por último la ejecución. Los ficheros *.ned deben pasarse a C++ con el compilador de NED nedtool (esquina superior izquierda). Todos los ficheros C++ se deben compilar para generar los ficheros objeto (*.obj), a los que se enlazarán (linking) los ficheros de librerías estáticas (parte superior derecha). Estos pasos darán lugar al programa de simulación y darán por terminada la tarea de construcción. El siguiente paso es la configuración de la simulación, o lo que es lo mismo, la creación del fichero omnetpp.ini. Una vez realizada la configuración se procede a la ejecución que dará lugar a los ficheros de salida (bloque inferior de la figura).

¹⁸ Figura extraída del Manual de Usuario de OMNET++.

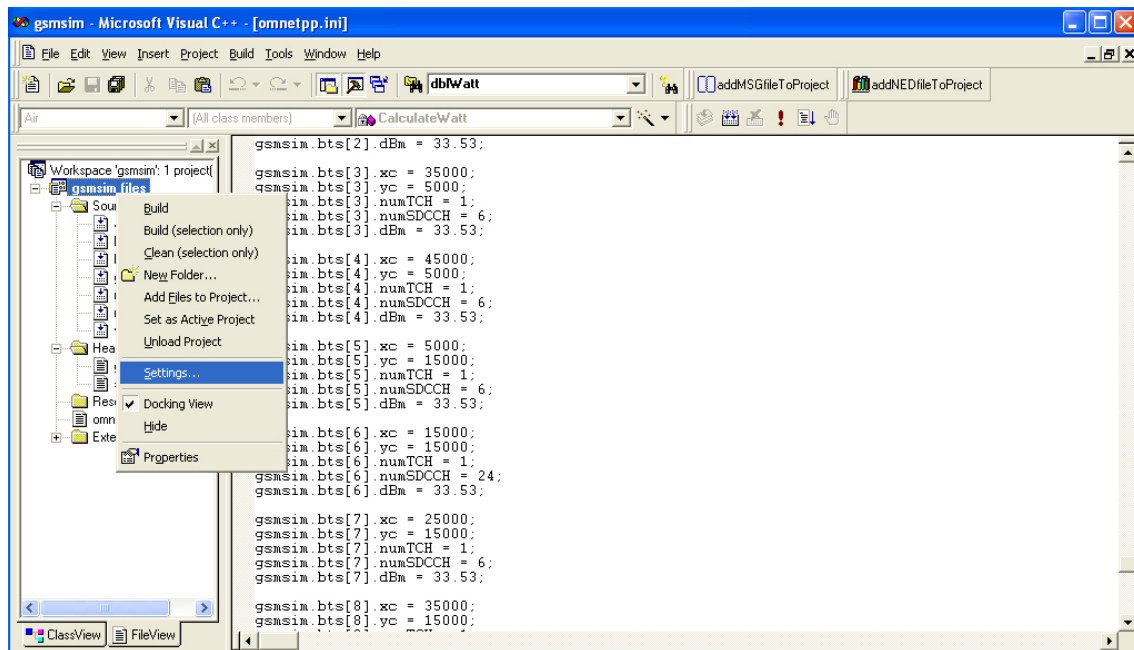


Figura 17 Ajustes del proyecto en Microsoft Visual C++ 6.0.

Para incluir las librerías en el enlazado de los objetos con este compilador de Microsoft se deben introducir en los ajustes del proyecto los nombres de las librerías. Para acceder a los ajustes, se debe pinchar con el ratón sobre el nombre del proyecto y a continuación pulsar el botón derecho para que salga el menú. La opción de ajuste (Settings-Project Settings) aparecerá en este menú y desplegará una ventana como la de la Figura 18.

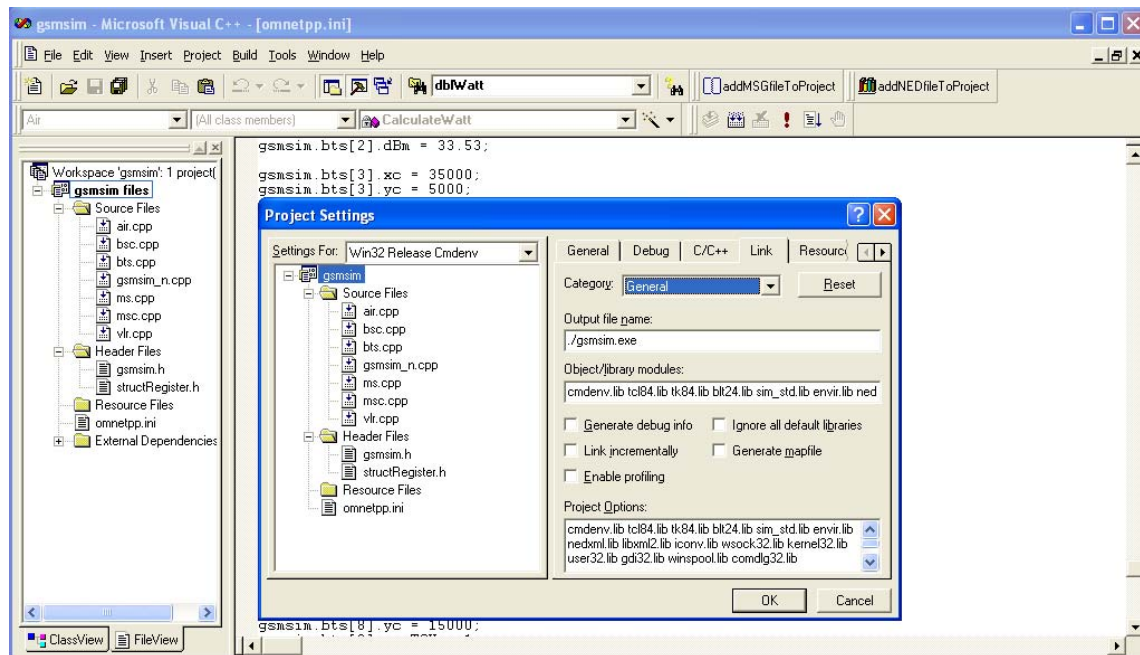


Figura 18 Ajustes del proyecto en Microsoft Visual C++ 6.0. Ventana de Project Settings.

En la casilla de Object/library modules se deben incluir los nombres de las librerías que se quieren emplear. En este caso emplea `cmdenv.lib`, `sim_std.lib`, `envir.lib`, entre otras. Merecen atención especial las librerías `tcl84.lib`, y `tk84.lib`. Van cambiando con las diferentes versiones de OMNET++ y en algunos casos puede ocurrir que sean incompatibles con la versión del compilador que se está empleando. Es recomendable descargarse las últimas versiones de estas librerías.

6.7.1 Configuración de simulaciones

Una vez comentada la construcción de simulaciones es conveniente pasar a explicar la configuración, o lo que es lo mismo, la creación y edición de un fichero `omnetpp.ini`. Para ello se pone como ejemplo el fichero de configuración empleado en una simulación del programa desarrollado para este proyecto. En algunos casos será necesario incluir otros ficheros de

configuración¹⁹ lo que se realiza mediante la directiva dentro de otro fichero *.ini:

```
#omnetpp.ini  
include nombre.ini
```

Las primeras líneas del fichero (sin tener en cuenta las líneas de inclusión) vienen seguidas de la palabra general entre corchetes. Esto indica que son parámetros generales de ajuste para la simulación. Como se puede ver a continuación en un ejemplo, las líneas hacen referencia a los ficheros de salida, tiempo de ejecución, tiempo simulado, avisos, avisos de inicialización, carga previa de ficheros NED, entre otros que pueden emplearse. Para dar una información más completa ir a la Tabla 6.

```
[General]  
preload-ned-files=*.ned  
network = gsmsim  
  
ini-warnings = no  
warnings = yes  
snapshot-file = gsmsim.sna  
output-scalar-file = gsmsim.sca  
output-vector-file = gsmsim.vec  
sim-time-limit = 3600s  
cpu-time-limit= 1000m
```

Parámetros generales [General]	
Nombre parámetro	Descripción
ini-warnings=yes	Permite la visualización de mensajes de aviso de inicialización
Preload-ned-files=	Permite la carga previa de los ficheros *.ned
network=	Nombre de la red que se simula

¹⁹ Para el caso del programa gsmsim no ha sido necesario crear más de un fichero de configuración.

snapshot-file=omnetpp.sna	Nombre de fichero de estado de la simulación
output-vector-file=omnetpp.vec	Nombre de fichero de salida para datos vectoriales
output-scalar-file= omnetpp.sca	Nombre de fichero de salida para datos escalares
pause-in-sendmsg=no	Sólo tiene sentido para la ejecución paso a paso
sim-time-limit=	Duración de la simulación en tiempo de simulación
cpu-time-limit=	Duración máxima de la simulación en tiempo de ejecución del programa
Parallel-simulation=false	Permite la simulación en paralelo, para ejecuciones mucho más rápidas

Tabla 6 Parámetros generales del fichero omnetpp.ini.

Son varios los parámetros generales que se pueden iniciar en la simulación, sin embargo aquí se reflejan los usados en la aplicación para GSM.

Después de los parámetros generales vienen los parámetros de configuración de Tkenv y Cmdenv, y seguido de éstos aparecen los valores de inicialización de la red y de los módulos que se quiere crear. Para los parámetros que corresponden a la red, la línea que se escribe debe tener el nombre de la red y a continuación el nombre del parámetro separado por un punto. El ejemplo siguiente inicia los valores del ancho de la superficie de simulación. Son dos coordenadas llamadas X e Y, que indican el ancho y el alto de la superficie mencionada.

```
gsmsim.xwidth = 1000;
gsmsim.ydepth = 1000;
```

El número de parámetros de la configuración de cada módulo depende de los parámetros que haya introducido el desarrollador en las clases creadas. Por ejemplo para la inicialización de cada módulo MS creado, que simula una estación móvil, hay que iniciar los parámetros de la

Tabla 7.

Parámetros módulo Mobile Station (clase MS)	
Nombre parámetro	Descripción
xc	Coordenada en el espacio en el eje de la X
yc	Coordenada en el espacio en el eje de la Y
vmod	Módulo del vector velocidad en m/s
angle	Ángulo del vector velocidad en Grados
pathType	Tipo de camino que sigue, si =0 lineal, si =1 aleatorio

Tabla 7 Parámetros módulo Mobile Station.

Los parámetros propios de este módulo son cinco. Esto no quiere decir que sean los únicos que contiene la clase, sino que son los únicos que le pertenecen únicamente a cada módulo y que debe ser inicializado con valores introducidos por el usuario.

Teniendo en cuenta estos parámetros, las líneas a necesarias en el fichero de configuración son las siguientes:

```
gsmsim.ms[0].xc = 200;  
gsmsim.ms[0].yc = 125;  
gsmsim.ms[0].vmod = 5;  
gsmsim.ms[0].angle = 315;  
gsmsim.ms[0].pathType = 1;
```

Para la inicialización de los módulos simples dentro de la declaración de una red se debe seguir este proceso. En caso de ser un módulo simple dentro de un módulo compuesto, las líneas de inicialización son parecidas. En el código desarrollado se han creado dos módulos simples dentro de un módulo compuesto. Estos módulos se denominan 'msc' y 'vlr', y el módulo compuesto es 'msc_vlr'. Para introducir los parámetros las líneas son las siguientes:

```
gsmsim.msc_vlr.msc.num_msc = 0;  
gsmsim.msc_vlr.msc.num_lai = 0;  
  
gsmsim.msc_vlr.vlr.num_vlr = 0;
```

```
gsmsim.msc_vlr.vlr.num_lai = 0;
```

En conclusión, la forma de introducir un parámetro es ir colocando el nombre de cada módulo simple, precedido por el módulo compuesto al que pertenece (si pertenece a alguno) y al nombre de la red, separados por un punto. Después del nombre del módulo simple se pone el nombre del parámetro separado por un punto. Para darle un valor se debe igualar mediante un signo de igual seguido del valor que se quiere asignar, y terminado por un punto y coma. En el caso de que sean módulos que formen parte de un array, debe ponerse entre corchetes el número de módulo a continuación del nombre y antes del operador de asignación. Éstos son ejemplos de estructuras que se pueden presentar:

```
nombreRed.nombreMódSimple.parámetro = valor;  
nombreRed.nombreMódSimple[número].parámetro = valor;  
nombreRed.nombreMódCompuesto.nombreMódSimple.parámetro = valor;  
nombreRed.nombreMódCompuesto.nombreMódSimple[número].parámetro = valor;
```

6.8 Animación y visualización

Una de las principales ventajas que ofrece OMNET++ frente a otros simuladores es sus grandes capacidades para la simulación. Pese a que ofrece una interfaz por línea de órdenes, la interfaz visual es la realmente útil e interesante. Por tanto, debe ofrecer una gran flexibilidad en esta visualización. Para aportar esta flexibilidad, el lenguaje NED permite el posicionamiento de los módulos en cualquier posición de la imagen indicada por dos coordenadas, x e y, o incluso indicar la topología de la red, en forma de anillo, fila, etc.

6.8.1 Asignación de características de visualización en el fichero ned

Para cada módulo definido en el fichero en lenguaje NED, se puede definir la posición, y las características de la visualización. Además se puede asignar un icono de los muchos que contiene el entorno. Estos iconos pueden ser añadidos desde el editor visual GNED, tienen diferentes formatos, pequeño(s de small), muy pequeño (vs very small), grande (l de large) y normal (ninguna letra). El icono se escribe detrás de la palabra “display” y precedido por comillas ‘i’ signo de igual. A continuación se coloca separado de un punto y coma la posición. Se debe tener cuidado con la colocación de las comillas. La línea debe cerrarse con comillas y punto y coma para que no dé errores de compilación.

```
bsc: BSC;  
  parameters:  
    numbts = number_bts,  
    phones = number_ms;
```

```
gatesizes:
    from_bts[number_bts],
    to_bts[number_bts];
display: "i=device/server2_s;p=500,10";
```

Las topologías de visualización, son en anillo, columna, fila, matriz, entre otras. También permite la coloración y el porcentaje de transparencia del icono, aunque eso son opciones avanzadas de configuración, aunque muy útiles a la hora de aportar la visualización. La Tabla 8 explica los parámetros de posición necesarios para cada tipo de visualización, ya sea columna, fila, etc. (para ver un ejemplo ir a la Figura 19 y la Figura 20)

Parámetros ajustables de visualización en el lenguaje NED	
Parámetros	Descripción
p=xpos,ypos	Posición de un módulo en las coordenadas X e Y
p=xpos,ypos,row,deltas	Posición del primer módulo en las coordenadas X e Y colocando el resto a partir de esta posición en una fila con una separación deltaX.
p=xpos,ypos,column,deltaY	Posición del primer módulo en las coordenadas X e Y colocando el resto a partir de esta posición en una columna con una separación deltaY.
p=xpos,ypos,matriz,módulosporfila,deltaX, deltaY	Posición del primer módulo en las coordenadas X e Y colocando el resto en una matriz de filas y columnas con un número de módulosporfila en cada fila, y con una separación de deltaX, y deltaY.
p=xpos,ypos,ring,width,height	Posición del primer módulo en las coordenadas X e Y colocando el resto en un anillo con un ancho width y un alto height.

$p=xpos,ypos,exact,deltaX,deltaY$	Cada módulo se coloca en las coordenadas $X + deltaX$ e $Y + deltaY$
$b=width,height,rect$	Rectángulo con anchura y altura determinadas ($width$ y $height$ respectivamente)
$b=width,height,oval$	Óvalo con anchura y altura determinadas ($width$ y $height$ respectivamente)

Tabla 8 Tipos de posiciones de los módulos.

Cada módulo puede visualizarse en columnas, filas, matrices, anillos, o de forma aleatoria si no se añade ningún valor de posición.

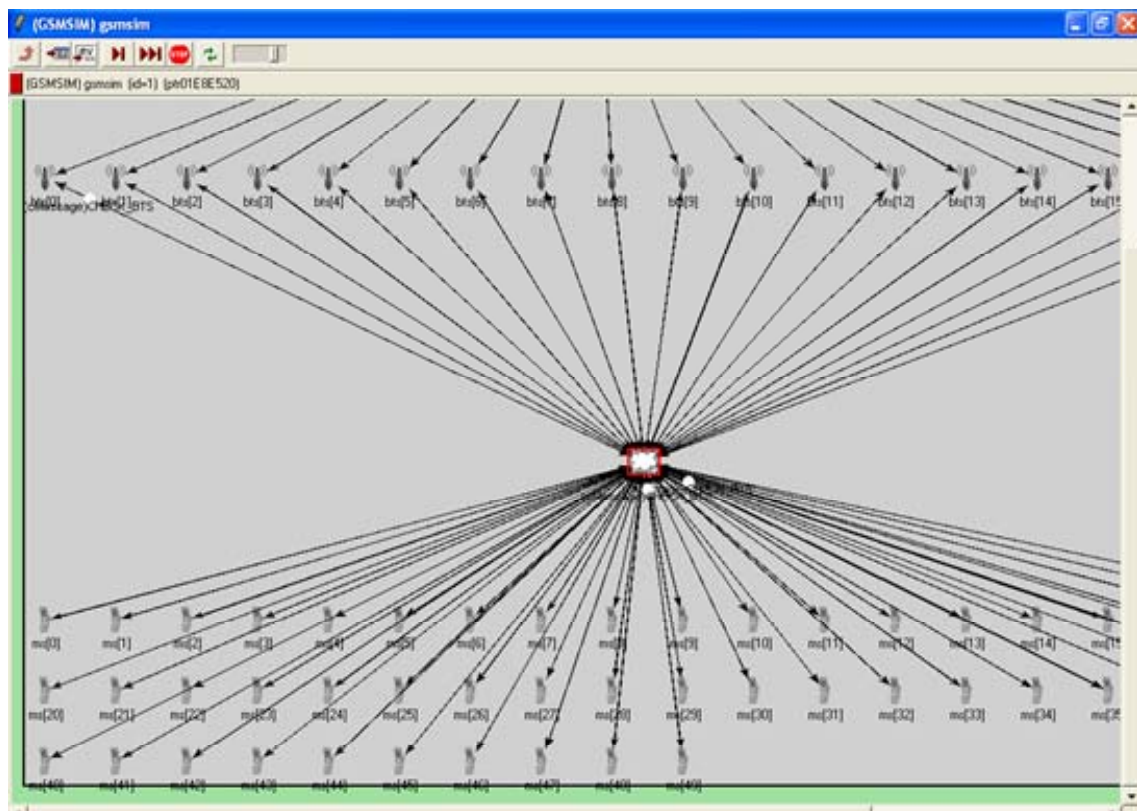


Figura 19 Posición de los teléfonos móviles en matriz.

Configuración aplicada a la simulación de la red GSM. En la imagen se pueden ver 50 teléfonos móviles organizados en matriz con 20 móviles por fila. Las estaciones base (BTS) se organizan en una fila.

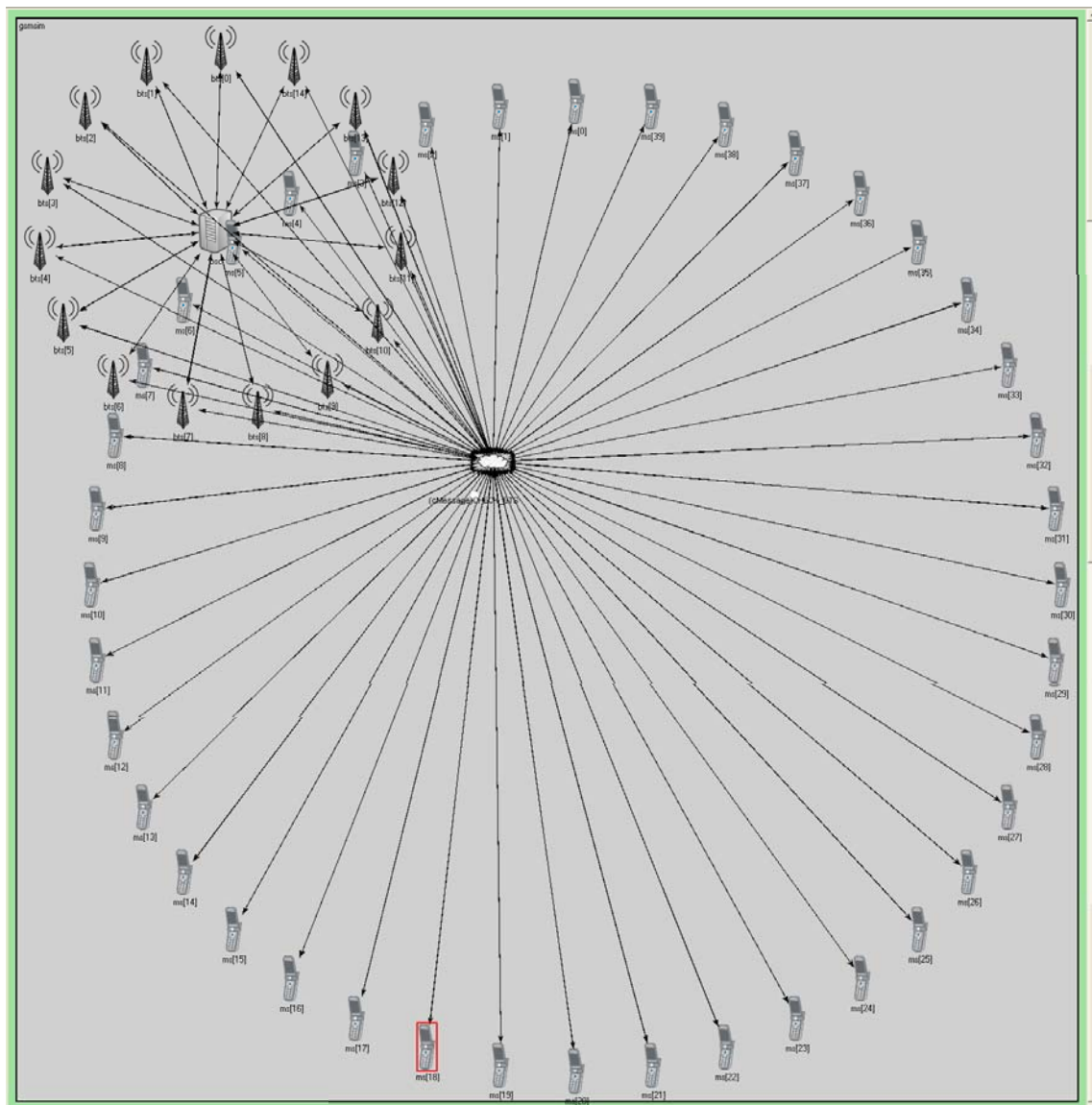


Figura 20 Posición de los módulos en anillo.

Configuración aplicada a la simulación de GSM. Tanto las estaciones móviles como las estaciones base se organizan en anillos. Es una opción que permite el lenguaje NED.

6.8.2 Función bubble

La función bubble() permite que aparezca una nube o bocadillo con una cadena de caracteres

en un módulo determinado. La cadena es pasada como variable a la función. Puede colocarse en cualquier parte del código C++, a tener en cuenta que se visualizará en el módulo en el que se llame la función de bocadillo y no en otro módulo. Si se llama a la función de la forma siguiente en el módulo 'bsc',

```
bubble("LAYER 3 DEDICATED CHANNEL MANAGEMENT");
```

el resultado visual es el siguiente:

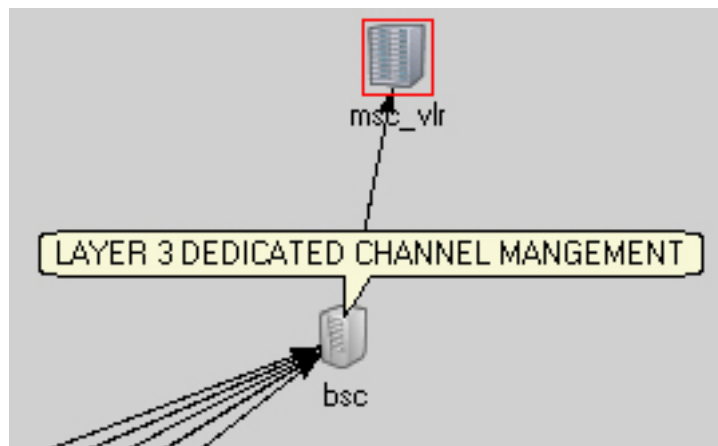


Figura 21 Resultado de la función bubble en la simulación.

La función bubble es llamada desde el módulo bsc programado con la clase BSC derivada de cSimpleModule. Indica el tipo de mensaje que ha llegado al módulo.

6.9 Análisis de los resultados

Como se ha explicado en la introducción, principalmente dos datos de salida, escalares y vectoriales. Éstos se escriben en ficheros con extensión *.sca, y *.vec respectivamente, y cuyo nombre se define en el fichero de inicialización. Para analizar los resultados que ha dado la simulación, el entorno dispone de dos herramientas denominadas Omnet++ Scalars y Omnet++ Plove (accesibles desde el escritorio o desde el menú inicio - Figura 22), aunque en el manual se pueden encontrar los nombres de otros programas con los que estos ficheros de estadísticas son compatibles, como GNU Plot.

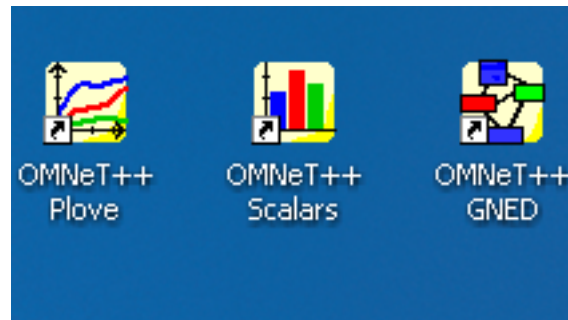


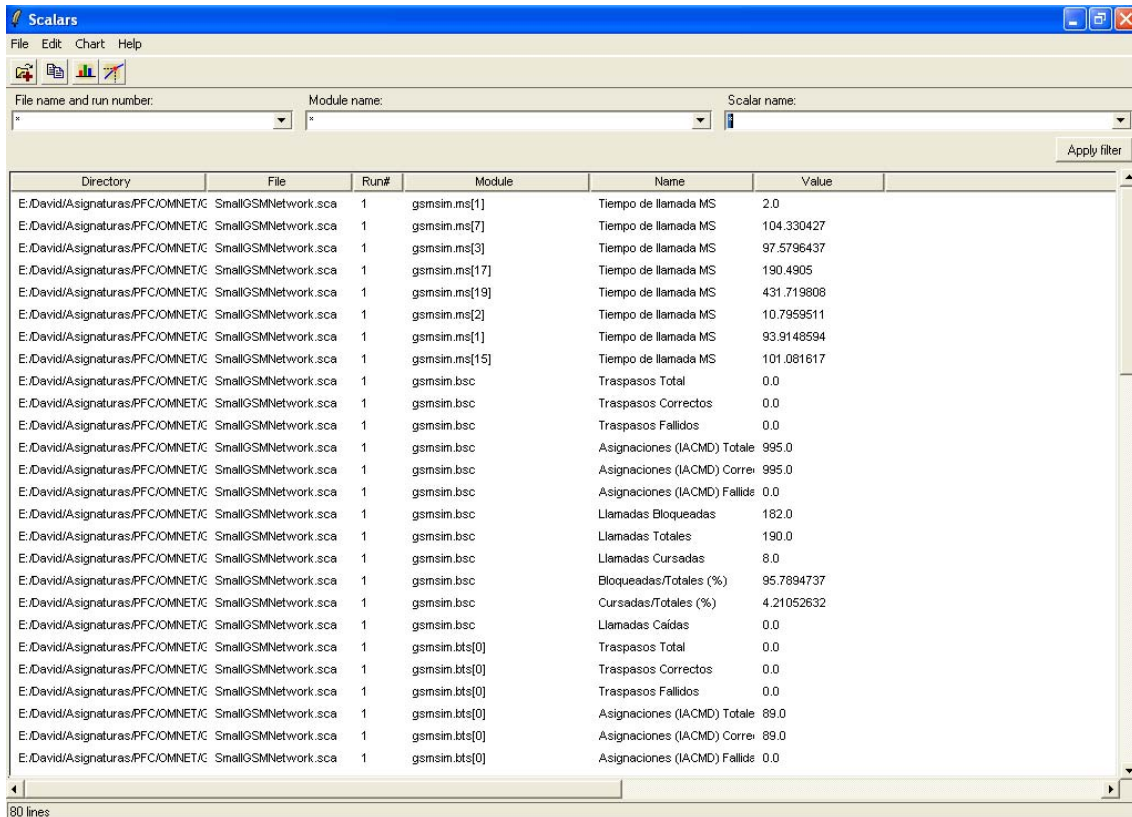
Figura 22 Iconos para el acceso a los programas de visualización de gráficas y para el editor de Lenguaje NED.

Iconos que se graban en el escritorio al instalar OMNET++ en el ordenador y que permiten un acceso rápido y cómodo desde el escritorio a los programas de visualización de gráficas.

El programa Scalars presenta en pantalla el nombre del parámetro, el módulo al que pertenece este valor, y el que ha sido asignado. Por ejemplo, si el dato escrito es el tiempo de llamada en la MS número 5, la línea presente en la ventana del programa será la ruta del fichero de datos, el número de ejecución, el módulo, en este caso `gsmSim.ms[5]`, el nombre del parámetro Tiempo de Llamada, y por último el valor que tiene. En la Figura se puede ver la ventana del programa.

Para visualizar sólo unos valores en concreto se ofrecen las herramientas de filtrado de selección. Las cajas que aparecen en la parte superior de la ventana, son menús desplegables que permiten la selección de un módulo, una ejecución o un valor en concreto. En la

Figura 24 se está realizando el filtrado del parámetro porcentaje de llamadas bloqueadas frente a totales.



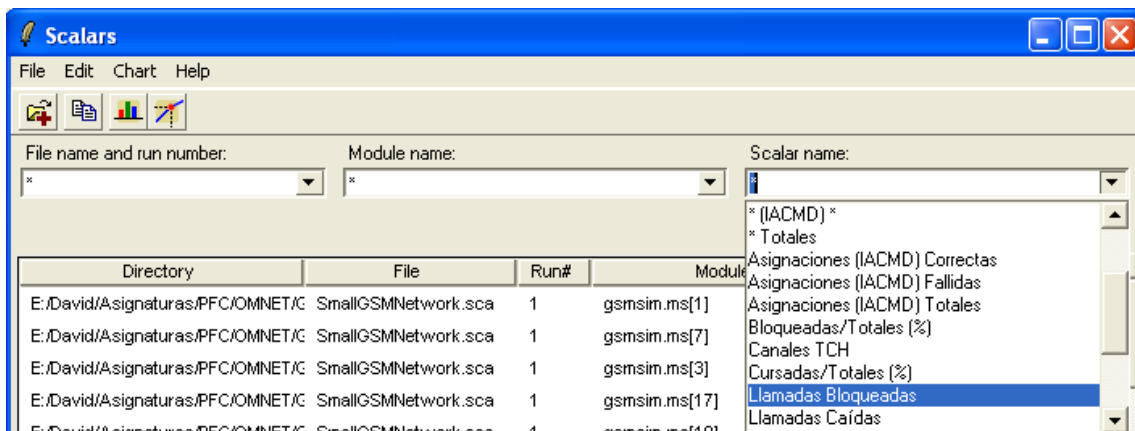
The screenshot shows the 'Scalars' window in Omnet++ with a table of simulation results. The table has columns for Directory, File, Run#, Module, Name, and Value. The data is filtered to show results for 'SmallGSMNetwork.sca'.

Directory	File	Run#	Module	Name	Value
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.ms[1]	Tiempo de llamada MS	2.0
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.ms[7]	Tiempo de llamada MS	104.330427
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.ms[3]	Tiempo de llamada MS	97.5796437
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.ms[17]	Tiempo de llamada MS	190.4905
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.ms[19]	Tiempo de llamada MS	431.719808
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.ms[2]	Tiempo de llamada MS	10.7959511
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.ms[1]	Tiempo de llamada MS	93.9148594
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.ms[15]	Tiempo de llamada MS	101.081617
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bsc	Trasposos Total	0.0
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bsc	Trasposos Correctos	0.0
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bsc	Trasposos Fallidos	0.0
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bsc	Asignaciones (IACMD) Totales	995.0
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bsc	Asignaciones (IACMD) Correctas	995.0
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bsc	Asignaciones (IACMD) Fallidas	0.0
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bsc	Llamadas Bloqueadas	182.0
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bsc	Llamadas Totales	190.0
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bsc	Llamadas Cursadas	8.0
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bsc	Bloqueadas/Totales (%)	95.7894737
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bsc	Cursadas/Totales (%)	4.21052632
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bsc	Llamadas Caídas	0.0
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bts[0]	Trasposos Total	0.0
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bts[0]	Trasposos Correctos	0.0
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bts[0]	Trasposos Fallidos	0.0
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bts[0]	Asignaciones (IACMD) Totales	89.0
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bts[0]	Asignaciones (IACMD) Correctas	89.0
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bts[0]	Asignaciones (IACMD) Fallidas	0.0

Figura 23 Visualización de resultados de simulación Omnet++ Scalars

En este programa se pueden visualizar parámetros escalares escritos durante la simulación para su visualización. Para ello se añade el nombre del módulo y el del elemento, además de disponer de filtros de selección.

Figura



The screenshot shows the 'Scalars' window in Omnet++ with a list of filters on the right side. The filters are: * (IACMD) *, * T totales, Asignaciones (IACMD) Correctas, Asignaciones (IACMD) Fallidas, Asignaciones (IACMD) Totales, Bloqueadas/Totales (%), Canales TCH, Cursadas/Totales (%), Llamadas Bloqueadas, and Llamadas Caídas. The 'Llamadas Bloqueadas' filter is selected.

Directory	File	Run#	Module
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.ms[1]
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.ms[7]
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.ms[3]
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.ms[17]
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.ms[19]
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.ms[2]
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.ms[1]
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.ms[15]
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bsc
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bsc
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bsc
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bsc
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bsc
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bsc
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bsc
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bsc
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bsc
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bts[0]
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bts[0]
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bts[0]
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bts[0]
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bts[0]
E:\David\Asignaturas\PFC\OMNET++\C	SmallGSMNetwork.sca	1	gsmSim.bts[0]

Figura 24 Filtros de Scalars.

En las cajas se pueden elegir los parámetros que se quieren representar y elegirlos en función del número de ejecución, nombre de módulo, y/o nombre del parámetro. En la figura se ha elegido el parámetro llamadas bloqueadas.

El último paso es el de crear la gráfica para la que se debe pulsar el icono que representa unas barras verticales en la parte superior de la ventana de la aplicación (Figura 25). Como se puede ver tanto para valores escalares como para valores vectoriales se añade un filtro de selección para que sea más fácil la visualización de valores concretos definidos por el usuario.

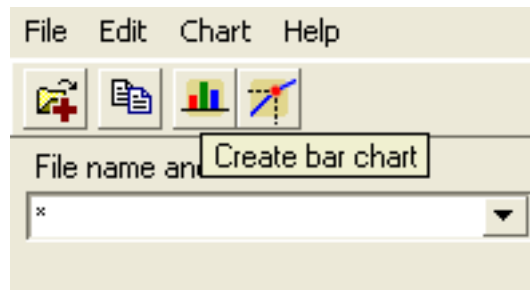


Figura 25 Iconos en Scalars.

El icono que representa las barras verticales permite la visualización de las gráficas. Abre una nueva ventana donde se eligen los ejes, y da lugar a una ventana que representa la gráfica.

La gráfica generada tiene la apariencia de la Figura 26 en la que se representa un ejemplo de una gráfica generada para la visualización del tiempo de llamada empleado por ocho móviles elegidos. La herramienta incluye un zoom que permite la visualización más exacta de los valores representados (Figura 27).

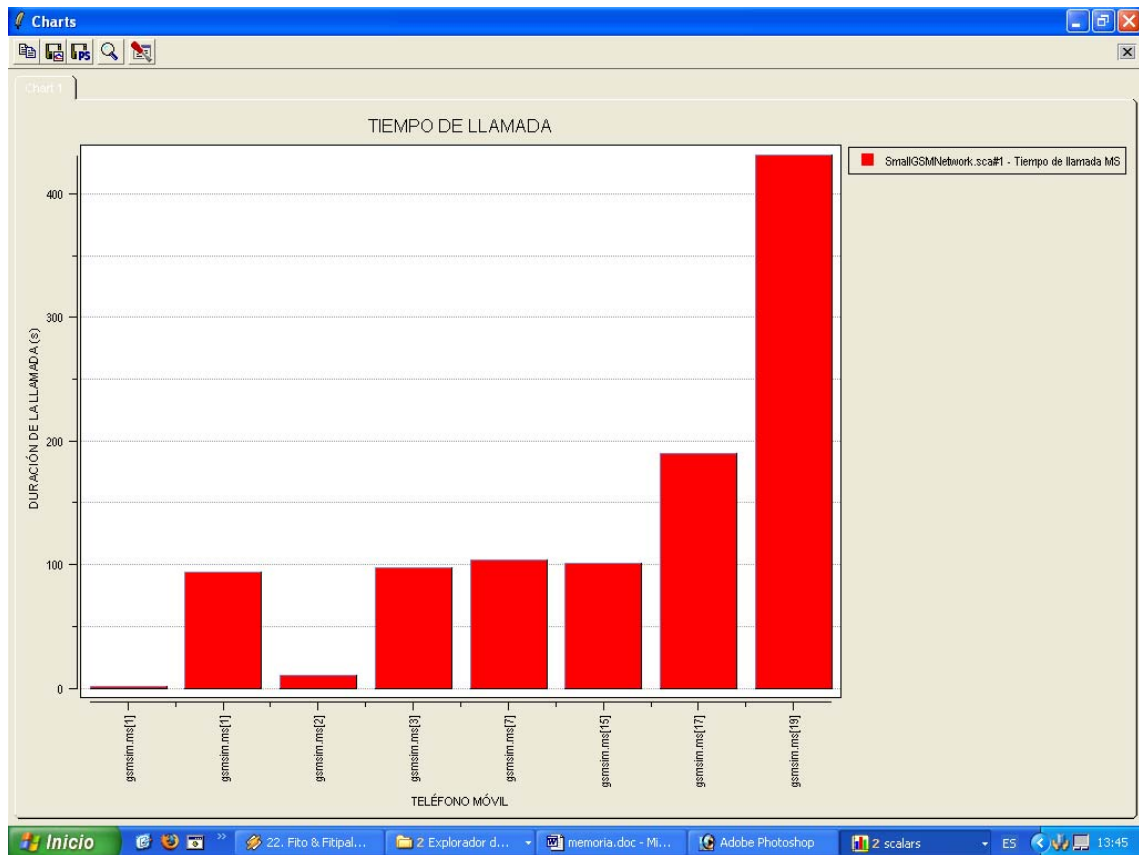


Figura 26 Tiempo de llamada.

En la figura se representa el tiempo de llamada de de ocho estaciones móviles. En el eje horizontal se representan los módulo, y en el vertical el tiempo de llamada.

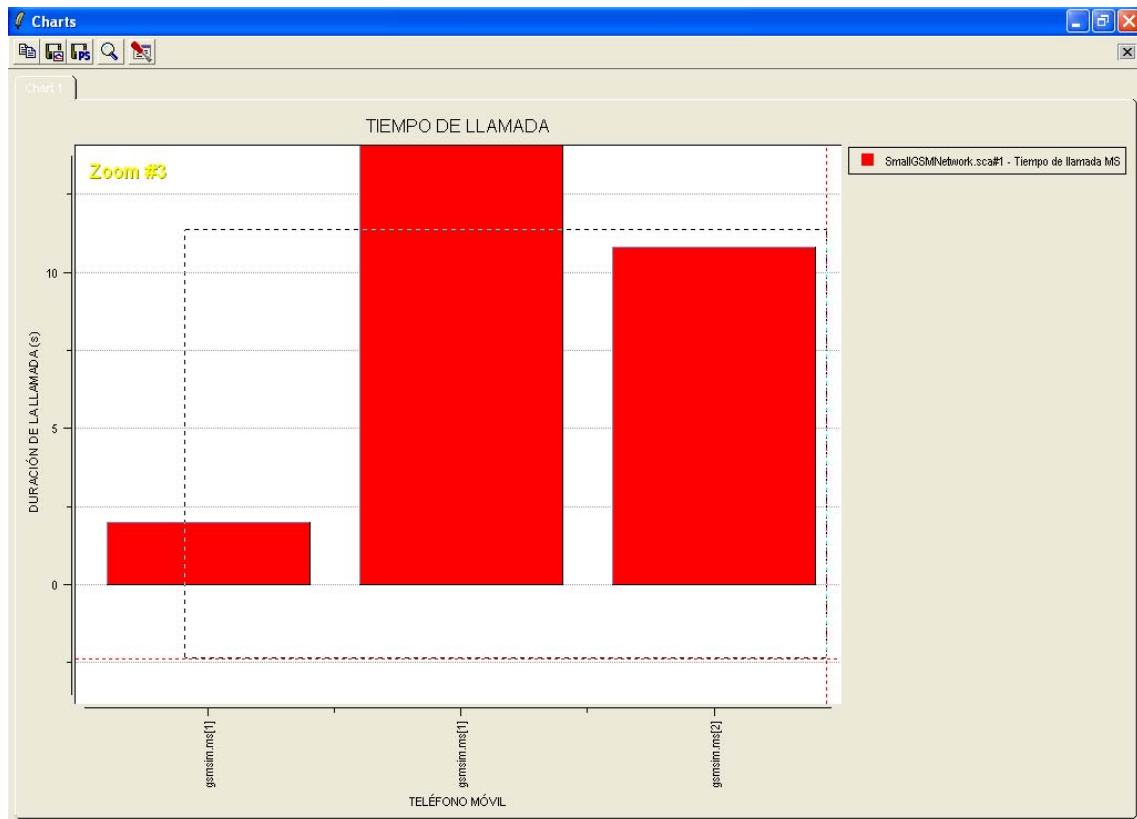


Figura 27 Aplicación de la herramienta de zoom a la gráfica.

Aplicación de la herramienta de zoom a la gráfica anterior de tiempo de llamada de los móviles. Para ello se debe seleccionar el área que se desee visualizar pichando en una esquina, tal y como si se seleccionasen varios archivos en una carpeta en los sistemas operativos Microsoft Windows.

La otra herramienta de visualización es la herramienta OMNET++ Plove. Ésta se diferencia de la anterior en que está destinada a la representación de valores vectoriales. El funcionamiento es similar al del programa Scalars y tiene la interfaz gráfica representada en la Figura 28. Ofrece un mayor número de herramientas que el programa para escalares aunque la función de representación gráfica es exactamente igual que para Scalars. Para crear una gráfica se debe pulsar en el icono de la parte superior que representa dos líneas con varios puntos rojos, y que se puede ver en la parte derecha de la Figura 29.

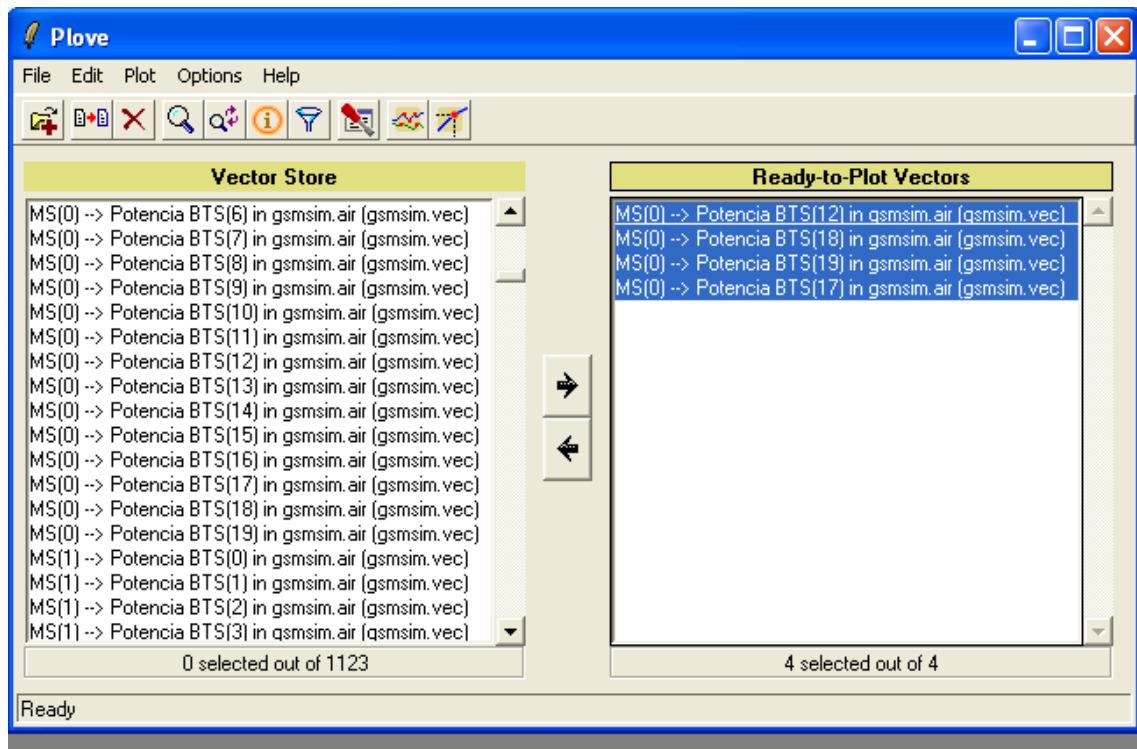


Figura 28 Programa Plove para visualizar resultados vectoriales.

Permite la representación gráfica de datos vectoriales. Tiene una interfaz similar a Scalars.

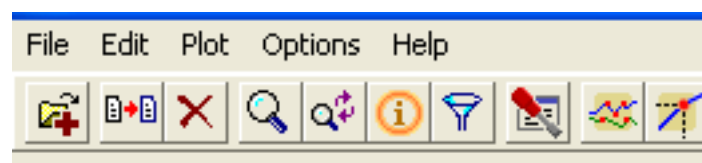


Figura 29 Herramientas del programa Plove.

Este programa aporta más herramientas que el programa para los escalares. La función de representación de gráficos es similar al programa para escalares, y su uso también. El icono es el que representa dos líneas con varios puntos rojos en la parte derecha de la ventana.

El procedimiento para sacar gráficos es muy sencillo e intuitivo, además de no complicar excesivamente al usuario en la tarea de sacar gráficos. La Figura 30 corresponde a una gráfica obtenida con este programa. Representa las potencias recibidas por una estación móvil en un

tiempo de simulación en segundos. El eje vertical representa la potencia en dBm detectada. Como se puede ver, la estación se encuentra a lo largo de su movimiento en diferentes zonas de cobertura, por lo que tendrá que realizar traspasos entre las diferentes antenas en el caso de que desee mantener una llamada telefónica de forma correcta.

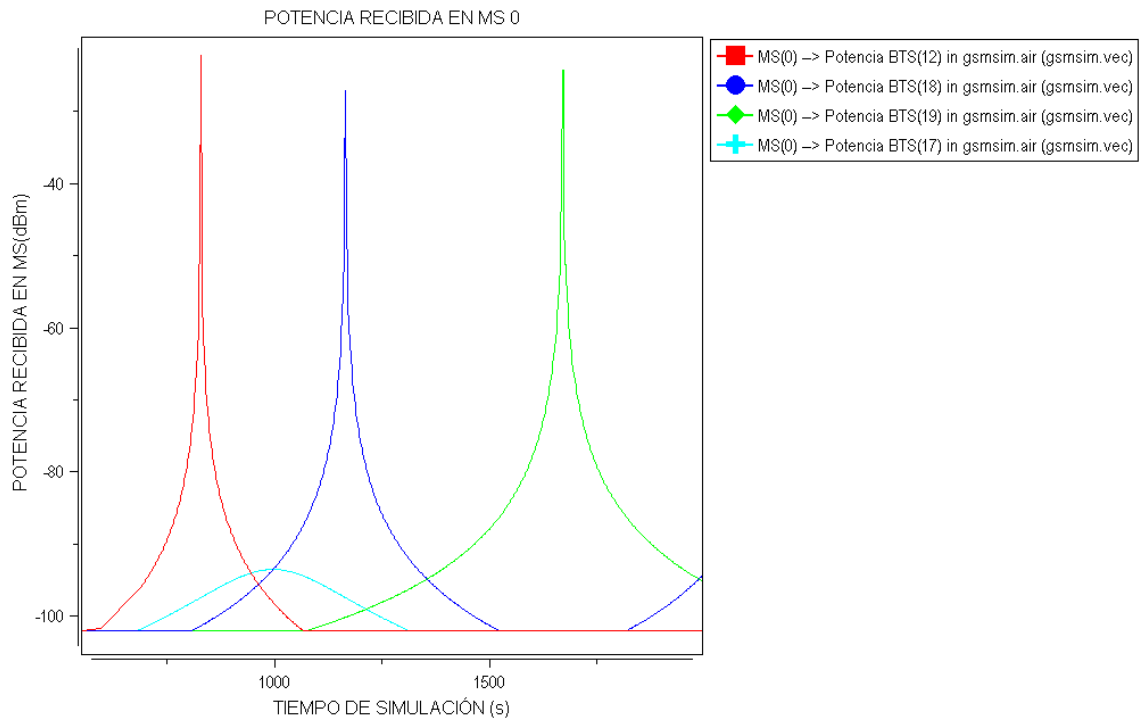


Figura 30 Gráfica de potencias de la estación móvil 0.

La gráfica representa las potencias recibidas de cuatro estaciones base (antenas en el tiempo) de simulación representado en el eje X.

7 OPENGL

OPENGL es un software que actúa de interfaz para el hardware gráfico. El interfaz consiste en diferentes órdenes que se usan para especificar los objetos y operaciones necesitadas para producir aplicaciones interactivas en tres dimensiones. OPENGL se puede usar en multitud de plataformas hardware. Esta librería no aporta funciones de alto nivel que permitan modelar complejos gráficos en 3D (tres dimensiones), sin embargo se puede modelar partiendo de pequeñas primitivas geométricas, puntos, líneas y polígonos. LA librería que aporta estos comandos es GLU siglas de OPENGL Utility Library. Las posibilidades de estas librerías son muy amplias. Para resumir su gran capacidad y potencia a la hora de crear gráficos, cabe destacar que han sido empleadas en la realización de animaciones e incluso se han introducido en juegos de ordenador. Quake es el nombre de uno de los juegos que emplearon OPENGL en su desarrollo. Todo su interfaz gráfico está realizado con esta librería y fue considerado en su época uno de los mejores juegos en cuanto a interfaz gráfica.

En este proyecto las librerías adoptan un papel de apoyo. No forman parte de la aplicación principal sino que se han introducido en otra aplicación para mejorar la simulación de la red

GSM. Se pasará a hablar detenidamente en qué se han empleado en capítulos posteriores. En éste se realizará tan sólo una descripción de las características esenciales de OPENGGL y las funciones para generar gráficos.

Ya que OPENGGL es totalmente independiente del hardware y del sistema operativo empleado, se necesitan otras librerías capaces de leer eventos de teclado, ratón o de gestionar ventanas. Como se ha comentado GLU es esta librería, sin embargo, es un estándar del que han surgido diferentes implementaciones. El usado en este código es GLUT u OPENGGL Utility Toolkit. Principalmente la funcionalidad se puede resumir en dos conceptos:

- Gestión de Ventanas
- Gestión de visualización

7.1 Gestión de Ventanas

Las gestión de ventanas se refiere al las características de la imagen y de la ventana en la que se representa. La funcionalidad de la gestión de ventanas la soportan cinco funciones:

- `glutInit(int *argc, char ** argv)`: es la primera función invocada (siempre la primera) en una aplicación de GLUT. Inicia GLUT y procesa la línea de órdenes.
- `glutInitDisplayMode(unsigned int mode)`: especifica las características de la pantalla, profundidad, color, buffer, etc.
- `glutIntiWindowPosition(int x,int y)`: especifica la posición de la ventana de la imagen en píxels desde la esquina superior izquierda.
- `glutInitWindowSize(int width,int size)`: especifica el tamaño en pixels de la pantalla donde se presenta la imagen.
- `glutCreateWindow(char * string)`: crea la ventana con el título especificado en la cadena string.

7.2 Gestión de visualización

Contiene las rutinas a ejecutar cuando se quiere modificar la pantalla, por lo que es en dónde reside el verdadero peso de la animación o de la creación de un gráfico.

- `glutDisplayFunc (void(*function)())` : especifica la función a la que se debe llamar cuando ocurre un evento de OPENGGL. Las rutinas que modifican pantalla deben colocarse en la función que se le pasa como parámetro.
- `glutMainLoop()`: la última función a llamar en la invocación de un programa. Permite la visualización de todas las funciones anteriormente mencionadas en pantalla. Es un bucle que no termina salvo que tenga una condición de salida dentro de las funciones de control de eventos.

7.3 Crear una aplicación

Como todo programa, debe contener una función principal llamada `main()`. En esta función se irán añadiendo el resto de funciones para dar el resultado final. Como se ha mencionado previamente, la primera función que debe ser llamada es la función `glutInit`.

A continuación se debe añadir la función `glutInitDisplayMode`, en la que se define el color a aplicar, el número de buffers para crear la imagen y la profundidad. La siguiente función es la que define la posición de la ventana en la pantalla del ordenador, seguida de la función que crea la ventana.

La siguiente función presente en el código es la función `glutDisplayFunc` que indica la función a llamar cada que se deba volver a dibujar la imagen ante un evento externo. Por último se llama a la función `glutMainLoop`.

7.3.1 Función de eventos de GLUT

La función incluida en `glutDisplayFunc()` es la que contiene el código de los objetos que se pretenden dibujar. Además de los objetos, debe incluir algunas otras funciones que borren la ventana de visualización, o cambien los buffers de visualización. Las funciones más comunes que se pueden incluir son:

- `glClear(...)`: realiza el borrado de la ventana anterior borrando el buffer de color y el de profundidad.
- `glPushMatrix()/glPopMatrix()`: Todos los cálculos de texturas, traslaciones, rotaciones de imágenes se hacen mediante el cálculo de matrices. Con estas funciones se crea una copia y se pone en la pila de matrices o se saca de la pila.
- `glColor4f(1.0,1.0,1.0,1.0)`: especifica el color en formato RGB, siendo el valor

1.0,1.0,1.0, el blanco y el 0.0, 0.0, 0.0 el negro. El 1.0, 0.0, 0.0 será el rojo (de Red, rojo en inglés). El segundo indicará el valor de verde (Green) y el tercer valor en azul (Blue).

- `glBegin(tipo de figura)`: indica el inicio de la descripción de una figura. Para definir el fin de la descripción se llama a la función `glEnd()`. En el interior de este bloque puede introducirse la creación de puntos con la función `glVertex(coordenada_x, coordenada_y, coordenada_z)` o una función de asignación de color. Por ejemplo si se quiere definir un triángulo, se deben crear tres vértices.
- `glutSwapBuffers()`: intercambia el buffer de la imagen cuando se trabaja con un doble buffer. Se emplea para evitar parpadeos en la generación de la imagen.

7.3.2 Área de proyección de la imagen

OPENGL es una librería para permitir la creación de imágenes en tres dimensiones (3D). Para establecer la imagen hay que definir el área de proyección de la imagen dentro de la función llamada desde `glutReshapeFunc()`.

La función `glutReshapeFunc()` debe ser incluida en la función `main()` de la aplicación. Las funciones que deben ser llamadas dentro de la función referenciada por `glutReshapeFunc()` son las siguientes:

- `glViewport(0,0,ancho,alto)`: indica la porción de ventana en que se va a dibujar.
- `glMatrixMode(GL_PROJECTION)`: dispone de tres tipos de matrices: `GL_PROJECTION`, `GL_MODELVIEW` y `GL_TEXTURE`, para proyección, textura, y modelo.
- `gluPerspective(num, alto/ancho, num, num)`: Esta función opera sobre la matriz de proyección y define el ángulo del campo de visión en sentido vertical (en grados), la relación entre la altura y la anchura de la figura (aspecto), el plano más cercano a la cámara y el plano más lejano de la cámara, respectivamente. Estos dos últimos son los planos de corte, que son los que se encargan de limitar el volumen de visualización por delante y por detrás de la figura. Todo lo que esté fuera del plano no será representado en la ventana.
- `glLookAt()`: esta función tiene 9 parámetros que representan la distancia en x, y, z de

los ojos del observador, las coordenadas de x, y, z, del punto de referencia a observar y los otros tres el vector de dirección de visualización.

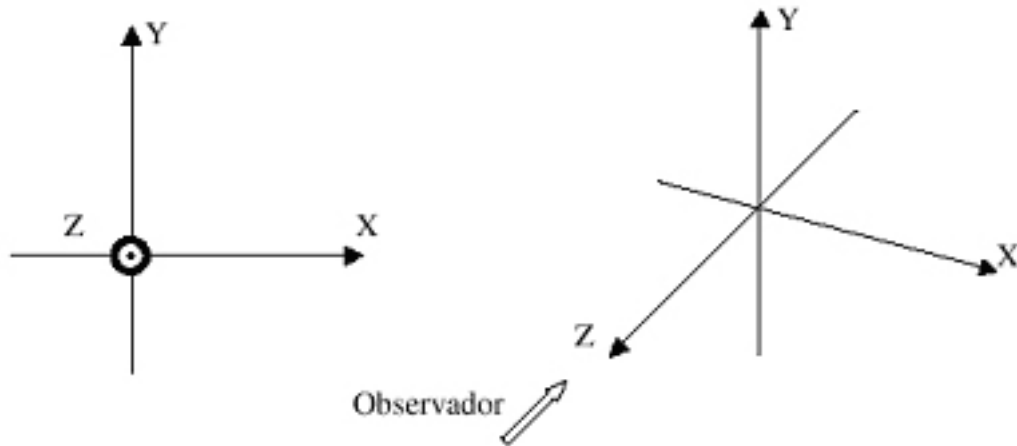


Figura 31 Coordenadas espaciales en OpenGL.

OpenGL es una aplicación para generar imágenes en 3D. Las imágenes se dibujan en pantalla desde un punto denominado observador definido en una función.

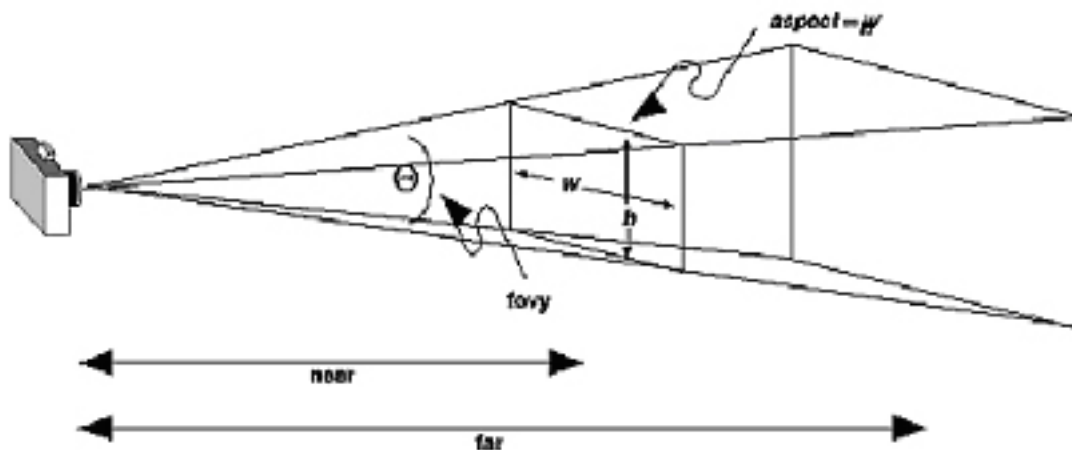


Figura 32 Representación de la imagen desde el punto de observación²⁰.

²⁰ Imagen extraída de Red Book (Libro Rojo llamado así por el color de la portada) cuyo título es OpenGL Programming Guide (Addison-Wesley Publishing Company). Se puede descargar la edición segunda de la versión 1.1 de OpenGL desde la página <http://www.opengl.org>. El libro impreso va por la edición quinta y explica la versión 2 de OpenGL.

El punto de observación se sitúa en dónde se representa la cámara fotográfica. Los límites de la visualización son los rectángulos.

7.3.3 Eventos de teclado

Como para la imagen, el teclado también tiene una función que se ejecuta cada vez que ocurre un evento de teclado. Esta función es `glutKeyboardFunc()` que llama a la función especificada como parámetro y que ha de ser colocada en la función `main()`. Gracias a esta función se pueden asociar eventos de visualización a teclas.

7.3.4 Eventos de ratón

El ratón también una función que se ejecuta cada vez que ocurre un evento de ratón. Esta función es `glutMouseFunc()` que llama a la función especificada como parámetro y que ha de ser colocada en la función `main()`. Gracias a esta función se pueden asociar eventos de visualización a botones del teclado. En el ejemplo siguiente ante un evento de pulsación del botón izquierdo del ratón imprime una cadena por pantalla.

```
If (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
{
    printf ("Evento de teclado");
}
```

7.3.5 Primitivas de dibujo

En OPENGL la unidad mínima de interpretación de dibujo se denomina primitiva. La unidad mínima es el vértice, definido en tres coordenadas espaciales. Las primitivas son:

- `GL_POINTS` Crea puntos individuales
- `GL_LINES` Crea líneas.
- `GL_LINE_STRIP` Crea líneas discontinuas.
- `GL_LINE_LOOP` Crea líneas cerradas.
- `GL_TRIANGLES` Crea triángulos
- `GL_TRIANGLE_STRIP` Crea triángulos discontinuos
- `GL_TRIANGLE_FAN` Crea triángulos en abanico.

- GL_QUADS Crea cuadrados.
- GL_QUAD_STRIP Crea cuadrados discontinuos
- GL_POLYGON Crea polígonos.

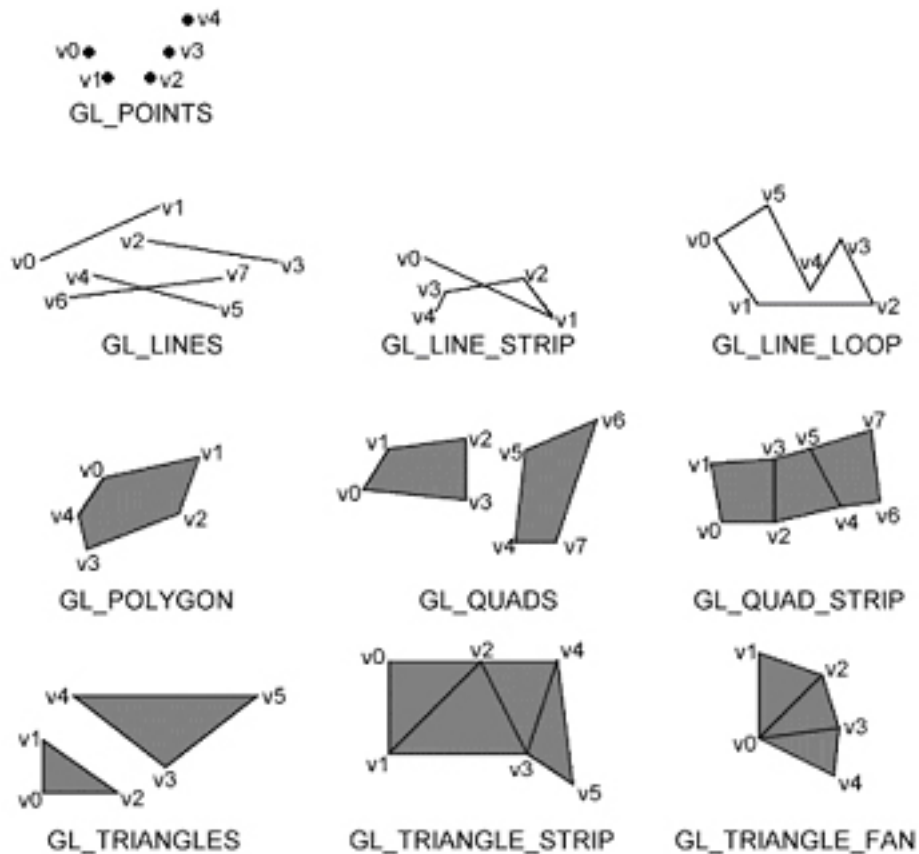


Figura 33 Tipos de primitivas y resultado²¹.

Se pueden crear puntos, líneas, polígonos irregulares, triángulos, etc.

Para crear un polígono se deben definir los vértices, ya sean de dos dimensiones o de tres dimensiones. Los puntos se definen dentro del bloque limitado por `glBegin()` y `glEnd()`.

²¹ Imagen extraída de Red Book. Para más información sobre el libro ver la nota al pie anterior.

7.3.6 Color

Los polígonos creados se pueden crear con un color determinado por el usuario, para ello simplemente hay que hacer una llamada a la función `glColor()` en la definición de cada polígono. Define el color del relleno actual en RGB y un cuarto parámetro opcional alpha.

Mediante la función `glShadeModel()`, se puede definir el tipo de relleno de color de los polígonos. Si el parámetro es `GL_FLAT`, rellena el polígono con el color activo en el momento definido, en cambio si es `GL_SMOOTH` rellena los polígonos con la interpolación de los colores activos en cada vértice.

7.3.7 Plantilla para la creación de programas con opengl

A continuación se presenta una plantilla básica para la realización de programas con OpenGL.

```
#include <GL/glut.h>
#include <stdio.h>
#include <stdlib.h>

void init ( void )
{
    // Inicialización de OpenGL
}

void display ( void )
{
    // Caracterización de la imagen en pantalla
}

void reshape ( int w, int h )
{
    // Código ejecutado ante una redimensión de la ventana. Define el área de proyección
}

void keyboard ( unsigned char key, int x, int y )
{
    // Tareas a ejecutar ante un evento de teclado.
}

void mouse ( int button, int state, int x, int y )
{
    /* Tareas a ejecutar ante un evento de ratón
}

int main ( int argc, char** argv )
{
    glutInit ( &argc, argv);
    glutInitDisplayMode (GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowSize (250,250);
```

```
glutInitWindowPosition (100,100);  
glutCreateWindow ("Título de la Ventana");  
init ( );  
glutReshapeFunc ( reshape );  
glutKeyboardFunc ( keyboard );  
glutMouseFunc ( mouse );  
glutDisplayFunc ( display );  
glutMainLoop ( );  
return 0;  
}
```


8 DESARROLLO

8.1 Programación de la aplicación.

La aplicación se ha desarrollado mediante diferentes módulos. Cada uno de los módulos de NED han sido desarrollados con C++ en ficheros separados. Los ficheros que se han introducido son los siguientes:

- omnetpp.ini: es el fichero de inicialización, con este se inicializan todos los parámetros que se emplean en la simulación de la red.
- gsmsim.ned: fichero que sirve para crear la red que se va a simular. A la hora de compilar toda la aplicación se pasa a C++ para dar lugar a la aplicación.
- gsmsim.h: fichero de cabecera con las definiciones de los mensajes y de variables que serán utilizadas por las clases que representan los modelos.
- ms.cpp: fichero que modela un equipo móvil.

- `bts.cpp`: fichero que modela una estación base.
- `bsc.cpp`: fichero que modela un controlador de estación base.
- `msc.cpp`: fichero que contiene la declaración y la definición de una clase llamada MSC derivada de `cSimpleModule` y que modela el funcionamiento de un equipo MSC.
- `vlr.cpp`: fichero que contiene la declaración y la definición de una clase llamada VLR derivada de `cSimpleModule` y que modela el funcionamiento de un equipo VLR.

8.1.1 Estructura de la red

En el programa de simulación se han desarrollado los equipos que se han mencionado con anterioridad. El equipo MS corresponde al equipo móvil y que se conecta al subsistema de estaciones base mediante el interfaz Um. Este interfaz se modela con un módulo llamado AIR. La MS va conectada con el equipo AIR y el módulo AIR se conecta a su vez con el módulo BTS. El subsistema BSS se compone de dos equipos, el BSC y la BTS. En la simulación se pueden conectar varios equipos BTS a un mismo BSC, aunque sólo se puede emplear un BSC durante la simulación. A este equipo va unido un equipo compuesto formado por un MSC y un VLR.

8.1.2 Movimiento de las estaciones móviles

Una de las partes más importantes de la aplicación es la de aportar el movimiento a los equipos MS. Para ellos se define una posición en función de dos ejes X e Y. A continuación se define un vector velocidad, con un módulo, y un ángulo que definirá la dirección. Estos valores darán lugar a dos componentes, una en el eje X y otra en el eje Y, que se multiplicarán por el tiempo entre cálculos de posición. Este tiempo es de 1 segundo, lo que quiere decir que cada segundo se actualiza la posición de la MS. El valor obtenido de la multiplicación de la velocidad por el tiempo se suma a la posición guardada con anterioridad. Para que se produzca esta modificación, es necesario que el móvil reciba y envíe mensajes de `MOVE_MS` cada segundo. Para ellos se emplean los automensajes. Con la llegada de un mensaje de este tipo la función de gestión de mensajes llama a la función de manejo de un mensaje `MOVE_MS`. Si el estado asociado que lleva el mensaje es `MOVE_MS`, entonces realiza el cálculo en esta función. Otros estados posibles son petición de llamada o espera de

conexión.

$$x_{actual} = x_{anterior} + v_m * t_{movimiento} * \cos(angulo)$$
$$y_{actual} = y_{anterior} + v_m * t_{movimiento} * \sin(angulo)$$

x_{actual} : nueva posicion en el eje x

y_{actual} : nueva posicion en el eje y

$x_{anterior}$: posicion anterior en el eje x

$y_{anterior}$: posicion anterior en el eje x

v_m : velocidad en modulo

$t_{movimiento}$: tiempo de actualizacion de posicion

$angulo$: angulo del vector velocidad

Ecuación 10 Cálculo de nueva posición de una MS

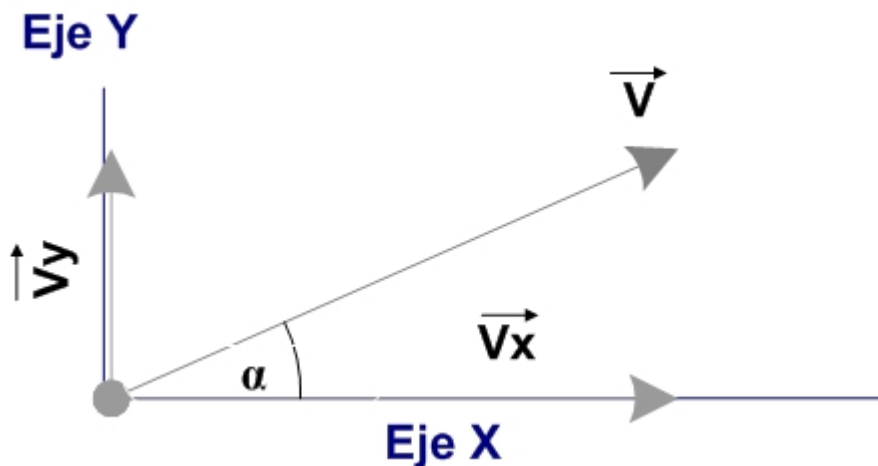


Figura 34 Velocidad de un equipo MS.

La velocidad dada en módulo y ángulo define dos compotentes a partir de las cuales se obtienen las nuevas posiciones.

La dirección que sigue depende de la forma en que se defina en el fichero de configuración. Existen dos opciones, movimiento aleatorio o lineal. En el movimiento lineal, los datos de velocidad se establecen durante la configuración inicial y permanecen hasta el final de la simulación, salvo en el caso de que alcance el límite de la zona definida para la simulación. Si ocurre esto, al ángulo inicial se le suma 180° y se comprueba que el ángulo nuevo está en el rango de 0° a 360° . Si no lo está, se divide por 360 y se guarda el resto en el valor del ángulo. El movimiento es como el de la Figura 35. Para el caso de la trayectoria aleatoria, se define un instante de cambio, un tiempo de vida, un módulo y un ángulo inicial. Cuando llega como antes un mensaje de MOVE_MS, la función de manejo comprueba que el tiempo de vida no haya acabado. Para ello se suma la marca del instante de tiempo de cambio con el tiempo de vida y se comprueba que no sea mayor que el valor devuelto por la función `simTime()`. Si este tiempo ha expirado, se cambia el ángulo con un valor aleatorio, se mantiene el módulo, se añade un nuevo tiempo de vida y en el instante de tiempo se da el valor devuelto por la función `simTime()`.

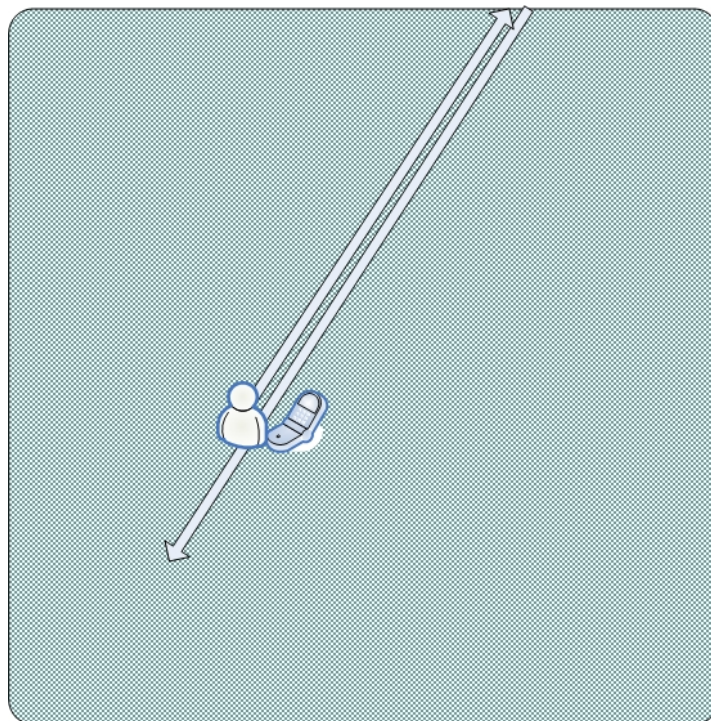


Figura 35 Movimiento lineal de MS

Cuando un móvil llega al límite de la zona de simulación, cambia su ángulo sumando 180° . De esta forma los equipos nunca se saldrán de la zona de simulación.

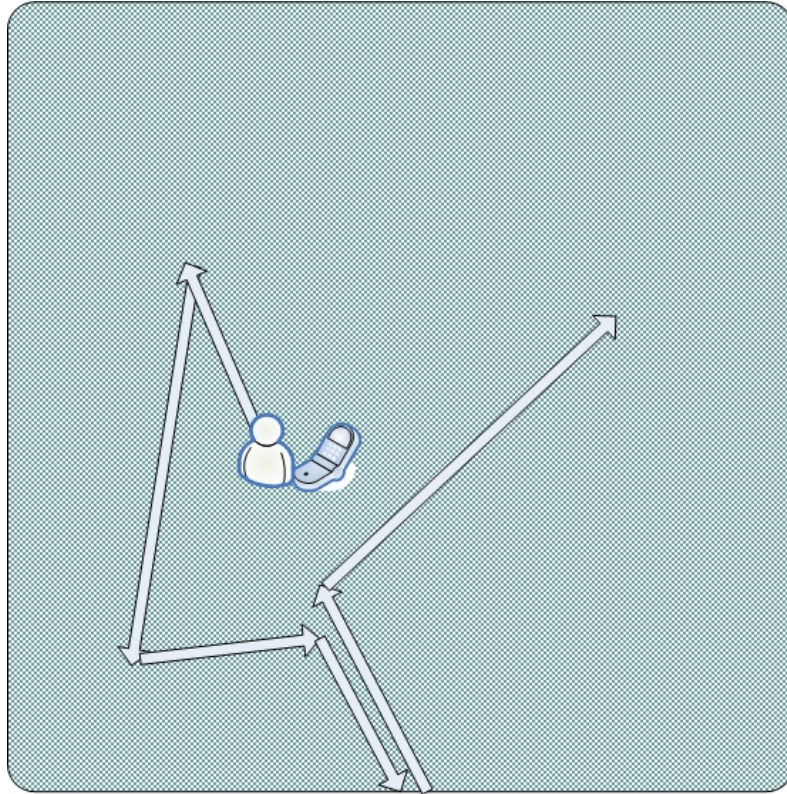


Figura 36 Movimiento aleatorio de MS.

Al igual que para los movimientos líneas, también se cambia el ángulo sumando 180° para evitar que se salga de la zona de simulación.

8.1.3 Inicio de funcionamiento de MS.

Cuando se inicia el módulo MS, el módulo VLR no tiene ninguna información de la MS, el móvil se encuentra apagado. En la función de inicialización se da un valor aleatorio entre 0 y 50 para la actualización de esta posición. Si por ejemplo el valor de este tiempo es 20, esto quiere decir que la estación debe esperar 20 segundos hasta realizar la primera actualización de posición. Hasta que no se realiza una actualización de posición, el equipo MS no puede recibir o establecer llamadas. Para despertar estos equipos móviles, la función de manejo de los mensajes de tipo MOVE_MS, comprueba si debe hacer una actualización de posición, si debe hacerlo inicia el procedimiento de actualización de posición enviando un mensaje de LUREQ hacia el VLR. Para conocer si debe establecer esta actualización, se llama a una

función llamada `lureq_query()`.

La función de inicialización despierta el funcionamiento del móvil gracias a un mensaje `MOVE_MS`. Este mensaje debe enviarse cada segundo para que se actualice la posición de los equipos móviles, y para que no termine la simulación. En OMNeT++ es necesario que se estén enviando continuamente mensajes para que no acabe la simulación. Para mantener los equipos despiertos, se mantiene y se envía cada segundo.

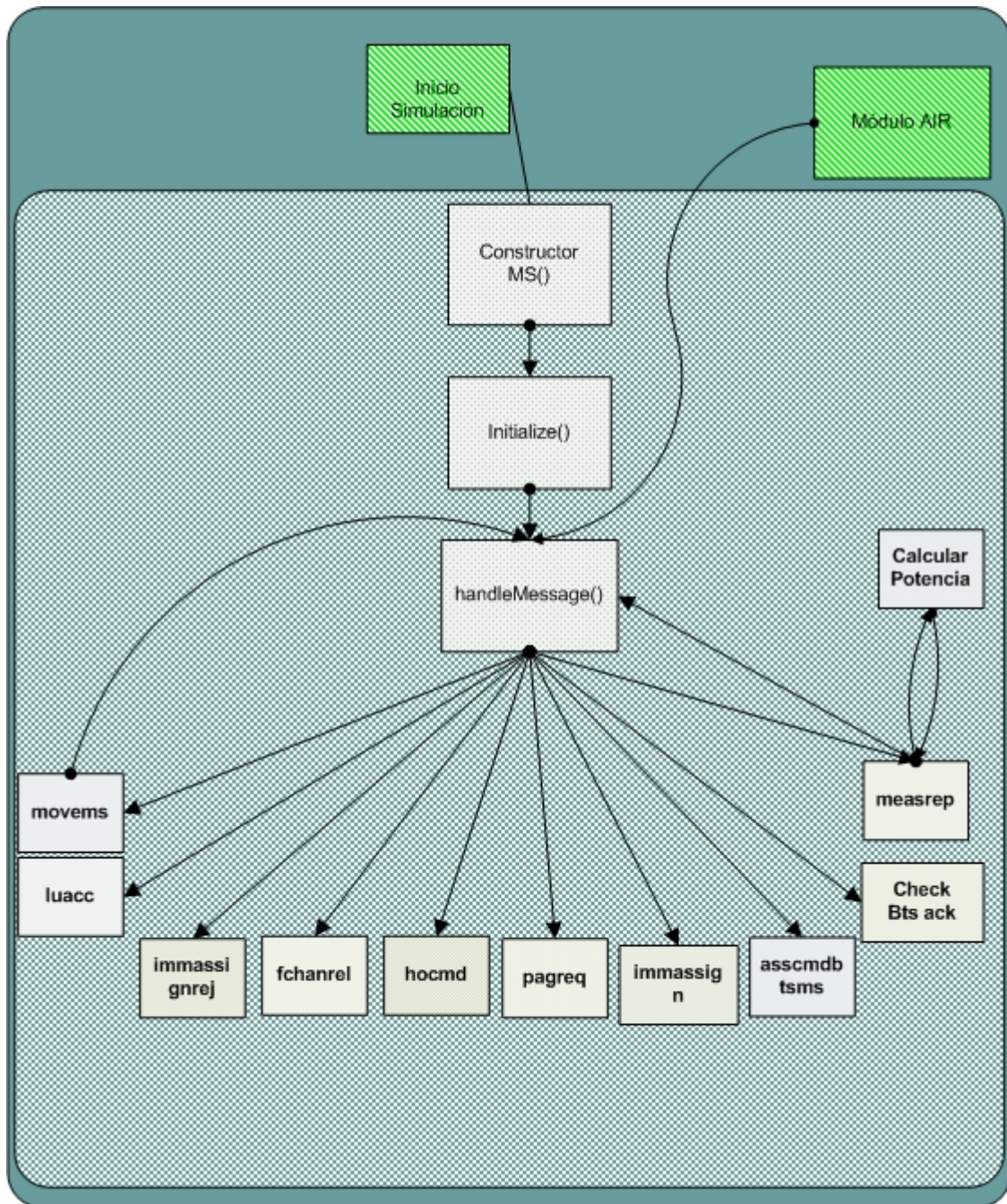


Figura 37 Inicio de un módulo MS.

El módulo MS se inicia cuando se inicia la simulación. Primero se llama a su constructor, y a continuación a la función de inicialización. Esta función envía un auto mensaje que hace despertar a la función handleMessage. Esta función llama a su vez a la función de tratamiento de MOVE_MS, que a su vez enviará otro mensaje de MOVE_MS hacia handleMessage. De esta forma, la simulación no se para hasta que llegue al límite definido por el usuario en el fichero de configuración, aún cuando no haya eventos en el

simulador. Ante la llegada de cualquier otro mensaje también se llamará a la función de gestión de mensajes. Estos mensajes vendrán desde el módulo AIR o serán automensajes. Provocaran eventos en el módulo, que serán respondidos o no con mensajes hacia el módulo AIR. La función de medidas es despertada por la asignación de un canal de tráfico. Según el tiempo definido en el fichero de configuración, un nuevo mensaje de potencias se generará en la estación móvil gracias a este automensaje.

8.1.4 Establecimiento de una llamada

Cuando el módulo MS recibe un mensaje MOVE_MS, no sólo actualiza la posición, sino que también comprueba otros datos. La función de manejo de este mensaje, llama a la función call_query() para ver si establece una nueva llamada. Esta función devuelve un valor verdadero o falso en función del valor que establece una función exponencial. Esta función exponencial devuelve una variable aleatoria con una media definida por el usuario en el fichero de configuración. La variable aleatoria define el tiempo entre llamadas en el móvil. Cada vez que se llama a la función de call_query(), ésta comprueba que no se ha superado el valor de tiempo entre llamadas. Si se ha superado, entonces el equipo MS devuelve un valor true.

Este valor de tiempo se calcula a partir de la suma del instante en que se llama a la función call_query() y el valor devuelto por la función exponencial. Esta suma da el instante en que se debe establecer la nueva llamada. Si esta suma es inferior a simTime() quiere decir que se debe establecer la llamada (devuelve true), si es superior debe esperar a que llegue el instante definido por la suma anteriormente mencionada (devuelve false). Este proceso se puede ver en la Figura 38.

El retorno de un valor verdadero en esta función, inicia el procedimiento de establecimiento de llamada en la MS. El módulo envía un automensaje MOVE_MS con el campo status igual a CHECK_BTS. Cuando este mensaje llega al módulo, envía un mensaje para chequear la BTS. Este mensaje origina un mensaje con el identificador de la estación base de la cual recibe más potencia. A este mensaje le sigue un mensaje en la MS de CHANREQ, iniciando así la petición de un canal de señalización. El mensaje es retransmitido hasta al BSC, donde se ordena a la BTS que reserve un canal para esa MS. Cuando llega el mensaje de asignación a la MS, el equipo móvil tiene asignado un canal de señalización. La confirmación de esta asignación desde la BTS origina la asignación de un canal de tráfico. Cuando el canal de tráfico queda establecido en la MS se procede a la liberación del canal de señalización.

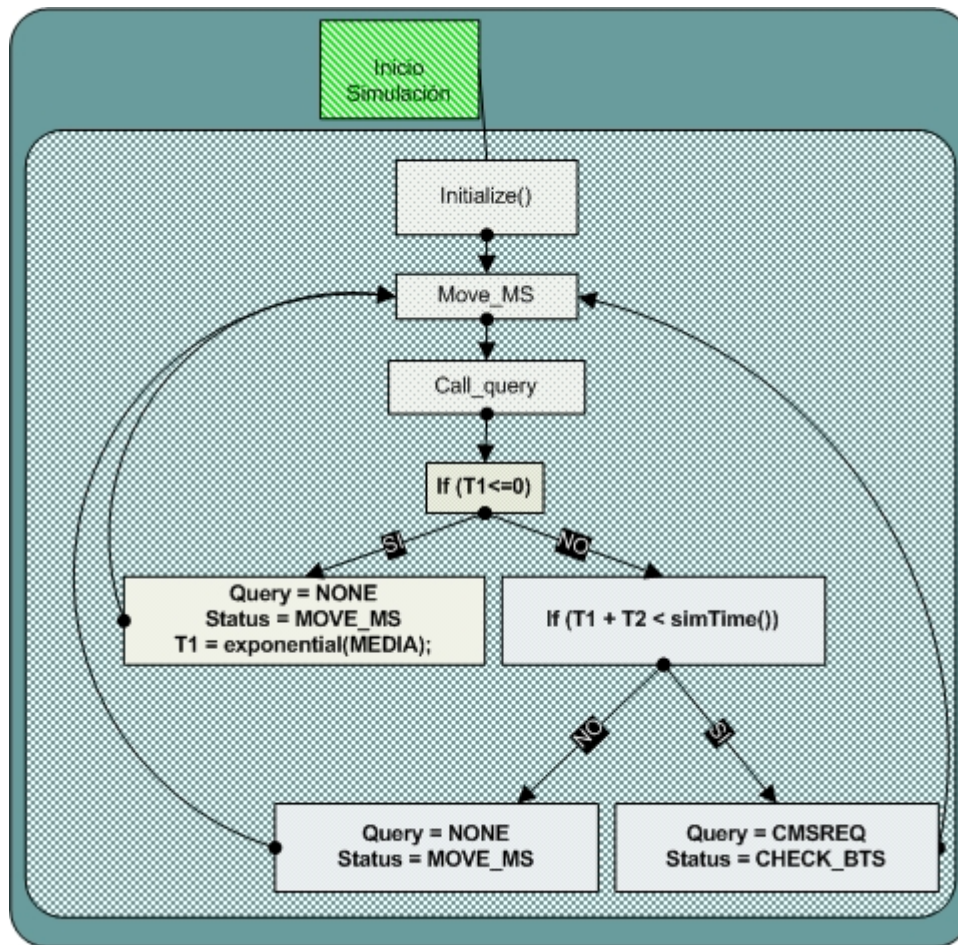


Figura 38 Función de establecimiento de llamada.

Cuando la simulación se inicia, se produce la llamada a la función initialize(). Esta función crea un auto-mensaje de tipo MOVE_MS que al llegar al módulo es tratado por la función move_MS() que a su vez llama a call_query(). En cada módulo se han establecido dos valores, T1 y T2. T1 almacena el tiempo de espera hasta la siguiente llamada en este módulo. T2 almacena el instante en que se cambio T1 y se le dio un valor definido por la función exponential(). Si se llega al instante en que debe producirse la llamada, entonces envía un mensaje MOVE_MS con la petición CMSREQ.

Para controlar que las MS tienen asignada un canal o no, se crean junto con la BSC unos vectores que llevan toda esta información. Los vectores almacenan el número de MS, la potencia recibida, la BTS a la que están conectados y el tipo de canal al que se encuentran conectados. El vector que almacena el número de MS actúa como un índice para el resto de vectores. En primer lugar se busca la posición que ocupa la MS en el vector y se almacena. Con la posición se accede al resto de elementos. La información queda como en la siguiente tabla:

Puertas de la BSC				
Posición	Identificador de MS	Potencia	BTS conectada	Canal
0	23	-100 dBm	1	1
1	2	-74 dBm	5	0
2	0	-	-1	-1

Tabla 9 Vectores en BSC.

En la tabla la posición 0 la ocupa el equipo 23, por lo que para acceder al resto de valores será necesario indicar esta posición.

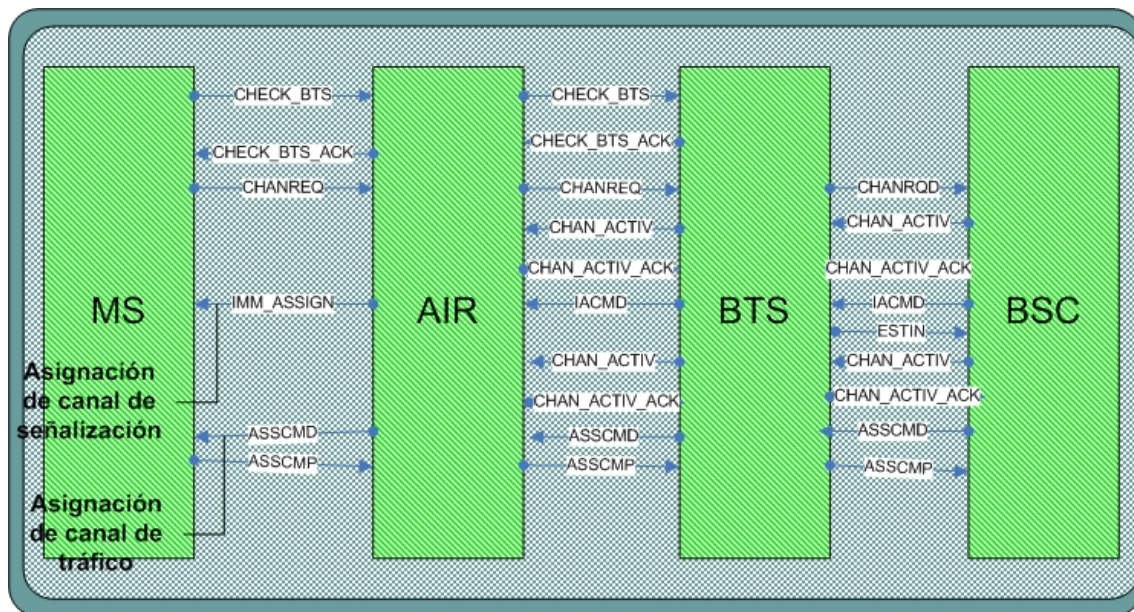


Figura 39 Establecimiento de llamada.

Los mensajes para el establecimiento de una llamada en la simulación son los que aparecen en la figura. Primero se asigna un canal de señalización, que es necesario liberar cuando se asigne el canal de tráfico. Una vez establecido el canal de tráfico se establece la llamada telefónica.

8.1.5 Duración de una llamada

Las llamadas tienen una duración definida por una función que devuelve una variable aleatoria que sigue una distribución exponencial con una media definida por el usuario en el fichero de configuración omnetpp.ini. Este valor se asigna cuando se termina de establecer el canal de tráfico. La función guarda dos valores, el instante en que se inicia la llamada y el tiempo de duración de la llamada.

Para controlar que la llamada se termina, cada vez que se envía un mensaje de medidas se controla que no se supera el tiempo de llamada asignado cuando se asignó el canal de tráfico. En caso de que la terminación se produzca por la red, por caída de la llamada, o por un mal traspaso, se libera el canal y se guarda en el fichero de escalares la duración de la llamada hasta el corte, en vez de almacenar la duración asignada.

8.1.6 Actualización de posición.

Cada vez que a la MS llega un mensaje MOVE_MS, se llama a la función de manejo de este tipo de mensajes. Lee el parámetro de estado que contiene el mensaje. Si el estado es el de MOVE_MS, llama a la función de lureq_query() para ver si debe actualizar la posición. En caso de que así sea, se almacena en el mensaje el estado CHECK_BTS, y se envía un automensaje MOVE_MS. A la llegada de este nuevo mensaje MOVE_MS, el código comprueba el estado, y al ver que se trata de un estado CHECK_BTS, comienza el establecimiento de una conexión de señalización enviando un mensaje CHECK_BTS.

El módulo AIR, recibe este mensaje y lo reenvía a la BTS con mayor potencia en la zona en que se encuentra la MS. La BTS responde con su identificador hacia la MS, y ya puede iniciar la petición de un canal. Para ello envía hacia BTS un mensaje de CHANNEL REQUIRED, con el parámetro de petición de actualización de posición. La BTS reenvía este mensaje hacia BSC. La BSC almacena los datos de identificación de la MS y de la potencia recibida por esta, y devuelve un mensaje de activación de canal hacia BTS. La BTS establece el canal y se lo comunica a BSC. Entonces la BSC envía un mensaje de establecimiento de canal de señalización entre MS y BTS. La BTS reenvía este mensaje hacia la MS, y devuelve un mensaje de información de establecimiento, que la BSC transforma en un mensaje hacia la MSC. Hasta aquí se describe el procedimiento seguido para el establecimiento del canal. Una vez establecido, el módulo MS cuando recibe el mensaje de asignación, envía un mensaje de actualización de posición. Este mensaje se pasa de forma transparente hasta la MSC. La MSC entonces inicia el diálogo con el equipo VLR.

El equipo MSC envía un mensaje de MAP_LOCATION_UPDATE_AREA hacia el registro de visitantes. El equipo responde con un mensaje de MAP_LOCATION_UPDATE_AREA_ACK, que origina hacia MS un mensaje de actualización correcta. Cuando este mensaje llega al módulo MS, se inicia el procedimiento de desconexión el canal.

8.1.7 Cálculo de potencias.

Para calcular las potencias es necesario que primero se calcule la distancia que separa al equipo móvil de la estación. En este programa se supone la ausencia de obstáculos que modifiquen el área de cobertura, es decir, se supone que se realiza la simulación en el espacio libre. Las estaciones se han programado como si tuviesen antenas omnidireccionales.

Para calcular la distancia, el programa utiliza las posiciones de ambos puntos y el teorema de Pitágoras (Figura 40). Si esta distancia es mayor que el radio de cobertura establecido para esta estación, entonces la MS no detectará señal de esta estación. El límite se calcula como el punto en que la potencia transmitida menos las pérdidas es menor que la sensibilidad de recepción de un móvil. Si la distancia es menor, entonces la MS calculará las pérdidas en función de esta distancia. La fórmula utilizada es la de pérdidas en el espacio libre, donde se define una atenuación a la distancia de un metro, y un exponente de pérdidas denominado con la letra 'n'. La frecuencia utilizada es la de 900MHz y la distancia, es la calculada anteriormente en metros.

$$L = L_0 + 10n \log(d)$$

Ecuación 11 Pérdidas en el espacio libre

L: atenuación

L_0 : atenuación a la distancia de un metro, para el espacio libre es $L_0 = 32,45 + 20 \log(f)$ con f frecuencia en GHz.

n: exponente de pérdidas

d: distancia entre transmisor y receptor en metros

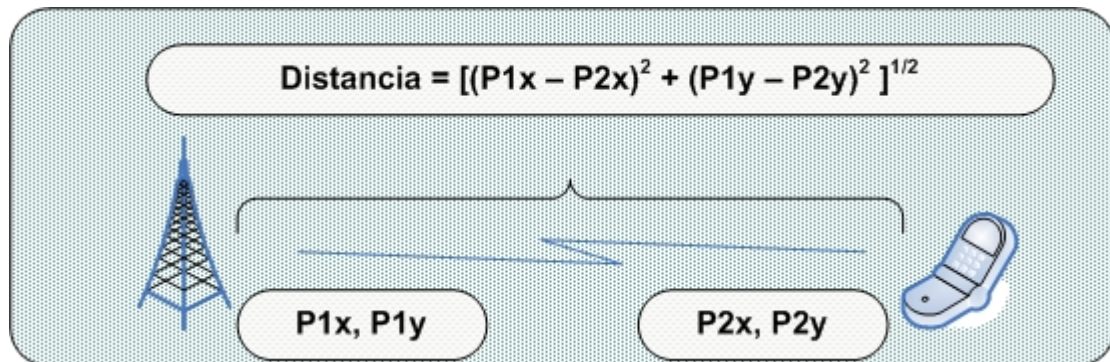


Figura 40 Cálculo de la distancia entre MS y BTS.

El programa utiliza el teorema de Pitágoras para el cálculo de la distancia entre ambos equipos.

El valor de potencia obtenido se utiliza para decidir cuál es la estación base de la que el equipo MS recibe más cobertura. En caso de establecimiento de un canal para llamada o señalización, se elige la BTS de la que se detecte una potencia más alta. Este cálculo se lleva a cabo en los módulos MS, y AIR, y se juega siempre con valores en dBm. Cuando se realiza en el módulo MS el objetivo es conocer cuáles son las BTS con mayor intensidad de recepción. En AIR se guarda esta información para la realización de gráficas de potencia.

En el caso de que se haya establecido una llamada, el módulo MS calcula según un intervalo de tiempo t definido por el usuario, las potencias actualizadas que recibe la MS. Estos datos se calculan en MS, y se envían a BTS y BSC en un mensaje denominado MEASREP. Será el equipo BSC, controlador de BTS el que decide la necesidad de realización de un traspaso en función de estos valores.

8.1.8 Elección de BTS.

La BTS mejor para la comunicación se elige en función de la potencia que reciba la MS. El módulo AIR conoce las posiciones de MS y BTS, por lo tanto es el que elige cuál es la BTS de la que obtiene más cobertura, y redirige el mensaje de establecimiento de conexión hacia el módulo BTS correspondiente. En caso de no tener ninguna BTS para la comunicación, el módulo AIR le informa al módulo MS que se encuentra fuera de cobertura, y que por tanto no se puede conectar a ningún otro equipo.

8.1.9 Establecimiento de canales.

El establecimiento de canales se ha programado de forma similar al establecimiento de

llamadas en el paso de mensajes entre equipos. De hecho el establecimiento de llamadas exige los dos tipos de establecimiento que se han implementado, para tráfico y para señalización.

8.1.10 Liberación de canales.

En el programa se pueden dar tres casos para la liberación de canales. El primer caso consiste en la liberación de un canal de señalización. El segundo de un canal de tráfico, y el tercero, la liberación de un canal de tráfico debido a un traspaso. En los tres casos se actúa de manera similar para liberar los recursos. Cuando llega el mensaje RF_REL_ACK al módulo BSC, la función que maneja este mensaje lee los vectores en busca del identificador de la MS y borra los contenidos que se han almacenado para esta MS. La llegada del mensaje CHANREL al equipo MS hace que se cambie el estado del móvil a NONE (cuando se encuentra conectado se encuentra en un estado CONN_REQ). También se cambia el valor de la variable connected y la de number_bts_connected a -1, y se ponen a cero los contadores de llamadas (en el caso de ser una liberación por llamada), después de grabar el último valor en el fichero de escalares. El borrado de estos datos se hace mediante el método erase() de la clase vector definida en la librería estándar de C++.

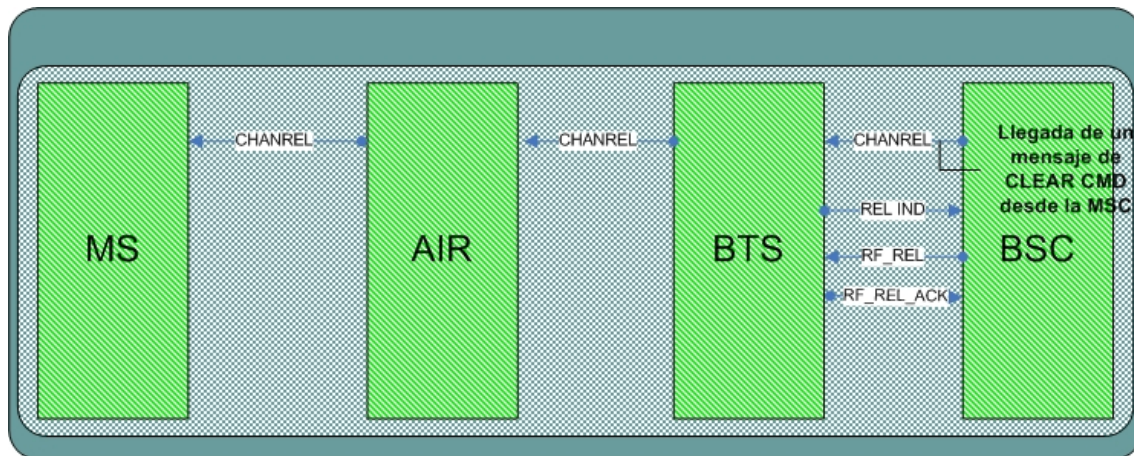


Figura 41 Liberación de canal.

Cuando llega un mensaje de CLEAR COMMAND al módulo BSC, éste envía un mensaje CHANNEL RELEASE hacia la MS. Con este procedimiento se libera la conexión entre MS y BTS.

8.1.11 Procedimientos MAP.

Los procedimientos MAP son los procedimientos que intercambian información entre los

equipos pertenecientes al sistema de conmutación. En el programa sólo se han añadido los equipos VLR y MSC, por lo que los procedimientos se reducen a los que involucran únicamente a los dos equipos. Se han incluido los procedimientos para actualización de posición y para el establecimiento de llamadas. El equipo MSC lleva dos vectores para almacenar los identificadores de las MS origen y destino para las que se establece la llamada. En caso de que la llamada tenga como destino un MS que no esté presente en la simulación, se almacena el valor -1. Cuando finaliza la llamada, se borran ambos valores en ambos vectores.

El módulo VLR contiene varios vectores que almacenan la posición de una MS. Para ello se almacena el identificador de la MS, el de BTS, el de BSC, MSC y el LAI. También guarda el MSISDN y el TMSI, aunque en esta programa el valor TMSI no se usa, se ha añadido para futuras ampliaciones del simulador.

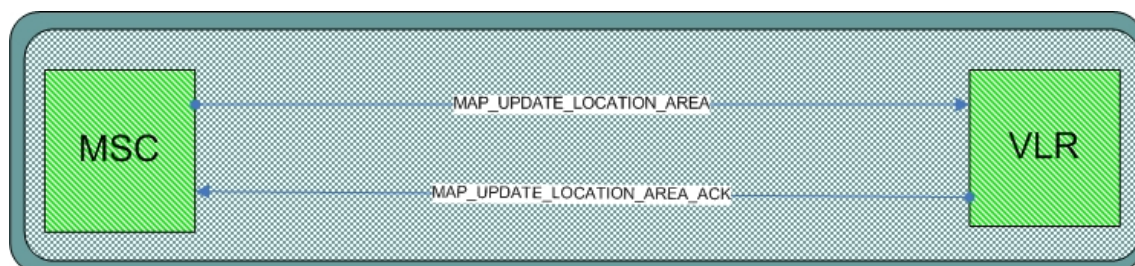


Figura 42 Procedimientos MAP para la actualización de posición.

Para la actualización se envían únicamente dos mensajes, uno en dirección VLR, y otro de VLR a MSC. El mensaje que llega al MSC se retransmite hacia la MS indicando si la actualización ha sido correcta (LUACC) o fallida (LUREJ).

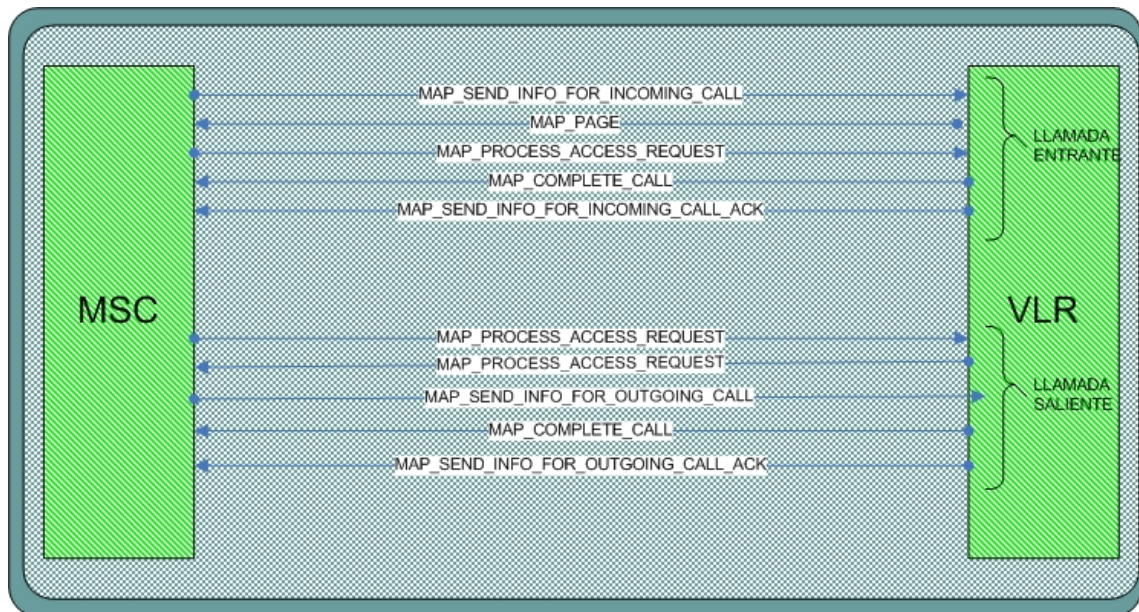


Figura 43 Procedimientos MAP para el establecimiento de llamada entrante (arriba) o saliente (abajo).

El procedimiento de establecimiento de una llamada se inicia en la MSC.

8.1.12 Realización de traspasos.

Los traspasos se controlan desde la BSC. Cuando se ha establecido un canal de tráfico, se envían medidas de potencia desde la MS hasta la BSC. La BSC observa estos parámetros y decide si es necesario un cambio o no. Si la potencia se acerca al valor umbral y no hay ninguna estación para realizar el cambio, se mantiene la comunicación hasta que se corta y no recibe. En este caso se trata de una llamada caída, y un traspaso erróneo, por lo que se modifican ambos valores en el fichero de escalares.

Si la MS se encuentra en el área de cobertura de varias estaciones base, la BSC siempre conecta a la que mejor cobertura tenga en el caso de que se encuentre disponible. Cuando la potencia recibida es menor que la de las estaciones de alrededor menos 9 dB, entonces es necesario que realice un cambio de BTS. En GSM se define el valor de -9dB de diferencia entre la señal y la señal interferente para el primer canal adyacente. Para el segundo canal adyacente es de -41 dB y para el tercero de -49 dBm. Al ser más restrictivo el primer valor, es el que se tiene en cuenta en la programación de la función para la realización de traspasos. Para las frecuencias cocanales este valor es de +9dBm.

Cuando se decide la realización de un traspaso el módulo BSC solicita a la nueva BTS la

asignación de un canal. Cuando queda establecido, el módulo BTS nuevo es el que sigue con la comunicación con el móvil. La realización de traspasos es muy interesante para esta aplicación, ya que se ha desarrollado de forma que sea visible desde la pantalla de visualización de la red como se cambia el paso de una estación a la otra.

Cuando se ha realizado correctamente, tras la llegada del mensaje HOCMP al módulo BSC, éste modifica el valor de la BTS a la que se encuentra el móvil conectado, y hace lo mismo en el VLR mediante una actualización de posición.

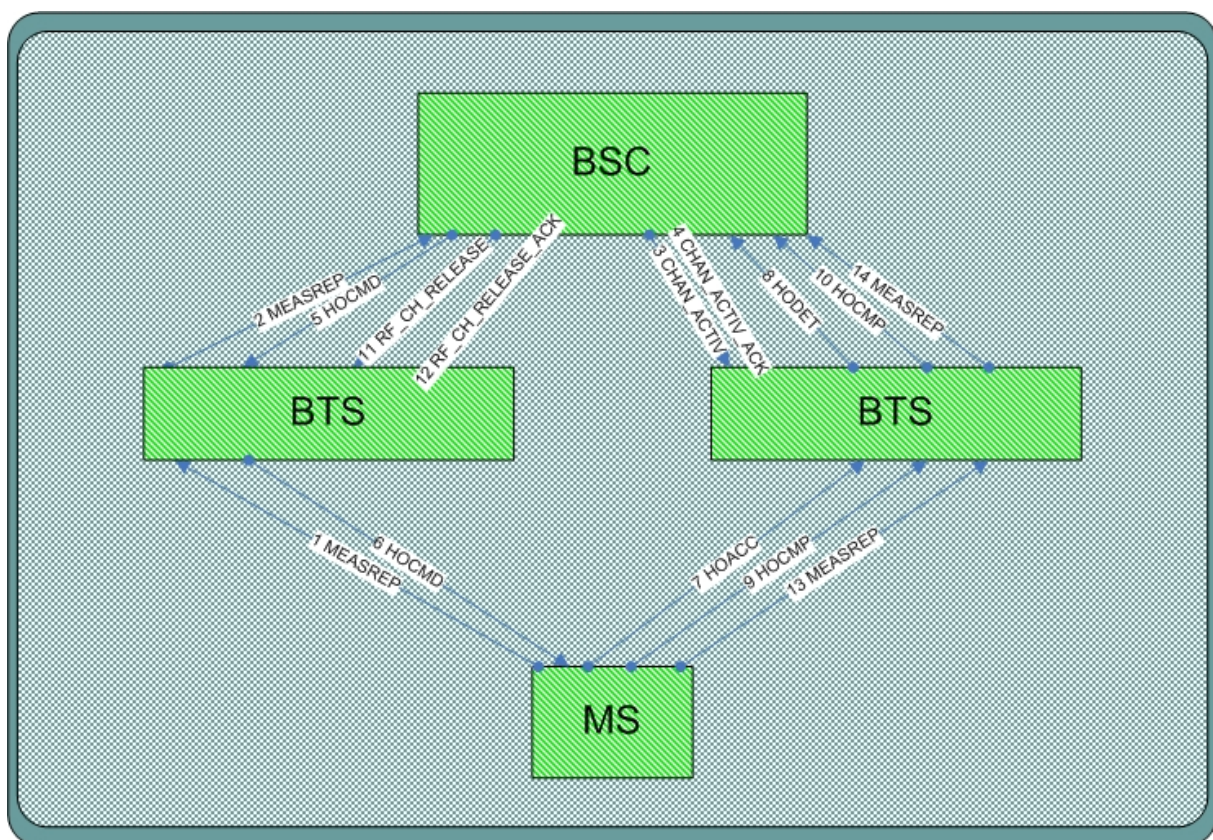


Figura 44 Realización de traspasos.

La BSC decide cuando debe realizar un traspaso en función de los valores de potencia que se envían en el mensaje de informe de potencias. El canal en la nueva BTS debe establecerse antes de que la BTS antigua pierda la comunicación con el equipo móvil. Cuando se ha establecido esta comunicación, se liberan los recursos de la antigua estación base.

8.2 Ficheros

8.2.1 El fichero “omnetpp.ini”

El fichero omnetpp.ini es el fichero que usan las aplicaciones desarrolladas con OMNET++ para iniciar sus parámetros en vez de ser introducidos por teclado. En el caso de esta aplicación se va a pasar a explicar como se compone este fichero y cuales son los parámetros que le pasa a la aplicación gsmsim.exe.

La estructura del fichero está formada por varias partes, indicadas por palabras encerradas entre corchetes. La primera que se observa es [General] en la cual se inician los elementos generales de la red.

La sentencia preload-ned-files sirve para indicar al compilador que cargue también los ficheros .ned durante la compilación de la aplicación. Esta línea no aparecerá en ficheros omnetpp.ini que pertenezcan a proyectos que usen versiones de OMNET anteriores a la 3.0. La razón se debe a que a partir de esta versión, no es necesario cargar manualmente el fichero .ned sino que realiza una carga dinámica del mismo, facilitando asimismo la tarea del programador.

Mediante la línea network se indica el nombre de la red que se va a ejecutar que para nuestro caso es gsmsim. Además de estas líneas, también contiene información de ficheros de salida de información, así como el tiempo máximo de ejecución (con dos opciones, de cpu y de simulación) para determinar el fin automático de la aplicación.

```
[General]
preload-ned-files=*.ned
network = gsmsim

ini-warnings = no
random-seed = 1
warnings = yes
snapshot-file = gsmsim.sna
output-vector-file = gsmsim.vec
sim-time-limit = 100s
cpu-time-limit= 1000m
total-stack-kb = 2048 ; 2MByte, increase if necessary
```

A continuación se encuentran los parámetros de inicialización de los entornos de simulación. El primero se corresponde a un entorno de comandos y el segundo al visual, que es el que se usará y en el que nos centraremos ahora. La línea default-run tiene como utilidad poder

seleccionar una ejecución entre varias posibles presentes en el fichero.

```
[Cmdenv]
module-messages = yes
verbose-simulation = yes
display-update = 0.5s
```

```
[Tkenv]
default-run = 1
use-mainwindow = yes
print-banners = yes
slowexec-delay = 300ms
update-freq-fast = 10
update-freq-express = 100
breakpoints-enabled = yes
```

La parte más importante del fichero es la zona en la cual se producen todas la inicializaciones de los parámetros de la red. Para ello se indica con la palabra Parameters dentro de corchetes. A continuación se pasa a comentar que significan cada uno de los parámetros de inicialización.

```
[Parameters]
```

Los dos primeros indican tanto el ancho como el alto de la red. Es decir, la posición espacial máxima de cada uno de los elementos de la red. Los parámetros que vienen a continuación son parámetros de la red gsm sim que indican la cantidad de estaciones base y estaciones móviles que habrá presentes en la red durante la simulación, y el parámetro exponencial de pérdidas para calcular las potencias. Otros valores no menos importantes son la probabilidad de que se genere una llamada entre dos equipos pertenecientes a la misma MSC, el retardo entre medidas de potencia, el retardo entre actualizaciones de posición, la longitud de llamada, la desviación estándar de la longitud de llamada, y el tiempo medio entre llamadas.

```
gsm sim.xwidth = 50000;
gsm sim.ydepth = 50000;
gsm sim.number_bts = 2;
gsm sim.number_ms = 2;
gsm sim.param_n = 2.6;

gsm sim.prob_intra_msc_call = 25;
gsm sim.delay_pwr_meas = 2.0;
gsm sim.time_lureq = 10.0;
gsm sim.call_length_mean = 120.0;
gsm sim.call_length_std_dev = 250.0;
```

```
gsmsim.call_interarrival_mean = 6000.0
```

Ahora se pasa a hablar de los parámetros de la red que sirven para inicializar cada uno de los módulos. Para el caso de la MS, los parámetros pasados son las posiciones dentro del espacio de simulación en función de dos coordenadas espaciales, y su velocidad de movimiento en módulo y ángulo. La velocidad será el método de simular el movimiento de una estación dentro de la zona espacial en la que se encuentra. El movimiento se realiza en una única dirección o en una dirección aleatoria durante toda la simulación. Los valores aleatorios conservan el módulo de la velocidad pero no el ángulo, y se cambian con un tiempo aleatorio. Ante la llegada a un límite, la estación móvil rebotará. Una posible mejora a aplicar sería añadir un algoritmo que permitiese además de cambiar de posición, cambiar también de dirección, haciendo por tanto una simulación bastante más real.

```
gsmsim.ms[0].xc = 0;  
gsmsim.ms[0].yc = 0;  
gsmsim.ms[0].vx = 30;  
gsmsim.ms[0].vy = 30;  
gsmsim.ms[0].pathType = 0;
```

```
gsmsim.ms[1].xc = 25000;  
gsmsim.ms[1].yc = 25000;  
gsmsim.ms[1].vx = -30;  
gsmsim.ms[1].vy = -30;  
gsmsim.ms[1].pathType = 1;
```

Siguiendo a la inicialización de los móviles, se encuentra la inicialización de los parámetros de la BTS. Cada módulo de la BTS recibe del fichero omnetpp.ini cinco parámetros, los cuales indican las coordenadas espaciales en dos dimensiones en las que se encuentra la estación dentro del espacio. El tercer y cuarto parámetros son los números de canales que tiene cada estación como máximo para tráfico y señalización respectivamente. El quinto es la potencia con la que puede transmitir esta estación.

```
gsmsim.bts[0].xc = 10000;  
gsmsim.bts[0].yc = 10000;  
gsmsim.bts[0].numTCH = 1;  
gsmsim.bts[0].numSDCCH = 32;  
gsmsim.bts[0].dBm = 33.53;
```

```
gsmsim.bts[1].xc = 35000;  
gsmsim.bts[1].yc = 35000;  
gsmsim.bts[0].numTCH = 1;  
gsmsim.bts[0].numSDCCH = 32;
```

```
gsmsim.bts[0].dBm = 38.11;
```

Tal y como se han iniciado los parámetros en este fichero .ini, la red representaría lo que viene a representar en el sencillo gráfico de ejemplo que se expone a continuación. En éste mismo se representan dos círculos de diferente color que representarían la zona de cobertura de la estación base, y cuatro puntos que indican la posición de las dos MS y las dos BTS.. El espacio sobre el que se simula es una hipotética zona de 50000 x 50000 metros de extensión donde las coordenadas que aparecen vienen indicadas en metros.

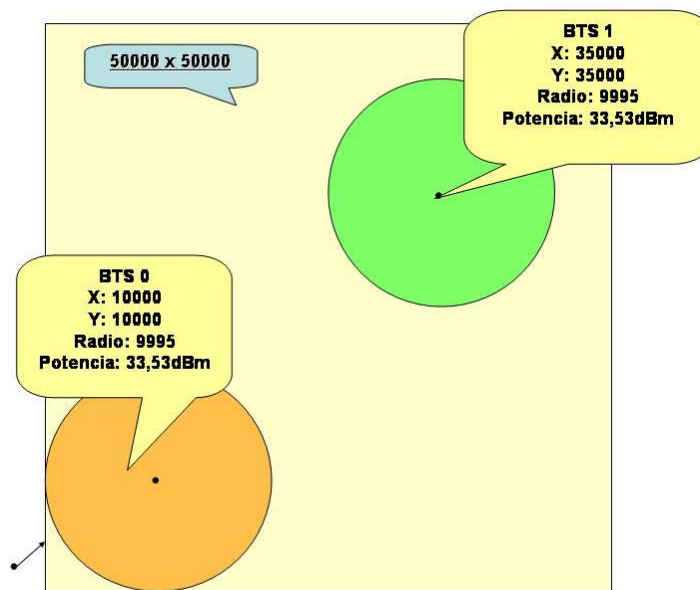


Figura 45 Plano espacial de posición de los equipos. Ejemplo 1.

Posición de las dos MS y las dos BTS en el plano. El ejemplo abarca una superficie de 50x50 Kilómetros

En un segundo caso, suponiendo que la BTS 1 emite con una potencia mayor En este ejemplo se reduce la zona de sombra entre ambas estaciones.

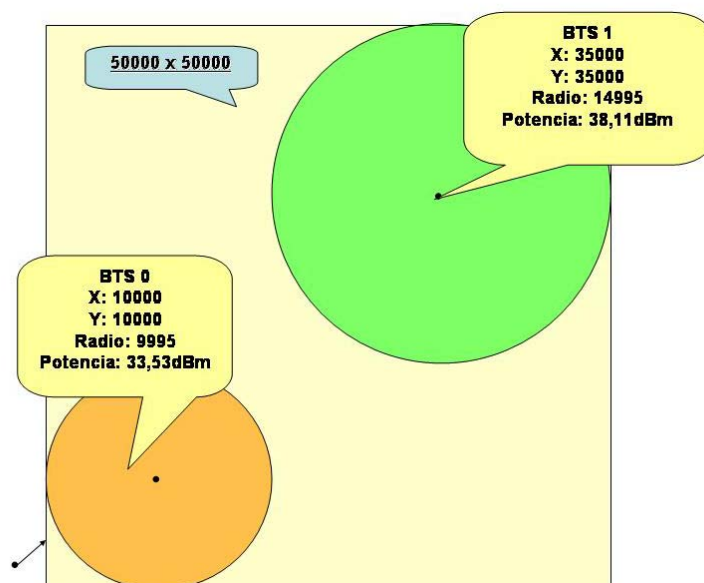


Figura 46 Plano espacial de posición de los equipos. Ejemplo 2.

Posición de las dos MS y las dos BTS. La BTS 0 emite con la misma potencia. La BTS1 emite con mayor potencia, aun que aún así queda una zona de sombra entre ambas estaciones.

8.2.2 El fichero "gsmSim.ned"

Los ficheros con extensión .ned son los que contienen la descripción de la red. En ellos se establecen las conexiones entre los diferentes módulos desarrollados en la red. Tenemos dos opciones para visualizar este tipo de archivo. Una es mediante el uso del bloc de notas mediante el cual podremos ver todo el código, y la otra opción es mediante la herramienta GNED que permite la creación de archivos de tipo .ned mediante un entorno visual. Este último tiene dos posibles opciones de visión, una es el modo código y la otra es en modo gráfico. La segunda permite ver con iconos y bloques los módulos de la red. En caso de no asignar una posición, será el entorno de simulación de OMNET el encargado de realizar todo el proceso de colocación de cada uno de los módulos. Se han definido seis módulos simples, Air, MS, BTS, BSC, MSC y VLR. Se ha definido un módulo compuesto que aune a MSC y VLR, y otro módulo compuesto que incluye todos los módulos anteriores. A partir de este módulo se define la red GSM.

```
simple Air
parameters:
/.../
```

```
    gates:
    /.../
endsimple

simple MS
  parameters:
  /.../
  gates:
  /.../
endsimple

/.../

simple MSC
  parameters:
  /.../
  gates:
  /.../
endsimple

simple VLR
  parameters:
  /.../
  gates:
  /.../
endsimple

module MSC_VLR
  parameters:
  /.../
  gates:
  /.../
  submodules:
    msc: MSC;
    parameters:
    /.../
    gatesizes:
    display: "p=174,97;i=device/server_1";
    vlr: VLR;
    parameters:
    /.../
    gatesizes:
    display: "p=41,95;i=device/mainframe_1";
  connections:
    msc.to_vlr --> vlr.from_msc;
```

```
        /.../  
    display: "b=238,171";  
endmodule  
  
module GSMSIM  
    parameters:  
        /.../  
    submodules:  
        /.../  
    connections:  
        /.../  
endmodule  
  
network gsmsim : GSMSIM  
    parameters:  
        number_bts = input(1,"Number of stations:"),  
        number_ms = input(1,"Number of cars:"),  
        xwidth = input(10000,"Width of the plane [m]:"),  
        ydepth = input(10000,"Depth of the plane [m]:");  
endnetwork
```

El resultado que queda al ejecutar la red con este fichero NED es el que se presenta en la Figura 47 . En la imagen se pueden ver los móviles en formando una matriz, y las BTS en una fila. Los otros elementos son la BSC y la BTS. Por último añadir que se ha añadido un elemento entre la BTS y la MS llamado AIR de forma que simplifique el desarrollo del código para la simulación de la comunicación a través de un medio aéreo. En el fichero .ned, además de definir cada uno de los módulos de la red, también se definen los iconos y características de visualización de cada uno de los módulos de la red.

Los iconos de los equipos MSC, VLR, y BSC son los únicos que llevan una posición de representación fija. Los equipos MS y BTS se colocan en función del número de equipos que haya a partir de unos valores introducidos. Los móviles se colocan desde el punto 0, 600 en una matriz con un máximo de 20 iconos por fila. Las BTS se colocan desde el punto 0, 200 en una fila. Estos iconos han sido obtenidos de la carpeta de iconos que proporciona OMNeT++, y se añaden junto con la posición de visualización.

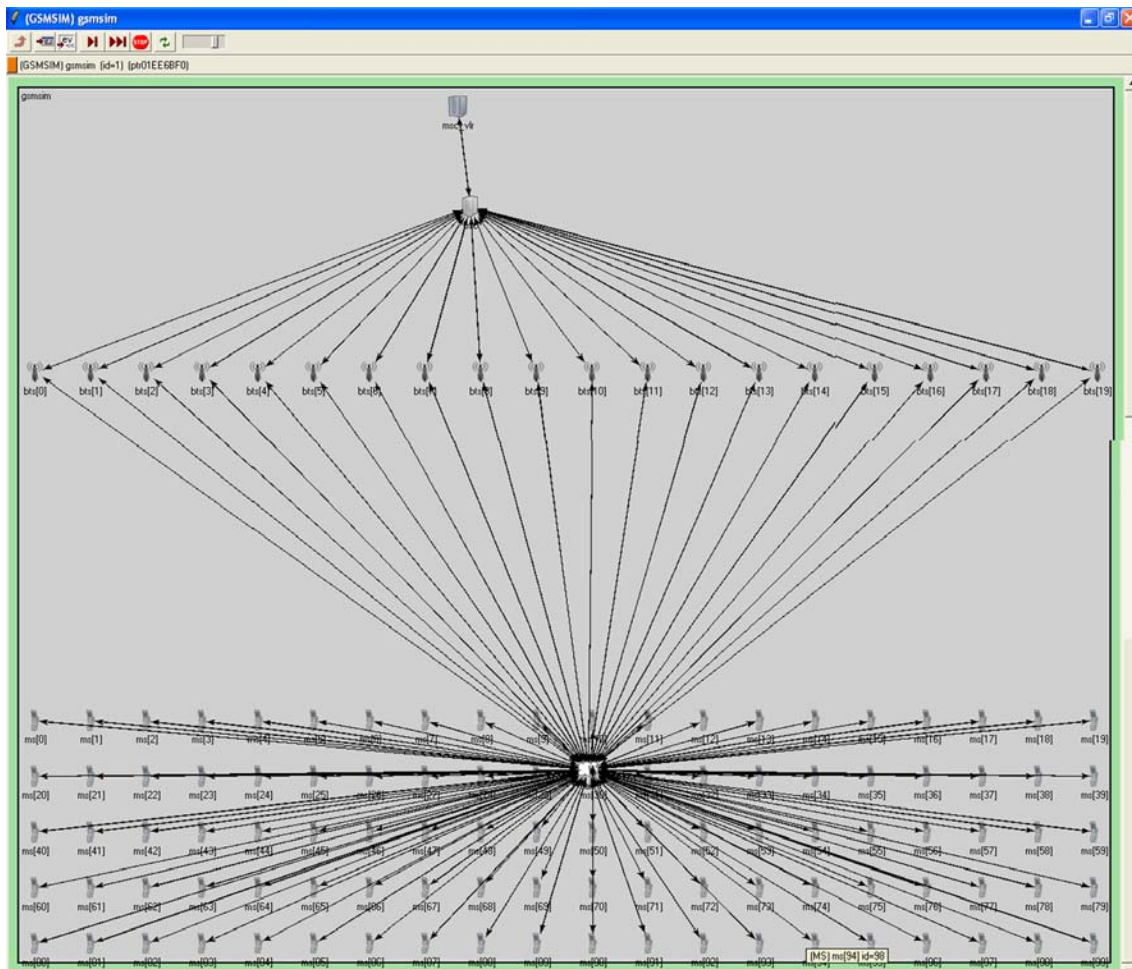


Figura 47 Ejemplo de simulación con GSMSIM.

Ejemplo de simulación con cien móviles, veinte estaciones base, una BSC, y un equipo conjunto con VLR y MSC. Las antenas se sitúan en la imagen en una fila. Los equipos móviles se sitúan en una matriz.

Son varios los parámetros que se introducen a los módulos desde el fichero de configuración. Para hacer más sencilla su localización, y su comprensión, se detallan en las siguientes tablas junto con los nombres y tipos de puertos que tiene cada equipo.

Parámetros de MS	
Nombre parámetro	Función

xc	Posición de la MS en el eje de las X
yc	Posición de la MS en el eje de las Y
vmod	Módulo de la velocidad
angle	Ángulo de la velocidad
pathType	Tipo de trayectoria
x_width	Ancho de la zona de simulación
y_depth	Alto de la zona de simulación
numbts	Número de BTS presentes en la red
n	Coefficiente exponencial de pérdidas
timeout	Tiempo
delay_pwr_meas	Retardo entre medidas de potencia
time_lureq	Tiempo entre actualizaciones de posición sucesivas
call_length_mean	Media del tiempo de llamada
call_length_std_dev	Desviación estandar del tiempo de llamada
call_interarrival_mean	Tiempo entre llamadas consecutivas

Tabla 10 Parámetros de MS

Parámetros de AIR	
Nombre parámetro	Función
phones	Guarda el número máximo de teléfonos.

numbts	Guarda el número máximo de BTS
n	Parámetro exponencial de pérdidas

Tabla 11 Parámetros de AIR

Parámetros de BTS	
Nombre parámetro	Función
phones	Número máximo de teléfonos
n	Parámetro exponencial de pérdidas
xc	Posición de la BTS en metros eje X
yc	Posición de la BTS en metros eje Y
numTCH	Número de canales TCH
numSDCCH	Número de canales de señalización

Tabla 12 Parámetros de BTS

Parámetros de BSC	
Nombre parámetro	Función
numbts	Número de BTS
phones	Número de teléfonos móviles

Tabla 13 Parámetros de BSC

Parámetros de MSC	
Nombre parámetro	Función
num_ms	Número de MS en la simulación

num_bts	Número de BTS en la simulación
num_msc	Número de MSC en la simulación
prob_intra_msc_call	Probabilidad de que una llamada que se genera en un móvil tenga como un destino un móvil perteneciente a la misma MSC.
num_lai	Número de identificación de área de localización.

Tabla 14 Parámetros de MSC

Parámetros de VLR	
Nombre parámetro	Función
num_ms	Número de MS en la simulación.
num_bts	Número de BTS en la simulación.
num_vlr	Número identificador de VLR
num_lai	Número identificador de área de localización.

Tabla 15 Parámetros de VLR

Parámetros de la red	
Nombre parámetro	Función
number_ms	Número de MS presentes en la red
number_bts	Número de BTS presentes en la red
xwidth	Ancho de la zona de simulación
ydepth	Altura de la zona de simulación

param_n	Parámetro exponencial de pérdidas
prob_intra_msc_call	Probabilidad de que una llamada sea dentro de una MSC
delay_pwr_meas	Tiempo entre medidas de potencias
time_lureq	Tiempo entre actualizaciones normales de posición.
call_length_mean	Tiempo medio de llamada
call_length_std_dev	Desviación estándar de tiempo de llamada
call_interarrival_mean	Tiempo entre llegada de llamadas

Tabla 16 Parámetros de la red

Puertas de la MS	
Nombre puerta y tipo	Conexión con
from_air (entrada)	AIR
to_air (salida)	AIR

Tabla 17 Puertas de la MS

Puertas de AIR	
Nombre puerta y tipo	Conexión con
to_ms[number_ms] (salida)	MS
from_ms[number_ms] (entrada)	MS
to_bts[number_bts] (salida)	BTS
from_bts[number_bts] (entrada)	BTS

Tabla 18 Puertas de AIR

Puertas de la BTS	
Nombre puerta y tipo	Conexión con
from_air (entrada)	AIR
from_bsc (entrada)	BSC
to_air (salida)	AIR
to_bsc (salida)	BSC

Tabla 19 Puertas de BTS

Puertas de la BSC	
Nombre puerta y tipo	Conexión con
from_bts[] (entrada)	BTS
to_bts[] (salida)	BTS
from_msc (entrada)	MSC
to_msc (salida)	MSC

Tabla 20 Puertas de la BSC

Puertas de la MSC	
Nombre puerta y tipo	Conexión con
from_vlr (entrada)	VLR
to_vlr (salida)	VLR
from_bsc (entrada)	BSC

to_bsc (salida)	BSC
-----------------	-----

Tabla 21 Puertas de la MSC

Puertas del VLR	
Nombre puerta y tipo	Conexión con
from_msc (entrada)	MSC
to_msc (salida)	MSC

Tabla 22 Puertas del VLR

Conexiones de la red	
Origen	Destino
ms.from_air (entrada)	air.to_ms[number_ms] (salida)
ms.to_air (salida)	air.from_ms[number_ms] (entrada)
air.to_bts[number_bts] (salida)	bts.from_air (entrada)
air.from_bts[number_bts] (entrada)	bts.to_air (salida)
bts.from_bsc (entrada)	bsc.from_bts[] (entrada)
bts.to_bsc (salida)	bsc.to_bts[] (salida)
bsc.from_msc (entrada)	msc.from_bsc (entrada)
bsc.to_msc (salida)	msc.to_bsc (salida)
msc.from_vlr (entrada)	vlr.from_msc (entrada)
msc.to_vlr (salida)	vlr.to_msc (salida)

Tabla 23 Conexiones de la red

8.2.3 El fichero "gsmsim.h"

La necesidad de este fichero de cabecera dentro del código se debe a que en él se definen los valores de todos los mensajes, además de ciertos parámetros interesantes para la simulación

como más adelante se ve. La primera línea tiene como objetivo incluir la cabecera de las librerías de OMNET++, que en este caso se encuentran en la ruta especificada. Conviene que esta línea sea revisada si se quiere recompilar todo el código ya que el directorio depende donde tenga cada ordenador instalada las librerías de OMNET++.

```
#include "C:/omnet++/include/omnetpp.h"
```

A continuación se definen parámetros para el reintento de llamadas. También se definen las frecuencias de GSM. En GSM se utilizan dos bandas, una en los 900MHz y otra en los 1800MHz. Para calcular las potencias, se utiliza sólo la banda de los 900MHz, ya que el simulador solo cuenta con esta banda. La definición de la segunda banda se ha realizado para futuras ampliaciones.

```
#define CALL_INTERRETRY_MEAN 30.0
```

```
#define FREQUENCY_GSM_900 900.0  
#define FREQUENCY_GSM_1800 1800.0
```

Además se definen valores de estado de la conexión para el móvil.

```
#define PHONE_STATE_CONNECTED 1  
#define PHONE_STATE_DISCONNECTED 2
```

Se define un valor de tiempo muy alto que se usará como infinito en alguna de las funciones que vayamos a usar.

```
#define INFINITY_TIME 99999999999.99
```

Para diferenciar el tipo de canal al que se refieren los mensajes se emplean dos definiciones, una para canales de tráfico y otro para canales de señalización.

```
#define ID_TCH_CHANNEL 1  
#define ID_SIGNAL_CHANNEL 0
```

Se definen las tasas de transmisión de información. Estas velocidades se dan en bits por segundo.

```
#define DATA_RATE_A 65536 // bit/s [64 Kbit/s]  
#define DATA_RATE_A_BIS 65536 // bit/s [64 Kbit/s]  
#define DATA_RATE_Um_CCCH 781 // bit/s  
#define DATA_RATE_Um_SDCCH 390 // bit/s
```



```
#define DATA_RATE_Um_SACCH 390 // bit/s
```

Se añade la longitud de los mensajes en bytes. Esta longitud se ha extraído de los estándares de GSM publicados por el ETSI.

```
#define LNG_ASSGN_CMP 3
#define LNG_MEASREP 18
/.../
#define LNG_LUREQ_UM 19
#define LNG_LUREJ_UM 3
```

También se incluyen identificadores para los módulos. Existen seis tipos de módulos en función de las clases. Estos módulos son AIR, MS, BTS, BSC, MSC y VLR.

```
#define ID_AIR 10000
#define ID_MS 10001
#define ID_BTS 10002
#define ID_BSC 10003
#define ID_MSC 10004
#define ID_VLR 10005
```

Las llamadas entrantes se definen con un 0 y las salientes con un 1.

```
#define INCOMING_CALL 0
#define OUTGOING_CALL 1
```

Se define un número base de MSISDN. Este número sumado al del identificador de la MS da como lugar el MSISDN de un móvil.

```
#define MSISDN_STD_NUMBER 600900000
```

También se realiza la definición de varios tipos para el uso de los diferentes módulos. Estos tipos son definidos a partir de las clases vector que se hallan en las librerías estándar de C++. Los tipos definidos son un vector para datos de tipo double, otro para datos de tipo entero, otro para datos de tipo long, y por último un vector para guardar vectores para datos de tipo double.

```
typedef vector<double> Vector_Double;
typedef vector<Vector_Double> Vector_Double_Vectors;
typedef vector<int> Vector_Int;
typedef vector<long> Vector_Long;
```

Los tipos de mensajes usados se han descrito dentro de un enumerado para que sea más

cómodo su manejo. Al no tener un nombre el enumerado, los elementos del mismo se acceden exactamente igual que si no fuesen parte de un enumerado pero con la ventaja de que son inicializados desde el valor cero (el primer elemento) incrementandose un valor hasta el último.

```

/.../
INIT,
NO_SIGNAL,
CONN_REQ,
CHECK_BTS,
CHECK_BTS_ACK,
MOVE_MS,
CALL_END,
    APAGADO,
/.../
```

A continuación se definen los tipos de acceso a la red que una MS puede hacer. El estado NONE es aquel en el cual la MS no desea realizar ninguna petición sobre la red. PAGRES corresponde a un acceso a la red como consecuencia de la llegada de un mensaje de PAGING desde la BTS que indica una llamada entrante o un deseo de establecer comunicación con el terminal por parte de la red. LUREQ corresponde a un caso en el cual la MS necesita actualizar su posición dentro de la red. CMSREQ indica el deseo de establecer una llamada y SMSREQ se usará para indicar el deseo de enviar un mensaje.

```

enum {
    NONE,           // NINGUNA ACCIÓN, NO ESTABLECE CANAL
    PAGRES,         // PARA RESPUESTA A UN PAGING
    LUREQ,          // PARA ACTUALIZACIÓN DE POSICIÓN
    CMSREQ,         // PARA LLAMADA ORIGINADA EN MS
    SMSREQ,         // PARA ENVÍO DE MENSAJE ORIGINADO EN MS
    HANDOVER        // PARA TRASPASO
};
```

8.2.4 El fichero "ms.cpp"

8.2.4.1 Atributos

own_addr: número identificador de MS.

`connected`: indica el estado del móvil. Si es -1 desconectado, 0 conectado a canal de señalización, y 1 conectado a canal de tráfico.

`num_ms`: número total de MS en la red.

`num_bts`: número total de BTS en la simulación.

`number_bts_connected`: identificador de BTS a la que se encuentra conectada. Si es -1 está desconectada.

`Counter_bts`: contador de BTS.

`counterRetry`: contador para la espera de un nuevo intento de llamada cuando se ha producido un bloqueo.

`pathType`: tipo de trayectoria.

`xc`: double que almacena la posición de la MS en el eje X.

`yc`: double que almacena la posición de la MS en el eje Y.

`vmod`: almacena el módulo de velocidad.

`angle`: almacena el ángulo del vector velocidad.

`xcLimit`: límite de movimiento en el eje X. .

`ycLimit`: límite de movimiento en el eje X.

`param_n`: parámetro exponencial de pérdidas.

`L0`: Pérdidas en el primer metro.

`Vect_NoBTS`: Vector de enteros para almacenar los números identificadores de BTS a la hora de calcular las potencias. Actúa de índice para el resto de vectores. Si por ejemplo se quiere conocer el valor de potencia de la BTS 5, busca el valor 5 en el vector y guarda la posición. A continuación va a la posición guardada en el vector de potencias y obtiene el valor.

`Vect_NoBTS_Rx`: Vector de enteros para almacenar los números identificadores de BTS a la hora de calcular las potencias.

`Vect_X`: Vector double que almacena la posición de la BTS en el eje X.

`Vect_Y`: Vector double que almacena la posición de la BTS en el eje Y.

`Vect_Radius`: Vector double que almacena los radios de cobertura.

`Vect_Watt`: Vector double que almacena las potencias.

`InfoPwr`: Vector double que almacena las potencias.

`tiempo_llamada`: tiempo de llamada. Valor aleatorio asignado según una distribución exponencial.

`inicio_llamada`: instante en que se inicia una llamada, controla que este valor más el de tiempo de llamada no sea superior al valor de tiempo devuelto por la función `simTime()`.

`marca_tiempo`: instante de tiempo.
`marca_interarrival`: instante de tiempo para realizar esperas entre llamadas.
`velVctrInit`: instante de tiempo en que se modifico el vector velocidad.
`velVctrLifeTime`: tiempo de vida del vector velocidad.
`retryCall`: tiempo en volver a realizar una llamada.
`lastTry`: instante de tiempo en que se produjo la última llamada.

8.2.4.2 Atributos de salida

`iMissedCalls`: llamadas perdidas o bloqueadas por la red.
`iCalls`: llamadas realizadas por la estación móvil.
`iBroken`: llamadas caídas durante la duración de la llamada debido a fallos de cobertura.
`iHandover`: número de traspasos que soporta una MS.

8.2.4.3 Métodos

`virtual void initialize()`: Realiza la inicialización de los parámetros de cada objeto de la clase MS. Además inicializa las variables para la impresión de los datos de salida.

`virtual void handleMessage(cMessage *msg)`: Es la función que se encarga de tratar y manipular los mensajes de tipo `cMessage`. Para ello lo que hace es a partir de los tipos de mensajes llamar a la función correspondiente que se encarga de tratarlo.

`virtual void finish()`: Es la función de finalización. Cuando es llamada además de finalizar todas las variables, escribe en un fichero los datos escalares. Borra el tamaño y el contenido de los vectores pertenecientes al objeto.

`virtual bool call_query ()`: Es la función que permite que un objeto MS realice la simulación de una llamada según un valor devuelto por una variable aleatoria con una distribución exponencial.

`virtual bool sms_query ()`: Es la función que permite que un objeto de la clase MS realice el envío de un mensaje.

`virtual bool lureq_query (int &status):` Permite que la MS actualice su posición de forma periódica o cada vez que se enciende.

`virtual bool turnon_terminal():` Función que enciende el terminal móvil.

`virtual void order_vector():` Función que ordena los vectores de potencia en función de la potencia de mayor valor a menor.

`virtual void func_movems (cMessage * msg):` Función de tratamiento del mensaje MOVE_MS. Este tipo de mensaje es un mensaje especial de la simulación. Se envía cada módulo a sí mismo con la función Schedule. Dentro del código se realizan diferentes operaciones en función del estado del móvil. Si está apagado se llama a la función `turnon_terminal` para que se encienda con una probabilidad.

`virtual void func_checkbtsack (cMessage * msg):` Función que emplea la MS ante la llegada de un mensaje CHECK_BTS_ACK, para que pueda pedir un canal de señalización para conectarse con la BTS.

`virtual void func_imassign (cMessage * msg):` Función para tratar los mensajes de asignación inmediata de un canal de señalización.

`virtual void func_asscmdbtsms (cMessage * msg):` función para tratar los mensajes de asignación de un canal de tráfico.

`virtual void func_measrep (cMessage * msg):` Función que se encarga del tratamiento y envío de los mensajes de informe de medidas. Cada vez que se llama a esta función, se envía un mensaje hacia el módulo aire con las potencias de las seis estaciones base que recibe con mayor potencia. Éstas son ordenadas de mayor a menos potencia con la función `order_vector()`.

`virtual void func_pagreq (cMessage * msg):` Es la función que se encarga de analizar los mensajes de tipo paging que vienen desde la MSC.

`virtual void func_hocmd (cMessage * msg):` Es la función que se encarga de tratar un mensaje de tipo handover command para que se realice una petición de handover.

`virtual void func_chanrel (cMessage * msg):` función para petición de liberación de canal.

`virtual double CalculateWatt(double dblMSX,double dblMSY,double BTSradius):` método para calcular las potencias que detecta cada estación móvil de las antenas que tiene a su alrededor. Para ello se le pasa como parámetros la coordenada en el eje X de la posición de la MS, y la del eje Y. También se le pasa el radio de cobertura de la BTS.

`virtual void func_immassignrej (cMessage * msg):` función para que rechaza la asignación de un canal.

`virtual void func_luacc (cMessage * msg):` función que recoge un mensaje de aceptación de actualización de posición.

8.2.5 El fichero "air.cpp"

El módulo air.cpp simplemente redirige los mensajes que se envían desde la BTS o la MS hacia los destinos correspondientes, y realiza la escritura de los datos de potencia en los ficheros de salida. Su declaración es la siguiente:

```
class Air : public cSimpleModule
{
    /.../
};
```

8.2.5.1 Atributos

`num_bts:` número real que almacena la posición de la BTS en el eje X.

`num_ms:` número que almacena la posición de la BTS en el eje Y.

`L0:` real doble que almacena la atenuación en el medio a la distancia de un metro.

`param_n:` entero que almacena el total de canales TCH usados.

`Vect_BTS_X:` Vector double que almacena la posición de una BTS en el eje X.

`Vect_BTS_Y:` Vector double que almacena la posición de una BTS en el eje Y.

`Vect_Radius`: Vector double que almacena el radio de cobertura de una BTS.

`Vect_Watt`: Vector double que almacena la potencia máxima en dBm de una BTS.

`VectMS_X`: Vector double que almacena la posición de la MS en el eje X.

`VectMS_Y`: Vector double que almacena la posición de la MS en el eje Y.

8.2.5.2 Atributos de salida

`ptrPowerReports`: puntero que almacena una matriz de dos dimensiones de punteros a objetos `cOutVector`. El tamaño de esta matriz es de número máximo de BTS x número máximo de MS.

8.2.5.3 Métodos

`virtual void initialize ()`: función que se emplea para la inicialización de los parámetros de los objetos de la red. Reserva espacio de forma dinámica para el puntero `ptrPowerReports`, y asigna un objeto a cada puntero perteneciente a la matriz. Asigna unos tamaños a los vectores atributo de la clase, siendo el tamaño el número máximo de MS.

`virtual void handleMessage (cMessage * msg)`: Extrae el valor de tipo de mensaje, y en función de éste, en un bloque switch llama a la función correspondiente.

`virtual void finish()`: función llamada cada vez que se va a finalizar un objeto de tipo BTS. Graba los datos escalares.

`~Air()`: destructor que se encarga de borrar todas las referencias de la matriz de punteros referenciada por `ptrPowerReports`. Libera también la memoria asociada. Es importante que esta porción de código sea llamada cada vez que se deja de usar un módulo AIR, ya que de lo contrario daría lugar a grandes lagunas de memoria.

`CalculateWatt(double dblMSX,double dblMSY)`: Función empleada para conocer la potencia que detecta una MS de una BTS. Calcula la distancia a la MS. Si la distancia es menor que el radio de cobertura, entonces se pasa a calcular la potencia recibida. En caso contrario se devuelve un valor por debajo del valor de sensibilidad umbral, evitando la realización de operaciones.

`virtual void func_checkbts(cMessage * msg)`: Función encargada de ver si se detecta suficiente potencia de una BTS. En caso contrario se devuelve hacia MS con un valor de potencia igual a el límite, que será interpretado por la estación móvil como fuera de cobertura.

`virtual void func_checkbtsack(cMessage * msg)`: Función encargada de ver si la BTS funciona correctamente. Le devuelve información hacia MS.

`virtual void func_init(cMessage * msg)`: Función de inicialización de valores. Guarda los valores en los vectores, que luego serán empleados en el cálculo de potencias.

`virtual void func_measOutVector (cMessage * msg):` para una MS introducida como parámetro guarda la potencia recibida en la matriz de punteros a `cOutVector`.

`virtual void func_luacc(cMessage * msg):` reenvía un mensaje de actualización de posición.

`virtual void func_pagreq(cMessage * msg):` reenvía un mensaje de búsqueda.

8.2.6 El fichero "bts.cpp"

Es el fichero en el que se encuentra implementada la clase BTS derivada de `cSimpleModule` y que a su vez se encarga de modelar el funcionamiento de la BTS. A continuación se comenta el código desarrollado. La clase BTS es una clase derivada de la clase `cSimpleModule`. Al ser una derivación pública, la clase hija hereda todos los atributos y métodos de la clase, además de tener acceso a todos los atributos públicos de la clase, aunque no así de los privados.

```
class BTS : public cSimpleModule
{
    //..
};
```

8.2.6.1 Atributos

`dblXc`: número real que almacena la posición de la BTS en el eje X.

`dblYc`: número que almacena la posición de la BTS en el eje Y.

`dblRadius`: radio máximo de cobertura, calculado en función de la potencia máxima, y de la atenuación.

`dblPwr`: potencia asignada a la BTS.

`number_of_bts`: entero que almacena el total de BTS empleadas en la simulación.

`L0`: real doble que almacena la atenuación en el medio a la distancia de un metro.

`iTrafficConnections`: entero que almacena el total de canales TCH usados.

`iSignalConnections`: entero que almacena el total de canales de señalización usados en una BTS.

`iConnections`: entero que almacena el total de conexiones.

`iTCHChannels`: entero que almacena el total de canales de tráfico existentes en una BTS.

`iSDCCHChannels`: entero que almacena el total de canales de señalización existentes en una BTS.

`iPhones`: entero que almacena el total de estaciones móviles empleadas en la simulación.

`iBTS`: entero que almacena el identificador de estación BTS.

`param_n`: coeficiente exponencial de pérdidas empleado para el cálculo de potencias..

`LAI`: entero que almacena el área de localización.

8.2.6.2 Atributos de salida

Estos atributos son almacenados durante la simulación, y al finalizarla se graban en ficheros de resultados para ser visualizados con los programas Plove o Scalars.

`handoverTotal`: entero que almacena el total de traspasos durante la simulación.
`handoverOK`: entero que almacena el total de traspasos correctos.
`handoverFailure`: entero que almacena la cantidad de traspasos fallidos.
`assignTotal`: entero que almacena el total de asignaciones de canal.
`assignOK`: entero que almacena el total de asignaciones correctas.
`assignFailure`: entero que almacena el total de asignaciones fallidas.
`llamadasBloqueadas`: entero que almacena el total de llamadas bloqueadas.
`llamadasCaidas`: entero que almacena el total de llamadas caídas durante la simulación.
`llamadasTotal`: entero que almacena el total de conexión establecidas para llamadas generadas en la BSC durante un periodo de tiempo.
`llamadasTotalSca`: entero que almacena el total de llamadas.
`llamadasCursadasSca`: entero que almacena el total de llamadas cursadas dentro de una BTS.
`llamadasEntrantes`: entero que almacena el total de llamadas entrantes dentro de una BTS.
`llamadasSalientes`: entero que almacena el total de llamadas salientes dentro de una BTS.
`TCHOcupados`: entero que almacena el número de llamadas salientes.
`TCH_Total`: entero que almacena el número de llamadas entrantes.
`TCH_Ocupados`: entero que almacena el número de llamadas entrantes.
`LlamadasTotal`: objeto de tipo `cOutVector` que almacena las llamadas establecidas en función del tiempo.
`LlamadasCursadas`: objeto de tipo `cOutVector` que almacena las llamadas salientes y las escribe en el fichero `gsmsim.vec`.
`LlamadasEntrantes`: objeto de tipo `cOutVector` que almacena las llamadas salientes y las escribe en el fichero `gsmsim.vec`.
`LlamadasSalientes`: objeto de tipo `cOutVector` que almacena las llamadas salientes y las escribe en el fichero `gsmsim.vec`.

8.2.6.3 Métodos

`virtual void initialize ():` función que se emplea para la inicialización de los parámetros de los objetos de la red. Es llamada cada vez que se crea un módulo de tipo BTS, y cada vez que se reinicializa la simulación. Envía un mensaje de inicio al módulo AIR. El mensaje es del tipo INIT. Escribe identificador, posición y radio en un fichero que pueda ser cargado por la aplicación SimoffLine.

`virtual void handleMessage (cMessage * msg):` función para manipular los mensajes que llegan a la BTS desde el módulo AIR o BSC. La función será llamada cada vez que la simulación detecte que ha llegado un mensaje a un objeto. Contiene un bloque para captura de excepciones. Si se produce una excepción dentro de un módulo creado a partir de esta clase, se recoge se avisa en la pantalla de eventos y la ejecución continúa.

`virtual void finish():` función llamada cada vez que se va a finalizar un objeto de tipo BTS. Graba los datos escalares.

`double CalculateWatt(double dblMSX,double dblMSY):` Función empleada para conocer la potencia que detecta una MS de una BTS. Es muy similar a la que se encuentra en la clase MS y AIR, aunque con unas pequeñas diferencias. Sin embargo, los resultados deben ser iguales en los tres objetos ante unos idénticos parámetros de entrada. Calcula la distancia a la MS. Si la distancia es menor que el radio de cobertura, entonces se pasa a calcular la potencia recibida. En caso contrario se devuelve un valor por debajo del valor de sensibilidad umbral, evitando la realización de operaciones.

`virtual void func_checkbts(cMessage * msg):` Función encargada de ver si la BTS funciona correctamente para a continuación enviar un mensaje de petición.

`virtual void func_chanreq(cMessage * msg):` función que trata un mensaje de tipo CHAN_REQ, se usa para petición de un canal de señalización SDCCH. La BTS responderá con un mensaje de tipo CHAN_RQD hacia BSC.

`virtual void func_chanactiv(cMessage * msg):` función que maneja la llegada de un mensaje de tipo CHAN_ACTIV para activación de un canal de tráfico. El módulo responde con un mensaje de tipo CHAN_ACTIV_ACK. Si no se puede realizar la

activación debido a la saturación de la BTS, devuelve un mensaje CHAN_ACTIV_NACK.

`virtual void func_iacmd(cMessage * msg):` función que responde ante un mensaje de tipo immediate assign command. Envía un mensaje de tipo immediate assign command a través del canal AGCH del interfaz Um. Hacia la BSC responde con un mensaje de tipo ESTIN.

`virtual void func_asscmdbscbts(cMessage * msg):` actúa ante la llegada de un mensaje de petición de asignación de canal entre bsc y bts.

`virtual void func_assgncmpmsbts(cMessage * msg):` actúa ante la llegada de un mensaje de asignación completa de canal entre ms y bts.

`virtual void func_rfchrel(cMessage * msg):` función que libera un canal de comunicación en el interfaz A bis después de que se reciba un mensaje de tipo RFCHRELEASE. Reduce el valor en uno de las llamadas totales o de los canales de tráfico o señal utilizados.

`virtual void func_callend(cMessage * msg):` función que es llamada ante la llegada de un automensaje de tipo CALL_END. Este mensaje es enviado desde la función `func_movems()` cuando detecta que ha superado el tiempo de llamada que se ha obtenido al inicio de la llamada a partir de una función aleatoria.

`virtual void func_measrep(cMessage * msg):` measrep son los mensajes que informan de las medidas de potencia a la BSC para que decida la necesidad de un traspaso, etc.

`virtual void func_hocmd(cMessage * msg):` función que indica a la BTS que una estación móvil necesita un traspaso. Para ello lo que hace es reenviarlo a la MS que siendo la BSC la que gestione el traspaso en caso de que sea intra-BSC o si es entre MSC o BSC se deja la gestión del traspaso al módulo MSC.

`virtual void func_hoacc(cMessage * msg):` función que maneja la llegada de un mensaje de tipo handover access desde la MS.

`virtual void func_hocmp(cMessage * msg):` función que maneja la llegada de un mensaje de tipo handover complete.

`virtual void func_chanrel(cMessage * msg):` función de petición de liberación de canal.

`virtual void func_immassrej(cMessage * msg):` función de aviso de asignación rechazada.

`virtual void func_luacc(cMessage * msg):` función de actualización de posición.

`virtual void func_pagcmd(cMessage * msg):` función de búsqueda de móvil.

8.2.7 El fichero "bsc.cpp"

El fichero contiene la declaración y la definición del módulo BSC que simula el funcionamiento de una BSC. Al igual que todas las anteriores, se deriva a partir de la clase `cSimpleModule`. Al ser una derivación pública, hereda todos los atributos y métodos de la clase base, es decir todos los que sean públicos para la clase base, también serán básicos para la clase derivada. Al igual que las otras, tiene una función constructora, una de inicialización, otra de finalización y otra de gestión de mensajes. Es conveniente implementar el destructor si dentro de la clase se realiza alguna reserva de memoria que puede dar lugar a molestas lagunas de memoria en caso de no ser finalizadas correctamente. En un principio esta clase se implementó con un destructor, ya que se reservaba memoria de forma dinámica. En la solución final se ha suprimido este destructor y eliminado la asignación de memoria dinámica. Si en algún caso se mantiene una asignación dinámica, en estos casos es mejor implementar el destructor que incluir el código en la función `finish()` ya que ocurren casos en los que la función de finalización no es llamada al cerrar el programa, mientras que el destructor se invoca siempre que el programa tenga que destruir un objeto creado.

```
class BSC : public cSimpleModule
{
    //...
```

};

8.2.7.1 Atributos

num_bts: entero que almacena el total de MS empleadas en la simulación.
own_addr: entero que almacena el total de BTS empleadas en la simulación.
LAI: entero que almacena el total de BSC empleadas en la simulación.
number_of_bsc: entero que almacena el total de MS empleadas en la simulación.
Vector_Pwr: entero que almacena el total de BTS empleadas en la simulación.
Search_Index: entero que almacena el total de BSC empleadas en la simulación.
Num_BTS: entero que almacena el total de MS empleadas en la simulación.
State_MS: entero que almacena el total de BTS empleadas en la simulación.

8.2.7.2 Atributos de salida

Estos atributos son almacenados durante la simulación, y al finalizarla se graban en ficheros de resultados para ser visualizados con los programas Plove o Scalars.

handoverTotal: entero que almacena el total de traspasos durante la simulación.
handoverOK: entero que almacena el total de traspasos correctos.
handoverFailure: entero que almacena la cantidad de traspasos fallidos.
assignTotal: entero que almacena el total de asignaciones de canal.
assignOK: entero que almacena el total de asignaciones correctas.
assignFailure: entero que almacena el total de asignaciones fallidas.
llamadasBloqueadas: entero que almacena el total de llamadas bloqueadas.
llamadasCaidas: entero que almacena el total de llamadas caídas durante la simulación.
llamadasTotal: entero que almacena el total de conexión establecidas para llamadas generadas en la BSC durante un periodo de tiempo.
llamadasTotalSca: entero que almacena el total de llamadas.
llamadasCursadasSca: entero que almacena el total de llamadas cursadas dentro de una BSC.
llamadasSalientes: entero que almacena el número de llamadas salientes.
llamadasEntrantes: entero que almacena el número de llamadas entrantes.
LlamadasTotal: objeto de tipo cOutVector que almacena las llamadas establecidas en función del tiempo.
LlamadasSalientes: objeto de tipo cOutVector que almacena las llamadas salientes y las escribe en el fichero gsmsim.vec.

LlamadasEntrantes: objeto de tipo cOutVector que almacena las llamadas entrantes y las escribe en el fichero gsmsim.vec.

8.2.7.3 Métodos

`virtual void initialize():` La función inicializa los atributos con los valores especificados en el fichero de configuración, pone a cero todos los atributos para salida de datos, y añade un nombre a los objetos de la clase cOutVector.

`virtual void handleMessage(cMessage *msg):` La función handleMessage tiene como parámetro recibido un mensaje que le llega al objeto BSC. Al igual que el resto de funciones handleMessage en el resto de módulos también se ha añadido los bloques try y catch para capturar y tratar una posible excepción que se pueda producir durante la ejecución. Graba datos en los objetos de cOutVector, y en algunos casos realiza llamadas a la función bubble().

`virtual void finish():` Vacía los vectores creados, y graba los datos escalares en el fichero gsmsim.sca.

`virtual void func_chanrqd (cMessage * msg):` En caso de llegar un mensaje de tipo CHANRQD, este contiene información acerca de la potencia con la que recibe el móvil la transmisión de esta estación base. Como parámetros se recibe el número de la MS y la potencia recibida. En caso de que sea menor que el límite de sensibilidad (-102dBm), esto indica que la estación MS no puede conectarse a ninguna BTS. En caso de que sea mayor que este valor, entonces, se comprueba que en el vector Vector_Pwr no hay ningún elemento y tampoco lo hay en Search_Index o en Num_BTS. En caso de estar vacíos se añade directamente sobre el primer elemento.

En caso de que los vectores no se encuentren vacíos, lo que hay que hacer es añadir un elemento a este vector con la información sobre esa estación. Los vectores definidos en la librería estándar de C++ a parte de ser una herramienta muy útil en el uso de matrices o vectores de información, proporcionan un buen lote de métodos con los que acceder a estos elementos y realizar búsquedas, etc. El único inconveniente que presentan es que necesitan más espacio que si se usasen punteros a matrices, sin embargo evitan gran número de errores de lagunas de memoria además de proporcionar un mecanismo que lanza una excepción del tipo out_of_range que indica que se accede a una posición del vector que no está siendo usada. En el caso de

punteros el mismo error puede dar lugar a resultados inesperados durante la ejecución mientras que con este método, el error puede ser tratado.

El bloque contenido en `else` lo que hace es buscar en `Search_Index` (el vector que almacena los números de las MS que se encuentran a disposición del BSC) el móvil que le ha pasado el mensaje de tipo `CHANREQ`. En caso de hallarlo dentro del vector, se comprueba en el vector `Num_BTS` (que almacena el identificador de BTS a la cual está conectado cada uno de los móviles que están bajo control de esta BSC), esta almacenado el mismo número que el número de BTS que se encuentra en el mensaje. En caso de no ser así, Se comprueba la potencia guardada y la que recibe de la nueva BTS y entonces decide realizar el traspaso de estación base. En caso de no ser así, sale del bucle `for`. En caso de que no sea distinto, simplemente añade los valores recibidos. `State_MS` es el único vector de los utilizados de los que todavía no se ha hablado. En este se almacenan los estados de cada teléfono, es decir si se encuentra sin ningún canal asignado (-1), si tiene un canal de señalización (0) o si tiene un canal de tráfico asignado (1). Por último es caso de no encontrarlo en el vector, se añade directamente como se puede ver en el bloque `else` al final de este bloque `for`.

Esta es la parte de código de la que se ha hablado antes en la cual se realiza la captura de la excepción de tipo `out_of_range`. Gracias a estas líneas en el supuesto de que nuestra ejecución pretenda acceder a una posición no permitida del vector, el programa no finalizará inadecuadamente, sino que se avisará al usuario del problema y seguirá la simulación.

Por último, la BSC responde a este mensaje de `CHANREQ` mediante el mensaje `CHANACTIV`, que lleva como información el número de BTS (`iDest`), el de MS (`iMS`) y el tipo de petición que ha recibido de acceso al canal (`Query`) que puede ser para llamada, actualización de localización, apagado-encendido de Terminal o para enviar un mensaje.

```
virtual void func_chanactivack (cMessage * msg):
```

A continuación si el establecimiento del canal ha sido correcto, entonces, la estación responde con un mensaje de asignación inmediata si se trata de asignación de un canal de señalización, o un mensaje de asignación para tráfico. Si se debe a un traspaso envía hacia BTS un mensaje de handover command.

`virtual void func_chanactivnack (cMessage * msg):` igual que el anterior pero con la diferencia de que recibe el aviso de una asignación errónea o de que no se puede realizar la asignación de canal a ese MS, devuelve un mensaje de asignación rechazada (reject).

`virtual void func_estin (cMessage * msg):` función que maneja la llegada de un mensaje establish indicator. Responde con el envío de un mensaje cl3i hacia la MSC.

`virtual void func_assgnreq (cMessage * msg):` cuando llega un mensaje de asignación requerida, la BSC responde con el envío de un mensaje de CHANACTIV hacia la BTS.

`virtual void func_assgncmpbtsbsc (cMessage * msg):` cuando llega un mensaje de asignación completa de canal entre bts y bsc se llama a esta función. Se inicia el procedimiento de liberación del canal de señalización.

`virtual void func_relind (cMessage * msg):` función de release indicator, informa de la necesidad de liberar un canal, envía un mensaje de liberación.

`virtual void func_rfchrelack (cMessage * msg):` función de tratamiento ante una liberación de canal correcta. Borra los datos almacenados de la MS en los vectores.

`virtual void func_measrep (cMessage * msg):` función que trata los informes de medidas. Si pierde la cobertura, borra los datos, si debe cambiar de BTS, envía a la BTS una petición de establecimiento de un canal.

`virtual void func_hocmp (cMessage * msg):` función que se encarga de recibir el mensaje de finalización correcta de traspaso. Envía un informe de aviso hacia la MSC si el traspaso es intra BSC, y solicita el procedimiento de desconexión para la BTS antigua.

`virtual void func_clearcmd (cMessage * msg):` función usada para liberar un canal entre BSC y BTS. Avisa de la liberación a MSC, envía un mensaje de liberación a MS y otro a BTS en caso de que sea debido a un traspaso.

`virtual void func_paging (cMessage * msg):` función usada para buscar una MS. Reenvía el mensaje hacia la BTS.

`virtual void func_callend (cMessage * msg):` función que indica el fin de una

llamada. Solicita liberación de un canal de tráfico.

`virtual void func_luacc (cMessage * msg):` función usada para informar de la actualización correcta de posición en VLR.

8.2.8 El fichero “msc.cpp”

El fichero contiene la declaración y la definición del módulo MSC que simula el funcionamiento de una MSC. Éste módulo junto con el módulo VLR forman el módulo compuesto MSC-VLR. De cara a la programación en C++ se programa los dos módulos simples de forma independiente.

```
class MSC : public cSimpleModule
{
Module_Class_Members(MSC,cSimpleModule,0)
~MSC();
};
```

8.2.8.1 Atributos

`num_ms`: entero que almacena el total de MS empleadas en la simulación.

`num_bts`: entero que almacena el total de BTS empleadas en la simulación.

`num_bsc`: entero que almacena el total de BSC empleadas en la simulación.

`MSOrig`: vector de enteros que almacena los identificadores de estaciones móviles que originan una llamada.

`MSDest`: vector de enteros que almacena los identificadores de estaciones móviles que son destino de una llamada telefónica.

8.2.8.2 Métodos

`virtual void initialize():` función de inicialización de un módulo MSC. Inicializa los atributos de esta clase.

`virtual void handleMessage(cMessage *msg):` función que se encarga de la gestión de mensajes que llegan a un módulo de la clase MSC.

`virtual void finish():` función que se encarga de finalizar un módulo. Pone a cero todos los atributos del objeto de la clase, y vacía los vectores.

`virtual void func_cl3i (cMessage * msg):` función que se encarga del tratamiento de un mensaje CL3I que llegue desde la BSC. Dependiendo del tipo de servicio requerido, esta función enviará un mensaje de respuesta u otro. Si el servicio solicitado en una llamada o la respuesta a un aviso de búsqueda, se envía hacia el VLR un mensaje de MAP_PROCESS_ACCESS_REQUEST. Si el servicio es la actualización de posición, el mensaje MAP_UPDATE_LOCATION_AREA se envía hacia el VLR.

`virtual void func_assgncmpbscmcs (cMessage * msg):` función que maneja la llegada de un mensaje de asignación completa.

`virtual void func_clearcmp (cMessage * msg):` mensaje que informa a la MSC de la liberación completa de un canal.

`virtual void func_clearreq (cMessage * msg):` función que solicita la liberación de un canal. Borra los datos correspondientes a la MS origen y MS destino en los vectores de la clase. Responde hacia la BSC con un mensaje de CLEAR_CMD.

`virtual void func_handoverperformed (cMessage * msg):` el mensaje de HANDOVER_PERFORMED informa de la realización de un traspaso intra BSC. Actualiza la posición almacenada en el VLR mediante el paso de un mensaje MAP_UPDATE_LOCATION_AREA hacia el registro de visitantes.

`virtual void func_mapupdatelocationareaack (cMessage * msg):` mensaje recibido cuando se ha actualizado la localización. Envía un mensaje de actualización de posición hacia el equipo BSC.

`virtual void func_mapsendinfoforincomingcallack (cMessage * msg):` Recoge un mensaje de tipo MAP_SEND_INFO_FOR_INCOMING_CALL_ACK enviado por un equipo VLR.

`virtual void func_mappage (cMessage * msg):` Recibe información del equipo VLR para la búsqueda de un equipo móvil. Si el equipo móvil se encuentra conectado el equipo envía un mensaje de PAGING hacia la BSC con destino la MS. Si el móvil está apagado, inicia el procedimiento de liberación del canal.

`virtual void func_mapprocessaccessrequestack (cMessage * msg):` Si la llamada es una llamada de salida, se envía hacia VLR un mensaje `MAP_SEND_INFO_FOR_OUTGOING_CALL`, y si el móvil destino está dentro del rango de equipos móviles que se está simulando, se envía también un mensaje `MAP_SEND_INFO_FOR_INCOMING_CALL`.

`virtual void func_mapcompletetecall (cMessage * msg):` ya sea para llamadas salientes como para llamadas entrantes se envía un mensaje para completar la llamada de asignación hacia el equipo BSC. El equipo BSC asigna un canal entre MS y BSC.

`virtual void func_mapsendinfoforoutgoingcallack (cMessage * msg):` es un mensaje de información que no genera ningún nuevo mensaje de respuesta.

`virtual void func_hocmp (cMessage * msg):` función que se encarga de liberar el canal en la BTS antigua una vez que se ha terminado de establecer un canal en la BTS nueva.

8.2.9 El fichero "vlr.cpp"

El fichero contiene la declaración y la definición del módulo VLR que simula el funcionamiento de una VLR. Se trata de una clase derivada de la clase base `cSimpleModule`.

```
class VLR : public cSimpleModule
{
Module_Class_Members(VLR,cSimpleModule,0)
/.../
};
```

8.2.9.1 Atributos

`num_ms`: entero que almacena el total de MS empleadas en la simulación.

`num_bts`: entero que almacena el total de BTS empleadas en la simulación.

`num_bsc`: entero que almacena el total de BSC empleadas en la simulación.

`VectMSnum`: vector de enteros que almacena el identificador de una MS.

`VectBTSnum`: vector de enteros que almacena el identificador de BTS dentro de la cual se encuentra la MS definida en `VectMSnum`.

`VectBSCnum`: vector de enteros que almacena el identificador de BSC.

`VectMSCnum`: vector de enteros que almacena el identificador de MSC.

VectLAI: vector de enteros que almacena el identificador de área de localización.

VectMSISDN: vector de 'long' que almacena el número de móvil dentro de la red.

VectTMSI: vector de enteros que almacena el número temporal asignado al equipo.

8.2.9.2 Métodos

`virtual void initialize()`: función de inicialización de un módulo VLR. Inicializa los atributos de esta clase. Asigna un tamaño a los vectores igual al número de móviles existentes en la simulación. Si se introduce por configuración un valor de 30 móviles, los vectores tendrán un tamaño de 30, siendo `VectorNombre.at(0)` el primer valor y `VectorNombre.at(29)` el último. Una vez dimensionados los vectores se les dan valores por defecto. El identificador de MS vendrá dado por el identificador que le asocie el simulador al crear el objeto MS, y el número MSISDN es la suma de este índice con un valor base asignado en el fichero de cabecera y cuyo valor es 600900000. El valor de este número no es de importancia, sólo se ha añadido para dar una numeración más similar a una red telefónica. Si el índice que usamos para identificar a un móvil es el 24, el número MSISDN y por tanto el que identifica al equipo para establecer llamadas es 600900024.

`virtual void handleMessage(cMessage *msg)`: función que se encarga de la gestión de mensajes que llegan a un módulo de la clase VLR. Tiene un bloque de código que permite la captura de excepciones en tiempo de ejecución. Está diseñada para que sea cual sea la excepción creada, sea capturada antes de abandonar la función, y se imprima por pantalla un mensaje de error.

`virtual void finish()`: función que se encarga de finalizar un módulo. Borra con el método `clear()` todo el contenido de los vectores y además elimina el dimensionamiento realizado con la función `initialize()`. Esto quiere decir que si se ha asignado una dimensión de 5 en la función de inicio. Cuando se llame a la función de finalización, la dimensión de los vectores será de cero.

`virtual void func_mapupdatelocationarea ()`: función de manejar un mensaje de petición de actualización de posición. Busca la información correspondiente al móvil indicado en el parámetro 'numms' del mensaje. Actualiza los datos en los vectores y devuelve un mensaje de confirmación hacia la MSC denominado

MAP_UPDATE_LOCATION_AREA_ACK. Imprime en la pantalla de eventos el identificador de MS modificado junto con las estaciones bajo las que se encuentra el equipo.

`virtual void func_mapcancellocation():` mensaje que borra la posición de un móvil en el VLR debido a un cambio de VLR o a un apagado del equipo.

`virtual void func_mapcancellocationack():` acepta el borrado de la información de un equipo.

`virtual void func_mapsendinfoforincomingcall():` mensaje que busca información sobre un móvil destino de una llamada y que genera un mensaje para aviso de búsqueda hacia la MSC que se reenviará hasta la MS destino. Busca los datos de posición del equipo MS dentro de los vectores de información.

`virtual void func_mapprocessaccessrequest():` acepta el procedimiento de acceso enviando hacia MSC un mensaje MAP_PROCESS_ACCESS_REQUEST_ACK, en caso de ser una llamada entrante, envía un mensaje para completar el establecimiento de llamada, y otro mensaje para aceptar la información de llamada entrante.

`virtual void func_mapsendinfoforoutgoingcall():` devuelve un mensaje para completar llamada hacia MSC y otro de aceptación de información de llamada saliente.

8.3 Mensajes incluidos en la aplicación.

Durante la simulación, los módulos intercambian mensajes entre sí. Estos mensajes siguen los procedimientos establecidos en los estándares de GSM. No se han incluido todos los mensajes presentes en este tipo de redes, ya que el número de mensajes es muy extenso. Por ello se incluyen en la aplicación una pequeña parte de los mensajes de nivel de red, los más representativos dentro de la comunicación entre equipos. Estos mensajes utilizados son principalmente los de establecimiento de canales, liberación, trasposos y control de potencias. Algunos de los mensajes que aparecen en la implementación no pertenecen realmente a una red GSM. De entre estos mensajes se destacan los denominados INIT y MOVE_MS. El

primero es empleado por el programa para realizar inicializaciones en la red, y el segundo para el cambio de posición del móvil durante la simulación.

Los mensajes se crean a partir de objetos de la clase base `cMessage`. Para añadir parámetros, se emplea la clase `cPar` y el método `addPar()` de la clase base `cMessage`. Cada uno de los mensajes es creado en origen mediante el operador “new” de C++, y destruido con el operador “delete” en destino. Si no se destruyesen los mensajes al final de su utilización en los módulos destino, la aplicación necesitaría un gran espacio en memoria para almacenarlos, además de correr el riesgo de crear lagunas de memoria. Además para los mensajes se han asignado unos valores de longitud en bytes definidos en los estándares de GSM. Estas longitudes se han definido en el fichero de cabecera “`gsmSim.h`”.

En función de esta longitud de mensaje y la tasa de transmisión de bits, se calculan los retardos en los canales. En el libro de [10] Comunicaciones Móviles GSM se presentan las velocidades de transmisión de bits de información por cada tipo de canal en el interfaz Um, y en el resto se ha supuesto una velocidad de 64Kbit/s.

A continuación se habla de los mensajes pertenecientes a los diferentes interfaces de GSM. En algunos casos aunque los mensajes pertenecen a diferentes interfaces, llevan el mismo nombre en ambos, ya que realizan la misma función o incluso pertenecen al mismo procedimiento pero tienen un nombre diferente dependiendo de la interfaz. Los primeros que se van a mencionar son los mensajes propios de la simulación. Estos mensajes no tienen importancia de cara al usuario, sólo de cara a la programación de la aplicación. Seguidamente se hablará de los mensajes del interfaz Um, Abis, A y B.

8.3.1 Mensajes propios de la aplicación.

Se han desarrollado cuatro mensajes propios de la aplicación para acciones que no tienen que ver propiamente con la una red GSM sino que se emplean en la simulación para que los módulos puedan realizar la simulación. Estos mensajes son los que se mencionan a continuación, junto con una explicación de los parámetros de los mensajes. En la Tabla 24 se tiene una breve descripción de estos mensajes y de sus parámetros.

- INIT: mensaje de inicialización. Generado en MS o BTS y recibido en AIR.
 - type : tipo de mensaje
 - src: número de equipo origen del mensaje

- xc: coordenada x en el plano de simulación.
- yc: coordenada y en el plano de simulación.
- radius: radio de cobertura, sólo para la BTS.
- MOVE_MS: mensaje de cambio de posición. Automensaje del módulo MS.
 - status: estado del móvil. Puede ser APAGADO, MOVE_MS, CHECK_BTS, CONN_REQ. El primero indica que el MS está desconectado de la red. El segundo que está a la espera de un mensaje de cambio de posición, o de cualquier mensaje de petición. El tercero indica que se encuentra a la espera de la respuesta de un chequeo de estación base por parte del módulo aire, y el último indica que se encuentra a la espera de una conexión o se encuentra conectado.
 - query: indica la petición de conexión de un móvil. Puede ser ninguna (NONE), de un canal para respuesta a un mensaje de búsqueda (PAGRES), de respuesta a actualización de posición (LUREQ), de petición de llamada (CMSREQ), de petición de envío de SMS (SMSREQ), o de traspaso (HANDOVER).
- CHECK_BTS: mensaje de chequeo de BTS.
 - src: indica el número de MS origen del mensaje
 - xc: indica la posición de la MS en el eje X
 - yc: indica la posición de la MS en el eje Y
 - query: indica la petición de un servicio al igual que el parámetro del mismo nombre de MOVE_MS.
- CHECK_BTS_ACK: mensaje de chequeo de BTS confirmado.
 - src: indica el equipo fuente
 - watt: indica la potencia recibida por el MS después de que haya sido calculada por el equipo AIR.

- numberbts: indica el número de BTS al que se debe conectar el equipo MS destino de este mensaje.
- query: igual que en mensajes anteriores sirve para indicar un tipo de petición.

Mensajes propios de la aplicación			
Nombre Mensaje	Parámetros	Dirección	Función
INIT	type, src, xc, yc, radius	MS→AIR BTS→AIR	Inicializar en el módulo aire valores de la red necesarios para el encaminamiento de mensajes o para el cálculo de potencias.
MOVE_MS	status, query,	MS→MS Automensaje	Mensaje utilizado para la actualización de la posición del móvil. Se envía cada segundo de tiempo simulado. También sirve para modificar el estado de una estación o la petición de canal.
CHECK_BTS	src, xc, yc, query	MS→AIR→BTS	Mensaje que emplea el simulador para que la MS avise al módulo AIR a que estación base debe conectarse. El módulo AIR reenvía el mensaje hacia la BTS.
CHECK_BTS_ACK	src, watt, numberbts, query	BTS→AIR→MS	Responde hacia el móvil con el número de BTS al que debe conectarse la MS.

Tabla 24 Mensajes propios de la simulación

8.3.2 Mensajes interfaz Um

El interfaz Um es el interfaz radio entre los equipos MS y BTS. En la simulación se ha introducido un módulo entre estos dos equipos denominado AIR y que representa el aire o

medio de transmisión aéreo. En la Tabla 24 se tiene una breve descripción de estos mensajes y de sus parámetros.

- ASSCMD_BTS_MS: asignación de canal. Generado en BTS y recibido en MS.
 - src: número de BTS fuente del mensaje.
 - query: indica la petición de conexión de un móvil. Puede ser ninguna (NONE), de un canal para respuesta a un mensaje de búsqueda (PAGRES), de respuesta a actualización de posición (LUREQ), de petición de llamada (CMSREQ), de petición de envío de SMS (SMSREQ), o de traspaso (HANDOVER).
- ASSGNCMP_MS_BTS: asignación de canal completa. Entre MS y BTS.
 - src: equipo MS que envía el mensaje.
 - dest: identificador de equipo destino del mensaje.
 - query: indica la petición.
- CHANREQ: mensaje de petición de canal.
 - src: indica el número de MS origen del mensaje
 - xc: indica la posición de la MS en el eje X
 - yc: indica la posición de la MS en el eje Y
 - query: indica la petición de un servicio al igual que el parámetro del mismo nombre de MOVE_MS.
 - dest: indica el número de BTS de destino.
- CHREL: mensaje de liberación de canal.
 - src: indica el equipo fuente
 - dest: indica el equipo destino del mensaje.
 - netrelease: indica si la liberación es llevada a cabo por la red o por el móvil.

- query: igual que en mensajes anteriores sirve para indicar un tipo de petición.
- HOACC: acceso a una BTS de una MS debido a un traspaso.
 - dest: indica el equipo destino del mensaje.
 - numms: identificador de MS.
- HOCMD: mensaje de petición de traspaso.
 - src: indica el equipo fuente
 - dest: indica el equipo al que va destinado el mensaje.
 - numms: indica el número de MS.
 - newbts: indica el identificador de la nueva BTS.
 - query: indica el tipo de petición de canal.
- HOCMP: mensaje finalización de traspaso.
 - dest: indica el equipo destino.
 - numms: numero identificador de MS.
 - xc: posición en el eje X.
 - yc: posición en el eje Y.
- IMM_ASSIGN: asignación de canal.
 - src: indica el equipo fuente
 - dest: identificador de equipo destino.
 - numms: número de MS
 - newbts: número de la nueva BTS.
 - query: tipo de petición.

- IMM_ASSIGN_REJ: asignación de canal rechazada.
 - src: indica el equipo fuente.
 - dest: indica el equipo destino.
 - query: igual que en mensajes anteriores sirve para indicar un tipo de petición.
- LUACC: mensaje de confirmación de actualización de posición.
 - numms: indica el equipo fuente.
 - numbts: indica el equipo BTS destino.
- MEASREP: mensaje de medidas de potencia.
 - src: indica el equipo fuente.
 - numms: indica el equipo fuente del mensaje.
 - query: indica el tipo de petición.
 - nobtsbase1: número identificador de BTS.
 - nobtsbase2: número identificador de BTS.
 - nobtsbase3: número identificador de BTS.
 - nobtsbase4: número identificador de BTS.
 - nobtsbase5: número identificador de BTS.
 - nobtsbase6: número identificador de BTS.
 - wattbase1: potencia recibida por la MS. Emitida por la BTS identificada en nobtsbase1.
 - wattbase2: potencia recibida por la MS. Emitida por la BTS identificada en nobtsbase2.
 - wattbase3: potencia recibida por la MS. Emitida por la BTS identificada en nobtsbase3.

- wattbase4: potencia recibida por la MS. Emitida por la BTS identificada en nobtsbase4.
- wattbase5: potencia recibida por la MS. Emitida por la BTS identificada en nobtsbase5.
- wattbase6: potencia recibida por la MS. Emitida por la BTS identificada en nobtsbase6.
- PAGREQ: mensaje de búsqueda de una MS.
 - numms: indica el equipo MS.
 - numbts: indica el equipo BTS.

Mensajes en el interfaz Um				
Nombre Mensaje	Parámetros	Dirección	Función	Longitud (Bytes)
ASSCMD_BTS_MS	src, query,	BTS→AIR →MS	Assignment command. Comando de asignación de canal entra BTS y MS	85
ASSGNCMP_MS_BTS	src, dest, query,	BTS→AIR →MS	Asignación completa entre MS y BTS	3
CHANREQ	xc, yc, dest, src, query	MS→AIR →BTS	Petición de canal	1
CHREL	src, dest, query, netrelease	BTS→AIR →MS	Channel release. Liberación de canal asignado.	6
HOACC	dest, numms	MS→AIR	Handover access.	1

		→BTS	Acceso de una MS a una nueva BTS debido a un traspaso	
HOCMD	src, dest, numms, newbts, query,	BTS→AIR →MS	Handover command.	118
HOCMP	dest, numms, xc, yc	MS→AIR →BTS	Handover complete. Traspaso finalizado	8
IMM_ASSIGN	src, dest, numms, newbts, query	BTS→AIR →MS	Asignación de canal	29
IMM_ASSIGN_REJ	src, dest, query	BTS→AIR →MS	Asignación de canal rechazada	29
LUACC	numms, numbts	BTS→AIR →MS	Aceptación de actualización de posición.	19
MEASREP	src, núms., query, wattbase1, nobtsbase1,watt base2, nobtsbase2,watt base3, nobtsbase3,watt base4, nobtsbase4,watt base5, nobtsbase5,watt	MS→AIR →BTS	Measures Report. Informes de medidas de potencia recibidas en la MS.	18

	base6, nobtsbase6,			
PAGREQ	numms, numbts	BTS→AIR →MS	Paging request. La red realiza la búsqueda del equipo móvil mediante este mensaje	40

Tabla 25 Mensajes interfaz Um

8.3.3 Mensajes interfaz Abis

El interfaz Abis es el interfaz entre los equipos BTS y BSC. En la Tabla 26 se tiene una breve descripción de estos mensajes y de sus parámetros.

- ASSCMD_BSC_BTS: mensaje de asignación de canal. Generado en BSC y recibido en BTS.
 - dest: número de equipo destino.
 - numms: número de MS a la que va destinado el mensaje.
 - query: indica la petición de conexión de un móvil. Puede ser ninguna (NONE), de un canal para respuesta a un mensaje de búsqueda (PAGRES), de respuesta a actualización de posición (LUREQ), de petición de llamada (CMSREQ), de petición de envío de SMS (SMSREQ), o de traspaso (HANDOVER).
- ASSGNCMP_BTS_BSC: asignación completa de canal.
 - src: equipo origen del mensaje.
 - numms: identificador de equipo destino del mensaje.
 - query: indica la petición.
- CHANREL: liberación de canal.

- dest: equipo origen del mensaje.
- numms: identificador de equipo destino del mensaje.
- netrelease: indica el tipo de liberación.
- query: indica la petición.
- CHANRQD: mensaje de petición de canal.
 - src: equipo origen del mensaje.
 - watt: potencia
 - handover: indica con un valor igual a 1 que se trata de una asignación para un trapaso.
 - numms: número de MS a la que se le va a asignar un canal.
 - query: indica la petición.
- CHANACTIV: mensaje de cambio de posición. Automensaje del módulo MS.
 - src: equipo origen del mensaje.
 - dest: identificador de equipo destino del mensaje.
 - numms: número de MS.
- CHANACTIVACK: mensaje de cambio de posición. Automensaje del módulo MS.
 - src: equipo origen del mensaje.
 - numms: número de MS.
 - handover: indica si la activación de un canal se debe a un handover.
 - query: indica la petición.
- CHANACTIVNACK: mensaje de cambio de posición. Automensaje del módulo MS.
 - src: equipo origen del mensaje.

- numms: número de MS.
- handover: indica si la activación de un canal se debe a un handover.
- query: indica la petición.
- ESTIN: mensaje de cabio de posición. Automensaje del módulo MS.
 - src: equipo origen del mensaje.
 - numms: identificador de MS.
 - query: indica la petición.
- HODET: mensaje de cabio de posición. Automensaje del módulo MS.
 - src: equipo origen del mensaje.
 - numms: identificador de MS.
 - query: indica la petición.
- IACMD: mensaje de cabio de posición. Automensaje del módulo MS.
 - dest: equipo origen del mensaje.
 - numms: identificador de MS.
 - query: indica la petición.
- IMM_ASS_REJ: mensaje de cabio de posición. Automensaje del módulo MS.
 - src: equipo origen del mensaje.
 - dest: identificador de equipo destino del mensaje.
 - query: indica la petición.
- LUACC: mensaje de cabio de posición. Automensaje del módulo MS. numms, numbts
 - numms: número de MS a la que va destinado el mensaje.
 - numbts: identificador de BTS en la que se encuentra la MS.

- MEASREP: mensaje de cabio de posición. Automensaje del módulo MS.
 - src: indica el equipo fuente.
 - numms: indica el equipo fuente del mensaje.
 - query: indica el tipo de petición.
 - nobtsbase1: número identificador de BTS.
 - nobtsbase2: número identificador de BTS.
 - nobtsbase3: número identificador de BTS.
 - nobtsbase4: número identificador de BTS.
 - nobtsbase5: número identificador de BTS.
 - nobtsbase6: número identificador de BTS.
 - wattbase1: potencia recibida por la MS. Emitida por la BTS identificada en nobtsbase1.
 - wattbase2: potencia recibida por la MS. Emitida por la BTS identificada en nobtsbase2.
 - wattbase3: potencia recibida por la MS. Emitida por la BTS identificada en nobtsbase3.
 - wattbase4: potencia recibida por la MS. Emitida por la BTS identificada en nobtsbase4.
 - wattbase5: potencia recibida por la MS. Emitida por la BTS identificada en nobtsbase5.
 - wattbase6: potencia recibida por la MS. Emitida por la BTS identificada en nobtsbase6.
- PAGCMD: mensaje de cabio de posición. Automensaje del módulo MS. numms, numbts
 - numms: número de MS a la que va destinado el mensaje.

- numbts: identificador de BTS en la que se encuentra la MS.
- RFCHREL: mensaje de cabio de posición. Automensaje del módulo MS.
 - dest: equipo origen del mensaje.
 - numms: identificador de MS.
 - chtype: tipo de canal que se libera, tráfico o señalización.
 - handover: indica si se libera debido a un traspaso con un valor igual a 1.
 - query: indica la petición.
- RFCHRELACK: mensaje de cabio de posición. Automensaje del módulo MS.
 - src: equipo origen del mensaje.
 - numms: identificador de equipo MS al que se refiere el mensaje.
 - chtype: tipo de canal.
 - handover: indica si se debe a un traspaso.
 - query: indica la petición.
- RELIND: mensaje de cabio de posición. Automensaje del módulo MS.
 - src: equipo origen del mensaje.
 - numms: identificador de equipo MS.
 - query: indica la petición.

Mensajes interfaz Abis				
Nombre Mensaje	Parámetros	Dirección	Función	Longitud (bytes)
ASSCMD_BSC_BTS	dest, numms, query	BSC→BTS	Asignación de	85

			canal	
ASSGNCMP_BTS_BSC	src, numms, query	BTS→BSC	Asignación de canal completa	3
CHANREL	dest, numms, netrelease, query	BSC→BTS	Liberación de canal	6
CHANRQD	src, watt, numms, query, handover	BTS→BSC	Solicitud de canal	1
CHANACTIV	dest, numms, src, query, handover, chtype	BSC→BTS	Activación de canal	45
CHANACTIVACK	numms, src, query, handover	BTS→BSC	Aceptación de activación de canal	7
CHANACTIVNACK	numms, src, query, handover	BTS→BSC	No aceptación de activación de canal	7
ESTIN	src, numms, query	BTS→BSC	Indicador de establecimiento	29
HODET	src, numms, query	BTS→BSC	Detección de handover	6
IACMD	dest, numms, query	BSC→BTS	Comando de asignación.	29
IMM_ASS_REJ	src, dest, query	BTS→BSC	Rechazo de asignación.	29
LUACC	numms, numbts	BSC→BTS	Aceptación de actualización de posición	25

MEASREP	src, núms., query, wattbase1, nobtsbase1,wattbase2, nobtsbase2,wattbase3, nobtsbase3,wattbase4, nobtsbase4,wattbase5, nobtsbase5,wattbase6, nobtsbase6,	BTS→BSC	Measures Report. Informes de medidas de potencia recibidas en la MS.	18
PAGCMD	numms, numbts	BSC→BTS	Mensaje de búsqueda PAGING	18
RFCHREL	dest, numms, query, chtype, handover	BSC→BTS	Liberación de canal.	4
RFCHRELACK	src, numms, query, chtype, handover	BTS→BSC	Aceptación de liberación de canal	4
RELIND	src, numms, query	BTS→BSC	Indicador de liberación	6

Tabla 26 Mensajes interfaz Abis

8.3.4 Mensajes interfaz A

El interfaz A es el interfaz entre los equipos BSC y MSC. En la Tabla 26 se tiene una breve descripción de estos mensajes y de sus parámetros.

- ASSGNCMP_BSC_MSC: mensaje que indica que se ha realizado la asignación completa de un canal.
- ASSGNREQ: mensaje de petición de asignación de canal.
 - dest: identificador de equipo destino del mensaje.

- numms: identificador de MS.
- src: origen del mensaje.
- bts_dest: identificador de BTS del móvil destino de la llamada dentro de la BSC.
- calltype: tipo de llamada. INCOMING (entrante) u OUTGOING (saliente).
- query: indica la petición.
- CL3I: mensaje de información de nivel 3.
 - src: equipo origen del mensaje.
 - dest: identificador de equipo destino del mensaje.
 - numms: identificador de MS.
 - bts_dest: identificador de BTS destino.
 - query: indica la petición.
- CLEAR_CMD: liberación de los recursos.
 - dest: identificador de equipo destino del mensaje.
 - numbts: identificador de BTS.
 - numms: identificador de MS.
 - netrelease: indica si la liberación se debe al móvil o a la red.
 - query: indica la petición.
- CLEAR_CMP: liberación completa de los recursos.
 - dest: identificador de equipo destino del mensaje.
 - query: indica la petición.
 - numbts: número de BTS.

- numms: número identificador de MS.
- CLEAR_REQ: petición de liberación de los recursos.
 - numms: equipo origen del mensaje.
 - numbts: identificador de equipo destino del mensaje.
 - Handover: indica si la liberación se debe a un traspaso.
 - query: indica la petición.
- HANDOVER_PERFORMED: mensaje de cabio de posición. Automensaje del módulo MS.
 - numms: equipo origen del mensaje.
 - numbts: identificador deBTS.
 - numbsc: identificador de BSC..
- LUACC: mensaje de cabio de posición. Automensaje del módulo MS.
 - numms: equipo MS.
 - numbts: equipo BTS..
- PAGING: mensaje de cabio de posición. Automensaje del módulo MS. numms, nummsorig, numbts
 - numms: equipo MS.
 - numbts: identificador de equipo BTS.
 - nummsorig: indica el identificador de MS que ha originado la llamada..

Mensajes interfaz A				
Nombre Mensaje	Parámetros	Dirección	Función	Longitud (bytes)

ASSGNCMP_BSC_MSC	- - -	BSC→MSC	Asignación completa.	3
ASSGNREQ	dest, numms, query, src, bts_dest, calltype	MSC→BSC	Solicitud de asignación	25
CL3I	dest, numms, query, src, bts_dest	BSC→MSC	Información de nivel 3	15
CLEAR_CMD	dest, numms, query, numbts, netrelease	MSC→BSC	Liberación de recursos	9
CLEAR_CMP	dest, numms, query, numbts	BSC→MSC	Liberación de recursos completa	1
CLEAR_REQ	numms, numbts, handover, query	BSC→MSC	Liberación requerida	5
HANDOVER_PERFORMED	numms, numbts, numbsc	BSC→MSC	Handover intra BSC realizado con éxito	-
LUACC	numms, numbts	MSC→BSC	Aceptación de actualización de posición	25
PAGING	numms, numorig, numbts	MSC→BSC	Mensaje de búsqueda de MS	25

Tabla 27 Interfaz A

8.3.5 Mensajes interfaz B

El interfaz B es el interfaz entre MSC y VLR. Este interfaz a nivel 3 utiliza el protocolo MAP. Estos son los mensajes que se han utilizado. En la Tabla 28 se presenta toda la información concerniente a nombre del mensaje, parámetros y origen y dirección del mensaje.

- MAP_COMPLETE_CALL: mensaje para completar la llamada.
 - src: número de equipo origen del mensaje
 - dest: identificador de equipo destino
 - numbsc: identificador de BSC
 - numms: identificador de MS
 - bts_dest: identificador de BTS.
 - calltype: tipo de llamada, entrante o saleinte
 - query: indica la petición de conexión de un móvil. Puede ser ninguna (NONE), de un canal para respuesta a un mensaje de búsqueda (PAGRES), de respuesta a actualización de posición (LUREQ), de petición de llamada (CMSREQ), de petición de envío de SMS (SMSREQ), o de traspaso (HANDOVER).
- MAP_PAGE: indica la petición de un aviso de búsqueda.
 - src: equipo origen del mensaje.
 - dest: identificador de equipo destino del mensaje.
 - nummsc: identificador de equipo MSC.
 - numbsc: identificador de BSC.
 - numbts: identificador de BTS.
 - numms: identificador de MS.
 - nummsorig: identificador de MS origen de la llamada.

- numbtsorig: identificador de BTS donde se encuentra el origen de la llamada.
- lai: área de localización del equipo MS.
- calltype: tipo de llamada, entrante o saliente.
- query: indica la petición.
- MAP_PROCESS_ACCESS_REQUEST: mensaje inicio del procedimiento de acceso.
 - src: equipo origen del mensaje.
 - dest: identificador de equipo destino del mensaje.
 - numbsc: identificador de BSC.
 - numms: identificador de MS.
 - bts_dest: identificador de BTS.
 - calltype: tipo de llamada, entrante o saliente.
 - query: indica la petición.
- MAP_PROCESS_ACCESS_REQUEST_ACK: acepta el procedimiento de acceso.
 - src: equipo origen del mensaje.
 - dest: identificador de equipo destino del mensaje.
 - numbsc: identificador de BSC.
 - numms: identificador de MS.
 - bts_dest: identificador de BTS.
 - calltype: tipo de llamada, entrante o saliente.
 - query: indica la petición.
- MAP_SEND_INFO_FOR_INCOMING_CALL: mensaje para intercambio de información para una llamada entrante.

- src: equipo origen del mensaje.
 - dest: identificador de equipo destino del mensaje.
 - numbsc: identificador de BSC.
 - numms: identificador de MS.
 - nummsdest: identificador de MS destino.
 - query: indica la petición.
 - bts_dest: identificador de BTS.
 - calltype: tipo de llamada, entrante o saliente.
- MAP_SEND_INFO_FOR_INCOMING_CALL_ACK: mensaje para aceptación del intercambio de información para una llamada entrante.
 - src: equipo origen del mensaje.
 - dest: identificador de equipo destino del mensaje.
 - numbsc: indica la petición.
 - numms: identificador de MS.
 - nummsdest: identificador de MS destino.
 - query: indica la petición.
 - bts_dest: identificador de BTS.
 - calltype: tipo de llamada, entrante o saliente.
 - MAP_SEND_INFO_FOR_OUTGOING_CALL: mensaje para intercambio de información para una llamada saliente.
 - src: equipo origen del mensaje.
 - dest: identificador de equipo destino del mensaje.
 - numbsc: indica la petición.

- numms: identificador de MS.
- nummsdest: identificador de MS destino.
- query: indica la petición.
- bts_dest: identificador de BTS.
- calltype: tipo de llamada, entrante o saliente.
- MAP_SEND_INFO_FOR_OUTGOING_CALL_ACK: mensaje para aceptación del intercambio de información para una llamada saliente.
 - calltype: indica tipo de llamada saliente o entrante.
- MAP_UPDATE_LOCATION_AREA: mensaje de actualización de localización de un móvil.
 - numms: equipo origen del mensaje.
 - numbts: identificador de la BTS bajo la cual se encuentra la MS.
 - query: indica la petición.
- MAP_UPDATE_LOCATION_AREA_ACK: mensaje de actualización de localización de un móvil aceptada.
 - numms: equipo origen del mensaje.
 - numbts: identificador de la BTS bajo la cual se encuentra la MS.
 - query: indica la petición.

Mensajes interfaz B			
Nombre Mensaje	Parámetros	Dirección	Función
MAP_COMPLETE_CALL	dest, numbsc, numms, query, src, bts_dest, calltype	VLR→MSC	Finaliza el procedimiento de establecimiento de llamada.

MAP_PAGE	calltype, nummsc, numbsc, numbts, numms, nummsorig, numbtsorig, lai	VLR→MSC	Solicita la búsqueda de una MS
MAP_PROCESS_ACCESS_REQUEST	dest, numbsc, numms, query, src, bts_dest, calltype	MSC→VLR	Solicita el procedimiento de acceso a la red por parte de una MS
MAP_PROCESS_ACCESS_REQUEST_ACK	dest, numbsc, numms, query, src, bts_dest, calltype	VLR→MSC	Acepta el procedimiento de acceso solicitado
MAP_SEND_INFO_FOR_INCOMING_CALL	dest, numbsc, numms, nummsdest, query, src, bts_dest, calltype	MSC→VLR	Envía información para una llamada entrante
MAP_SEND_INFO_FOR_INCOMING_CALL_ACK	dest, numbsc, numms, nummsdest, query, src, bts_dest, calltype	VLR→MSC	Acepta la información para una llamada entrante
MAP_SEND_INFO_FOR_OUTGOING_CALL	dest, numbsc, numms, query, src, bts_dest, calltype	MSC→VLR	Envía información sobre llamada saliente
MAP_SEND_INFO_FOR_OUTGOING_CALL_ACK	calltype	VLR→MSC	Acepta la información sobre llamada saliente

MAP_UPDATE_LOCATION_AREA	query, numms, numbts	MSC→VLR	Realiza la actualización de posición de una MS
MAP_UPDATE_LOCATION_AREA_ACK	query, numbts, numms	VLR→MSC	Acepta la actualización de posición de la MS

Tabla 28 Interfaz B

8.4 Aplicación de apoyo: SimOffLine.exe

8.4.1 Objetivo de la aplicación.

La aplicación de apoyo pretende ayudar a la visualización de las trayectorias simuladas en la aplicación principal. Para ello lee unos ficheros con extensión “gsmsim” que genera la aplicación principal. Este programa da a lugar una pantalla en la que se representan las zonas de cobertura de las estaciones y las trayectorias de cada equipo móvil.

8.4.2 Programación

El programa se ha desarrollado con la misma estructura que cualquier otra aplicación OPENGL. Para conocer mejor que estructura siguen este tipo de programas, y las funciones que es necesario implemetar, leer el apartado 7.3 en el que se explica como desarrollar una aplicación con OPENGL.

8.4.2.1 init()

Función que borra los colores de la ventana, y que ajusta los parámetros de la visualización.

8.4.2.2 read_file()

Lee el fichero btsPosition.gsmsim que contiene las posiciones de las BTS, los radios de cobertura, los límites de la zona de simulación y el número máximo de equipos móviles.

8.4.2.3 read_MS-file()

Lee el fichero msPosition?.gsmsim que contiene las posiciones de las MS a lo largo de la simulación. Se guarda el tiempo y las coordenadas del eje X y eje Y en las que se encuentra la estación móvil en el instante de tiempo mencionado.

8.4.2.4 display()

Función que dibuja en la ventana. Primero dibuja las líneas de coordenadas. Se dibuja una cada 1000 metros de zona de simulación. Si la zona simulada es de 10Km x 10Km entonces se dibuja una malla de 11x11 líneas que cubre toda la zona.

A continuación se dibuja el cuadrado de fondo de la ventana y los círculos de cobertura de una estación base. Cada vez que se dibuja una zona de cobertura nueva, se cambia el color en función de tres variables. Cada una de estas variables son de tipo flota, y su valor oscila entre 0.0 y 1.0. Los valores de coloración corresponde a RGB, es decir el primer valor será el porcentaje de rojo, el segundo de verde y el tercero de azul.

Por último se dibuja la trayectoria definida por los ficheros de trayectoria de cada MS. El usuario puede elegir desde la pantalla de comandos que trayectorias visualizar y cuales no.

8.4.2.5 reshape()

La función ajusta los valores de la cámara para visualizar las imágenes y ajusta los límites de visualización.

8.4.2.6 keyboard()

Define los eventos de teclado que van a dar resultados en la visualización.

8.4.2.7 mouse()

Define los eventos de teclado que van a dar resultados en la visualización.

8.4.2.8 main()

Llama a todas las anteriores funciones, y crea un bucle del que la aplicación sólo sale cuando se impone una condición de salida.

9 HERRAMIENTAS

Para el desarrollo del proyecto se han empleado varias herramientas, aunque principalmente lo que se necesita son las librerías de OMNET++ y un compilador de C++ compatible con el entorno OMNET++. Como se ha dicho ya en el inicio, las librerías se pueden descargar sin necesidad de pagar desde la página oficial. En cuanto al entorno de desarrollo, el elegido ha sido el Visual Studio 6.0 de Microsoft. La razón de elegir este entorno frente a otros como el .NET es en primer lugar que se ha desarrollado sobre un sistema operativo Windows XP instalado en un ordenador personal, con lo cual se debía utilizar un compilador para Windows, y en segundo lugar que es una herramienta muy potente para la creación de proyectos en C++. En un principio se planteó la posibilidad de usar una plataforma posterior a esta y también distribuida por Microsoft, la plataforma .NET. Sin embargo finalmente se adoptó la Visual Studio 6.0 por dos razones, la primera es que es una plataforma sobre la que se ha trabajado en un mayor número de proyectos con OMNET++ y por tanto con una mejor depuración de la integración entre Visual y OMNET y la segunda razón porque es una herramienta que emplea el departamento además de encontrarse ampliamente distribuida. En principio el paso de un entorno a otro no debería causar ningún problema pero no se ha probado con esta aplicación. En el sitio de la Comunidad OMNET++ se mencionan ciertos

problemas con determinadas versiones de las librerías de .NET y consejos para solucionarlos. La ventaja del entorno Visual Studio frente a otros es que en éste, se puede integrar perfectamente el desarrollo del código con un proyecto de tipo OMNET. Los ficheros .ini, y .ned son añadidos al proyecto e incluidos en la compilación.

Para una mejora del proyecto, y como una aplicación secundaria de apoyo a la principal se ha desarrollado un pequeño programa con la librería GLUT para OPENGL. Estas librerías se pueden descargar desde Internet. Son librerías que permiten la realización de gráficos.

En cuanto al desarrollo de las memorias y documentos, se ha empleado el programa Microsoft Word para la redacción. Los programas PDFCreator, Adobe Acrobat Reader 7.0 Professional para pasar los ficheros a formato pdf. En cuanto a planificación y organización se ha empleado el programa Microsoft Project, y el programa Microsoft Excel. Para los dibujos y diapositivas se han usado diferentes programas, Microsoft Visio 2003, Microsoft PowerPoint, y Photoshop.

Por último hay que mencionar la utilización de un analizador de protocolos para una red GSM como apoyo durante el inicio del proyecto. El programa se llama K1205 Protocol Tester de la empresa Tektronics.

Los ordenadores que se han empleado son ordenadores personales. El que se ha utilizado para toda la parte del desarrollo ha sido un ordenador portátil con un microprocesador Intel Centrino A 1.5GHz y también se ha utilizado para las pruebas y redacción un ordenador con procesador Intel Pentium 4 a 3.0GHz.

Por último, en este capítulo se van a explicar los procesos de instalación de las tres herramientas más importantes para el desarrollo del proyecto. Éstas son Microsoft Visual 6.0, OMNET++ y OPENGL.

9.1 Instalación de Visual Studio 6.0

La instalación de Visual es muy sencilla. Debe realizarse antes que la instalación de OMNET++. Normalmente viene distribuida con un CD que se ejecuta automáticamente. Únicamente hay que seguir los pasos de instalación que se van solicitando. Es como la instalación de otras aplicaciones de Microsoft, muy sencilla e intuitiva.

9.2 Instalación de OMNET++

En primer lugar hay que descargarse el ejecutable desde la página oficial. Cuando se ejecuta sigue los pasos de instalación de cualquier otra aplicación. El instalador incluye el entorno OMNET++ en Visual Studio C++. A partir de aquí se pueden desarrollar proyectos de tipo OMNET++ (). Las librerías se encuentran en C:\OMNeT++\lib en dos carpetas para MSVC llamadas vc6-debug y vc6-release.

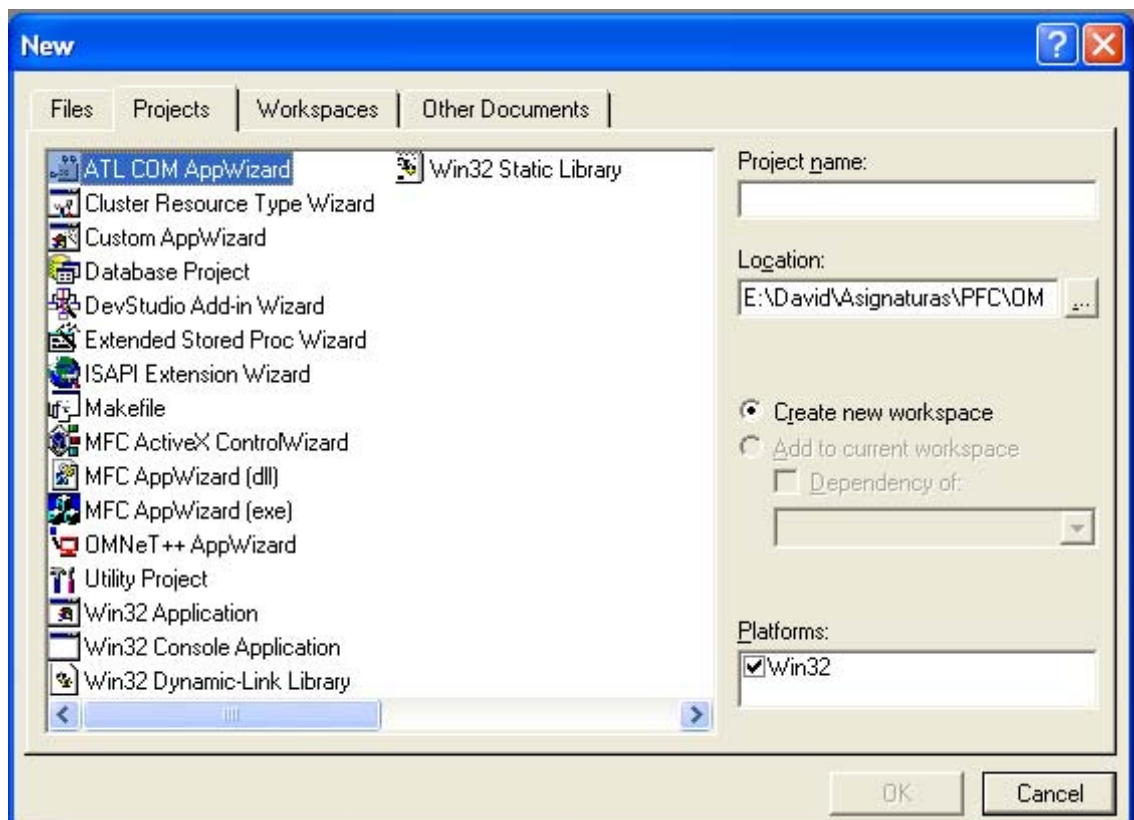


Figura 48 Crear un proyecto OMNET++.

Gracias a la integración OMNET++ - Visual Studio C++, se pueden crear nuevos proyectos OMNET++ AppWizard.

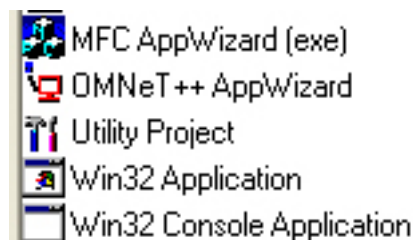


Figura 49 OMNET++ AppWizard.

Un nuevo proyecto OMNET++ se denomina OMNET++ AppWizard.

9.3 Instalación de OPENGL GLUT

Las librerías de GLUT para Windows suelen distribuirse dentro de un fichero comprimido, generalmente de tipo zip. El fichero debe incluir los ficheros glut32.dll, glut32.lib, glut.h, glut.def, y generalmente un fichero que describe la instalación. Para comenzar a trabajar hay que copiar estos ficheros y colocarlos en los directorios que se presentan a continuación:

Instalación de GLUT	
Nombre fichero	Ruta
Glut32.dll	C:\windows\System
Glut32.lib	Directorio VC++ 6.0\..\VC96\lib
Glut.h	Directorio VC++ 6.0\..\VC96\include\GL
Glut.def	Directorio VC++ 6.0\..\VC96\include\GL

Tabla 29 Instalación de Glut.

Directorios de instalación de los ficheros y librerías de GLUT para desarrollar programas con OPENGL.

Para comenzar a trabajar sólo falta añadir a los ficheros del código fuente del programa para OPENGL la cabecera de GLUT que se ha grabado antes. La línea a introducir es `#include <gl/glut.h>`.

Otra posibilidad para las librerías es descargarse el código fuente y compilarlo desde Microsoft Visual C++ 6.0. El código fuente incluye un archivo de espacio de trabajo para Visual que puede ser abierto y ejecutado.

10 RESULTADOS

En este apartado se presentan ejemplos en los que se ha utilizado la aplicación desarrollada. Estos ejemplos se encuentran en el CD que se adjunta con esta memoria.

10.1 Ejemplo 1

En el primer ejemplo se ha simulado una red con 5 BTS y 20 MS. La zona de simulación es de 1000 x 1000 metros. Se han realizado dos simulaciones diferentes, en la primera, a cada estación BTS se le ha asignado un número bajo de canales, de forma que llegue a la saturación y tenga que bloquear llamadas. En la segunda se ha asignado un número de canales suficiente como para que no lleguen a bloquearse. Se ha vuelto a simular esta situación pero con una menor potencia por estación. Los resultados son muy parecidos debido a la cercanía entre las estaciones base, que no da lugar a zonas de sombra. A continuación se pasa a presentar los resultados obtenidos en forma de gráficas.

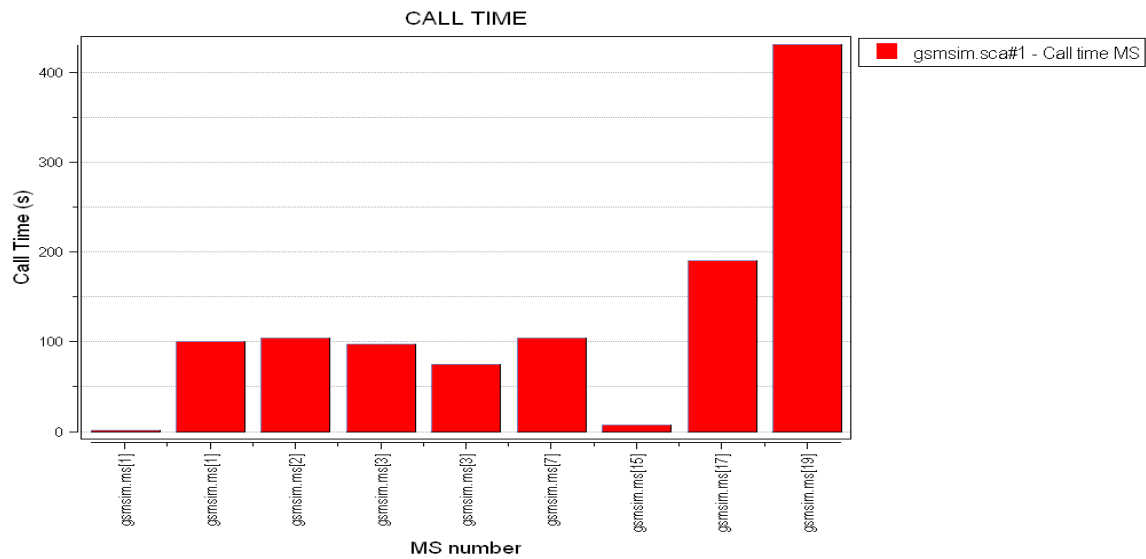


Figura 50 Tiempo de llamada.

Durante la simulación se han establecido nueve llamadas. El módulo 19 es el módulo que produce la llamada más larga, de más de 400 segundos.

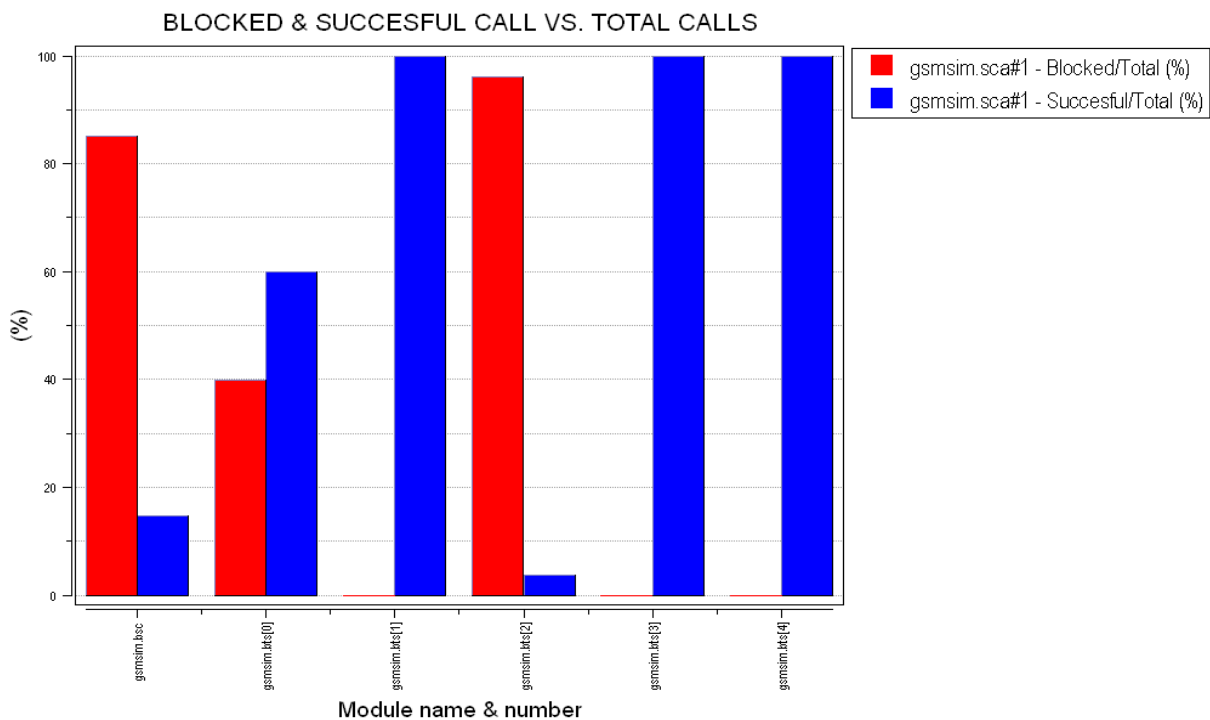


Figura 51 Porcentaje de llamadas.

En esta simulación se han usado cinco BTS. A cada una se le han dado pocos canales de tráfico, por lo que los porcentajes de bloqueo son muy altos, sobre todo en el caso de la BTS número 2 la del medio, la cual llega a un 90% de llamadas bloqueadas.

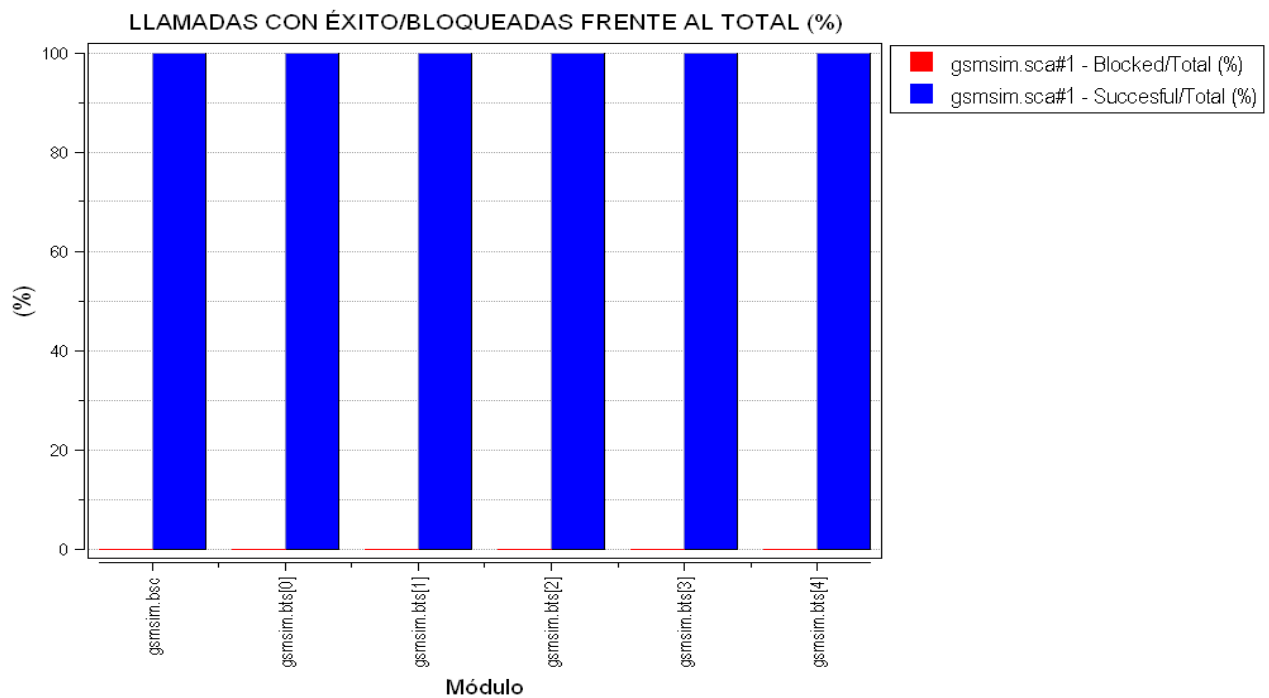


Figura 52 Porcentaje de llamadas.

Esta gráfica corresponde a la misma simulación pero aplicando a cada BTS un número suficiente de canales de tráfico para que no llegue a haber bloqueo en la red. Si se compara con la gráfica anterior, la calidad de servicio que ofrece la red es mucho mejor.

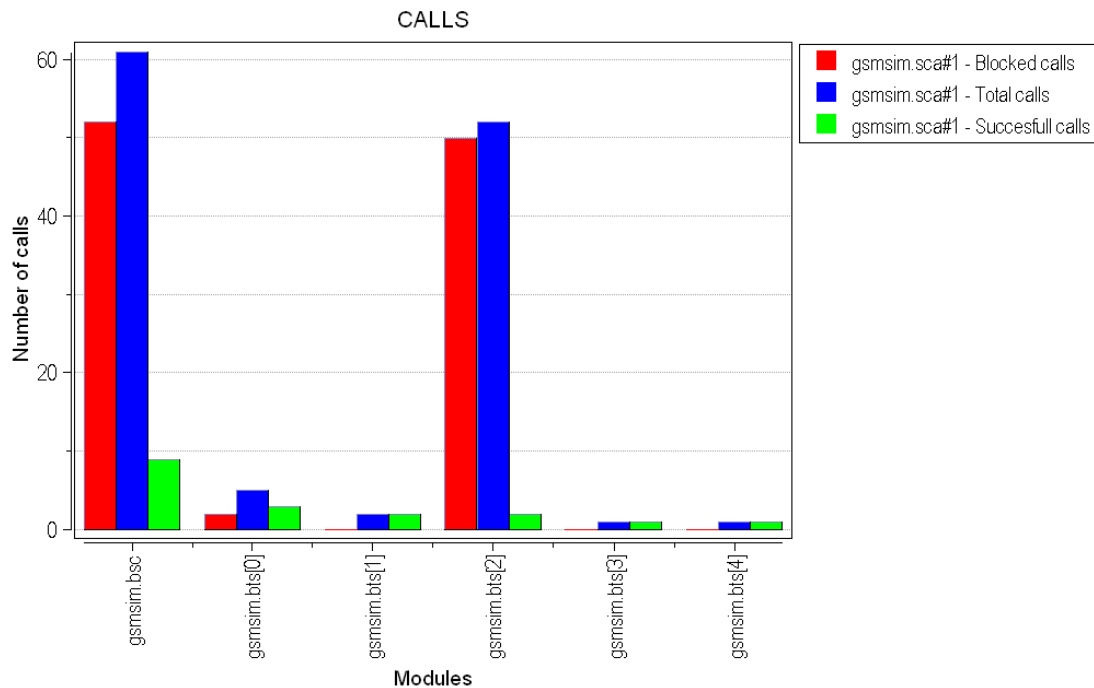


Figura 53 Llamadas con éxito y bloqueadas.

Como se puede ver hay un gran número de llamadas bloqueadas en las estaciones base.

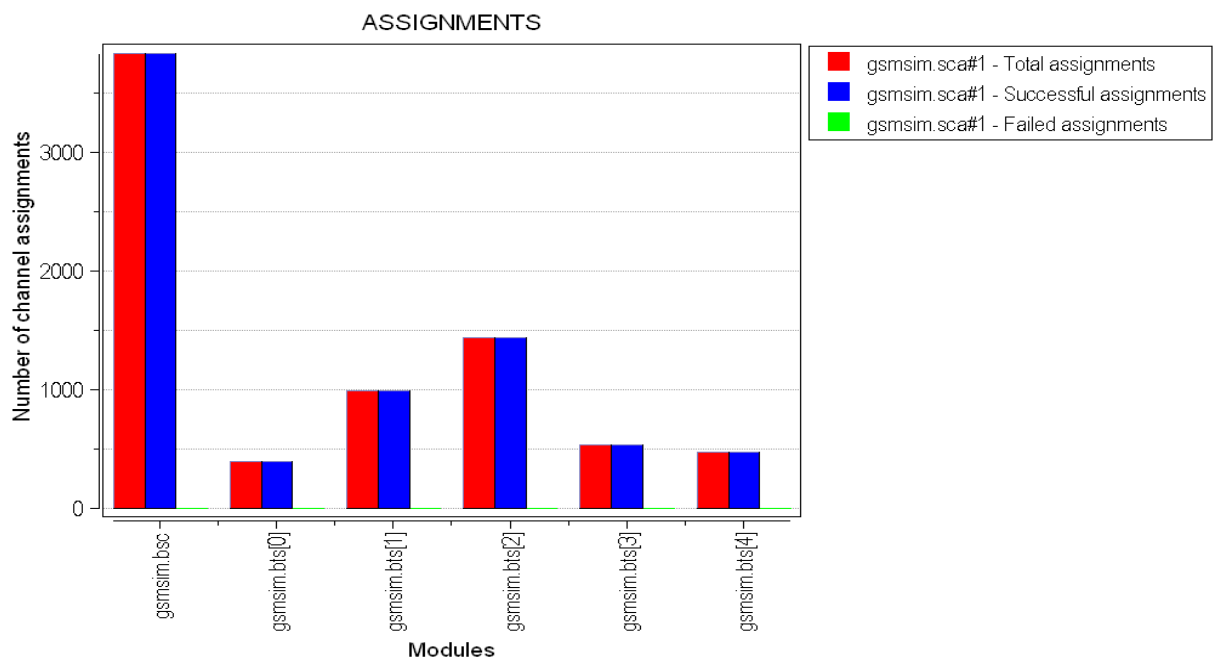


Figura 54 Asignaciones.

La figura representa el número de asignaciones de canal que se han producido por cada estación BTS durante la simulación. La BTS número 2 es la que más tráfico soporta.

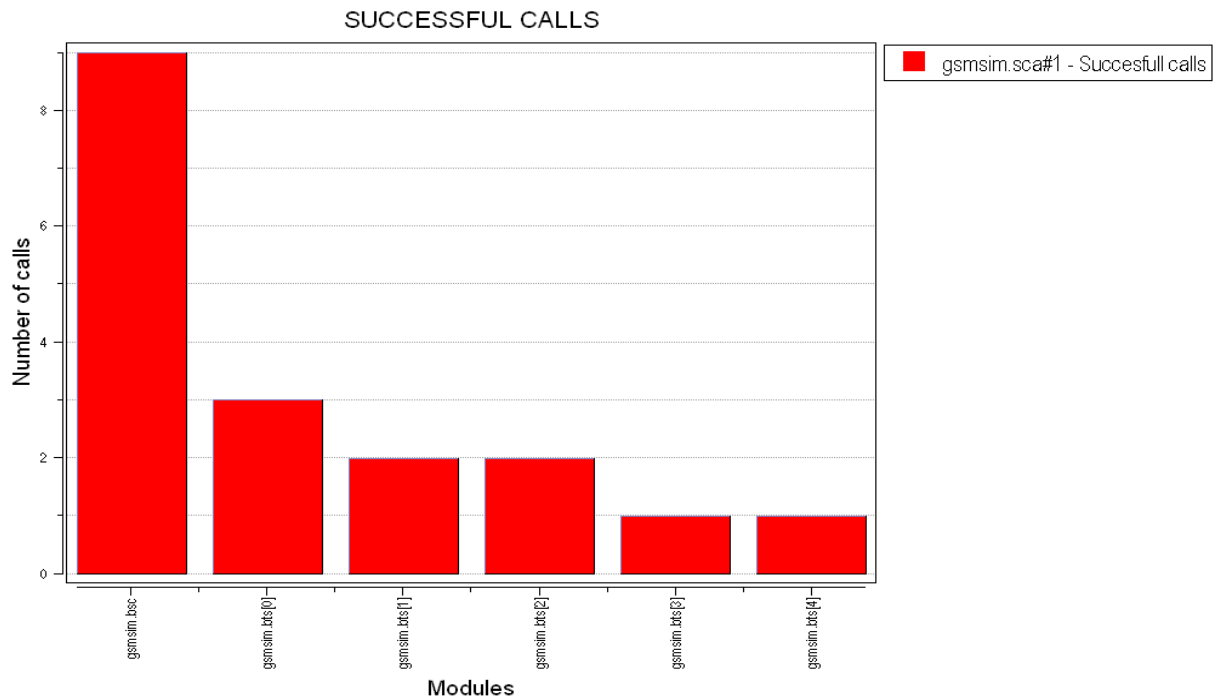


Figura 55 Llamadas con éxito.

En la figura se pueden ver el total de llamadas correctas que se han producido durante la simulación.

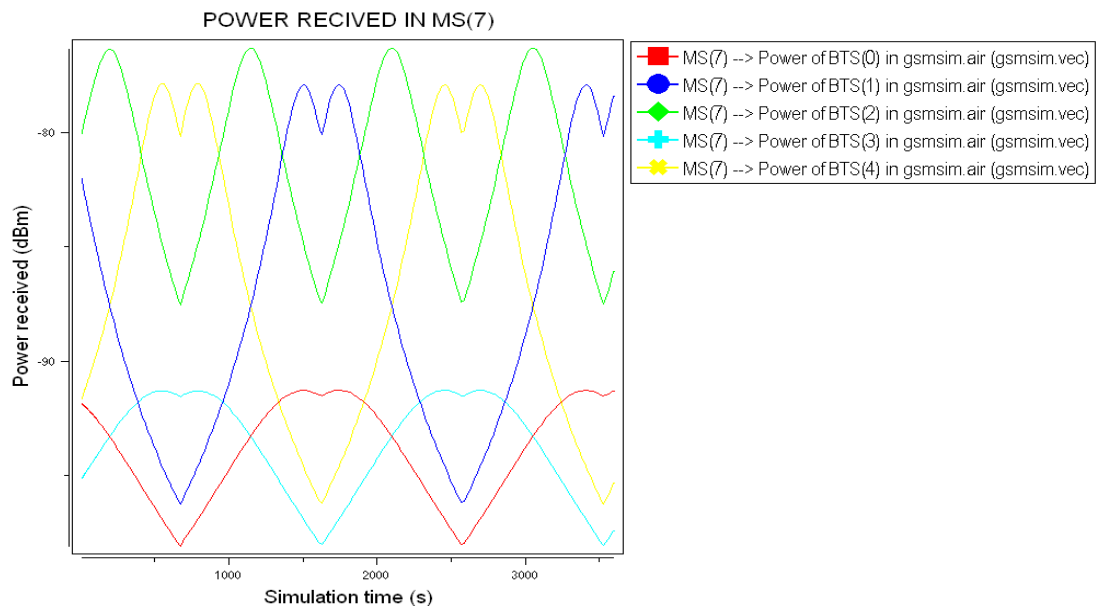


Figura 56 Potencias en MS(7)

La MS 7 se mueve de forma lineal, y además rebota en varias ocasiones con los límites de la simulación, de ahí que la gráfica tenga determinadas zonas simétricas.

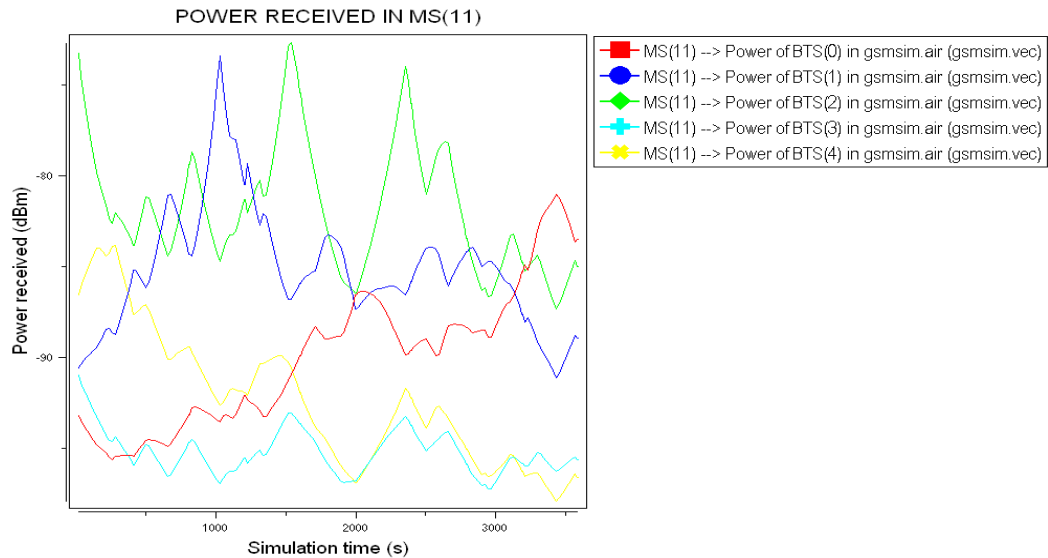


Figura 57 Potencias en MS(11)

La MS lleva un movimiento aleatorio. La estación bajo la que se encuentra va variando con la simulación. Al inicio de conecta con la estación número 2, pasando a continuación a depender de la BTS 1.

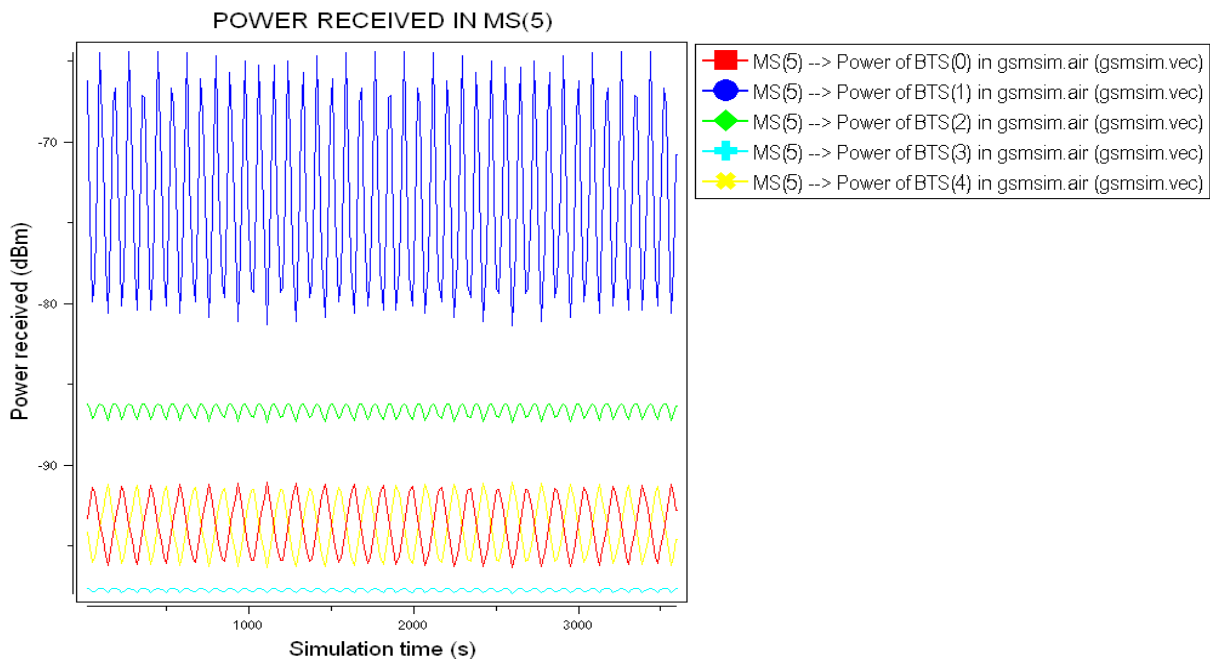


Figura 58 Potencia en MS(5).

La MS se mueve en una zona muy cercana a la BTS 1 por lo que durante toda la simulación se encuentra bajo la cobertura de esta estación. Recibe además las potencias de las otras cuatro estaciones, pero mucho más débiles.

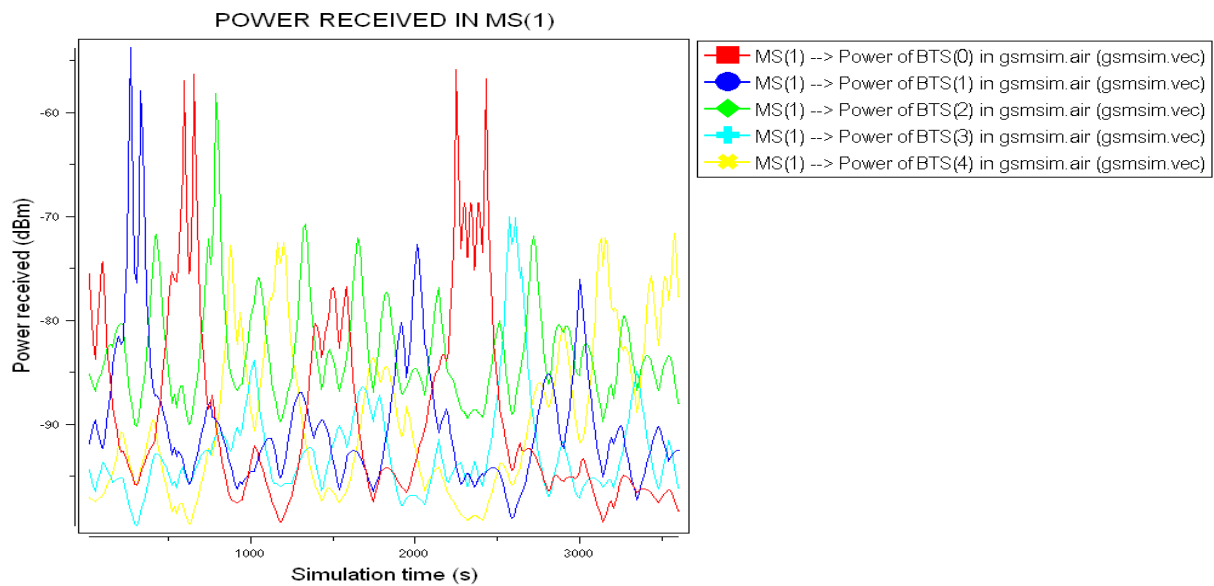


Figura 59 Potencia en MS(1)

La estación lleva un movimiento aleatorio, por lo que las potencias irán variando durante toda la simulación.

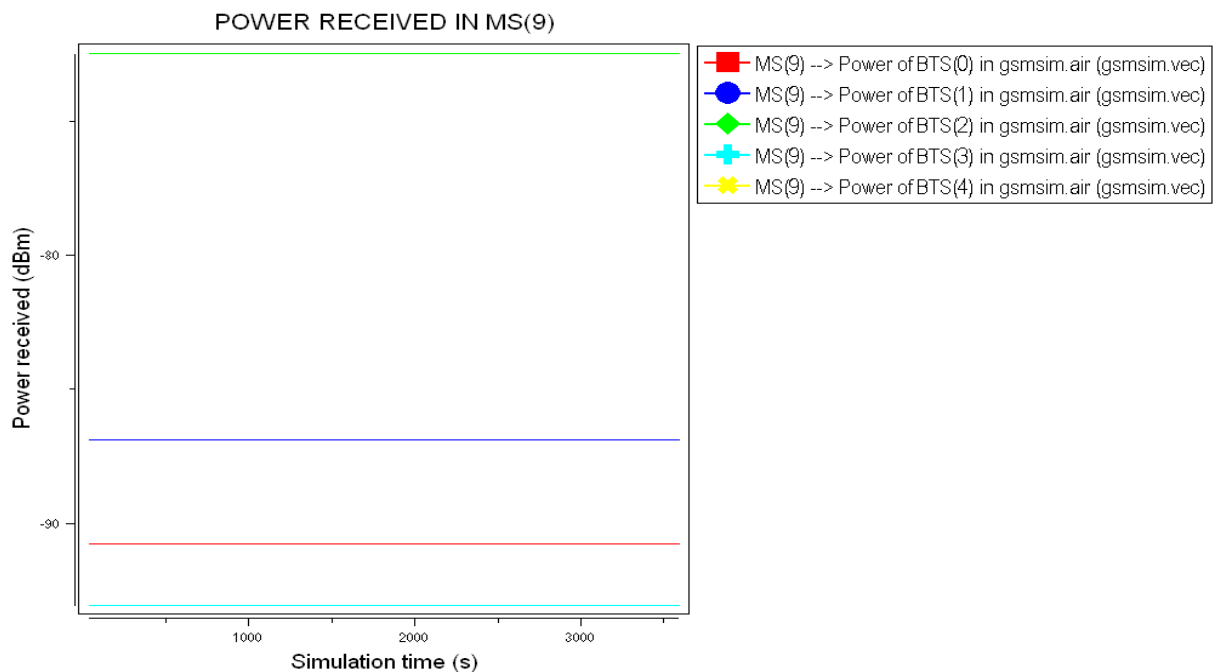


Figura 60 Potencia en MS(9)

La estación móvil se encuentra parada durante toda la simulación, por lo que no varía su valor durante la simulación.

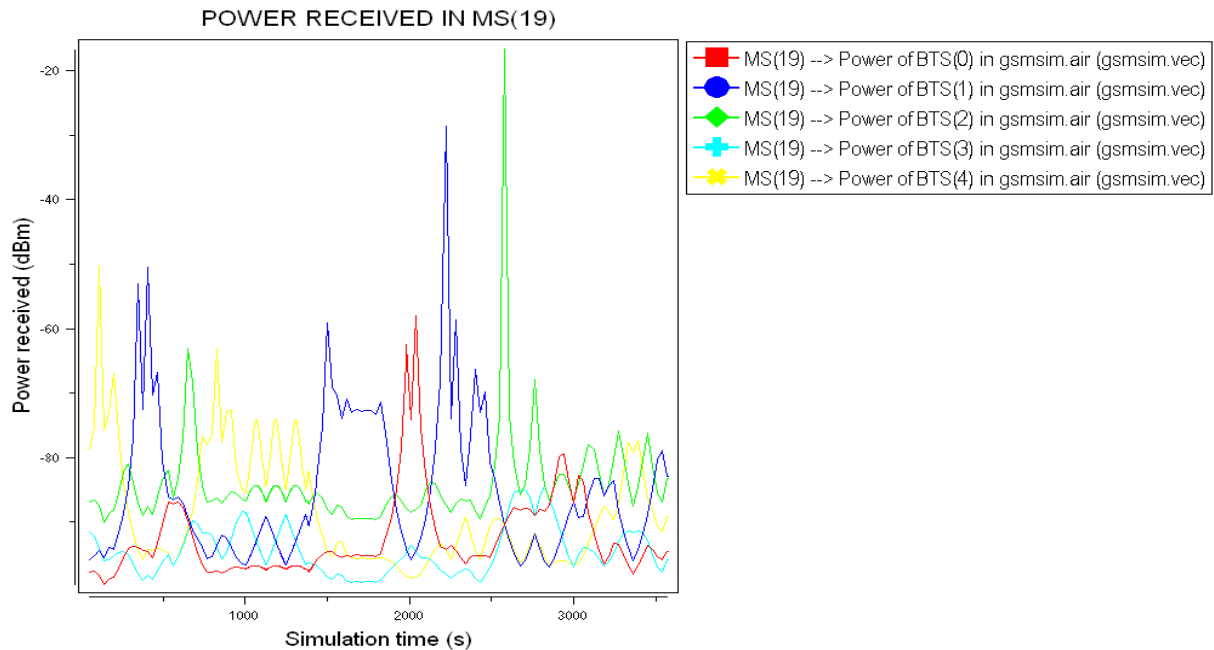


Figura 61 Potencia recibida en MS (19)

La MS también lleva un movimiento aleatorio, encontrándose bajo la cobertura de diferentes estaciones a lo largo de la simulación. Se puede observar que en ningún momento la estación móvil se llega a quedar sin cobertura. Esto quiere decir que la red está bien configurada en la planificación celular, evitando posibles zonas de sombra.

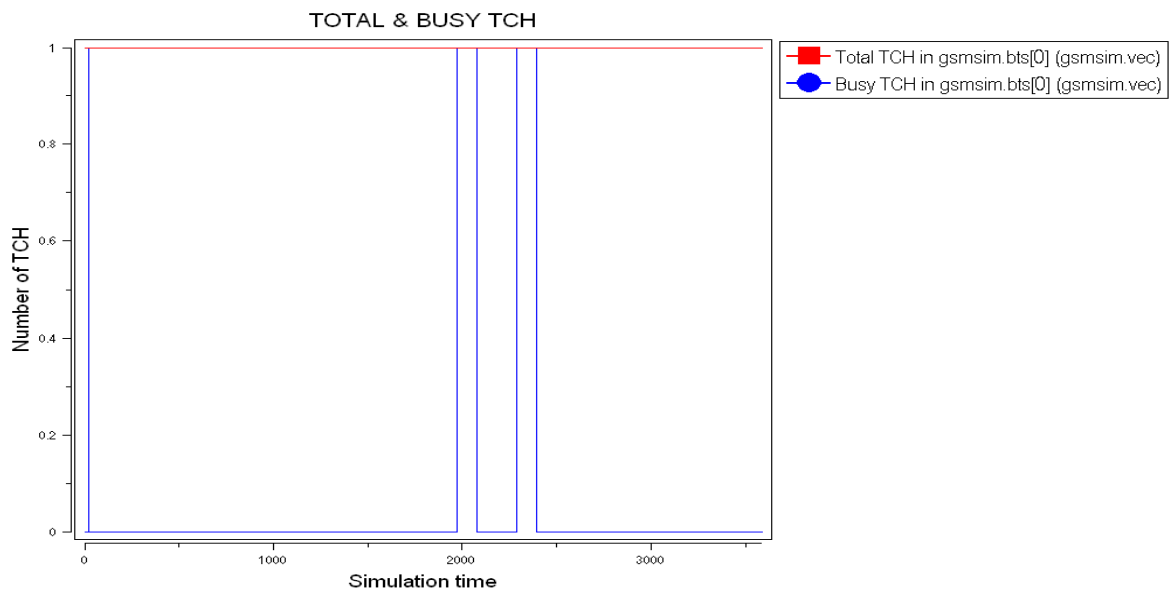


Figura 62 Canales usados en la BTS 0 durante la simulación.

Como se puede ver en la imagen, los canales en esta estación no están ocupados la mayoría del tiempo.

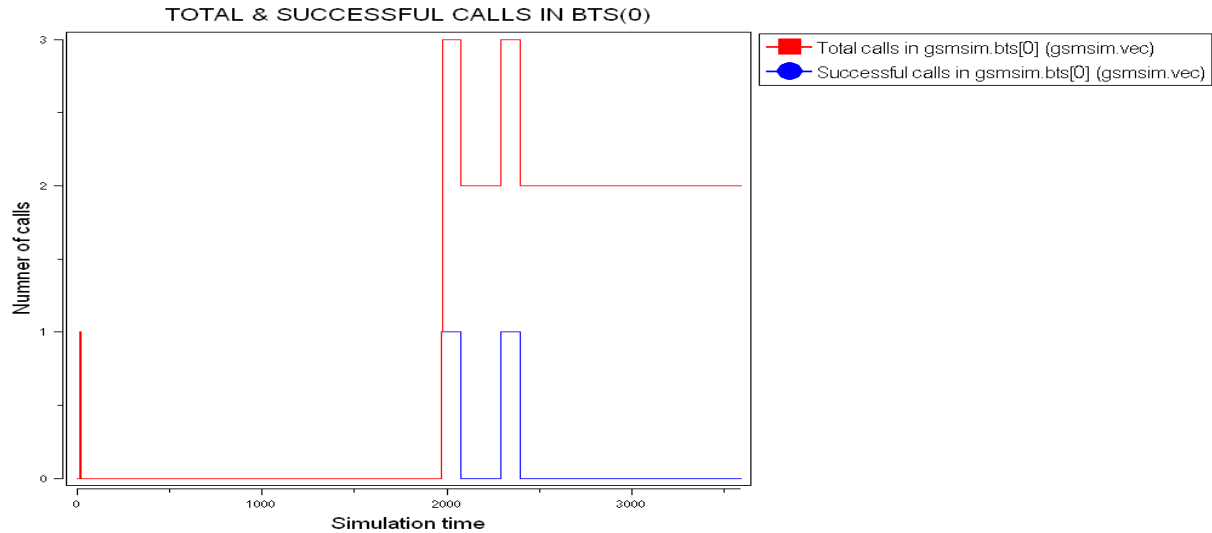


Figura 63 Llamadas totales y con éxito en la BTS(0).

Como se puede ver esta estación empieza a estar ocupada a partir de los 2000 segundos de simulación. Durante la primera mitad de la simulación, bajo la BTS no se producen llamadas.

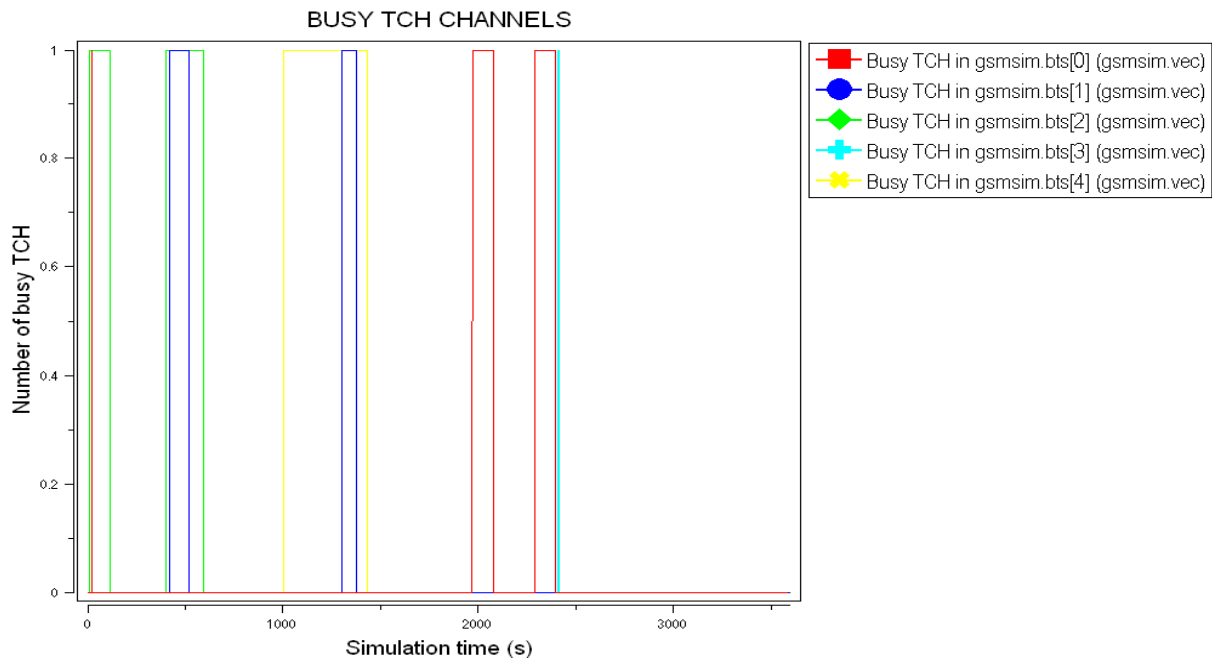


Figura 64 Uso de canales TCH.

En la imagen se puede ver la utilización de los canales de todas las BTS durante la simulación.

10.2 Ejemplo 2

En este segundo ejemplo se ha simulado una red con 100 MS y 20 BTS. La zona de simulación se ha establecido en 10000 metros x 10000 metros, y se han dejado estaciones con un menor radio de cobertura que otras, dejando algunos espacios en los que las estaciones móviles no son capaces de recibir una potencia superior a la umbral. Se han obtenido las siguientes figuras.

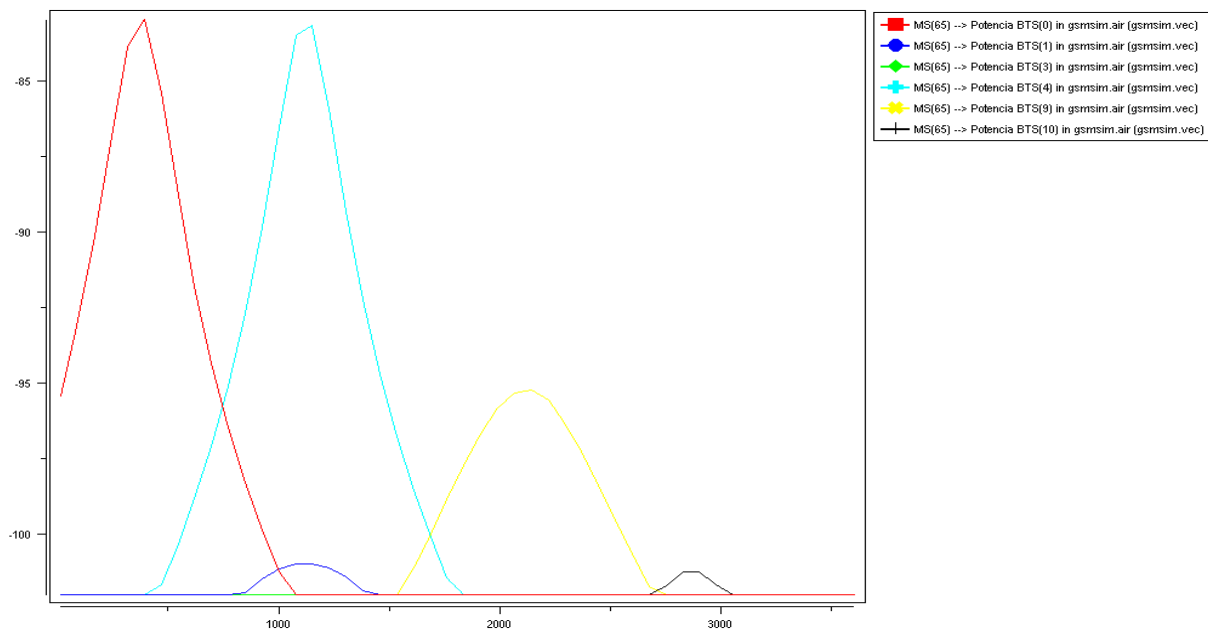


Figura 65 Potencia en MS(65)

La estación recibe potencia hasta el segundo 3000 más o menos. A partir de este punto la estación se queda sin cobertura. En la configuración de la simulación, se han distribuido las estaciones base de forma que se puedan crear zonas en las que no hay cobertura. La MS 65 estaría a partir del instante 3000 en una de estas zonas con ausencia de cobertura.

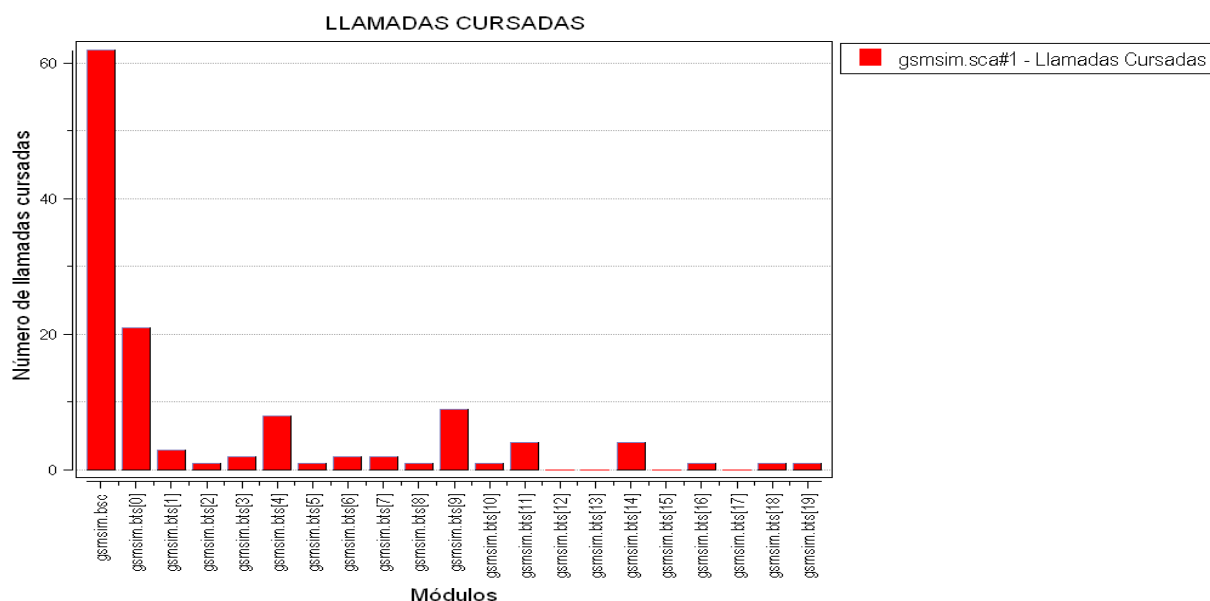


Figura 66 Número de llamadas cursadas.

En la simulación se han utilizado 20 BTS. Como se puede ver hay estaciones bajo las cuales se producen más llamadas y otras en las cuales no se producen llamadas.

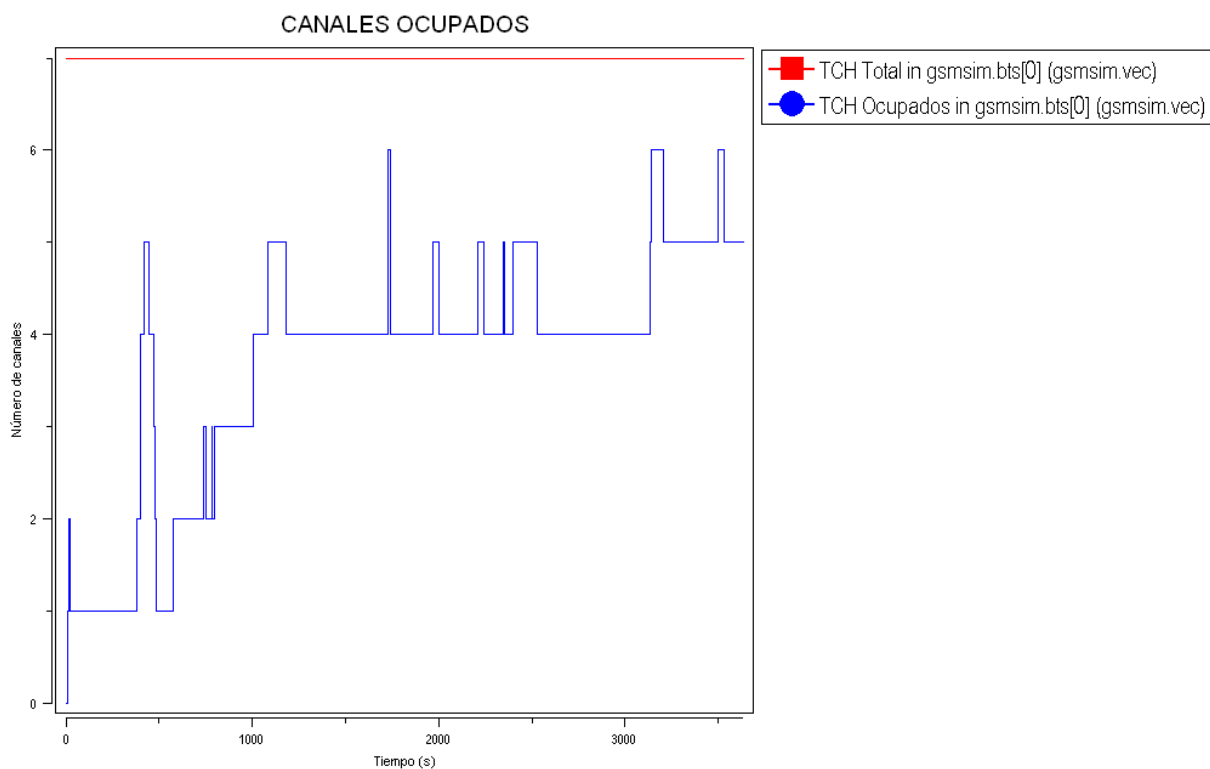


Figura 67 Canales ocupados en la BTS(0).

El número de canales ocupados aumenta según transcurre la simulación. Esto se debe a que un gran número de las estaciones móviles se han colocado bajo esta estación. En concreto se han colocado 18 MS en las cercanías de esta estación, que tiene una capacidad de 7 canales de tráfico. Si aumenta el número de llamadas, pueden llegar a rechazarse o bloquearse debido a que la BTS no sea capaz de soportar el tráfico.

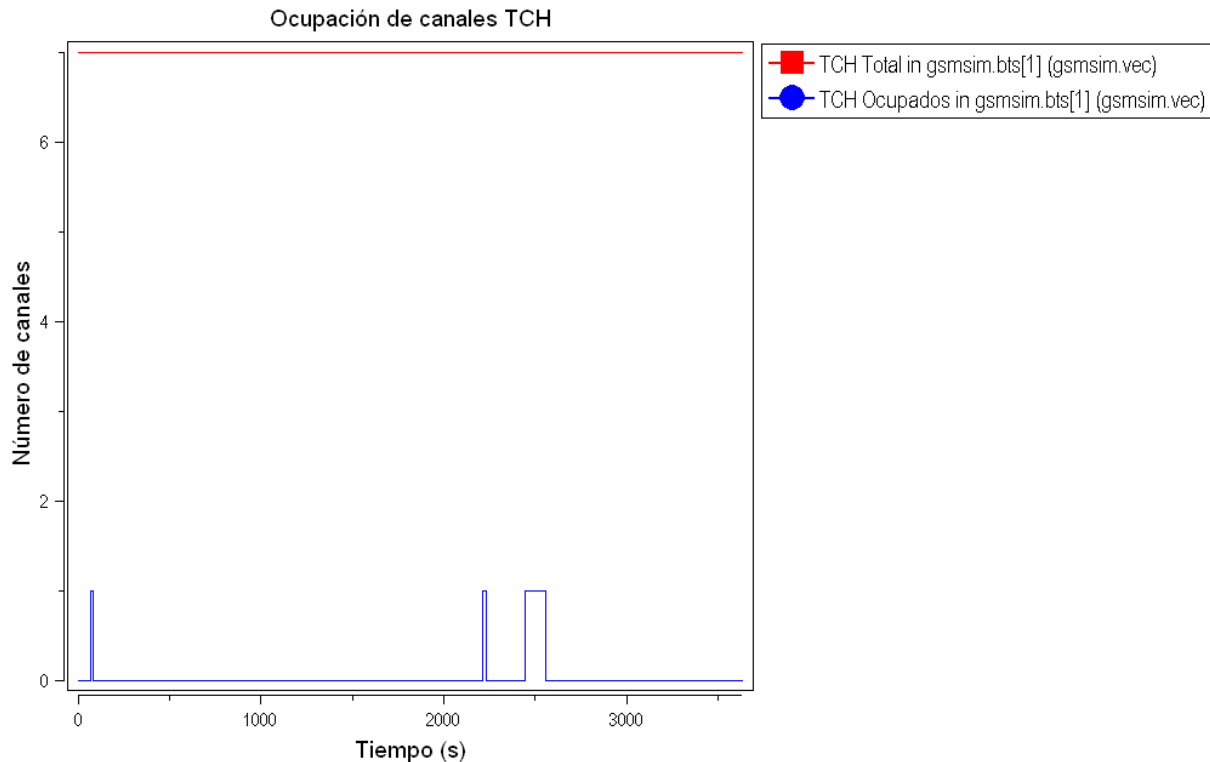


Figura 68 Ocupación de canales en BTS(1).

La BTS tiene menor tráfico. En comparación con la gráfica anterior, esta estación apenas tiene móviles que generen llamadas. No llegarán a bloquearse las llamadas, ya que el porcentaje de uso es muy inferior al de la gráfica anterior, llegando a ser prácticamente nulo.

10.3 Ejemplo 3

En este tercer ejemplo se ha simulado una red con 100 MS y 3 BTS. La zona de simulación se ha reducido con respecto a la del ejemplo número 2. Esta zona es de 1000 metros x 1000 metros. Los móviles se han localizado principalmente en tres zonas, y se les ha asignado una velocidad muy baja, para que no se salgan excesivamente de su zona. Equivaldría a una zona urbana comercial, con un gran número de usuarios en una pequeña zona espacial, y con un

movimiento lento, a la velocidad a la que anda una persona. El movimiento que se le ha aplicado es un movimiento aleatorio en la mayoría de los casos.

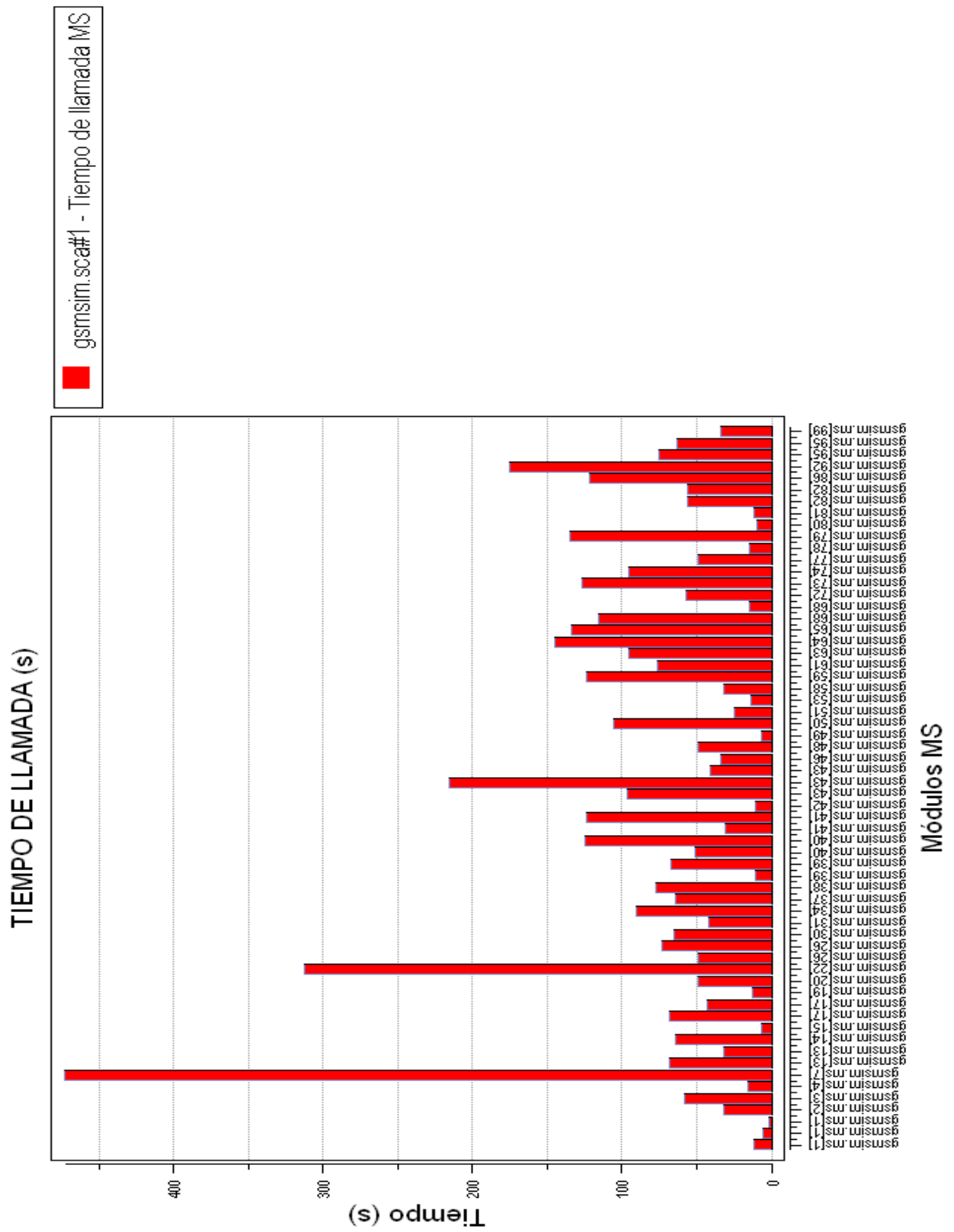


Figura 69 Llamadas en el ejemplo 3

En la imagen se pueden ver la duración de las llamadas y el equipo que las ha producido. La duración de la llamada más larga se acerca a los 500 segundos.

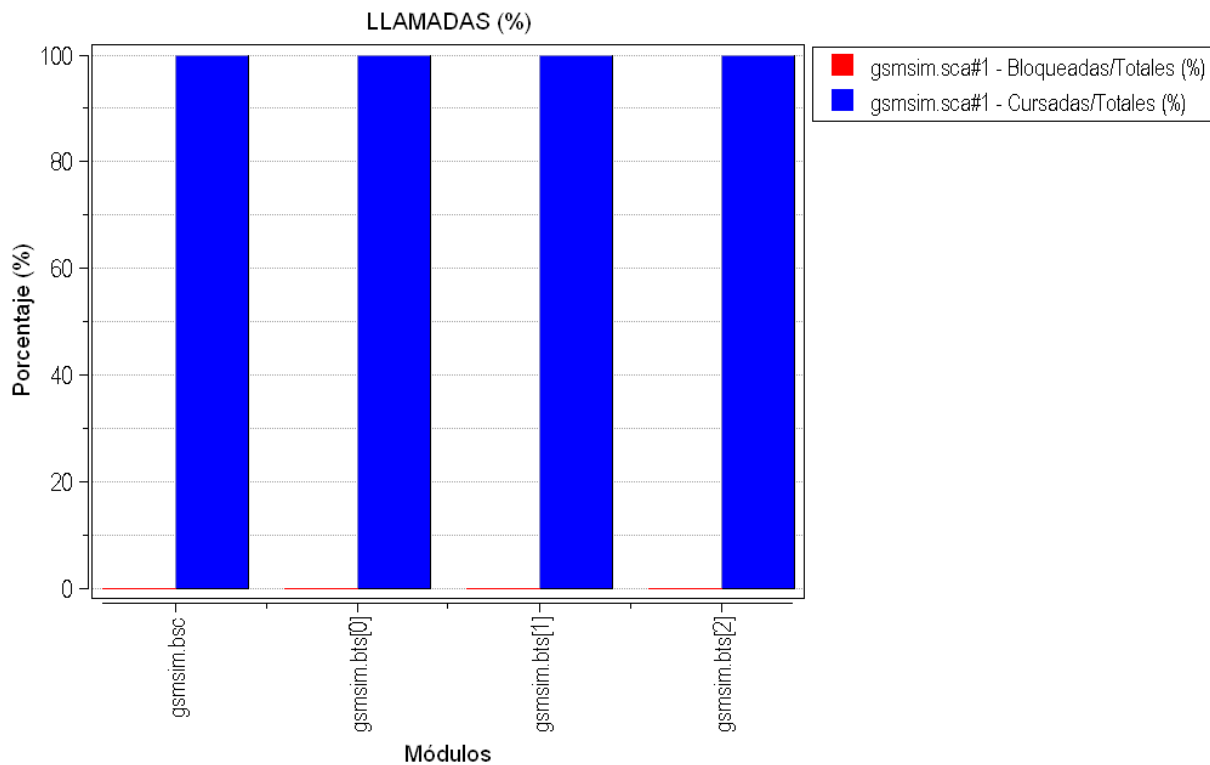


Figura 70 Porcentaje de llamadas con éxito en el ejemplo 3.

En este ejemplo las llamadas cursadas se acerca al 100% mientras que las bloqueadas es del 0%. Esto quiere decir que la planificación celular se ha dimensionado correctamente, y no se pierden llamadas en la simulación debido a saturación en las BTS

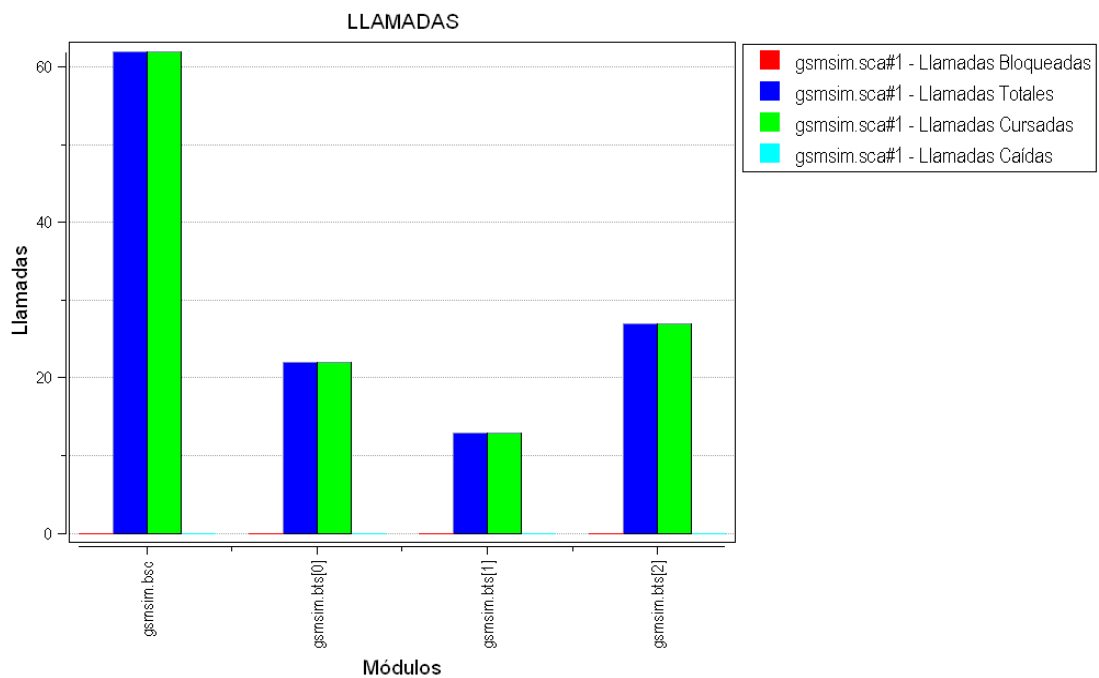


Figura 71 Llamadas en el ejemplo 3.

Durante la simulación, no se producen llamadas caídas ni se producen bloqueos. Esto se debe a que la red está bien dimensionada, y no se presentan sombras de cobertura que hagan perder la comunicación, ni tampoco se producen más llamadas que canales disponibles.

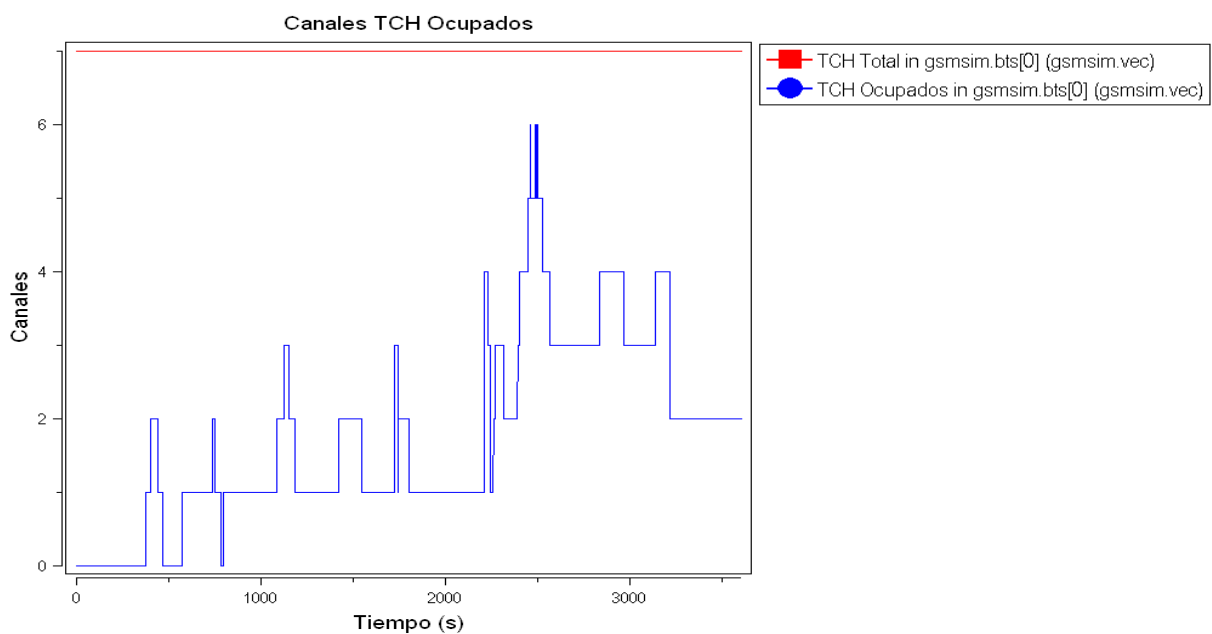


Figura 72 Canales TCH ocupados

Esta estación (BTS 0) tiene una ocupación media de canales, observándose una gran variación conforme transcurre la simulación. Si se compara con la Figura 73, esta estación tiene un menor uso durante la simulación. La figura siguiente corresponde a la estación BTS 1, la cual tiene una ocupación media alta. Esto se debe a que hay un mayor número de equipos bajo su zona de cobertura que en otras estaciones. Si se compara ahora con la Figura 74 que representa el uso de canales en la BTS 2, se ve que se sitúa a medio camino entre la BTS 1 y la BTS 2. La BTS 2, tiene un intervalo de tiempo en el cual no se emplea ningún canal de tráfico, y su media de uso es bastante baja. Esto quiere decir que la densidad de tráfico en esta celda es bastante menor que la densidad que se presenta en las otras BTS.

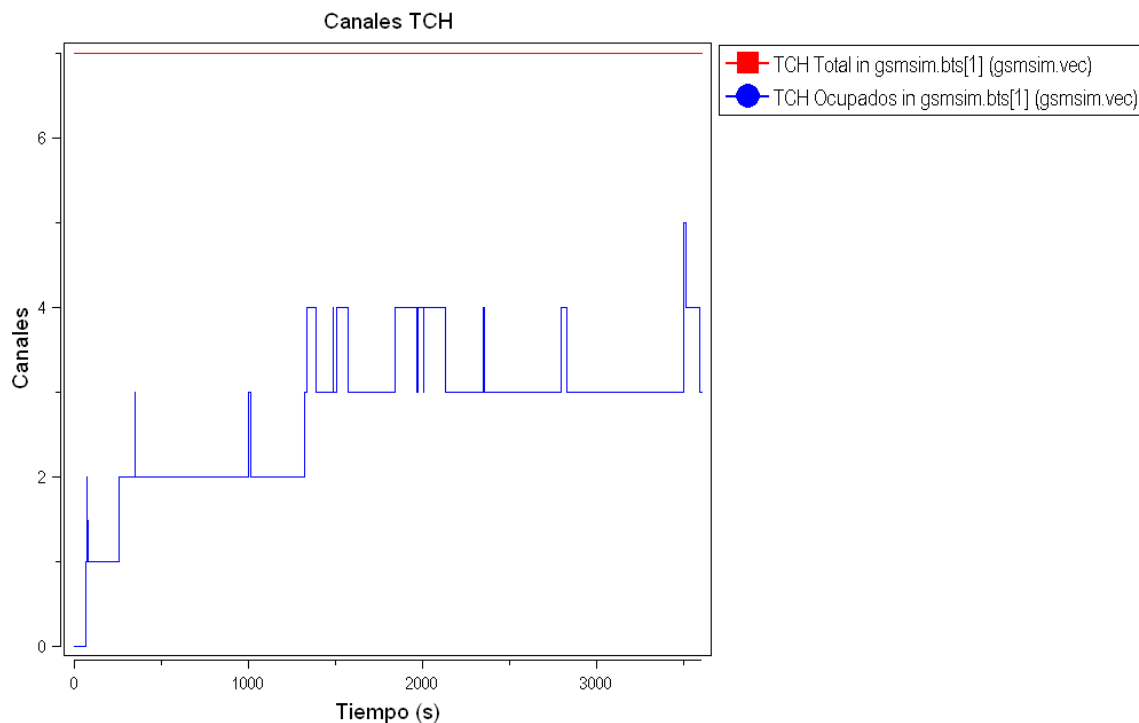


Figura 73 Canales TCH Ocupados

La estación BTS 1 tiene una ocupación media alta durante la simulación. De las tres BTS que se han introducido, esta es la que tiene una mayor densidad de tráfico, y una mayor densidad de usuarios según indican las gráficas. La mayor parte de las estaciones móviles usadas durante la simulación se han colocado en la zona alrededor de esta estación.

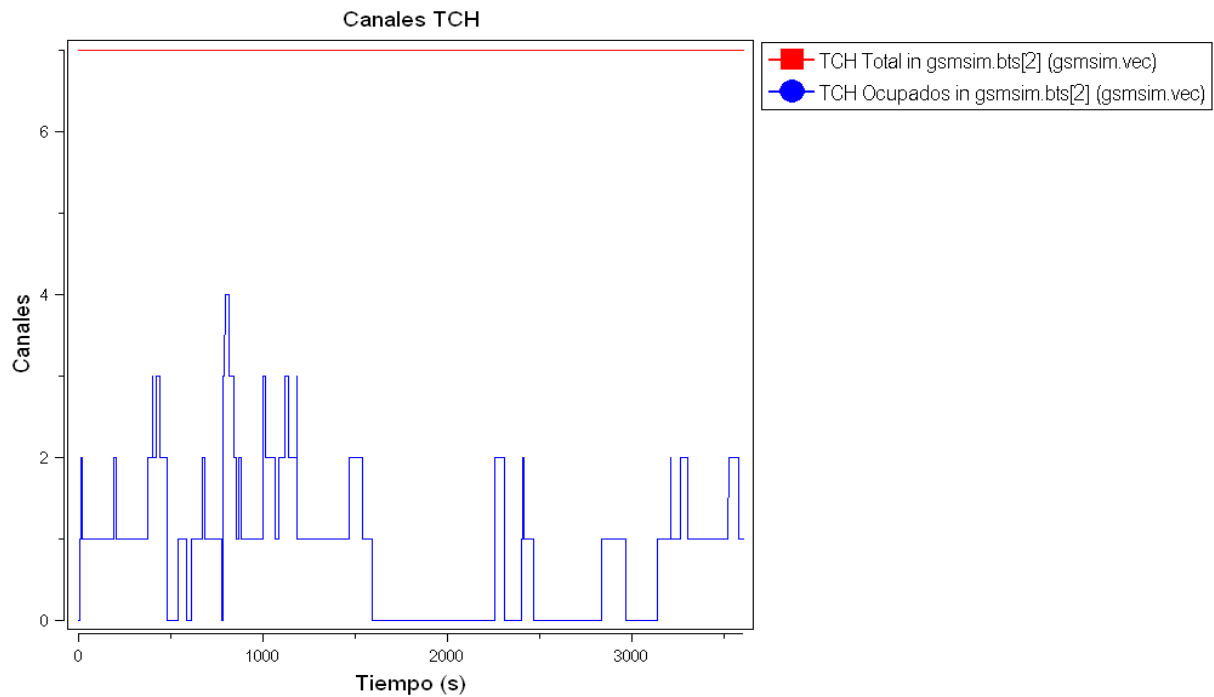


Figura 74 Canales TCH Ocupados

Esta estación tiene una variación de cantidad de tráfico bastante considerable. En primer lugar, el uso de canales es muy bajo, llegando a ser nulo durante ciertos instantes de tiempo. Al inicio de la simulación es cuando sufre un mayor tráfico.

11 MANUAL DE USUARIO

11.1 *Funcionamiento gsmsim.exe*

Para iniciar la aplicación se debe ejecutar el programa que se denomina gsmsim.exe. En el mismo directorio se deben encontrar otros ficheros para que la ejecución sea correcta. El fichero que debe acompañar siempre a este ejecutable es un fichero de configuración o de inicialización que tiene el nombre de omnetpp.ini. Desde este fichero se cargan las configuraciones y parámetros introducidos en la red de simulación. Nada más iniciarse la ejecución y tras un pequeño tiempo de inicialización, se presenta en pantalla la imagen de inicio de la simulación. Esta primera imagen es la pantalla principal en la cual se encuentran las herramientas de opciones de simulación (velocidad, imágenes, etc.) y la activación y acceso a otras ventanas. Esta ventana se compone de dos ventanas más pequeñas y un menú de herramientas (Figura 75). En la ventana blanca se escriben los eventos de la simulación, y en el cuadro de la izquierda de tono rosáceo se puede acceder a los parámetros de módulos, mensajes y conexiones establecidas. Dentro de este cuadro, el icono cuadrado con cuadrados azules en su interior representa la red. Si se despliega el menú aparecen los diferentes módulos o submódulos de la red. Si se pulsa dos veces con el ratón se puede acceder a la

información contenida en sus parámetros.

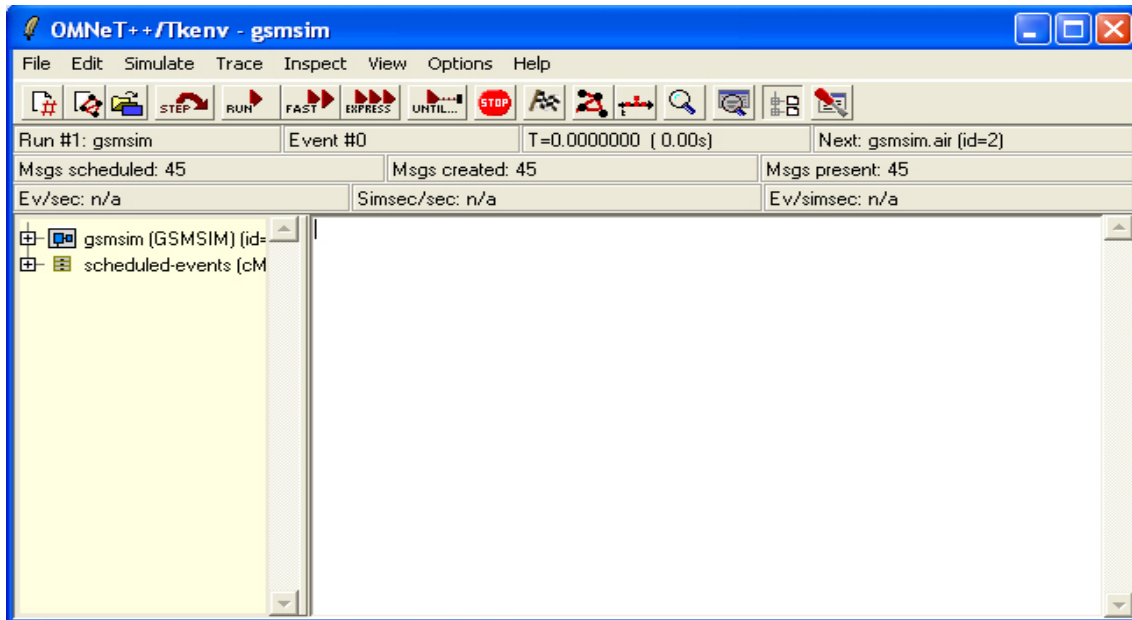


Figura 75 Ventana principal del simulador.

La ventana principal permite el acceso directo y sencillo a las herramientas de que dispone. A partir de esta ventana se pueden abrir el resto de ventanas que dan una visualización más detallada.

11.1.1 Herramientas del simulador

La aplicación incluye numerosas herramientas, a las cuales se puede acceder desde los menús desplegables de la parte superior. De todas formas, las principales herramientas y las más útiles se encuentran accesibles desde los iconos de la parte superior de la pantalla principal que se ve en la Figura 76.

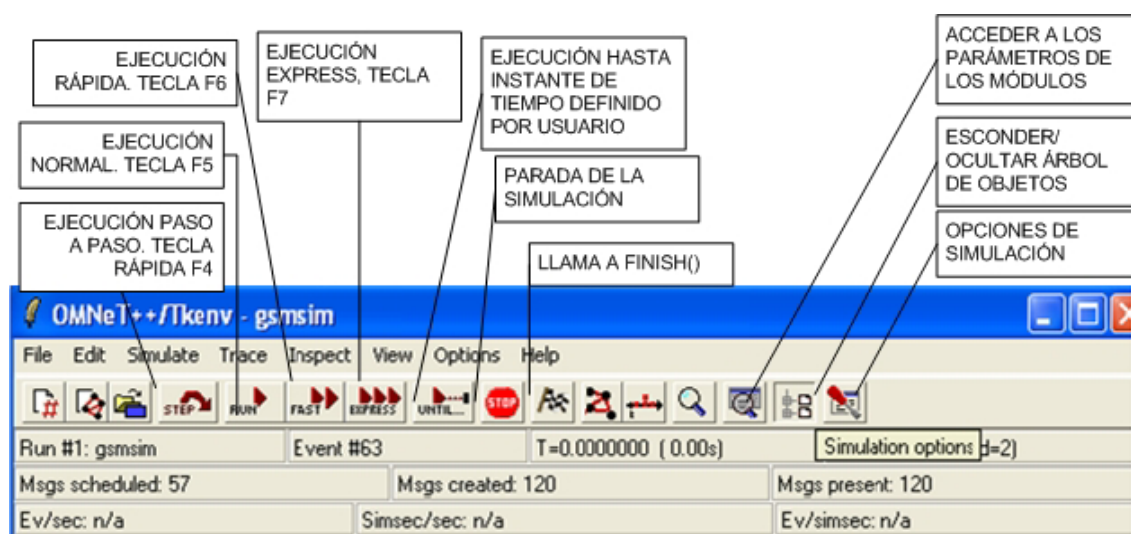


Figura 76 Herramientas del simulador.

De entre las herramientas que incluye, se puede acceder a las más importantes desde los iconos de acceso rápido de la parte superior de la aplicación. Los más destacados son los de velocidad de ejecución de la red.

Los tres iconos de la izquierda de la figura mencionada corresponden a las herramientas para abrir una ejecución (Run), abrir una determinada red (Network), y abrir un determinado fichero de descripción de red en lenguaje NED. Los siete iconos siguientes corresponden a la ejecución. Con ellos se puede definir la velocidad de simulación, las paradas, y el fin de la simulación. Las diferentes velocidades que se pueden emplear son ejecución paso a paso, ejecución normal, ejecución rápida, ejecución Express, o ejecución hasta punto definido por el usuario. Para las dos últimas se puede emplear la herramienta STOP para realizar la parada. Cuando se ha terminado de emplear la red o cuando se quiere volver a cargar es necesario pulsar el icono de llamada a finish, representado por una bandera a cuadros. Gracias a esta herramienta, se puede llamar a las funciones de finalización de todos los módulos sin necesidad de cerrar la aplicación y volver a abrirla.

Los tres iconos siguientes permiten la apertura de nuevas ventanas de visualización destinadas a detalles de la red. El primero de ellos activa la visualización de la ventana de red y módulos (Figura 77). Esta ventana reproduce la estructura de la red definida en el fichero NED y muestra el paso de mensajes. Si se quiere obtener una simulación rápida, sin tener en cuenta las imágenes, es aconsejable cerrar esta ventana. La gestión de imágenes y eventos de esta herramienta reduce en una gran cuantía la velocidad de simulación.

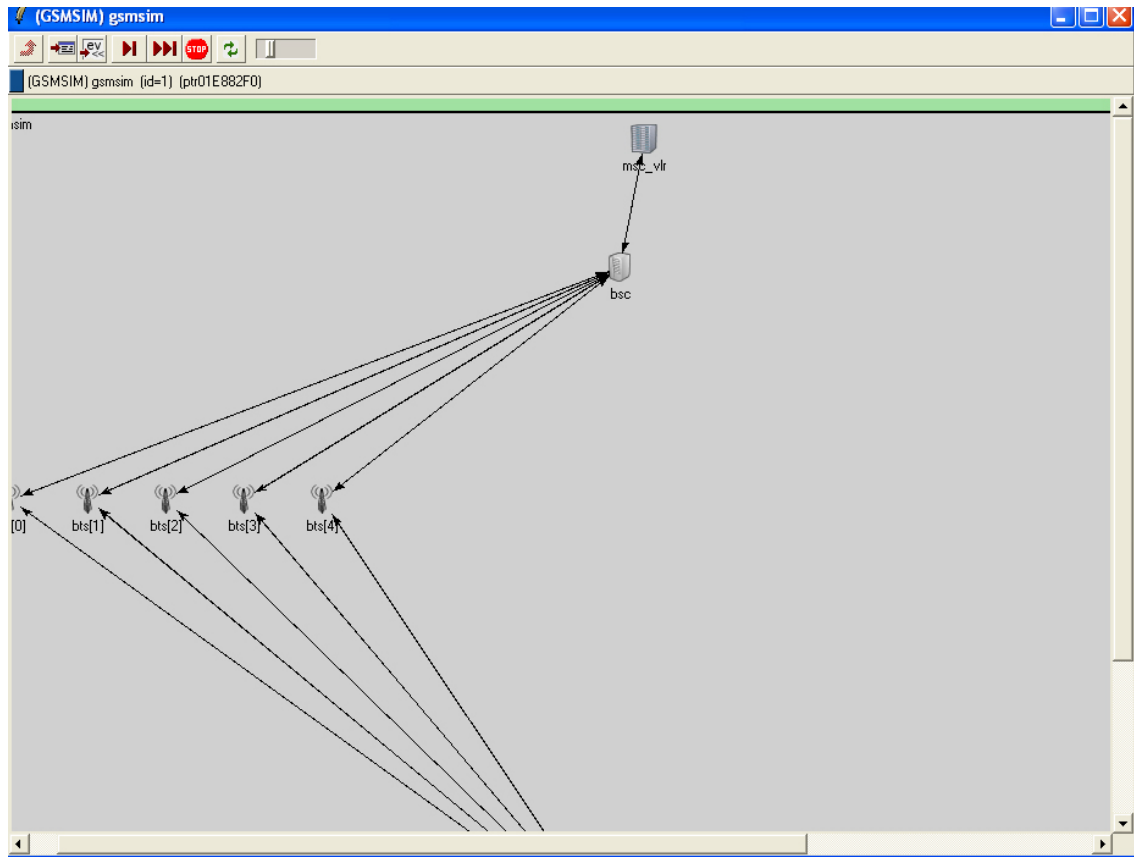


Figura 77 Ventana de representación de la red.

Ventana que visualiza la red. Aparecen los módulos, las conexiones, y los mensajes enviados entre equipos.

El segundo de los iconos de este grupo permite acceder a los mensajes creados en la red y que esperan llegar al destino indicado. Es una herramienta que sirve para ver que mensajes están siendo enviados a través de la red junto con información de tipo y de tiempo de llegada.

Class	Name	Info	Pointer
cMessage	ASSCMD_BSC_BTS	T= 8 (8.00s) src=gmsim.bsc (id=3) dest=gmsim.bts[2] (id=26)	ptr01EB16B0
cMessage	MAP_SEND_INFO_FOR_OL	T= 8.1 (8.10s) src=gmsim.msc_vlr (id=31) dest=gmsim.msc_vlr.msc (id=30)	ptr01EB4950
cMessage	MOVE_MS	T= 9 (9.00s) selfmsg for gmsim.ms[0] (id=4)	ptr01EBE790

Figura 78 Ventana de mensajes.

En esta ventana se puede ver el tipo de mensaje, tanto en tipo de clase a la que pertenece el objeto, como al nombre que recibe el mensaje de parte del usuario. También aparece el tiempo de llegada, y un puntero al mensaje enviado.

Con el tercer icono del grupo, que representa una lupa se puede realizar búsquedas en función de caracteres o de cadenas de caracteres dentro de la pantalla de representación de eventos.

Por último, los tres iconos de la derecha permiten la búsqueda de módulos dentro de la simulación, mostrar u ocultar el árbol de módulos, y acceder a las opciones de simulación.

11.1.2 Ventanas de detalles

Ya se han comentado las dos ventanas más importantes, la ventana principal y la ventana de visualización de la red. Además de estas dos existen otras ventanas que muestran detalles más concretos de la red, de los módulos y de los mensajes.

11.1.2.1 Ventana de visualización para el módulo MSC-VLR

El módulo MSC-VLR es un módulo compuesto formado por dos módulos simples. Gracias a esto, se puede abrir otra ventana (Figura 79) en la que aparezcan ambos módulos, y sus conexiones, tal y como aparece en la visualización principal de la red. Incluye herramientas para abrir la ventana superior de visualización (ventana principal de visualización), la ventana de inspección de parámetros, y la ventana de inspección de eventos. También permite el cambio en la velocidad de simulación mediante la herramienta de la derecha. Arrastrando la barra central en el eje se cambia la velocidad de representación de eventos. Para acceder a ella es necesario pulsar dos veces con el puntero del ratón sobre el icono que representa al módulo MSC-VLR. Esto funciona así para todos los módulos complejos presentes en OMNET++. Pulsando dos veces sobre su icono aparece una ventana nueva con los módulos simples que lo componen. Si se hace lo mismo sobre módulos simples aparece el cuadro de parámetros del módulo.

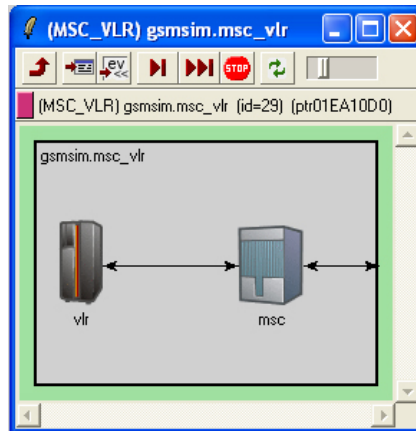


Figura 79 Ventana de visualización de MSC-VLR.

La ventana tiene las mismas herramientas que la ventana de visualización principal de la red.

11.1.2.2 Ventana de visualización de conexiones

Al igual que se pueden representar los módulos con iconos, también se pueden representar las conexiones establecidas entre ellos. Para esto se debe pulsar con doble clic sobre las líneas que representan las conexiones entre módulos. Las ventanas que se abren son ventanas en las cuales aparecen bloques que representan a los módulos, cuadrados amarillos y blancos que representan las puertas de entrada o de salida, y la línea de conexión que se visualiza. El objeto de estas ventanas es ver con mayor detalle la llegada de mensajes a los módulos. En las figuras siguientes (Figura 80, Figura 81, Figura 82, Figura 83) se presentan ejemplos de ventanas de visualización de conexiones utilizadas durante una ejecución de gsmsim.exe.

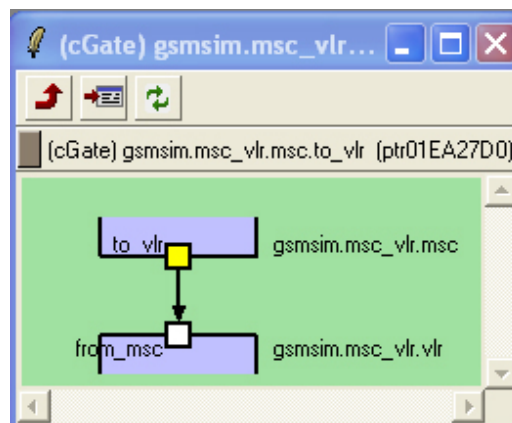


Figura 80 Ventana de conexión MSC hacia VLR.

Conexión entre MSC y VLR, con puerta de salida en MSC y puerta de entrada en VLR.

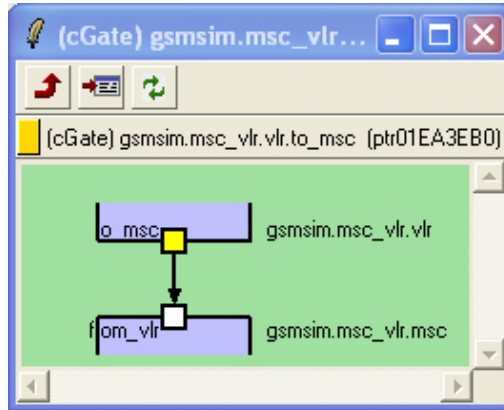


Figura 81 Ventana de conexión VLR hacia MSC.

Conexión entre MSC y VLR, con puerta de salida en VLR y puerta de entrada en MSC.

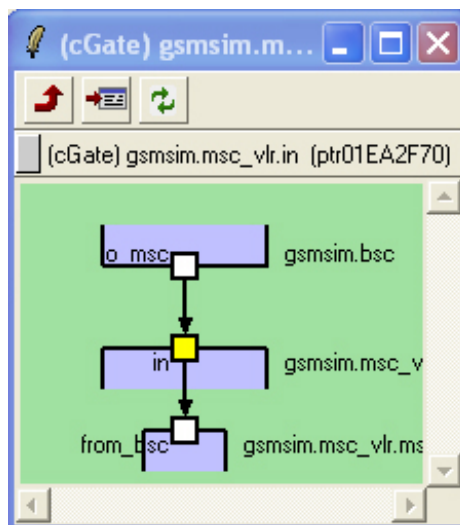


Figura 82 Ventana de conexión BSC hacia MSC.

Conexión entre BSC y MSC, con puerta de salida en BSC y puerta de entrada en MSC. Aparece también la puerta de entrada del módulo compuesto MSC_VLR al que pertenece el MSC. Los módulos compuestos actúan de forma transparente hacia el exterior para los módulos que se encuentran dentro de él. La puerta simplemente recoge el mensaje y lo envía a la puerta del módulo simple MSC.

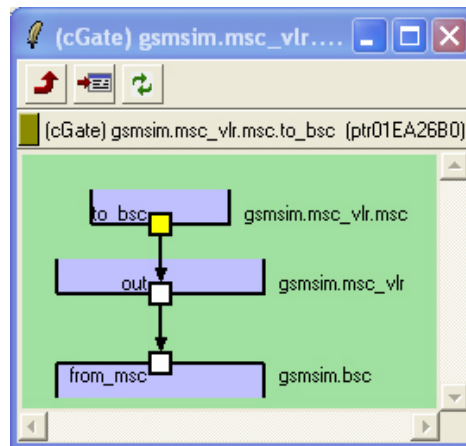


Figura 83 Ventana de conexión MSC hacia BSC.

Conexión entre BSC y MSC, con puerta de salida en MSC y puerta de entrada en BSC. La conexión es parecida a la imagen anterior, aunque ahora en dirección contraria. El módulo compuesto en esta conexión también actúa de forma transparente.

11.1.2.3 Ventana de visualización de eventos

La ventana de visualización de eventos se encuentra ya incluida en la ventana principal de la aplicación, aunque sin embargo hay ocasiones en las que interesa usar esta ventana de forma separada, ya sea por comodidad para el usuario o para una mejor visualización. Para acceder a esta ventana se debe pulsar con el puntero del ratón sobre el icono que aparece en las ventanas de visualización de los módulos y que contiene una flecha con las letras “ev” (Figura 84).



Figura 84 Icono de visualización de eventos.

Icono presente en las ventanas de visualización de módulos y red para abrir la ventana de presentación de eventos.

La ventana () imprime los eventos ocurridos durante la simulación, y el instante de simulación en el que ocurrieron. Además aparece el módulo en el que ocurrieron dichos eventos juntos con el número. Es en esta pantalla donde aparecen las cadenas que se quiere que salgan por pantalla, y que se han debido de incluir en el objeto “ev” tal y como se hace en C++ con “cout”.

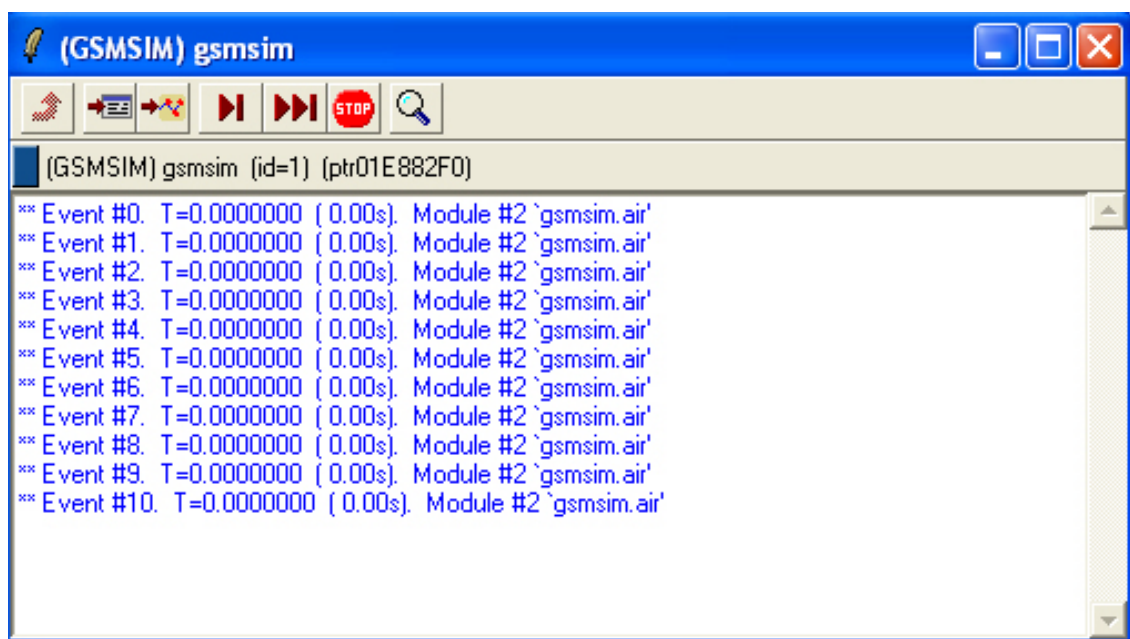


Figura 85 Ventana de eventos.

En esta ventana se imprimen todos los eventos, y además las cadenas de caracteres o mensajes que el programador haya incluido en el código fuente a través del objeto “ev”. Este objeto se emplea igual que el “cout” en C++. Para una mayor referencia ver el capítulo de la memoria que se centra en OMNET++.

11.1.2.4 Ventana de parámetros de módulos

Antes se ha mencionado que mediante la doble pulsación del puntero del ratón sobre un módulo compuesto se puede abrir una ventana donde aparezca una representación de sus módulos simples. Para los módulos simples, esta doble pulsación, permite la apertura de un cuadro con los parámetros de cada módulo. La ventana que aparece es la de la Figura 86. En ella se puede acceder a información del módulo, a las puertas que tiene, a los parámetros y a los submódulos en caso de que los tenga. La pestaña principal es la de información general que muestra el nombre del módulo, y el icono asignado. Las otras pestañas Params, y Gates se pueden ver en la Figura 87 y en la Figura 88 respectivamente. La pestaña Contents muestra el nombre de los vectores de datos de salida que se están usando para que después sean leídos por el programa Plove. Además muestra también el número de datos grabados. Desde la figura de parámetros se pueden abrir otros cuadros que representen los parámetros de cada módulo de forma individual. Cada cuadro abierto representa únicamente un parámetro, su valor, su tipo y su nombre (Figura 89).

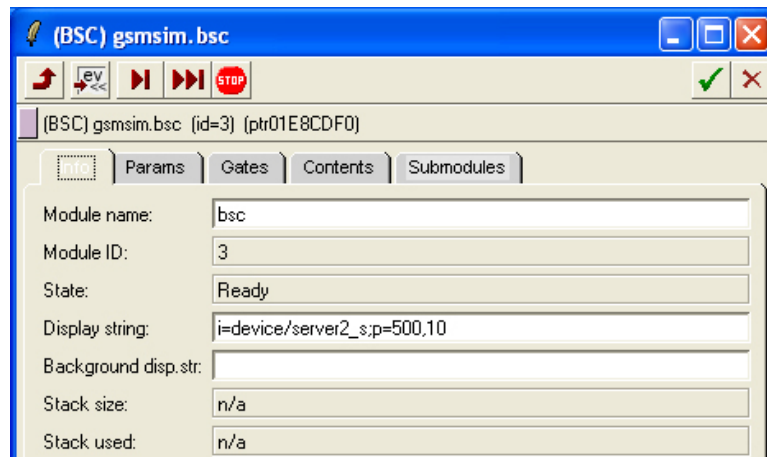


Figura 86 Ventana de detalles del módulo BSC.

Información general. Esta pestaña contiene información acerca del nombre del módulo, el identificador dentro de la red, el estado, el icono asignado, y el fondo asignado.

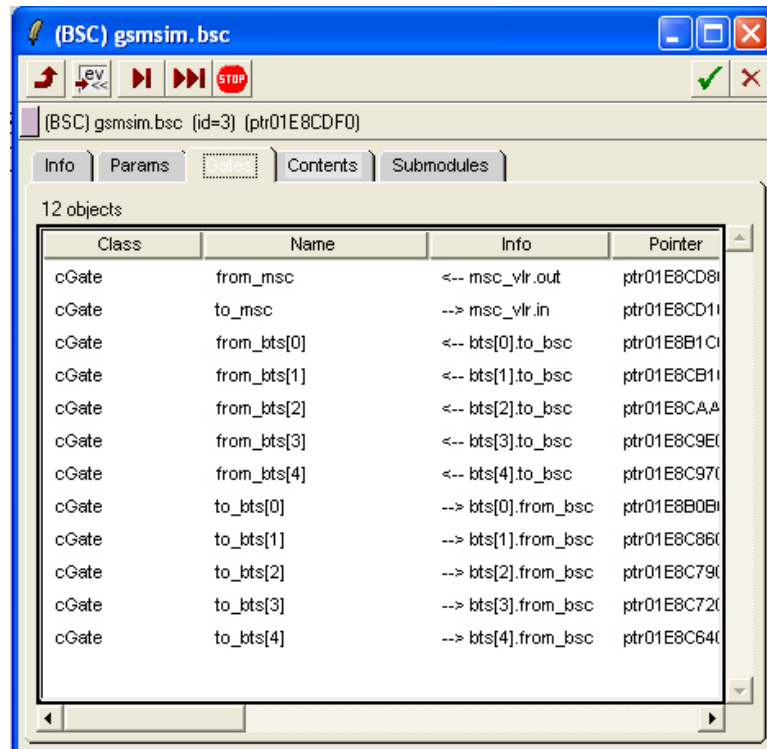


Figura 87 Ventana de detalles del módulo BSC. Información de puertas.

La pestaña contiene información de cada una de las puertas que contiene un módulo. Este módulo BSC tiene doce puertas para doce conexiones con otros módulos de la red, MSC y BTS. De estas doce, seis son conexiones de entrada y las otras seis de salida.

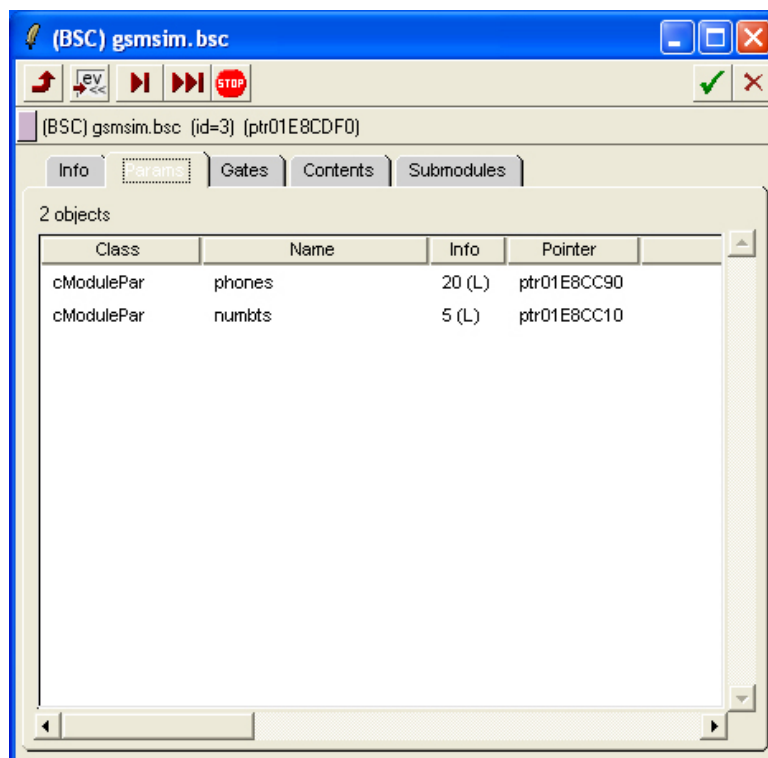


Figura 88 Ventana de detalles del módulo BSC. Información de parámetros.

La pestaña de parámetros presenta los parámetros pertenecientes a un módulo BSC, en este caso se ven dos, “phones” y “numbts”. El primero es el número de estaciones móviles o teléfonos móviles de la red, veinte, y el segundo el número de estaciones base, cinco.



Figura 89 Ventanas de parámetros de un módulo BTS.

Se representan varias ventanas de detalles de parámetro. Los datos van en este orden y de izquierda a derecha, posición en el eje X de la antena, posición en el eje Y de la antena, número de canales TCH en la antena, potencia emitida por la antena.

11.1.2.5 Ventana de parámetros de la red

Al igual que para los módulos, para las redes también se pueden abrir las ventanas de parámetros. En NED, las redes se definen a partir de instancias de módulos compuestos que incluyen a todos los equipos de la red. Las pestañas que contiene son de información general (Info), de submódulos (Figura 91), de parámetros (Figura 90) y de puertas.

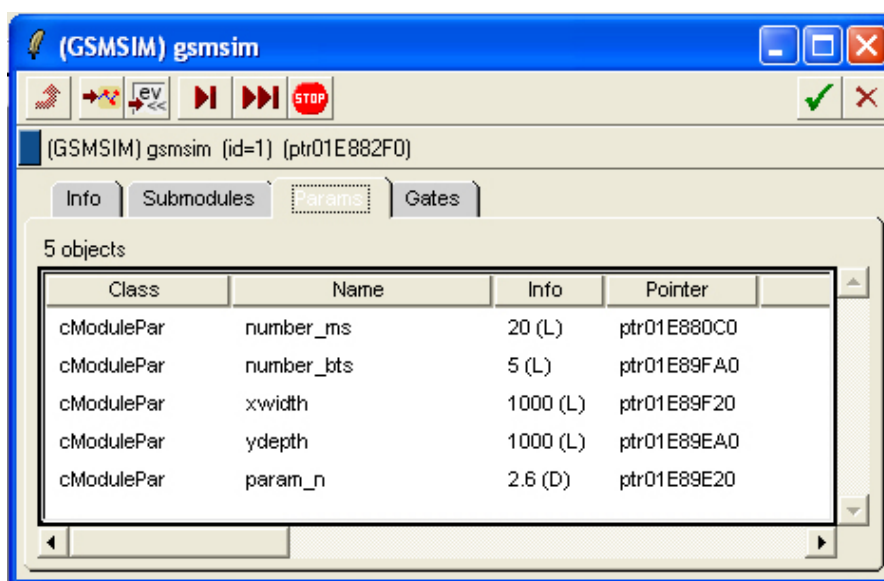


Figura 90 Ventana de información de red. Parámetros.

Los parámetros de la red son el número de estaciones móviles, el número de estaciones BTS, el ancho y el alto del área simulada y el coeficiente exponencial de pérdidas. Estos valores se introducen desde el fichero de configuración de la simulación.

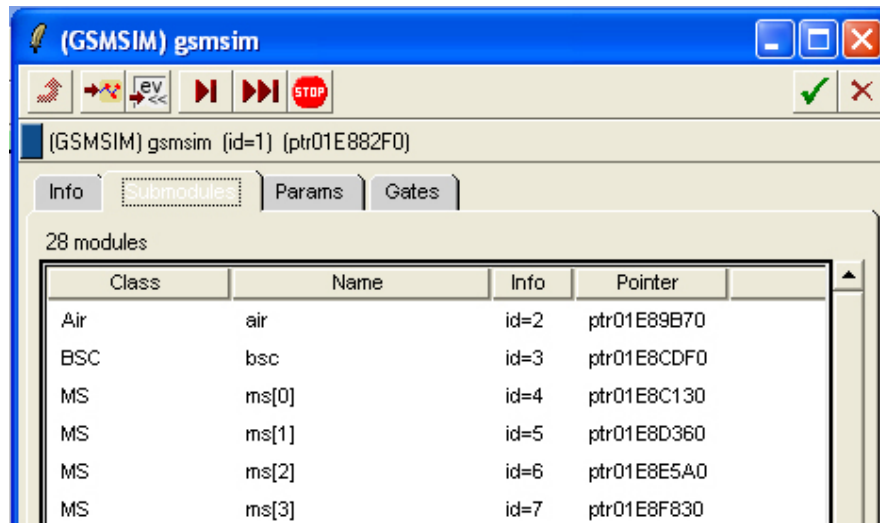


Figura 91 Ventana de información de red. Módulos.

Éstos son algunos de los módulos presentes en la red. Air es un módulo que representa al medio físico empleado en la comunicación. BSC es la estación base controladora y el MS el equipo móvil. La pestaña indica el tipo de módulo, el nombre del módulo en la red, el identificador, y el puntero en memoria a este módulo.

11.1.2.6 Ventana de parámetros de mensajes

Desde el marco izquierdo de la ventana principal del programa se puede acceder a las ventanas de detalles de los mensajes. Cada mensaje creado se coloca en el esquema desplegable de scheduled-events, y a través de su nombre se puede acceder a esta ventana. En la (Figura 92) se muestra un ejemplo de un mensaje de indicador de establecimiento (ESTIN). Las pestañas que aparecen son de información general, de tiempo de envío/llegada, de información de control y de parámetros. En la pestaña de tiempo de envío/llegada se puede ver el instante en tiempo de simulación en que un mensaje fue enviado por la puerta de salida, y el instante en que llega al módulo destino.

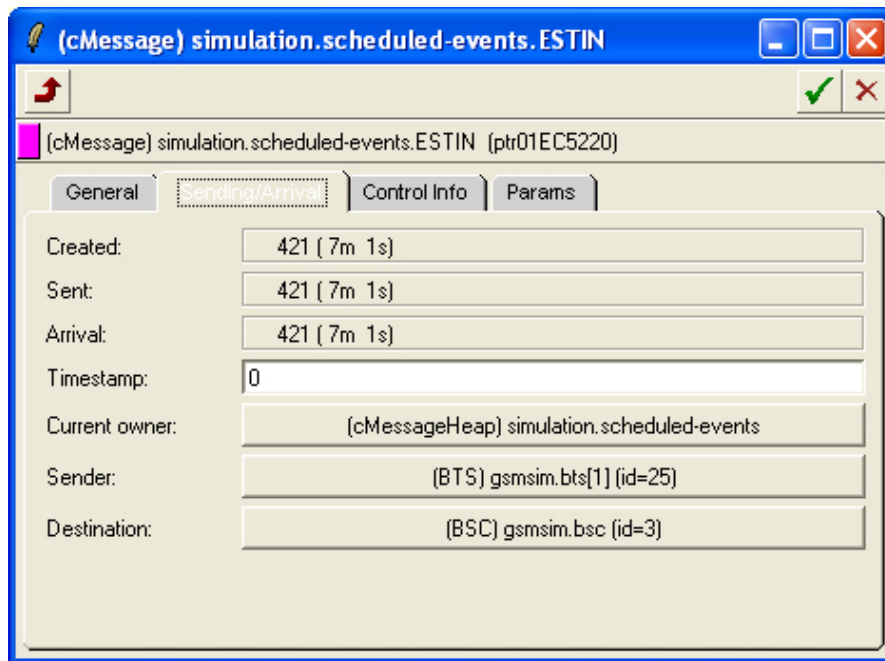


Figura 92 Ventana de detalles de mensaje.

El mensaje del que se visualizan los datos se crea en el instante 421 (7m 1s), y se envía y se recibe en este mismo instante. El origen es el módulo BTS(1) y el destino es el módulo (BSC). El mensaje después de ser leído, pasa a la cola de eventos.

11.1.2.7 Ventana de gráficas para ver diferencias de valores

Existen unas ventanas que permiten ver los valores de los parámetros en una gráfica durante la simulación. Para ello almacena el último valor modificado de ese parámetro, y el valor modificado en el instante de visualización, y realiza una gráfica. En la Figura 93 se ha empleado para ver la variación de potencia detectada por una estación móvil. La imagen de la izquierda representa la potencia recibida por la MS 17 de la BTS 0. Como se puede ver el valor es creciente, por lo que se va acercando a la antena. Para la imagen de la derecha se representa la misma MS, pero con la BTS 1. Como se puede ver el valor se ha reducido y por tanto se está alejando de la antena.

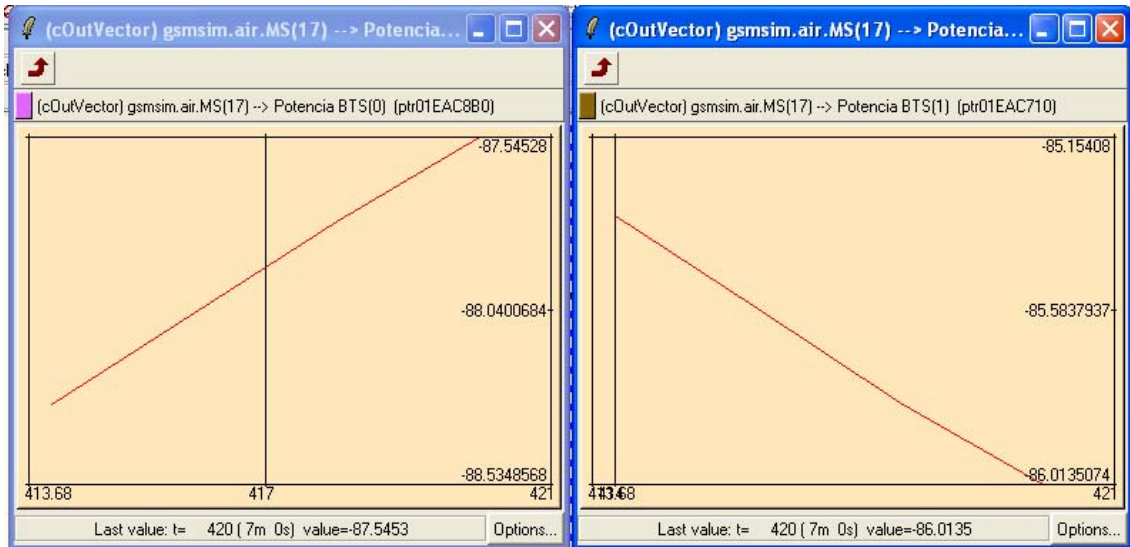


Figura 93 Gráficas de variación de potencia recibida en MS(17).

Los instantes de captura de datos son en 413.60 y el 421. En la imagen izquierda, aumenta la potencia recibida en la MS, en la de la derecha se reduce.

11.1.3 Proceso de ejecución

La primera imagen que se obtiene en la ventana de visualización de la red al empezar a ejecutar la simulación es la de la red con el módulo AIR rodeado por mensajes de color rojo. Son mensajes que la aplicación emplea para la inicialización de la simulación por lo que no deben ser considerados por el usuario durante la simulación. Estos mensajes reciben el nombre de mensajes INIT. Los campos que contiene son valores de posición de cada equipo dentro del área de simulación y potencias de las estaciones base, para realizar el cálculo de potencias y el encaminamiento de mensajes en la red.

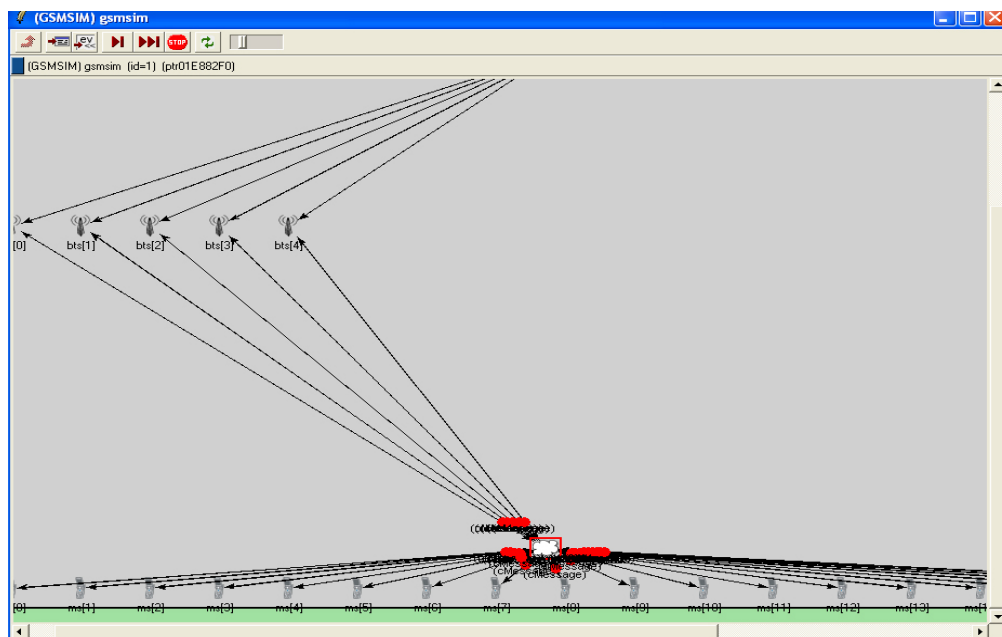


Figura 94 Inicio de la ejecución. Mensaje INIT.

El mensaje INIT se emplea para que el módulo AIR grabe los valores de posición de las MS y las BTS. A partir de estos valores, puede luego calcular las potencias y encaminar los mensajes por la red a los destinos.

Para realizar la simulación se tienen varias opciones. Estas opciones van desde la ejecución paso a paso de la simulación, a la ejecución en modo FAST. Si se quiere ejecutar paso a paso para ver con calma la ejecución de la simulación pues lo que se debe realizar es pulsar la tecla F4. Si lo que se pretende es ejecutar la simulación con una mayor rapidez pues la opción es pulsar la tecla F6 o hacer clic en el icono FAST del entorno de simulación. Las otras opciones son el modo RUN pulsando F5 con la ejecución normal, el modo EXPRESS con F7 para la ejecución muy rápida y el modo UNTIL para la ejecución muy rápida hasta un punto de parada definido por el usuario o hasta que el usuario pulse el STOP (Figura 95). Con la pantalla de visualización de módulos, la simulación es mucho más lenta. Si se prueba a cerrar esta ventana durante la simulación, el tiempo de espera hasta el final de la misma se verá reducido muy considerablemente. Por tanto, si lo que interesa es obtener resultados y gráficas, directamente cerrar las ventanas de visualización de módulos, y dejar únicamente la ventana principal.

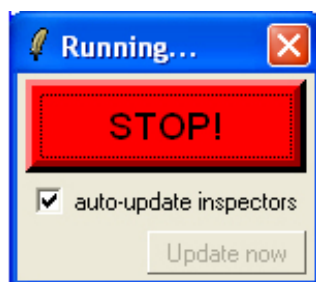


Figura 95 Ventana de STOP.

Ventana empleada para la parada de la simulación cuando se emplea el modo EXPRESS o el modo UNTIL.

A continuación pulsar alguna de las opciones de ejecución que anteriormente se han mencionado con lo que dará comienzo la comunicación de mensajes entre equipos. Según van saliendo los mensajes, en la pantalla de eventos se realiza la impresión de información de eventos. Entre ellos se verán los “INFORMES DE POTENCIAS” en los que se saca por pantalla la información de potencias (Figura 96). En caso de que una estación se quede sin cobertura o se le asigne un canal, también aparecerá esta información en la pantalla de eventos (Figura 97), o en caso de que se realicen traspasos también se informará.

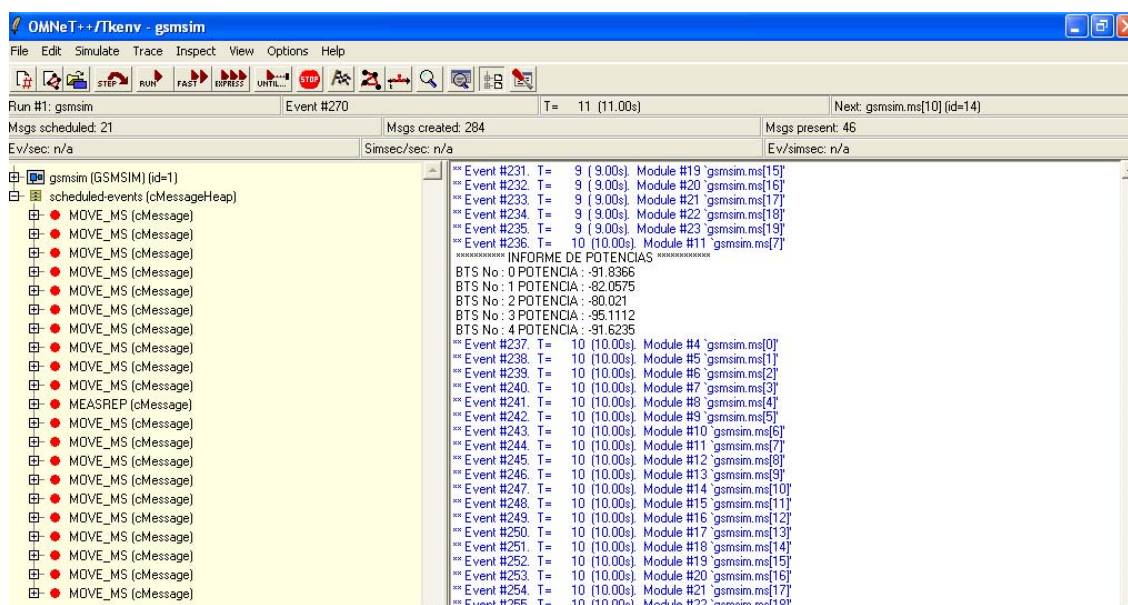


Figura 96 Informe de potencias en pantalla de eventos.

Información de potencias en la pantalla de eventos. Esta información se graba en el fichero de salida vectorial.

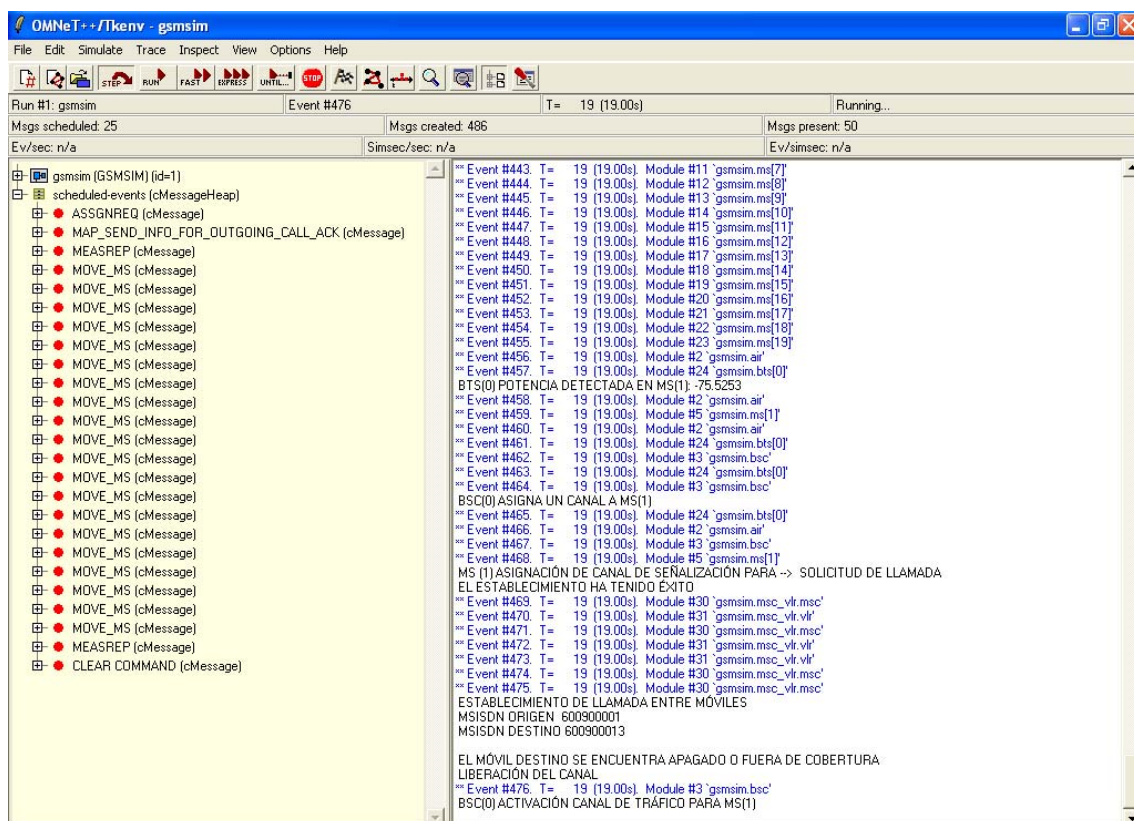


Figura 97 Información de asignación de canales en la pantalla de eventos.

Informa de la asignación de un canal y del establecimiento de una llamada entre dos móviles, uno con MSISDN 600900001 (origen) y el otro con 600900013. El VLR indica además que el destino se encuentra apagado o fuera de cobertura.

Los mensajes enviados se representan mediante círculos en las conexiones por las que se envían. Ejemplos de esta visualización se pueden ver en la Figura 98 y en la Figura 99. La primera representa el mensaje de medidas de potencia MEASREP enviado entre la MS y la BTS. La segunda figura mencionada envía un mensaje de actualización de posición entre MSC y VLR dentro del módulo compuesto MSC-VLR.

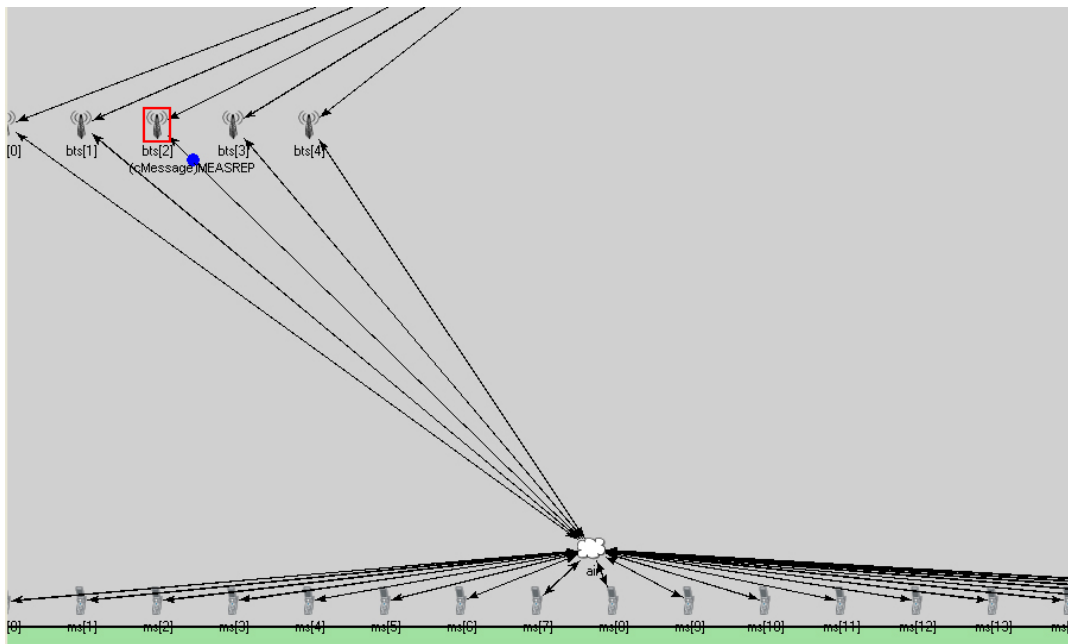


Figura 98 Mensajes en la pantalla de visualización de red.

El módulo de medidas de potencia se envía desde la MS hacia la BTS. El color asignado es azul, y el nombre es MEASREP.

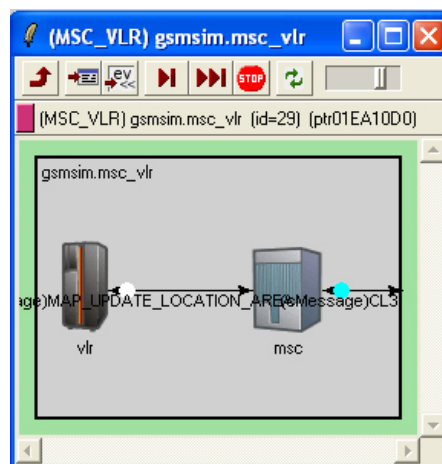


Figura 99 Mensajes en la ventana de visualización del módulo MSC_VLR.

El módulo compuesto MSC-VLR recibe un mensaje del exterior llamado CL3I y de color azul cyan. El módulo MSC envía un mensaje de actualización de posición al VLR (MAP_UPDATE_LOCATION_AREA).

Cuando llegan los mensajes al equipo BSC, se llama a la función Bubble. Esta función permite la visualización de un mensaje de cadena de caracteres durante la simulación. El resultado es el de la Figura 100.



Figura 100 Representación de la función Bubble().

La función permite visualizar en la ventana de representación de módulos una cadena de caracteres definida por el usuario e introducida como parámetro en la función. En este caso al llegar un mensaje de indicación de establecimiento se visualiza el grupo de mensajes al que pertenece. Los mensajes ESTIN según las recomendaciones GSM son del tipo Layer 3 Radio Link Layer Management, es decir un mensaje de nivel 3 de gestión de la capa de enlace radio.

El usuario si quiere puede acceder a los parámetros de los módulos o de los mensajes desde el cuadro izquierdo presente en la ventana principal del simulador. Este cuadro se denomina árbol de módulos y mensajes.

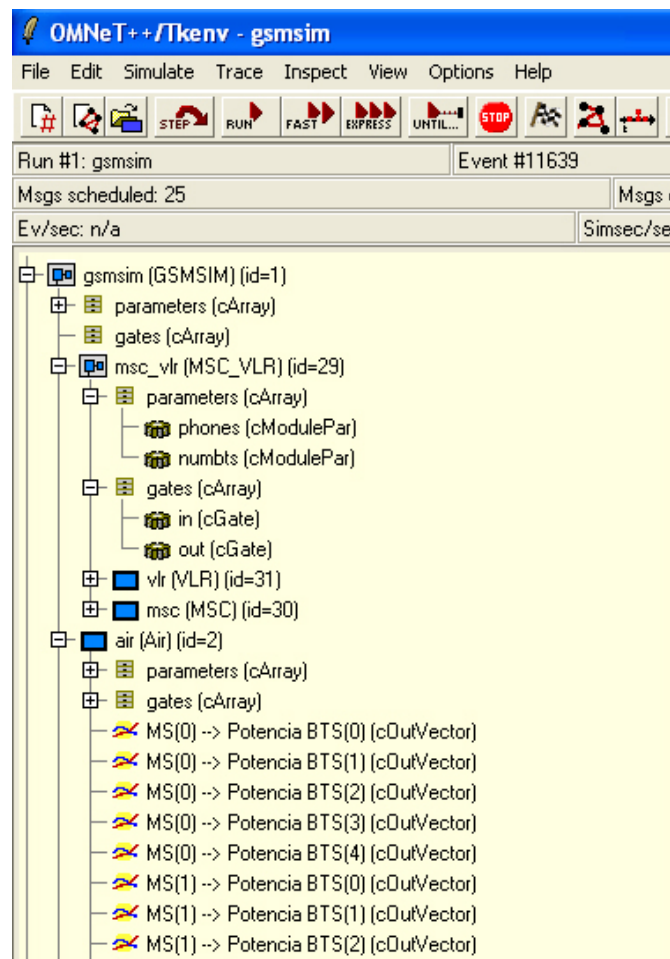


Figura 101 Acceso a los parámetros de un módulo desde el árbol de la red.

Desde este cuadro se puede acceder a los parámetros de los módulos, además de a la información de puertas y a los valores contenidos en los objetos cOutVector para la escritura de vectores.

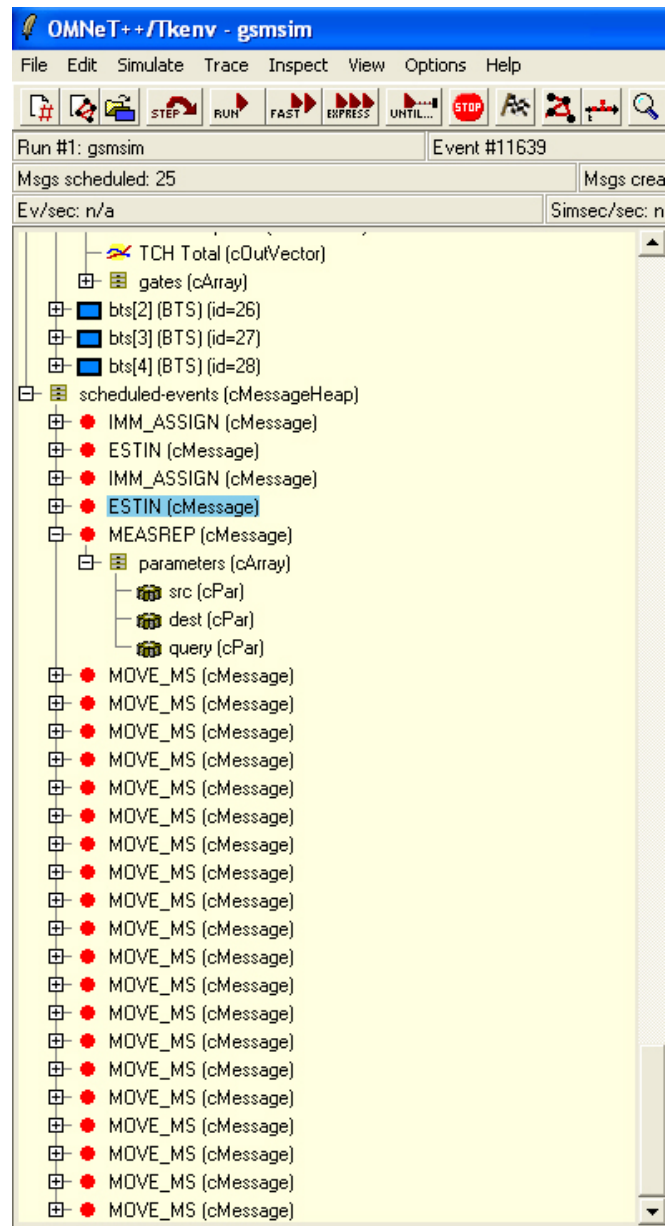


Figura 102 Acceso a los mensajes de un módulo a través del árbol de la red.

Desde este cuadro se puede acceder a los parámetros de cada uno de los mensajes creados por la simulación. En la imagen se ha desplegado el árbol para un mensaje de informe de medidas MEASREP.

La aplicación finaliza cuando llegue al límite definido por el usuario en el fichero de configuración o cuando el usuario ordena al programa que finalice. Una vez ejecutada la aplicación, el programa pregunta si se quiere que se llame a la función finish(). Esta función es la que escribe en dos ficheros los resultados de la simulación. Los datos que se presentan

son de dos tipos, escalares y vectoriales como se ha comentado ya. Para obtener más información sobre los programas de visualización de datos, ver la sección 11.3 donde se explica el funcionamiento de las mismas. Los datos que se graban son los de tiempo de llamada, número de traspasos totales, número de traspasos fallidos, número de asignaciones correctas, número de asignaciones fallidas, etc. En cuanto a los datos de salida vectoriales, lo que se guardan son las informaciones de potencias de estaciones y de número de llamadas en función del tiempo. Probablemente las gráficas de potencias sean las más importantes, ya que en éstas se puede ver si la estación móvil o MS se queda sin cobertura o cambia de estación base. En la siguiente sección se explica el funcionamiento de una aplicación añadida al proyecto inicial para mejorar la visualización espacial de los equipos móviles y de las antenas.

Como se ha podido comprobar en esta sección, el uso de la aplicación `gmsim.exe` es muy sencillo, con una interfaz muy útil para el usuario y con un gran número de posibilidades de visualización para dar mayor o menor detalle a los valores que se quieren observar.

11.2 *Funcionamiento SimOffLine.exe*

11.2.1 Objetivo de la aplicación

La aplicación `SimOffLine.exe` tiene como objetivo proporcionar una herramienta de ayuda a la aplicación principal. La carencia principal que presenta el simulador es la falta de la posibilidad de visualización del plano o área de simulación. Las herramientas de OMNET++ no permiten que sea capaz de visualizar en una pantalla el seguimiento de los movimientos de cada móvil y de cada estación, por lo que al usuario le resulta bastante complicado conocer cuales son las trayectorias que ha seguido el equipo (sobre todo si la trayectoria es aleatoria). Este ejecutable proporciona una representación gráfica muy sencilla para ayudar al usuario a visualizar estos movimientos.

Para ello la aplicación principal escribe unos ficheros en el mismo directorio en el que se encuentra. La aplicación de apoyo, debe encontrarse también en este directorio, y carga automáticamente estos archivos cuando se ejecuta. Estos ficheros, que tienen la extensión `.gmsim`, almacenan la posición en dos coordenadas de cada estación móvil en un instante de tiempo también almacenado. Este fichero tiene el nombre de `msPositionM.gmsim`, con M el número de estación móvil. El otro archivo llamado `btsPosition.gmsim` almacena la posición de todas las estaciones BTS de la red GSM simulada, los parámetros del área de simulación, y

el número de estaciones móviles.

La aplicación se debe visualizar después de que se haya simulado la red, y no durante la ejecución de la misma. Cuando se estén visualizando las gráficas de potencias generadas por el ejecutable, esta herramienta será muy útil para comprender los datos que representan estas gráficas. La forma de uso será utilizar a la vez el programa SimOffLine.exe y OMNET++ Plove, e ir consultando las trayectorias y las gráficas de potencia resultantes para así llegar a conclusiones sobre el movimiento del móvil, o sobre la calidad de cobertura de la red.

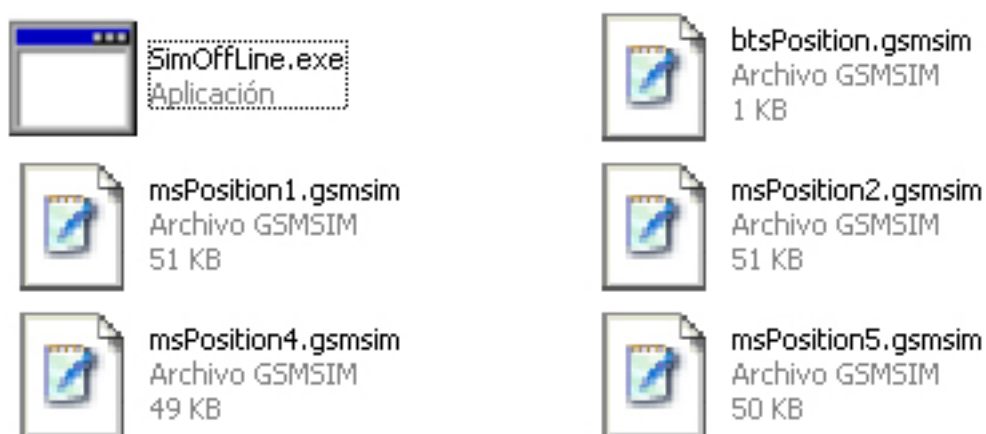


Figura 103 SimOffLine y archivos de posición.

La aplicación de apoyo tiene el nombre de SimOffLine.exe. El fichero de posición de BTS, en la imagen parte superior derecha, contiene la posición de las BTS y su radio de cobertura en el área de simulación. Los ficheros de MS contienen las posiciones de cada estación móvil y la marca de tiempo para cada posición. Estos ficheros son escritos por la aplicación gsmsim.exe cuando se ejecuta.

11.2.2 Funcionamiento de la aplicación

Cuando se inicia la aplicación aparece una pantalla similar a la de la Figura 104. Presenta dos ventanas, una es una interfaz de comandos para dar órdenes a la aplicación, y la otra es una ventana para la visualización gráfica del área de simulación. En esta última ventana, los círculos representan las áreas de cobertura de cada estación. Son circulares puesto que no se tiene en cuenta la presencia de obstáculos. Las líneas negras que crean una malla limitan el área de simulación. Para que sea más sencilla la simulación, se dibuja una línea por lo que equivalen a 1000 metros. En este dibujo, el área corresponde a una superficie de 50000 x 50000 metros por lo que se dibuja una malla de 50 x 50 líneas. Las líneas blancas que se

encuentran dentro de esta superficie son las trayectorias de los móviles. Como se puede ver, al llegar al límite de la red de líneas, que también es el límite de área de simulación, el móvil rebota e impide que salga del área. Las trayectorias de los móviles a visualizar se pueden seleccionar desde la ventana de órdenes.

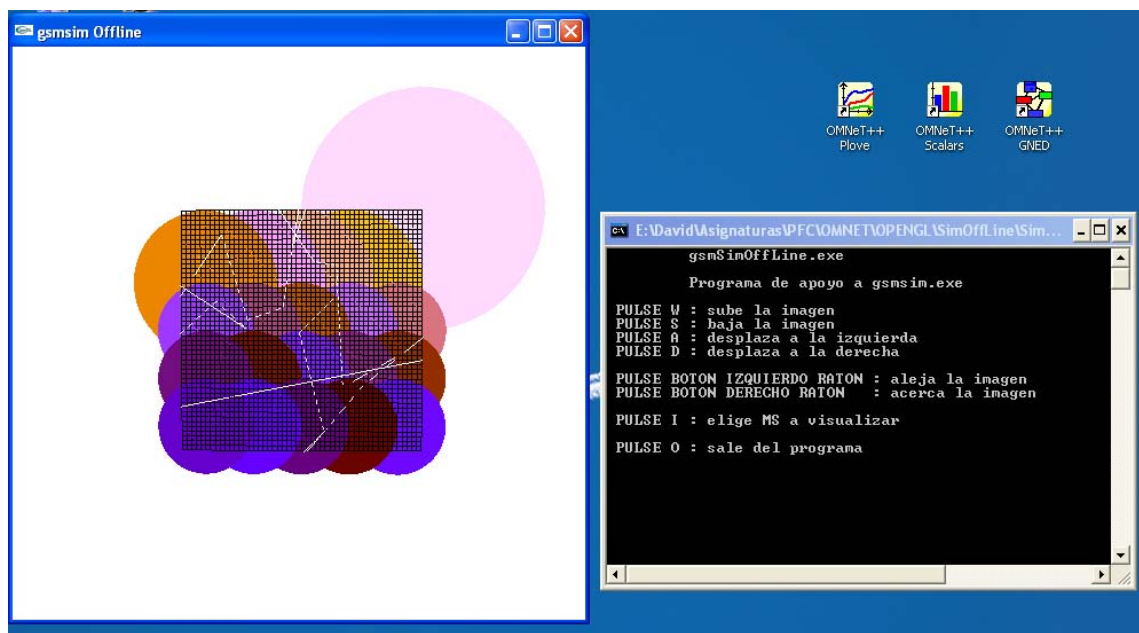


Figura 104 Pantalla de SimOffLine.exe.

El ejecutable despliega dos ventanas, una gráfica en la que presenta la imagen del área de simulación, y otra ventana de comandos para darle órdenes a la aplicación

La ventana de órdenes permite varias opciones, que son las siguientes:

- Tecla 'w': sube la representación de la imagen.
- Tecla 's': baja la representación de la imagen.
- Tecla 'a': desplaza la imagen a la izquierda.
- Tecla 'd': desplaza la imagen a la derecha.
- Tecla 'i': selección de MS a visualizar.
- Tecla 'o': salida del programa.
- Botón derecho ratón: acerca la imagen.
- Botón izquierdo ratón: aleja la imagen

La pulsación de alguna de estas teclas despierta una acción en la gráfica. Si se pulsan los

botones del ratón, el programa reaccionará aumentando o acercando la imagen, actuando como un “zoom”. Es importante que la selección de estas teclas se realice con la ventana de la imagen seleccionada. Aunque esta ventana de órdenes forma parte de aplicación, la parte principal es la imagen, y es la que despierta y reacciona ante los eventos de teclado o ratón. La ventana de comando sólo sirve para representar las opciones, y para la selección de estaciones móviles. En la Figura 105 se puede ver la ventana de comandos que se despliega con el inicio de la aplicación. La aplicación no distingue entre mayúsculas y minúsculas, para que sea más sencilla la utilización de cara al usuario.

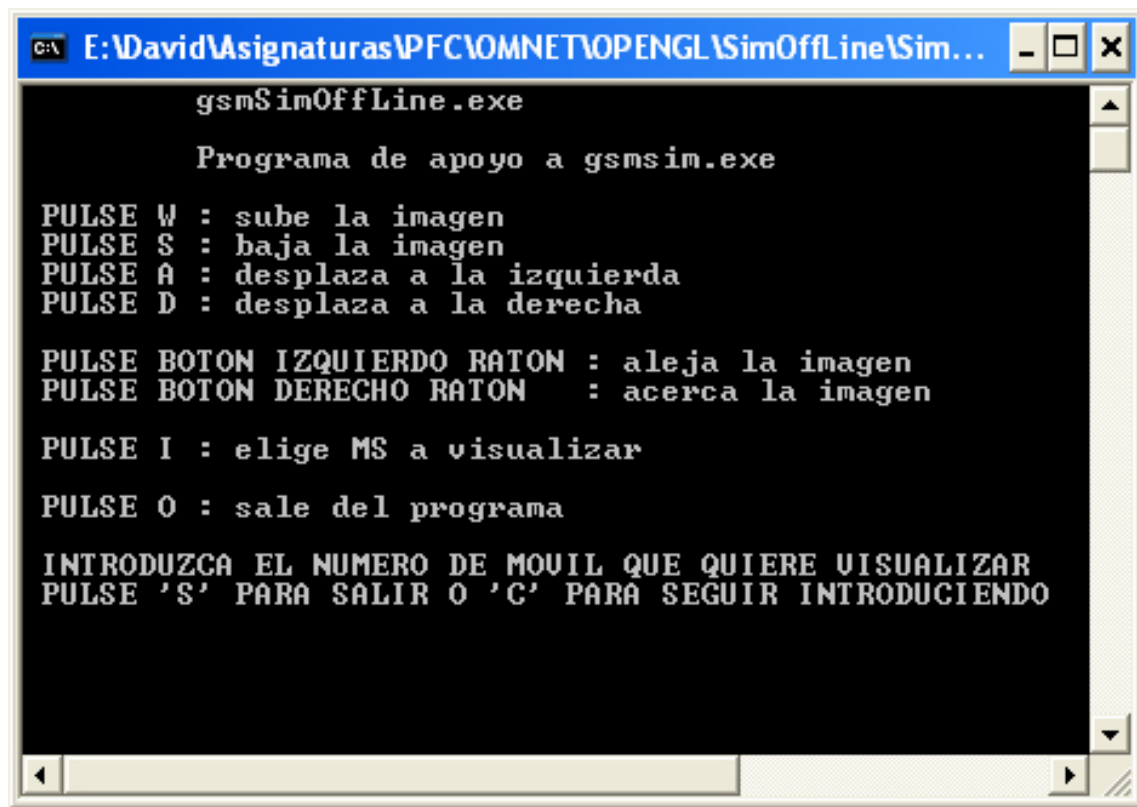
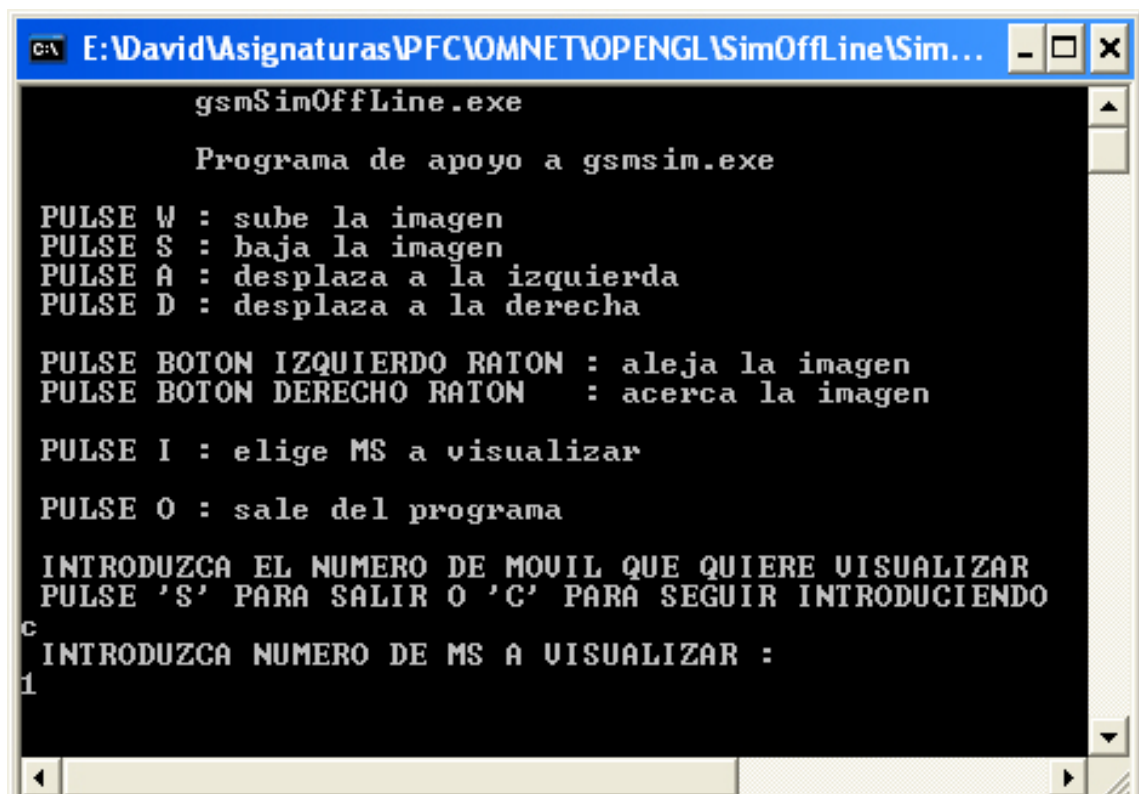


Figura 105 Menú de SimOffLine.exe.

El menú de la aplicación indica las teclas y botones del ratón que producen eventos. Estos eventos modifican el contenido en la imagen.

En el caso de la opción de selección es necesario pulsar la tecla ‘I’. El ejecutable solicita

entonces que se pulse una tecla, 'S' para no introducir ninguna y 'C' para introducir por lo menos una. A continuación solicita la entrada de un número de MS, que deberá estar comprendido entre el '0' y el número máximo de MS representado menos 1 (Figura 106).



```
gsmSimOffLine.exe

Programa de apoyo a gsmSim.exe

PULSE W : sube la imagen
PULSE S : baja la imagen
PULSE A : desplaza a la izquierda
PULSE D : desplaza a la derecha

PULSE BOTON IZQUIERDO RATON : aleja la imagen
PULSE BOTON DERECHO RATON : acerca la imagen

PULSE I : elige MS a visualizar

PULSE 0 : sale del programa

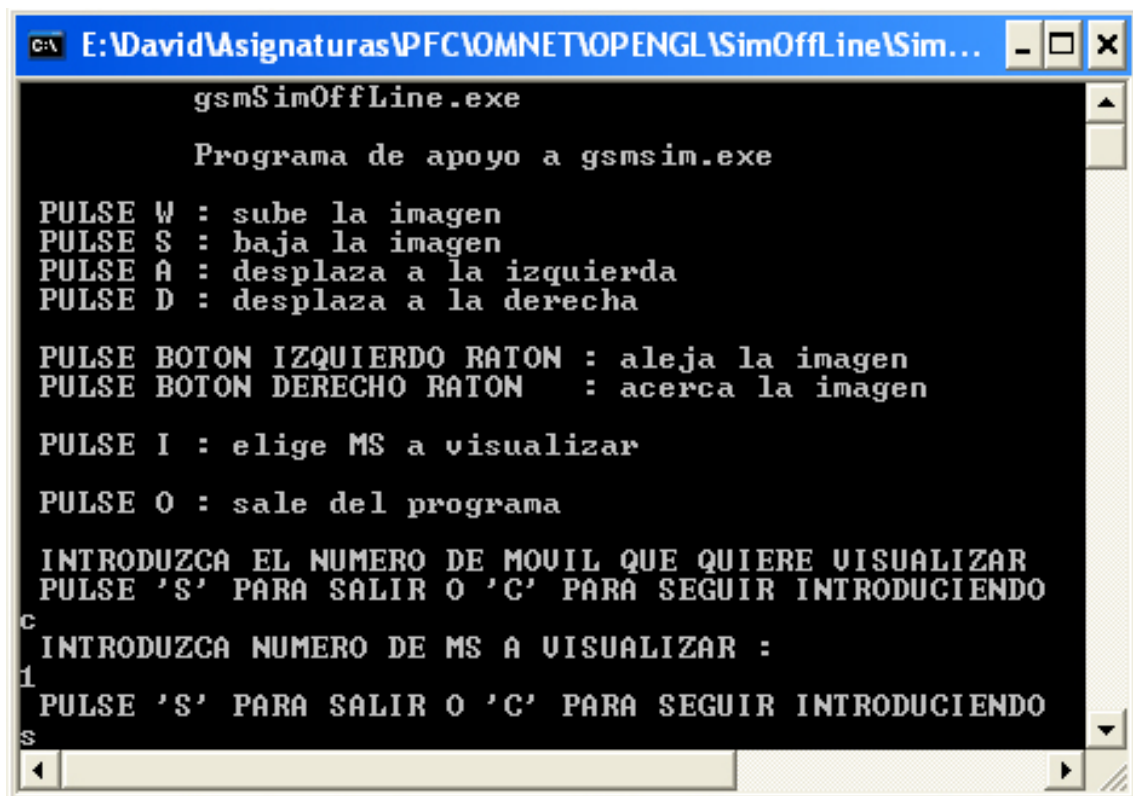
INTRODUZCA EL NUMERO DE MOUIL QUE QUIERE UISUALIZAR
PULSE 'S' PARA SALIR O 'C' PARA SEGUIR INTRODUCIENDO
C
INTRODUZCA NUMERO DE MS A UISUALIZAR :
1
```

Figura 106 Selección de MS en SimOffLine.exe.

Para la selección de una MS determinada, se debe pulsar la tecla 'I'. A continuación solicita la tecla 'C' para continuar con el proceso de introducción, y por último se solicita la estación de la cual se quiere representar la trayectoria.

El siguiente paso es decidir si introducir más estaciones MS o dejar de introducir. Para ello vuelve a solicitar la tecla 'C' para continuar introduciendo o la tecla 'S' para salir, como se ve en la Figura 107. Si se pulsa la tecla 'C' se vuelve a las pantallas de selección mencionadas con anterioridad. Si se pulsa la tecla 'S' representa por pantalla las estaciones que se van a visualizar. Cada línea aporta información de una MS. Si el valor asociado es '0', la MS no será representada. Si el valor es '1' se representará. Ver un ejemplo en la Figura 108. En éste

se ha seleccionado la estación MS número 1. En la pantalla gráfica, después de haber aplicado la selección y el ‘zoom’, se representará la imagen que aparece en la Figura 109. Los círculos representan las áreas de cobertura de una estación base, y las líneas blancas son las trayectorias de movimiento. En esta figura la estación rebota tres veces debido a que tres son las ocasiones en que llega al límite del área definida. Además se ve que pasa por el área de cobertura perteneciente a 10 estaciones base.



```
gsmSimOffLine.exe
Programa de apoyo a gsmSim.exe

PULSE W : sube la imagen
PULSE S : baja la imagen
PULSE A : desplaza a la izquierda
PULSE D : desplaza a la derecha

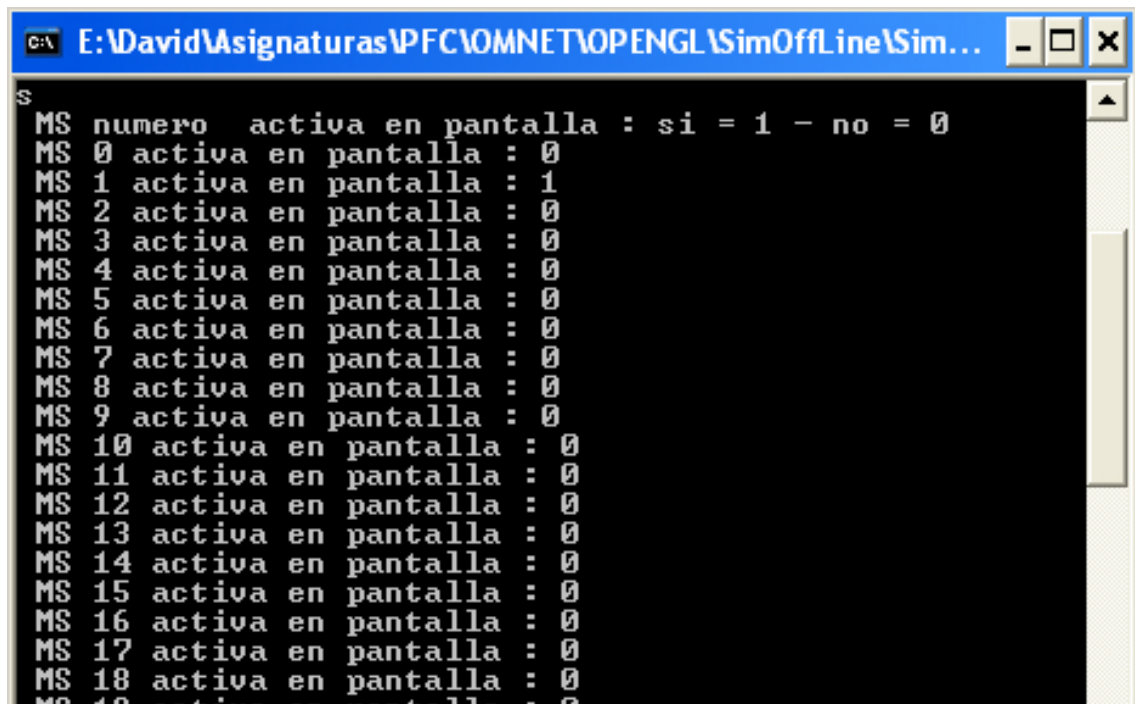
PULSE BOTON IZQUIERDO RATON : aleja la imagen
PULSE BOTON DERECHO RATON : acerca la imagen

PULSE I : elige MS a visualizar
PULSE O : sale del programa

INTRODUZCA EL NUMERO DE MOUIL QUE QUIERE VISUALIZAR
PULSE 'S' PARA SALIR O 'C' PARA SEGUIR INTRODUCIENDO
c
INTRODUZCA NUMERO DE MS A VISUALIZAR :
1
PULSE 'S' PARA SALIR O 'C' PARA SEGUIR INTRODUCIENDO
s
```

Figura 107 Finalización de selección de MS en SimOffLine.exe.

Cuando se termina la selección, la aplicación pregunta si se quieren introducir más, tecla ‘C’, o si se quiere terminar, tecla ‘S’.



```
S
MS numero activa en pantalla : si = 1 - no = 0
MS 0 activa en pantalla : 0
MS 1 activa en pantalla : 1
MS 2 activa en pantalla : 0
MS 3 activa en pantalla : 0
MS 4 activa en pantalla : 0
MS 5 activa en pantalla : 0
MS 6 activa en pantalla : 0
MS 7 activa en pantalla : 0
MS 8 activa en pantalla : 0
MS 9 activa en pantalla : 0
MS 10 activa en pantalla : 0
MS 11 activa en pantalla : 0
MS 12 activa en pantalla : 0
MS 13 activa en pantalla : 0
MS 14 activa en pantalla : 0
MS 15 activa en pantalla : 0
MS 16 activa en pantalla : 0
MS 17 activa en pantalla : 0
MS 18 activa en pantalla : 0
MS 19 activa en pantalla : 0
```

Figura 108 Fin de selección de MS en SimOffLine.exe.

Cuando se finaliza, se presenta en pantalla información de las MS y su trayectoria que se va a representar. Si el valor es 1 es que va a representarse, si el valor es 0 es que no.

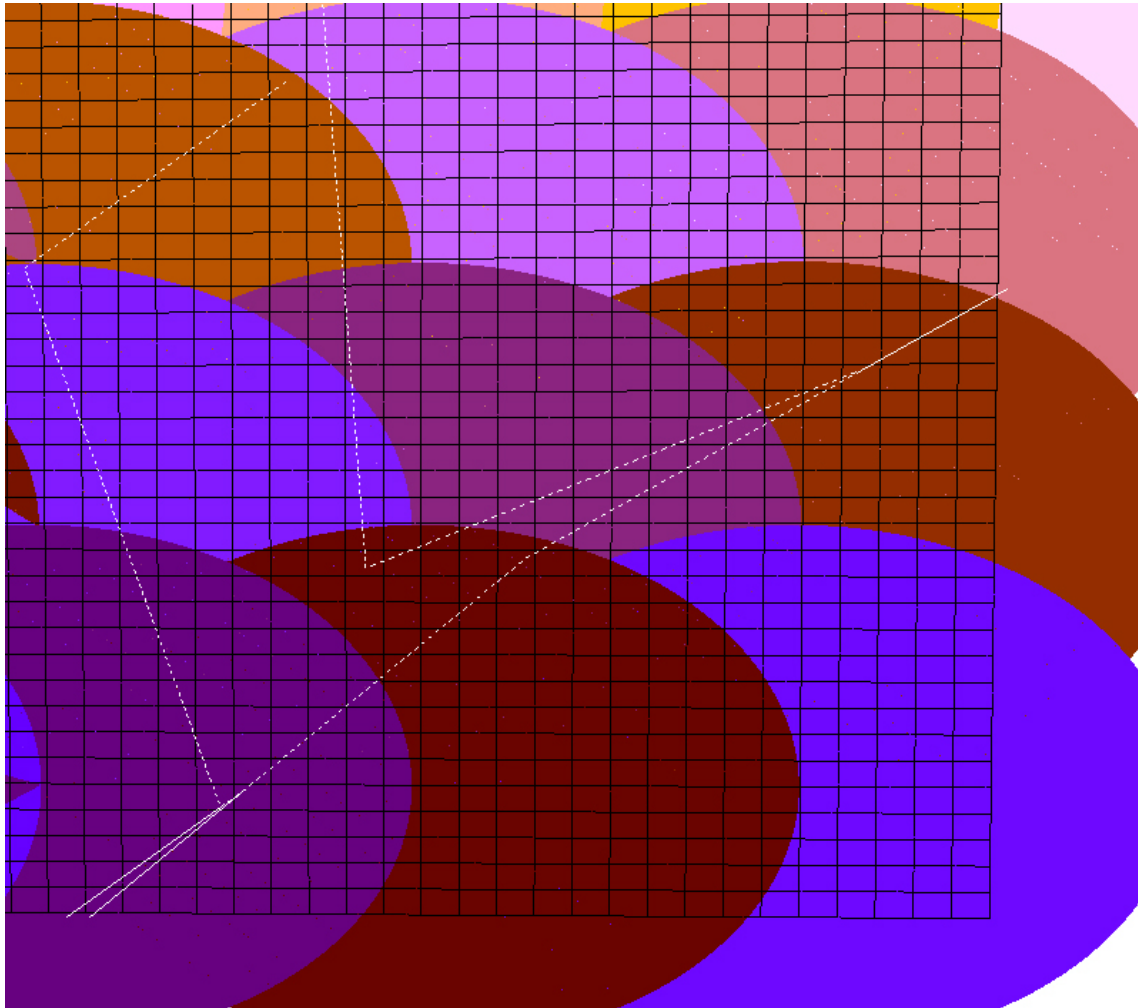


Figura 109 Trayectoria de un móvil.

Las líneas indican la trayectoria que sigue el móvil durante el tiempo de simulación dentro del área definida. Si llega al límite de esta área, el móvil cambia su ángulo sumando al original 180°.

11.3 Funcionamiento OMNeT++ Scalars y OMNeT++ Plove.

Como se ha mencionado en apartados anteriores, las funciones de visualización de gráficas son muy sencillas de usar además de muy útiles a la hora de generar imágenes a partir de los resultados. Aunque Scalars y Plove son programas con una funcionalidad muy parecida, y herramientas prácticamente iguales, aquí se van a reproducir los pasos necesarios para ambas

en la generación de gráficas.

11.3.1 OMNeT++ Scalars.

Este programa permite la visualización de ficheros que contengan datos de tipo escalar. Para abrir uno de estos archivos con extensión *.sca, simplemente hay que hacer doble clic sobre el icono y a continuación aparecerá una pantalla como la de la Figura 110.

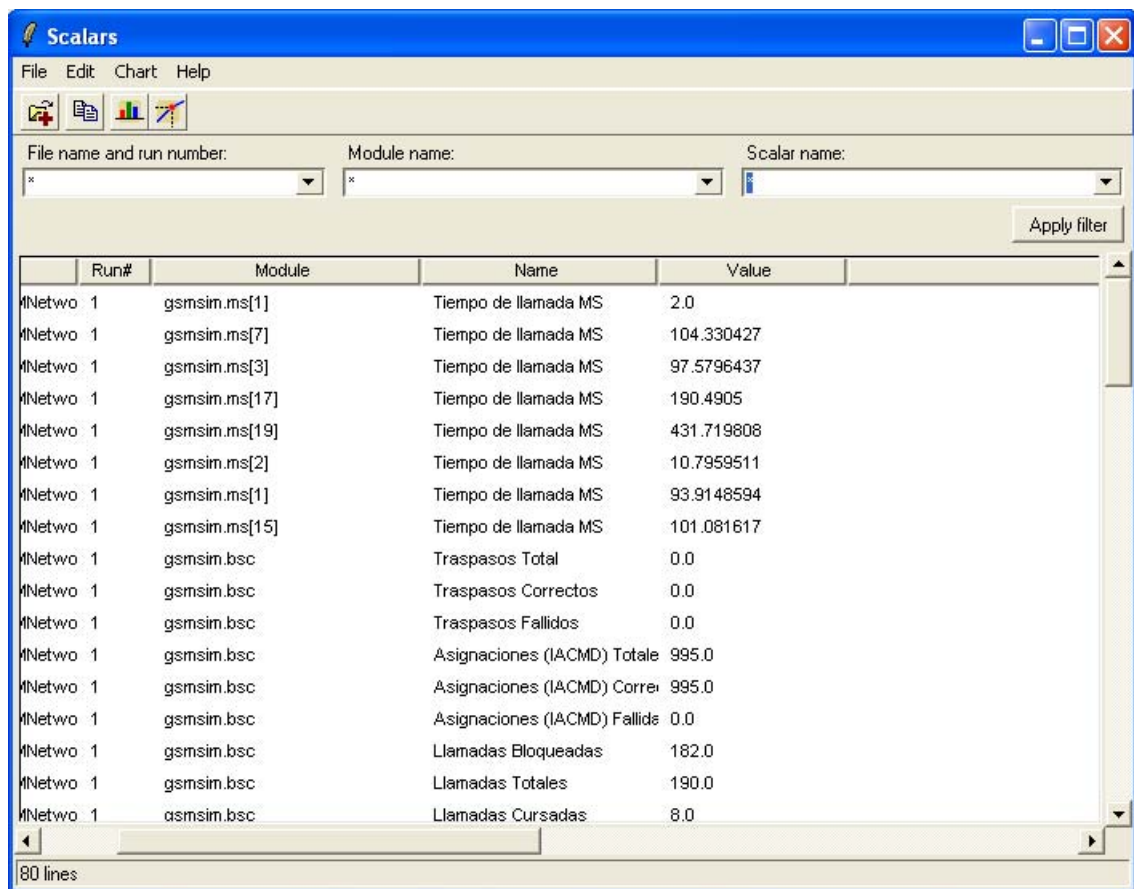


Figura 110 Pantalla de datos de Scalars.

Es la primera pantalla que aparece del programa. Presenta todos los datos escalares que se han almacenado durante la simulación.

Se va a explicar el proceso de obtención de gráficas con un ejemplo: una gráfica del número de llamadas cursadas totales en cada una de las antenas. La simulación se ha realizado con cinco antenas y una BSC.

En primer lugar hay que emplear la herramienta filtro de la aplicación. Para ello se despliega

el cuadro en el que pone “Scalar Name” (Figura 111). Ahora se presentará un menú con diferentes palabras a seleccionar. Se seleccionará Llamadas Cursadas y se cerrará el menú (Figura 112). A continuación se pulsará el botón de nombre “Apply Filter” en la esquina superior derecha de la ventana, y se aplicarán las selecciones realizadas. Los otros cuadros sirven para seleccionar ejecución o un módulo de todos los creados en la red.



Figura 111 Cuadros de selección de filtrado.

Los bloques centrales permiten la selección de 1 tipo de filtrado que se pretende realizar. El botón de la esquina inferior derecha aplica el filtro. En la parte superior izquierda está el icono que permite la salida de gráficas.

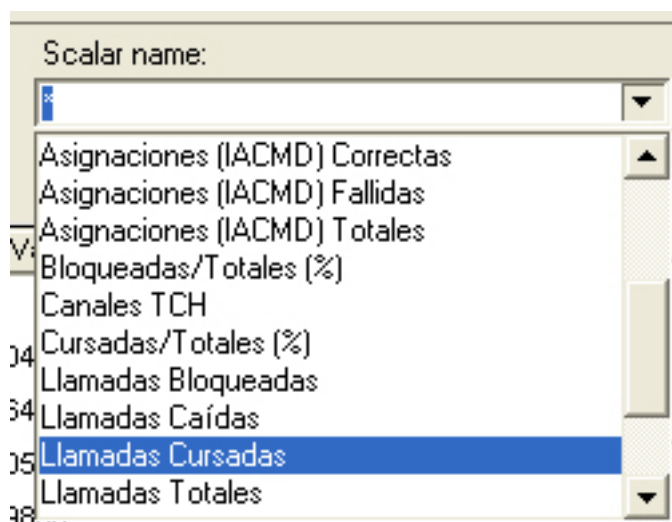


Figura 112 Menú desplegable de filtrado.

El filtro por nombre escalar realiza la selección de parámetros en función del nombre que haya establecido el usuario.

El último paso es el de generar la gráfica. Para ello pulsar el icono que representa unas barras, y saldrá un menú de diálogo en el que se solicitará un nombre para la gráfica. Se pondrá el nombre de “LLAMADAS CURSADAS” (Figura 113). Se puede elegir el tipo de dato generado en el eje X entre módulos o número de ejecución. En este caso se elegirá el segundo para que cada módulo aparezca de un color. Se pueden cambiar las fuentes de las gráficas, o

incluso también añadir nombres a los ejes. Sólo basta con pulsar con el botón derecho del ratón sobre la gráfica (Figura 114).

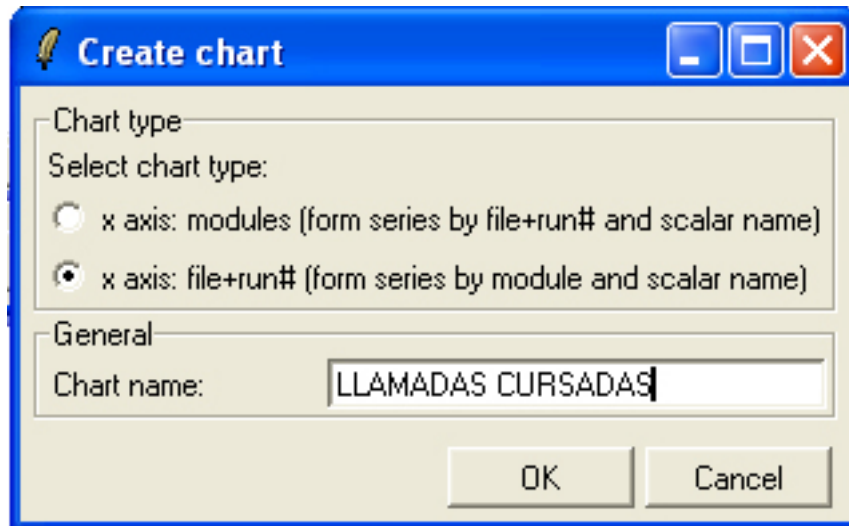


Figura 113 Cuadro de diálogo. Generación de gráficas.

Cuadro que aparece durante la generación de gráficas. Permite elegir entre representación por módulos o por fichero y ejecución. También permite la escritura de un nombre para la gráfica.

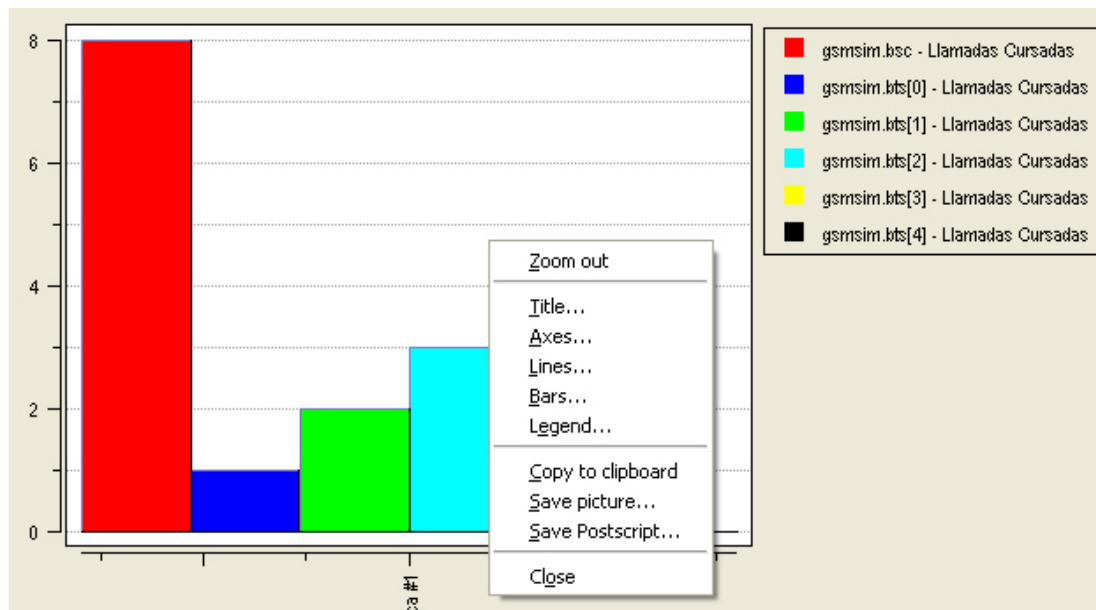


Figura 114 Menú desplegable para opciones de gráfica.

Con la pulsación del botón derecho del teclado sobre la gráfica se puede acceder al menú de modificación de la gráfica. Se pueden modificar la características de título, ejes, líneas, barras y leyenda. Incluso se puede guardar a archivo desde este menú.

La herramienta de zoom consiste en pinchar en una parte de la imagen y seleccionar el área de la imagen que se quiere visualizar. El último paso consiste en la generación de ficheros de imagen con extensión *.gif. o *.ps. Para ello hay que pulsar en los iconos de la barra superior que representan un disquete (Figura 115). El que contiene unas líneas da como salida archivos en modo imagen *.gif, y el que contiene las letras PS da como salida ficheros Postscript.



Figura 115 Iconos para guardar.

Los iconos para guardar archivos de salida vienen representados por la imagen de un disquete. El de la izquierda permite guardar en modo imagen y el de la derecha en formato postscript.

Durante el uso de esta aplicación se ha encontrado un problema con la herramienta de obtención de ficheros gif. Este fallo consiste en que sólo se pueden generar ficheros de salida en la ruta de trabajo, y con el nombre de graph.gif. Probablemente sea un problema de la distribución con la que se ha trabajado para este proyecto, y en sucesivas distribuciones se vea resuelto.

11.3.2 OMNeT++ Plove.

Este programa se emplea para la visualización de vectores. La forma de utilizarlo es similar al programa Scalars. La pantalla inicial presenta los datos grabados a la izquierda y los datos a visualizar en la derecha (Figura 116). Para seleccionar datos para la visualización se deben marcar en la pantalla izquierda, y pulsar la tecla con una flecha en dirección a la derecha. Con esto se añaden valores. Para eliminarlos basta utilizar la flecha de sentido contrario. Para visualizar todos los datos presentes en el cuadro de la derecha hay que seleccionarlos, quedando marcados de azul (Figura 117). Si no se seleccionan, aunque estén en el cuadro de visualización, éstos datos no se representan.

Para crear la gráfica se debe pulsar sobre el botón de gráficas representado por un icono con líneas y puntos rojos en la parte derecha de la barra de herramientas (Figura 118). El resultado

es una gráfica que representa los datos donde el eje X representa el tiempo de simulación.

Este programa tiene las mismas herramientas que en el programa anterior para modificar la leyenda, barras, etc. Pulsando el botón derecho del ratón se despliega un menú exactamente igual que con el programa para escalares. Los ficheros de salida pueden ser de imagen GIF o de Postscript (PS).

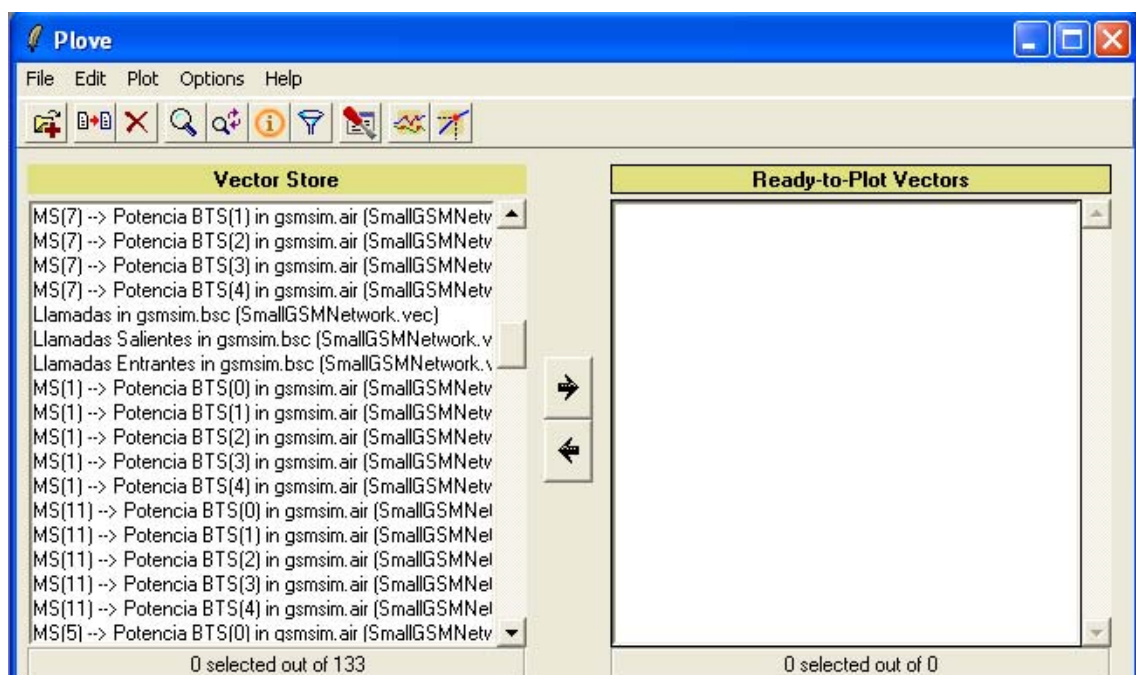


Figura 116 Pantalla OMNeT++ Plove.

Imagen de inicio de OMNeT++ Plove. En el cuadro de la izquierda aparecen los parámetros que se han almacenado durante la simulación. En el cuadro de la derecha se deben añadir los parámetros a visualizar.

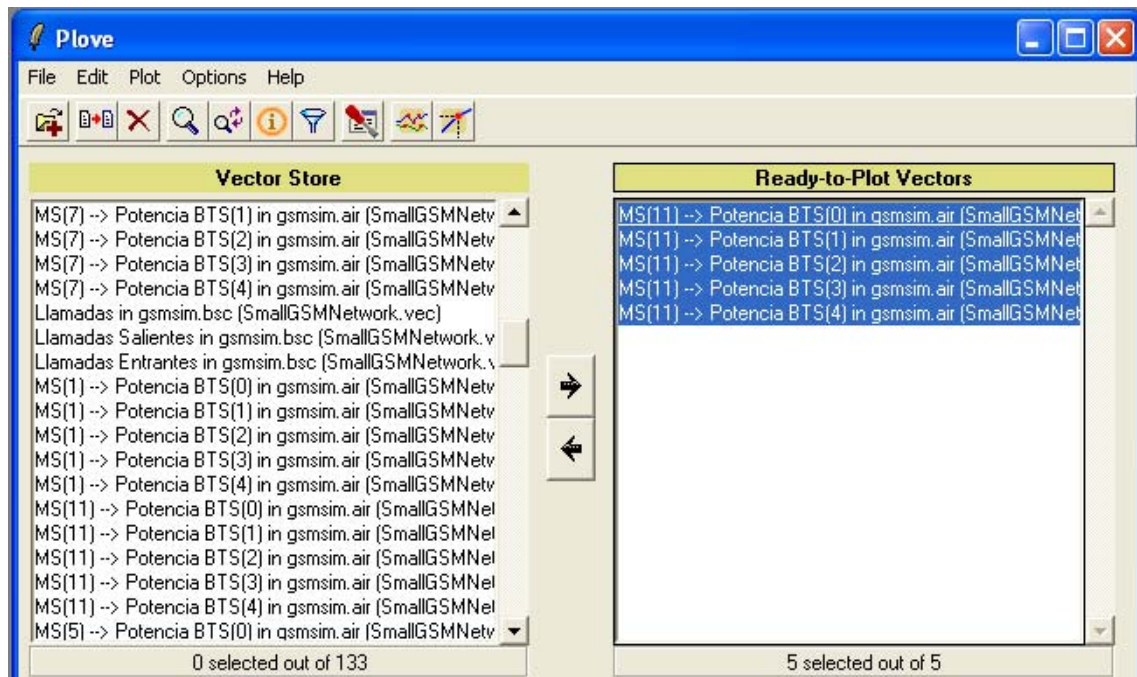


Figura 117 Selección de datos.

Los datos seleccionados se pasan al cuadro de la derecha. En este caso los datos de la MS(11) sobre potencia recibida.



Figura 118 Herramientas Plove.

La herramienta de la derecha representada por el icono con líneas y puntos permite la obtención de gráficas donde el eje X es el eje de tiempo de simulación.

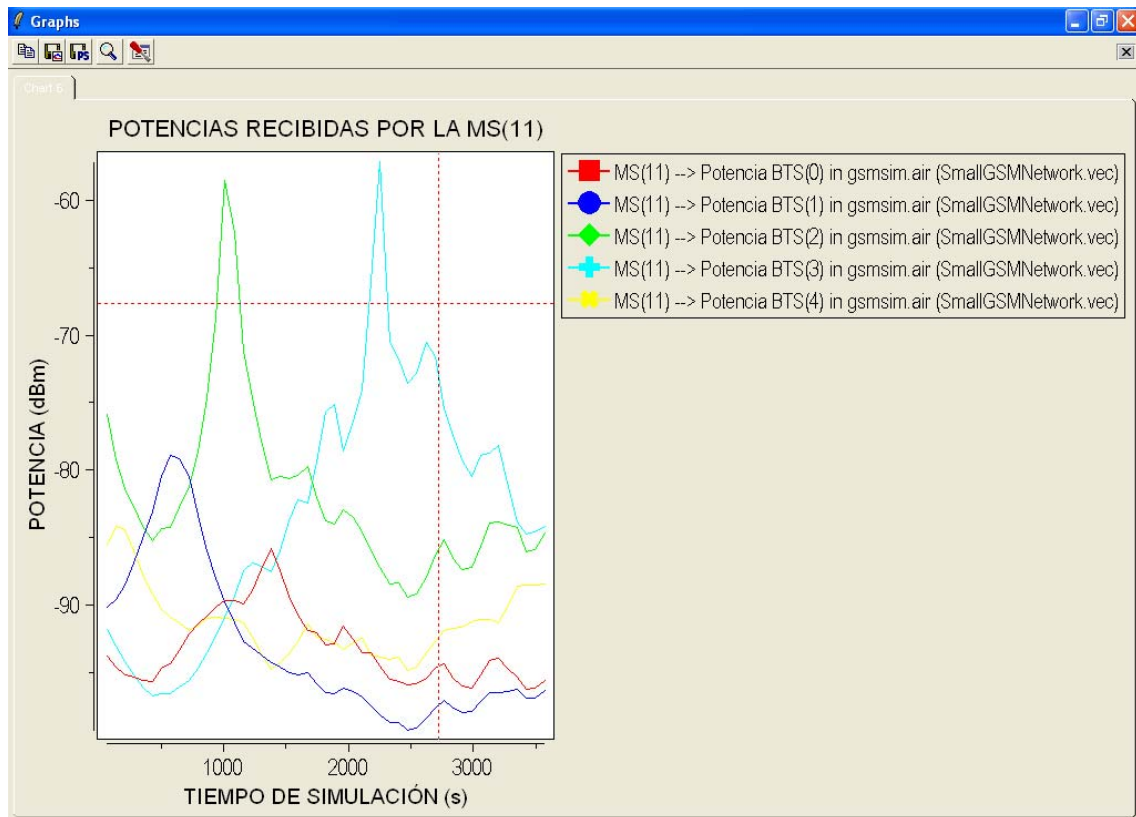


Figura 119 Presentación de datos con Plove.

La imagen presenta las potencias recibidas por la estación móvil número 11. Cada una de las líneas representa la potencia en dBm de una de las BTS. En la simulación se han empleado cinco BTS. Durante la simulación, al principio recibe más potencia de la BTS 2, después de la BTS 1 y la segunda parte de la simulación se encuentra bajo el área de cobertura de la BTS 3.

12 PLIEGO DE CONDICIONES

Para la ejecución de este proyecto se han necesitado herramientas muy accesibles, es decir se han empleado equipos o materiales muy comunes. En cuanto a hardware, se ha necesitado un equipo PC sencillo. Gracias a las características del entorno, no ha sido necesario un equipo muy potente o una tarjeta gráfica de última generación. Es importante que este ordenador disponga de conexión a Internet. En cuanto a software, se ha necesitado un sistema operativo, las librerías del entorno OMNeT++, y un compilador compatible con éstas. Las librerías del entorno tienen la ventaja de ser gratuitas, por lo que no aportan ningún gasto al proyecto. A continuación se enumeran los materiales necesarios:

1. Ordenador PC Portátil con un procesador Intel® Pentium® M 715 a 1,50 GHz con tecnología Intel® Centrino™, una memoria DDR RAM de 512 MB, un disco duro de 40 GB, y tarjeta inalámbrica con tecnología Wireless LAN (802.11b/g).
2. Sistema Operativo Windows XP Home Edition.
3. Microsoft Visual Studio 6.0.
4. Entorno OMNeT++ versión 3.2.

5. Microsoft Office 2003, para la redacción de documentos y realización de transparencias.
6. Microsoft Visio 2003, para la realización de dibujos.
7. Microsoft Project 2003, para la planificación de las tareas.
8. Adobe Acrobat Profesional 7.0, para pasar los documentos a formato PDF.
9. Dispositivo de memoria USB 2.0 de 1GB, como elemento de memoria auxiliar.

13 COSTES Y PRESUPUESTO

La mayor parte de los costes del proyecto se deben a la mano de obra. El coste de material está en torno a los 2000 € gracias al tipo de herramientas que se emplea. Al no necesitar herramientas muy específicas, el coste del equipo y del software, y por tanto el coste total de material no es muy alto.

13.1 *Mano de obra*

CATEGORÍA	TIPO DE TRABAJO	HORAS ESTIMADAS	PRECIO/HORA (€)	TOTAL (€)
Ingeniero Técnico de Telecomunicaciones	Programación de la aplicación	300	60.10	18030
	Documentación	180	12.02	2163.6
			SUMA	20193.6

Tabla 30 Coste de mano de obra.

13.2 *Material*

TIPO DE MATERIAL	UNIDADES	PRECIO/UNIDAD (€)	TOTAL (€)
Ordenador PC Portátil	1	1100.0	1100.0
Dispositivo de memoria USB 2.0 de 1GB	1	35.0	35.0
Microsoft Windows XP Home Edition	1	150.0	150.0
Entorno OMNeT++	1	0	0
Microsoft Office 2003 (incluye Visio y Project)	1	160.0	160.0
Adobe Acrobat 7.0	1	404.84	404.84
Encuadernación e impresión	3	200.0	200.0
Material de oficina (folios, bolígrafos, etc.)			20.0
		SUMA	2069.84

Tabla 31 Coste de material

13.3 Costes totales

CONCEPTO	TOTAL (€)
Mano de obra	20193.6
Material	2163.6
SUMA	22263.44

Tabla 32 Costes totales

13.4 Presupuesto

CONCEPTO	TOTAL (€)
Precio Base	22263.44
IVA (16%)	3562.1504
SUMA	25825.5904

Tabla 33 Presupuesto

El presupuesto total del proyecto asciende a la cantidad de *veinticinco mil ochocientos veinticinco Euros con cincuenta y nueve céntimos de euros*, 25825, 5904 €.

Madrid a 19 de Mayo de 2006.

Fdo.
Ingeniero Técnico de Telecomunicación

14 CONCLUSIONES Y TRABAJOS FUTUROS

Después de trabajar con OMNET++, se puede destacar que es un entorno muy útil para la implementación de cualquier aplicación de simulación. Es muy flexible, por lo que se pueden añadir tantas características como el desarrollador sea capaz de imaginar. Es un entorno también muy útil para los docentes, ya que estos simuladores se pueden aplicar en las prácticas de laboratorio incluidas en los planes de estudio de las asignaturas. La sencillez de desarrollo que caracteriza a OMNET++ permite que no sean necesarios unos conocimientos muy profundos en programación, sino unos conocimientos básicos del lenguaje C++ orientado a objetos.

La realización de un proyecto basado en OMNET++ para la simulación de una red GSM es muy interesante a la hora de poder aplicarlo a laboratorios. La gran facilidad de manejo de su entorno, y la sencilla interfaz con el usuario permite que pueda ser aplicado sin ningún problema en los laboratorios de la Escuela Politécnica. Además no exige una instalación muy complicada, y tampoco unos requisitos muy restrictivos en cuanto a características de ordenador por lo que puede ser aplicado a la mayoría de los ordenadores que se tienen actualmente en la escuela o en la casa de cualquier usuario.

Gracias a la facilidad de desarrollo y a las características de la aplicación este proyecto no es un proyecto cerrado. Se pueden aplicar nuevas mejoras para aportar nuevas capacidades al entorno, o incluso ampliar el número de equipos en la red. Con este proyecto la aplicación simula hasta los equipos MSC y VLR. En el desarrollo del código fuente se ha seguido en todo momento un desarrollo modular para permitir una modificación sencilla a la hora de aplicar nuevas ideas al proyecto. En cierto modo este proyecto puede ser considerado como una primera fase en la realización de una aplicación de simulación de una red GSM que incluya todos los equipos presentes en esta red junto con los elementos de conexión con otras redes.

Se pueden tener varios trabajos futuros a partir de este proyecto, que se pueden clasificar en mejoras de la aplicación realizada en este proyecto y mejora y creación de aplicaciones de apoyo. El primer bloque corresponde a aquellas mejoras que se pueden realizar directamente sobre la aplicación de simulación. Se han pensado tres trabajos futuros. El primero consiste en la modificación y añadido de trayectorias y módulos de movilidad. La segunda mejora consiste en añadir nuevos modelos de propagación que incluyan la presencia de obstáculos, y la tercera es la mejora y ampliación de módulos y mensajes.

En cuanto al segundo bloque referente a las aplicaciones de apoyo, se puede mejorar el programa de visualización de la posición de las estaciones y de los móviles. El programa desarrollado en este proyecto como una mejora a la aplicación principal, puede ser mejorado para dar nuevas características gracias a que se ha implementado con unas librerías gráficas muy potentes que permiten un amplio abanico de posibilidades. El otro trabajo futuro perteneciente a este bloque es la realización de un programa de inicialización para el fichero de configuración de la simulación.

Por último, se puede hablar de otro trabajo futuro a parte de los anteriores, aunque se saldría ya de la red GSM. Este trabajo futuro consistiría en aplicar los conocimientos adquiridos en este proyecto para desarrollar un simulador para UMTS (Universal Mobile Telecommunications System), o telefonía móvil de tercera generación.

14.1 *Trabajos futuros*

14.1.1 Trayectorias. Módulos de movilidad.

La generación de movilidad es una parte importante de la aplicación. En el código desarrollado, se establece un módulo y un ángulo para un vector velocidad y una posición en

el espacio en función de dos coordenadas X e Y en un plano bidimensional. Como se ha explicado en las características de este simulador, se han desarrollado dos tipos de movimientos, un movimiento lineal y un movimiento aleatorio. El movimiento lineal consiste en que se mantiene en todo momento el módulo y el ángulo de la velocidad hasta que se toca el borde del área de simulación, momento en que se invierte el ángulo. El movimiento aleatorio se basa en establecer un módulo y un ángulo con un tiempo de vida aleatorio entre 1 y 200 segundos. Cuando finaliza el tiempo de vida o se llega al límite del área el ángulo se invierte.

La aplicación de nuevos modelos de movilidad puede aplicar movimientos más reales, como el modelo Manhattan. Este modelo se aplica a las grandes ciudades, en las cuales los usuarios siguen un movimiento en función de la disposición de las calles de la ciudad como puede verse en la Figura 120 .

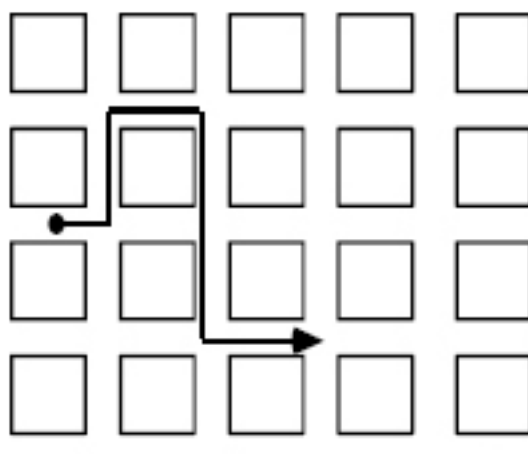


Figura 120 Modelo de movimiento Manhattan

Otro modelos de movimiento que se puede aplicar es el movimiento de vehículos en una carretera, que siguen velocidades altas y pocos cambios de dirección, o el movimiento en el interior de edificios, donde el movimiento se ve limitado por las paredes.

14.1.2 Modelos de propagación y obstáculos

En el desarrollo de la aplicación se ha tenido en cuenta únicamente el espacio libre, sin incluir posibles obstáculos, ni modelos de propagación que incluyan estos obstáculos. Los obstáculos

hacen que la cobertura de las estaciones se reduzca y además hace que las pérdidas de transmisión sean mayores. La introducción de nuevos modelos es muy sencilla, durante la implementación se decidió utilizar funciones independientes del resto de los módulos para el cálculo de potencias. Estas funciones reciben como entrada unos valores de posición y dan a la salida la potencia recibida restando las pérdidas en el camino radioeléctrico. Para dar nuevos modelos en la simulación simplemente hay que modificar estas funciones.

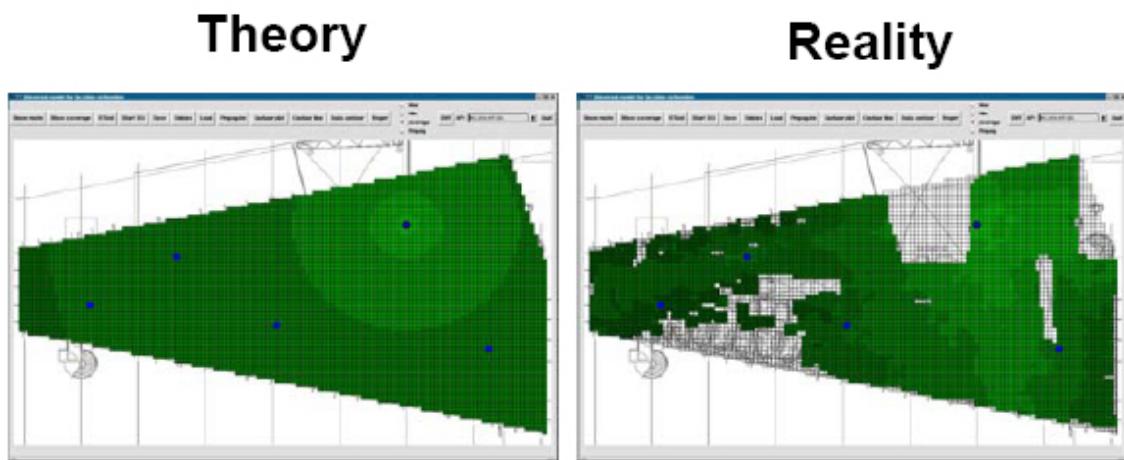


Figura 121 Cobertura de estaciones base, de la teoría a la práctica.

Como se puede ver en la figura, la cobertura supuesta en la teoría difiere enormemente de la cobertura real. Se debe a la presencia de obstáculos tales como montañas, edificios, etc.

Para añadir obstáculos tal vez sería interesante definir ficheros de texto que contengan la descripción del área a simular, con posibles edificios, montañas, túneles, etc., y que estos datos fuesen leídos por las funciones de cálculo de potencia a la hora de ver la potencia que recibe una estación móvil. Tal vez con esta solución se llegaría a un problema de eficiencia y rendimiento de la aplicación, ya que el programa debería abrir, leer, y cerrar un archivo cada vez que se llamase a estas funciones.

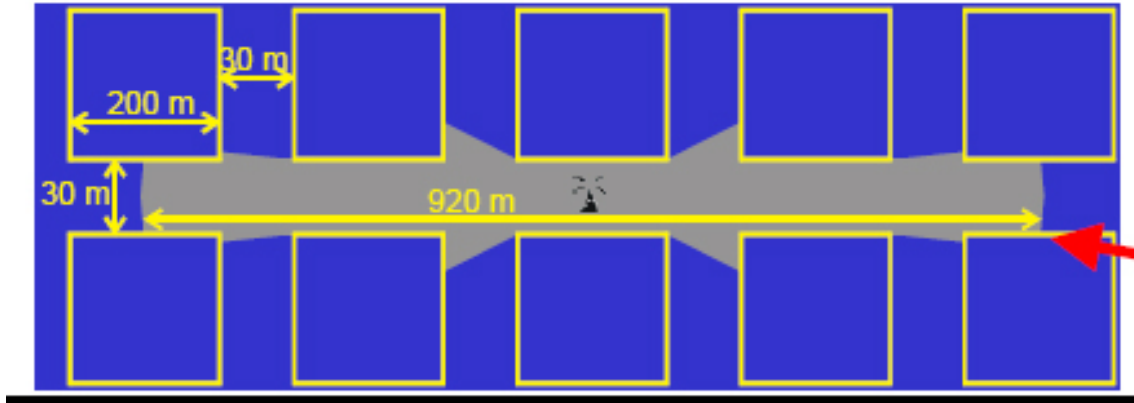


Figura 122 Cobertura de microceldas en una ciudad.

En las ciudades los edificios actúan como obstáculos para la cobertura de las estaciones base. En algunos casos como en el de la figura, la antena sólo irradia potencia al área perteneciente a una calle y con una longitud de 920 de metros con 30 de ancho. En las ciudades grandes se pueden establecer redes de antenas que den cobertura a determinadas calles.

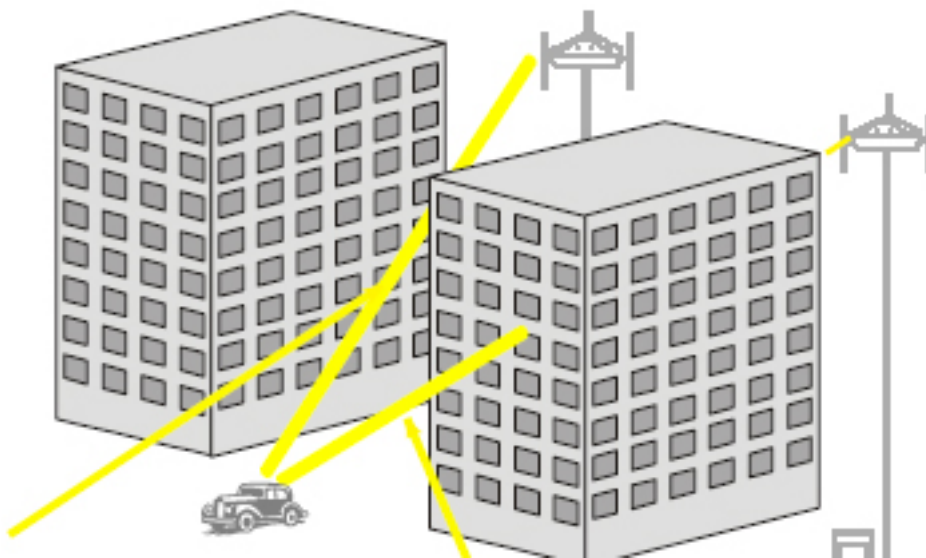


Figura 123 Los edificios actúan como obstáculos.

Los edificios interfieren en la comunicación entre una antena y un dispositivo móvil. Esto es muy importante ya que puede dar lugar a cortes en la comunicación, por lo que se degradaría la calidad del servicio ofertado por un operador.

14.1.3 Mensajes y módulos

Otra de los futuros trabajos que se proponen es la ampliación del número de módulos. En este proyecto se abarca la simulación de los equipos que van desde la estación móvil hasta el equipo VLR. Una siguiente fase se puede encargar de desarrollar los equipos HLR, el equipo de autenticación AuC, el equipo de gestión de mensajes SM-SC, el equipo de identificación de terminales EIR o las conexiones con redes externas a la red GSM. También se puede ampliar el número de mensajes empleado en la simulación, ya que el presente proyecto no utiliza todos los mensajes que se emplean en una red GSM, sino algunos de los más relevantes. En cuanto a la característica de los mensajes en la simulación tal vez sería interesante el desarrollo de nuevos tipos de mensajes independientes de la clase `cMessage` del entorno. Esto permitiría añadir nuevos parámetros y estructuras sin que el rendimiento de la aplicación se viese mermado por el uso de la clase `cPar` para añadir parámetros. Además se puede crear una estructura jerárquica de tipos de mensajes en cuanto a protocolos y niveles del modelo basándose en la característica de derivación, jerarquía y polimorfismo que aporta el lenguaje orientado a objetos C++.

14.1.4 Programa de inicialización

La configuración de simulaciones con OMNET++ se basa en la creación de un fichero de inicialización con la extensión `*.ini`. Para este simulador, la configuración de este archivo puede resultar una tarea muy tediosa cuando se trabajan con redes con un gran número de módulos. Para reducir este trabajo la solución planteada es la creación de un nuevo programa de apoyo que cree este fichero automáticamente.

Este programa podría tener una presentación en ventanas, y apoyándose en las librerías de `OPENGL` o cualquier otra librería gráfica, podría introducir una interfaz gráfica en la que el usuario tuviese la posibilidad de introducir nuevos módulos y moverlos de posición simplemente con arrastrar el ratón. Otra opción que se podría usar es introducir los valores mediante pestañas o cuadros de diálogo. El resultado sería un fichero `.ini` que permitiese al usuario acudir directamente al programa simulador sin tener que abrir en modo texto este fichero y escribir uno a uno todos los parámetros necesarios.

14.1.5 Programa de visualización de posiciones

En este proyecto se ha desarrollado una mejora basada en la creación de una aplicación que a partir de unos ficheros dibuje el movimiento de la estación. Estos ficheros son generados por

el simulador y contienen la información de posición de una estación en un instante determinado. Además contienen los radios de cobertura y las posiciones de las antenas de forma que trayectorias y coberturas pueden ser reproducidos en un mismo dibujo. Se ha desarrollado con la librería OpenGL, por lo que las mejoras que se pueden aplicar a esta aplicación de apoyo son muy variadas.

En primer lugar, sólo se realiza un dibujo en dos dimensiones, viendo el área de simulación desde la vertical. Si se introdujese una tercera dimensión y se añadiesen funciones de dibujo de coberturas se podrían hacer dibujos como el de la Figura 124. En segundo lugar, se pueden cambiar las características de presentación, haciéndolas más atractivas o más interesantes para el usuario, con zonas de coberturas transparentes, etc.

En cuanto a las opciones de visualización, el ejecutable permite la selección de determinadas estaciones móviles, suprimiendo la representación de las que no se crean necesarias. Esta interfaz se ha desarrollado en línea de órdenes. El paso de esta funcionalidad y de toda la aplicación a un programa basado en ventanas sería una mejora interesante.

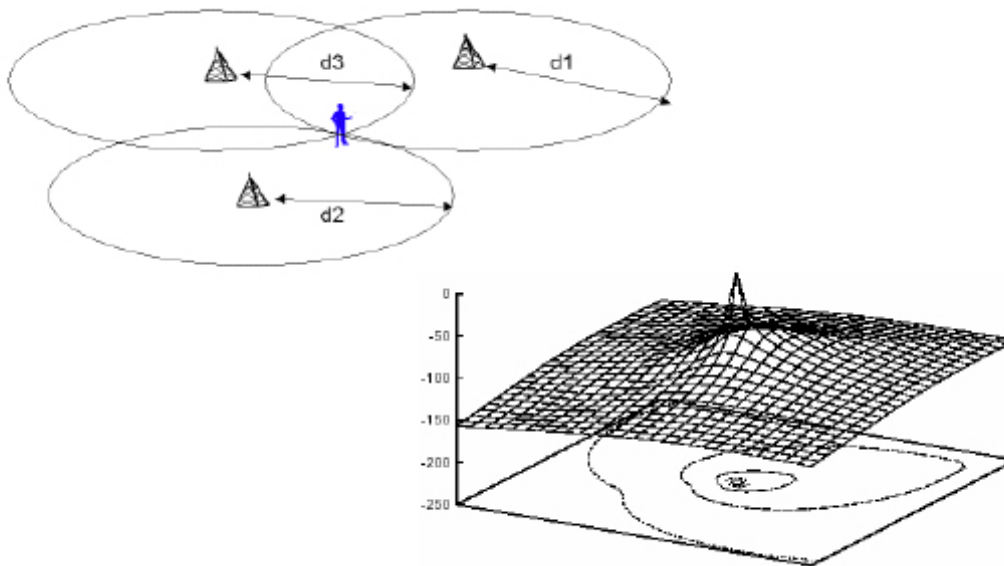


Figura 124 Potencia transmitida por una antena en 3D.

Una de las posibles mejoras de la aplicación de ayuda consistiría en la visualización de gráficas en tres coordenadas. Actualmente este programa sólo permite visualizar dibujos en 2 dimensiones, como si se vieses desde la vertical. Para ello habría que cambiar las funciones de dibujado en este programa, añadiendo funciones que permitan la creación de gráficas atendiendo a los modelos de propagación usados en el simulador.

14.1.6 Realización de un simulador de UMTS.

UMTS es la denominada telefonía móvil de tercera generación o 3G, y se prevee que esta tecnología vaya creciendo en usuarios durante los próximos años y sustituya poco a poco a la tecnología GSM. Una de sus principales cualidades es que aumenta la velocidad de transmisión de datos, llegando a los 2Mbit/s frente a los 9600bps que es capaz de ofrecer GSM. Gracias a que UMTS es una tecnología muy similar a GSM, se puede aprovechar el desarrollo realizado en este proyecto y modificarlo adaptandolo a las exigencias de UMTS para dar un simulador aplicado a esta nueva tecnología.

15 BIBLIOGRAFÍA

La bibliografía se presenta estructurada en cinco apartados en función del contenido y de la parte para la que han sido consultados. Estos apartados son OMNET++, red GSM, estándares GSM, Programación C++ y páginas web.

15.1 OMNET ++:

- [1] C. Mallandal, A. Suri, V. Kunchakarra, S.S. Iyengar, R. Kannan, A. Durrezi, “Simulating Wireless Sensor Networks with OMNeT++” Sensor Network Research Group, Department of Computer Science, Louisiana State University – University of Akron, Akron, Ohio. 24 de Enero de 2005. *version 1.0*
- [2] OMNET++ Community, “OMNeT++ 3.2 API Reference” OMNET ++ Community Site, 2005. Versión 3.2.
- [3] OMNET++ Community, “Simulation with OMNeT++”
- [4] OMNET++ Community Site, “Tic-Toc Tutorial for OMNET ++” OMNET++ Community Site, 2005. Versión 3.2
- [5] Varga, András. “OMNeT++ Discrete Event Simulation System. Version 3.1 User Manual”. 29 Marzo 2005. Páginas: 247.

15.2 Red GSM

- [6] Boggie, G., Camarda, P., D'Alconzo, A., De Biasi, A., Siviero, M. "Drop Call Probability in Established Cellular Networks: from data Analysis to Modelling" IEEE, Politecnico di Bari, Italia 2005. Páginas: 5.
- [7] Fourneau, J. M., Kloul, L., Valois, F. "Performance Modelling of Hierarchical Networks using PEPA" Laboratoire PRiSM, Université de Versailles, Francia, 2000. Páginas: 24.
- [8] Giambene, Giovanni. "DCS1800 & UMTS mobility analysis". Dipartimento di Ingegneria dell'Informazione, University of Siena, Italia. Páginas: 7.
- [9] Herradón Díez, Rafael, Jiménez Muñoz, Florentino. "Caracterización estadística de la propagación para comunicaciones móviles en el interior de fábricas". Departamento de Ingeniería Audiovisual y Comunicaciones, EUIT de Telecomunicaciones, Universidad Politécnica de Madrid, 2000. Páginas: 9.
- [10] Hernando Rábanos, J. M. "Comunicaciones Móviles GSM". Fundación Airtel. 1ª edición. Páginas: 751. ISBN: 8493029823.
- [11] de Miguel Ambite, Enrique. "Proyecto Fin de Carrera Simulador 2Gsim." Escuela Politécnica. Universidad de Alcalá de Henares. 2001.
- [12] Myllymäki, Petri, Tirri, Henry. "Graphical models on Manhattan. A probabilistic approach to mobile device positioning" Complex Systems Computation Group, University of Helsinki & Helsinki Institute for Information Technology, Finlandia. Ver http://cosco.hiit.fi/Teaching/GraphicalModels/Fall2003/material/gm_in_positioning_2up.pdf
- [13] Redl, Siegmund M. "An introduction to GSM". Artech House, Boston London, 1995. Páginas: 379. ISBN: 0890067856.
- [14] Redl, Siegmund M. "GSM and personal communications handbook". Artech House, Boston, 1998. Páginas: 526. ISBN: 0890069573.
- [15] Winter, Thomas; Correia, Luis M.; Fledderus, Eric R.; Meijerink, Ellen; Perera, Ranjit; Serrador, Antonio; Türke, Ulrich; Van Uiter, Miranda J. G. "Identification of relevant parameters for traffic modelling and interference estimation" IST, 2001. Ver <http://momentum.zib.de/paper/momentum-d21.pdf>

15.3 Estándares GSM de ETSI:

- [16] GSM 01.02 General description of a GSM Public Land Mobile Network (PLMN).
- [17] GSM 03.02 Network architecture.
- [18] GSM 03.30 Radio network planning aspects.
- [19] GSM 04.01 Mobile Station – Base Station System (MS – BSS) interface. General aspects and principles.
- [20] GSM 04.02 GSM Public Land Mobile Network access reference configuration
- [21] GSM 04.03 Mobile Station – Base Station Subsystem (MS – BSS) interface; Channel structures and access capabilities.
- [22] GSM 04.04 Layer 1 General requirements.

- [23] GSM 04.05 Data Link layer General aspects.
- [24] GSM 04.06 Mobile Station – Base Station Subsystem (MS – BSS) interface Data Link layer specification.
- [25] GSM 04.07 Mobile radio interface signalling layer 3; General aspects.
- [26] GSM 04.08 Mobile radio interface layer 3 specification.
- [27] GSM 04.13 Performance requirements in the mobile radio interface.
- [28] GSM 05.01 Physical layer on the radio path; General description.
- [29] GSM 05.05 Radio transmission and reception.
- [30] GSM 05.22 Radio link management in hierarchical networks.
- [31] GSM 08.01 Base Station System – Mobile services Switching Centre (BSS – MSC) interface; General aspects.
- [32] GSM 08.02 Base Station System – Mobile services Switching Centre (BSS – MSC) interface; Interface principles.
- [33] GSM 08.04 Base Station System – Mobile services Switching Centre (BSS – MSC) interface; Layer 3 specification.
- [34] GSM 08.08 Mobile services Switching Centre (BSS – MSC) interface; Layer 3 specification.
- [35] GSM 08.51 Base Station Controller – Base Transceiver Station (BSC – BTS) interface; General aspects.
- [36] GSM 08.52 Base Station Controller – Base Transceiver Station (BSC – BTS) interface; Interface principles.
- [37] GSM 08.54 Base Station Controller – Base Transceiver Station (BSC – BTS) interface; Layer 1 structure of physical circuits.
- [38] GSM 08.58 Base Station Controller – Base Transceiver Station (BSC – BTS) interface; Layer 3 specification.
- [39] GSM 09.02 Mobile application part (MAP) specification.
- [40] GSM 09.10 Information element mapping between Mobile Station – Base Station System (MS – BSS) and Base Station System – Mobile services Switching Centre (BSS – MSC) signalling procedures and the Mobile application part (MAP).

15.4 Programación C++:

- [41] Ceballos Sierra, F.J. “C/C++ curso de programación” Editorial RA-MA 2ª edición, Madrid, 2002. Páginas: 669. ISBN: 8478974806.
- [42] Ceballos Sierra, F.J. “Programación orientada a objetos con C++” Editorial RA-MA 3ª edición, Madrid, 2003. Páginas: 617. ISBN: 8478975705.

15.5 Páginas web:

- [43] Página web comunidad OMNET++. <http://www.omnetpp.org>
- [44] Página web OMNEST. <http://www.omnest.com>

APÉNDICE A. CÓDIGO FUENTE SIMOFFLINE.EXE

APÉNDICE B. CÓDIGO FUENTE DE GSMSIM.EXE

APÉNDICE C. CONTENIDO DEL CD

En el CD que se adjunta con la memoria se añade el código fuente, librerías, y varios documentos sobre el proyecto. También se añaden los manuales de OMNeT++ y OPENGL. El objetivo es que en este CD se encuentren toda la documentación utilizada para el proyecto, para que en caso de que se quiera modificar en un futuro, sea sencillo para un desarrollador encontrar toda la información. El CD se estructura en las carpetas que se comentan a continuación:

- “..**codigo fuente**\\” : contiene los códigos fuentes de las aplicaciones.
 - “..**codigo fuente**\\gsmsim\\español” : código fuente de la aplicación principal en ingles.
 - “..**codigo fuente**\\gsmsim\\español” : código fuente de la aplicación principal en español.
 - “..**codigo fuente**\\simoffline” : código fuente de la aplicación de apoyo.

- “..\documentos\” : contiene los códigos fuentes de las aplicaciones.
 - “..\documentos\memoria\pdf\” : memoria del proyecto en formato pdf.
 - “..\documentos\memoria\doc\” : memoria del proyecto en formato doc.
 - “..\documentos\memoria\doc\imágenes\” : imágenes utilizadas en la memoria.
 - “..\documentos\papers\” : documentos que se han utilizado como apoyo.
- “..\ejecutables\” : contiene los ejecutables de las aplicaciones.
 - “..\ejecutables\gsmsim\español” : ejecutable de la aplicación principal en ingles.
 - “..\ejecutables\gsmsim\español” : ejecutable de la aplicación principal en español.
 - “..\ejecutables\simoffline” : ejecutable de la aplicación de apoyo.
 - “..\estandares GSM\” : contiene los estándares de GSM.
- “..\ejemplos \” : contiene los ejemplos que se comentan en la memoria.
 - “..\ejemplos\ej1\” :
 - “..\ejemplos\ej2\” :
 - “..\ejemplos\ej3\” :
- “..\estandares GSM\” : contiene los estándares de GSM publicados por el ETSI.
- “..\manuales\” : contiene los manuales usados en el desarrollo.
 - “..\manuales\omnet++\” : código fuente de la aplicación principal en ingles.
 - “..\manuales\opengl\” : código fuente de la aplicación principal en español.
- “..\omnet++\” : contiene el programa de instalación de OMNET++ y su código fuente.
- “..\opengl\glut\” : contiene las librerías para desarrollo de programas con OPENGGL.