DISEÑO DE UN EMULADOR RRM PARA REQUERIMIENTOS DE CALIDAD DE SERVICIO (QoS) EN UNA RED UMTS

ANEXOS

OSCAR FERNANDO VELANDIA PEDROZA

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERIA ELECTRONICA Y TELECOMUNICACIONES
DEPARTAMENTO DE TELECOMUNICACIONES
GRUPO I+D EN NUEVAS TECNOLOGIAS EN TELECOMUNICACIONES
POPAYAN
2004

DISEÑO DE UN EMULADOR RRM PARA REQUERIMIENTOS DE CALIDAD DE SERVICIO (QoS) EN UNA RED UMTS

ANEXOS

OSCAR FERNANDO VELANDIA PEDROZA

Trabajo de grado presentado como requisito para obtener el título de Ingeniero en Electrónica y Telecomunicaciones

Director
ALEJANDRO TOLEDO TOVAR
INGENIERO EN ELECTRONICA Y TELECOMUNICACIONES

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERIA ELECTRONICA Y TELECOMUNICACIONES
DEPARTAMENTO DE TELECOMUNICACIONES
GRUPO I+D EN NUEVAS TECNOLOGIAS EN TELECOMUNICACIONES
POPAYAN
2004

ANEXO 1. ATM EN LA RED DE ACCESO: ALL2

1. INTRODUCCIÓN.

La implantación de la red móvil de Tercera Generación supondrá la introducción de nuevos servicios con tráficos de naturaleza heterogénea, cuyos caudales binarios pueden oscilar desde las decenas de kbps hasta los Mbps, y con diferentes niveles de calidad de servicio.

Con el fin de hacer frente a estos nuevos requerimientos, los fabricantes de equipos de redes móviles propusieron la utilización del modo de transferencia asíncrono (ATM, Asynchronous Transfer Mode) en la infraestructura de la red de acceso dada su inherente habilidad de proporcionar calidad de servicio y la capacidad de tratar con tráfico multimedia. A medida que ha ido creciendo la demanda de nuevos tipos de servicios en el ámbito de las comunicaciones celulares, ha crecido la aceptación de este mecanismo de transmisión como la tecnología ideal para la red de transporte de la redes celulares de Tercera Generación.

ATM se ha convertido, pues, en una atractiva alternativa a los sistemas tradicionales de transferencia síncrona (STM, Synchronous Transfer Mode) en que se basan las redes de Segunda Generación.

ATM ofrece un servicio orientado a conexión, lo que implica que antes de que se produzca la transferencia de información es necesario el establecimiento de una conexión virtual con los recursos necesarios. Los tiempos de establecimiento y liberación que maneja son suficientemente cortos, de modo que puede manejar las situaciones de soft handover, asegurando los niveles de calidad de servicio que se requieren. Pero, además, trabaja en modo paquete con paquetes de tamaño relativamente reducido (53 bytes) y fijo, llamados células, garantizándose de este modo

niveles de retardo y jitter pequeños, lo que la convierte en una tecnología idónea para transportar servicios en tiempo real.

Otra de las ventajas que ofrece frente a los sistemas convencionales basados en conmutación de circuitos es la eficiencia con la que se utilizan los recursos de transmisión. Esta eficiencia viene principalmente caracterizada por el concepto de multiplexación estadística. En las redes STM el ancho de banda requerido es simplemente la suma de las tasas fijas de todos los usuarios. Sin embargo, en una red ATM no se hace una reserva estática de ancho de banda si no que se va ajustando a lo largo del tiempo, lo que permite dar servicio a un mayor número de usuarios, haciendo un uso más eficiente de los recursos. Pero esto conlleva la incorporación de un sistema de control de acceso (CAC) con el fin de asegurar el nivel de calidad de servicio solicitado por cada usuario. El ejemplo más claro se encuentra en el servicio vocal. Durante una conversación telefónica existen periodos de silencio en los que no se transmite ninguna información. En los sistemas basados en conmutación de circuitos el canal asignado para cada usuario no sería utilizado durante ese periodo de silencio, utilizando ineficientemente el ancho de banda. En cambio, al utilizar multiplexación estadística, durante un periodo de silencio puede estar transmitiéndose información procedente de otros usuarios.

Por todo ello, el 3GPP ha elegido ATM como el compañero ideal para WCDMA en la red de acceso UMTS.

2. NIVEL DE ADAPTACIÓN ATM TIPO 2: AAL2.

El comité de estandarización para UMTS propone que se utilice AAL2 como protocolo de transporte, a través de la UTRAN, para el tráfico de los canales de transporte.

AAL2 permite la transmisión de servicios de bajo régimen binario y estrictos requisitos de retardo, como la voz, a través de una red ATM. Cuando la carga es más pequeña que la que puede transportar una célula ATM, la utilización de los niveles de adaptación AAL1, AAL3/4 o AAL5 llenarían parcialmente una célula o reunirían paquetes consecutivos de un mismo usuario en una misma célula, lo que introduciría o bien una

ineficiente utilización del ancho de banda (en el primer caso) o bien un retardo considerable (en el segundo caso). En definitiva, conllevaría un ineficiente funcionamiento del sistema.

La utilización de AAL2 permitiría solucionar todos estos inconvenientes mediante la multiplexación de varios usuarios, que pueden generar tráficos de características variables, dentro de un mismo VCC (Virtual Channel Connection). Esto se consigue permitiendo que la carga procedente de diferentes usuarios puedan compartir una misma célula, consiguiéndose de esta manera una mayor eficiencia en el uso del ancho de banda, al no enviar células parcialmente llenas, con un aceptable retardo de empaquetamiento.

El nivel de adaptación ATM, tal y como se define en la recomendación ITU-T I.363, está dividido en dos subniveles:

- ✓ El subnivel de segmentación y reensamblaje SAR (Segmentation And Reassembly). Su función principal es la de segmentar los datos procedentes de capas superiores, dando lugar a un conjunto de segmentos cuyo tamaño es igual al de la carga útil transportada por una célula.
- ✓ El subnivel de convergencia de servicios CS (Convergence Sublayer). Su misión principal es la de actuar de interfaz entre los servicios solicitados por el usuario y los servicios ofrecidos por el nivel inferior SAR. Por tanto, el papel que desempeñe dependerá en gran medida del servicio demandado.

A su vez, el nivel CS está dividido en otras dos capas:

- ✓ La capa CPCS (Common Part Convergence Sublayer). Esta capa es responsable de la delimitación de los datos (mediante el uso de cabeceras), de su secuenciamiento, de la reserva de memoria en los extremos, de la detección de errores, etc.
- ✓ La capa SSCS (Service Specific Convergence Sublayer). Cubre funciones tales como la segmentación/reensamblaje (para los bloques de mayor tamaño), corrección de errores, bloqueo/desbloqueo, control de flujo, etc. Existe la posibilidad de que este nivel sea nulo, es decir, no desempeñe ninguna función.

Esta es la estructura general del nivel de adaptación; pero en el caso del nivel de adaptación ATM tipo 2 (AAL2), esta división no es respetada. Se divide también en dos

subniveles, pero la correspondencia con la división anterior no es directa. El nivel inferior, llamado CPS (Common Part Sublayer), desempeña funciones propias del nivel SAR y del subnivel CPCS, mientras que el nivel superior SSCS sólo desempeña las funciones propias de esta capa en el subnivel CS de la estructura general. Esta correspondencia queda reflejada en la Figura 1.

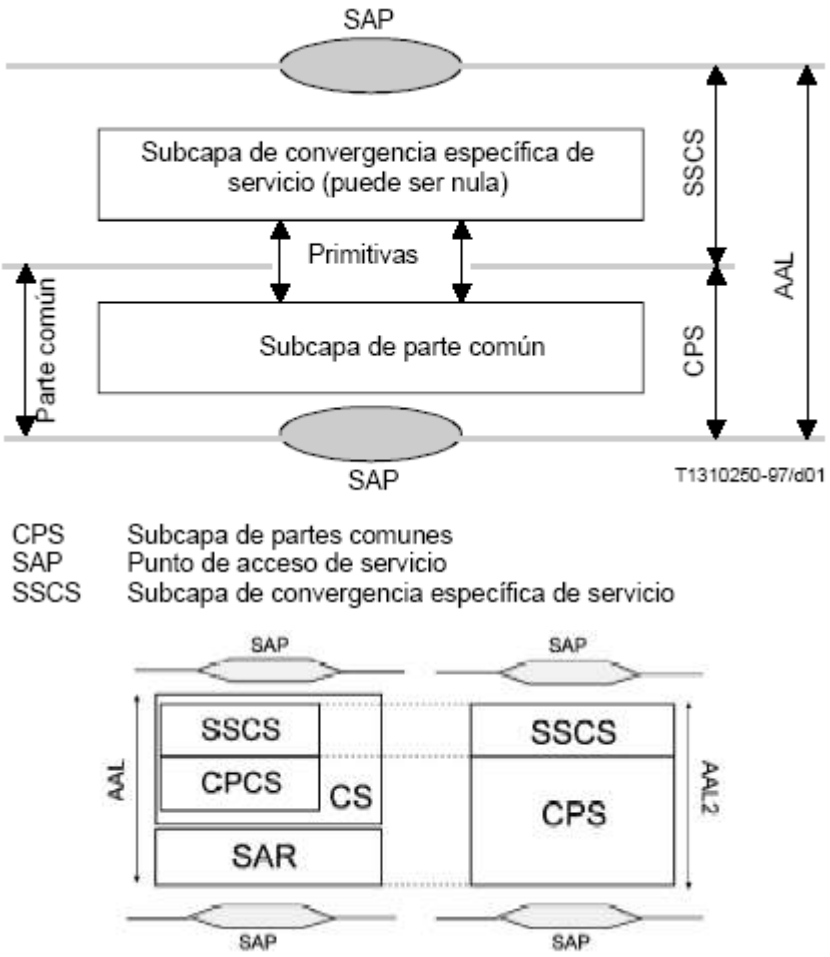


Figura 1. Arquitectura del nivel AAL2.

Cada uno de los subniveles en que se divide el nivel de adaptación ATM tipo 2 se describe en una recomendación diferente. El nivel CPS viene detallado en la recomendación I.363.2; el nivel SSCS viene detallado en las recomendaciones I.366.1 e I.366.2.

2.1. El subnivel CPS.

El nivel CPS ofrece los siguientes servicios:

- ✓ Transferencia de bloques de datos (llamados CPS SDUs), cuyo tamaño no excede de 45 octetos (valor por defecto), aunque pueden negociarse hasta 64 octetos. En el caso de que se perdiera alguna CPS SDU, ésta no sería retransmitida.
- ✓ Multiplexación y demultiplexación de varios canales AAL2 en una misma conexión VCC.
- ✓ Integridad secuencial de CPS SDUs en cada canal AAL2.

Extremo a extremo, una conexión AAL2 consiste en la concatenación de canales AAL2, cada uno de los cuales representa un usuario (o entidad SSCS). Cada uno de estos canales AAL2 es un canal virtual bidireccional. Estas "miniconexiones" AAL2 pueden ser establecidas sobre conexiones ATM permanentes (PVC) o conmutadas (SVC). Pero además, cada canal AAL2 que va multiplexado en un circuito virtual puede ser multiplexado en otros VCC diferentes a lo largo de la red ATM. Cobra sentido, pues, la conmutación a este nivel, lo que da un toque de originalidad a AAL2 respecto a otros tipos de AAL [Isn00-a]. Esta idea queda reflejada en la Figura 2.

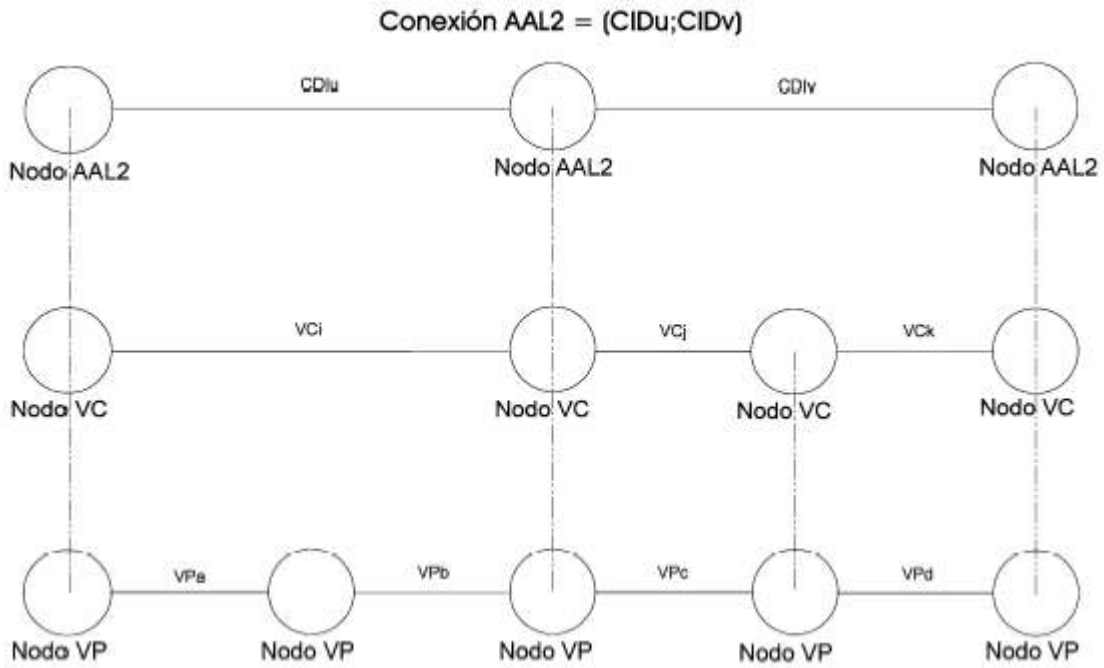


Figura 2. Descomposición de una conexión AAL2 en canales AAL2

Según la QoS de la aplicación que se va a transportar, cada usuario AAL2 puede seleccionar un AAL-SAP distinto, que va asociado a un nivel de QoS. Será posible multiplexar fuentes que acceden al nivel AAL a través de SAPs con distintos niveles de QoS dentro de un mismo VCC, como se ilustra en la Figura 3.

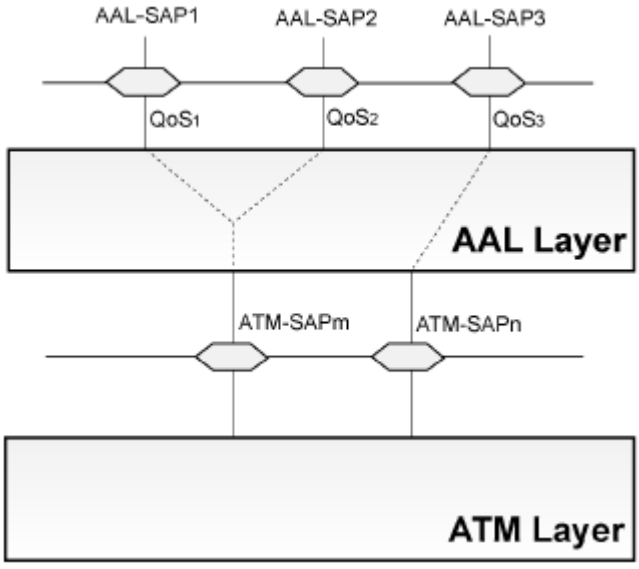
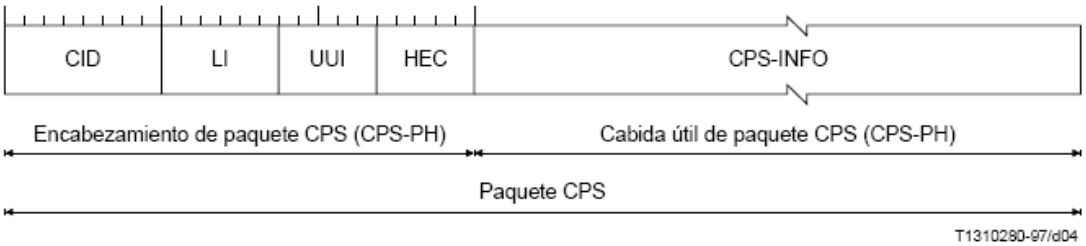


Figura 3. Puntos de acceso al servicio y QoS.

Cada paquete procedente del nivel superior es encapsulado en un paquete CPS. Dicho paquete está constituido por una cabecera de 3 octetos y la carga útil correspondiente a un segmento procedente de SSCS. El tamaño de la carga útil está acotado por un máximo de 45 o 64 octetos (el valor por defecto es igual a 45). A continuación se detalla e ilustra la estructura de un paquete CPS.



CID	Identificador de canal	(8 bits)
LI	Indicador de longitud	(6 bits)
UUI	Indicación usuario a usuario	(5 bits)
HEC	Control de error de encabezamiento	(5 bits)
CPS-INFO	Información	(1 .. 45/64 octetos)

Figura 4. Paquete CPS.

La cabecera del paquete está constituida por:

- ✓ Un identificador de canal AAL2 de 8 bits (CID, Channel IDentifier), que define unívocamente a cada usuario. Al ser canales bidireccionales, el CID deberá coincidir en ambas direcciones. El valor del CID es inferior a 255. El CID = 0 no se utiliza, ya que el cero se emplea como relleno; el CID = 1 se utiliza en procedimientos de gestión de capas; CID comprendidos entre el 2 y el 7 están reservados para un futuro uso. El resto de valores son reservados para los usuarios (8-255). Esto implica que en cada VCC se pueden multiplexar hasta 248 usuarios CPS. La gestión de canales AAL2 estará asegurada por la capa CPS mediante el uso de este campo.
- ✓ Un indicador de longitud (LI, Length Indicator) de 6 bits que permite conocer el tamaño variable de la carga útil.
- ✓ UUI de 5 bits, utilizado sólo por los usuarios y que no es interpretado por la capa CPS. Permite distinguir tipos de usuarios como, por ejemplo, entre entidades SSCS y entidades de gestión de capa. Un valor de UUI igual o inferior a 27 indica que se trata de un usuario SSCS. Los valores 30 o 31 son reservados para O&M. El resto de valores están reservados para una futura estandarización.
- ✓ Un campo de control de errores (HEC, Header Error Control) que permite detectar errores eventuales en la transmisión. El transmisor calcula, en primer lugar, el producto de los diecinueve primeros bits de la cabecera CPS-PH por el monomio x5. A continuación divide (módulo 2) el resultado por el polinomio generador de grado cinco x5 + x2 + 1. Los coeficientes del polinomio resto se insertan en el campo HEC con el coeficiente del término x4 como bit más significativo.

Una vez se ha encapsulado cada CPS-SDU en un paquete CPS (cuya longitud puede variar de un usuario a otro; incluso, un mismo usuario puede tener conexiones simultáneas con tamaños diferentes), se procede a multiplexar varios de estos paquetes en una misma CPS-PDU, hasta ocupar una longitud de 47 octetos. De esta forma se consigue que el tráfico procedente de varias fuentes pueda ser multiplexado en una misma célula ATM.

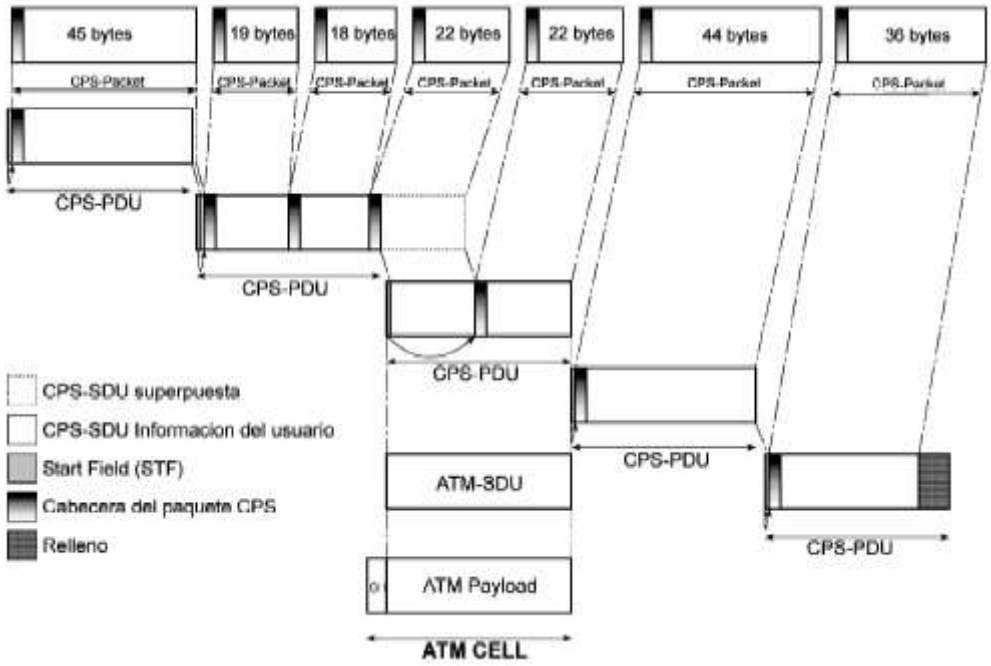


Figura 5. Multiplexación de paquetes de usuario con longitud variable.

A continuación se forma la CPS-PDU mediante la introducción de una cabecera de 1 octeto de longitud. Al multiplexar, puede suceder que los límites de un paquete CPS no coincidan con los límites de la carga de una célula ATM. Esto conlleva a que fragmentos de un mismo paquete sean transportados en células diferentes. Por tanto, dado que un paquete puede empezar en cualquier parte dentro de los últimos 47 octetos de la carga de la célula, se hace necesaria la inclusión de un campo que permita localizar el comienzo del paquete. En eso consiste la principal función que lleva a cabo la nueva cabecera, denominada Start Field (STF). El offsetde inicio del paquete dentro de la carga de la célula ATM se almacena en el campo OSF (OffSet Field), de 6 bits de longitud. Esta cabecera además incluye otros dos campos: número de secuencia (SN, 1 bit), que permite detectar la pérdida de células, y un campo de paridad (P), para detectar errores en la cabecera STF. Todo este proceso de multiplexación queda recogido por la Figura 5.

Con el fin de limitar el tiempo de espera en emisión necesario para completar una CPSPDU se recurre al uso de un mecanismo de temporización. Se hace necesario con el fin de llegar a un compromiso entre retardo y eficiencia del enlace, como se comentó anteriormente. En un capítulo posterior se entrará en detalle en el funcionamiento de

este temporizador, y las repercusiones que tiene en las prestaciones ofrecidas por el enlace ATM.

2.2. El subnivel SSCS.

Se pueden definir diferentes subcapas dentro del nivel SSCS para soportar servicios de usuario AAL2. Aunque también es posible que no exista ninguna, desempeñando el nivel SSCS meras funciones de transformación de primitivas en otras equivalentes del nivel CPS.

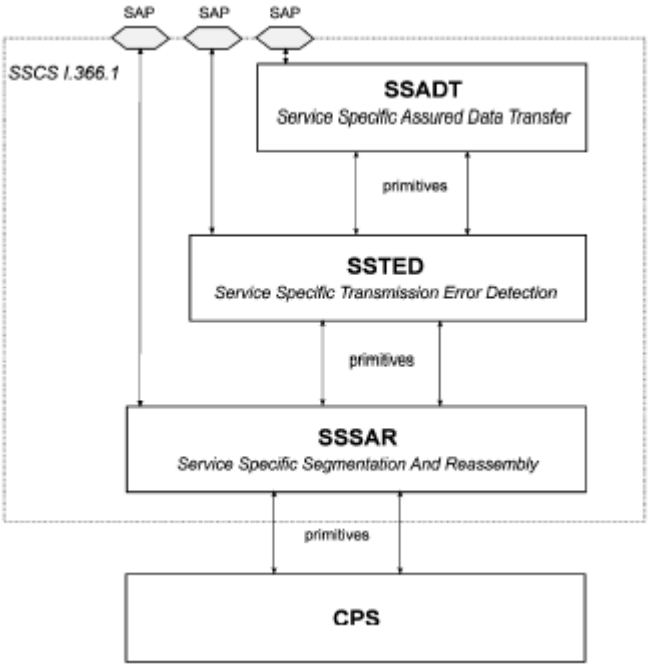


Figura 6. Composición de la capa SSCS según la Recomendación ITU-T I.366.1.

El subnivel SSCS, tal y como se define en la recomendación ITU-T I.366.1, se descompone en tres subcapas, como muestra la Figura 6:

- El servicio mínimo ofrecido por SSCS es una función de segmentación y de reensamblaje del que se hace cargo la capa SSSAR (Service Specific Segmentation And Reassembly).
- De manera opcional, se ofrece un servicio de detección de errores, que lleva a cabo la capa SSTED (Service Specific Transmission Error Detection).
- Cuando el mecanismo de detección de errores de transmisión está activado, éstos pueden ser corregidos gracias a las funciones de la capa SSADT (Service Specific

Assured Data Transfer). En realidad, la subcapa SSADT es idéntica a la capa SSCOP (Service Specific Connection Oriented Protocol) definida en la recomendación ITU-T Q.2110.

La subcapa SSSAR ofrece los servicios de transferencia de paquetes donde el tamaño puede variar hasta 65568 octetos y asegurar la integridad secuencial (heredada de CPS). El mecanismo de segmentación y reensamblaje de la subcapa SSSAR es bastante rudimentario, como muestra la Figura 7.

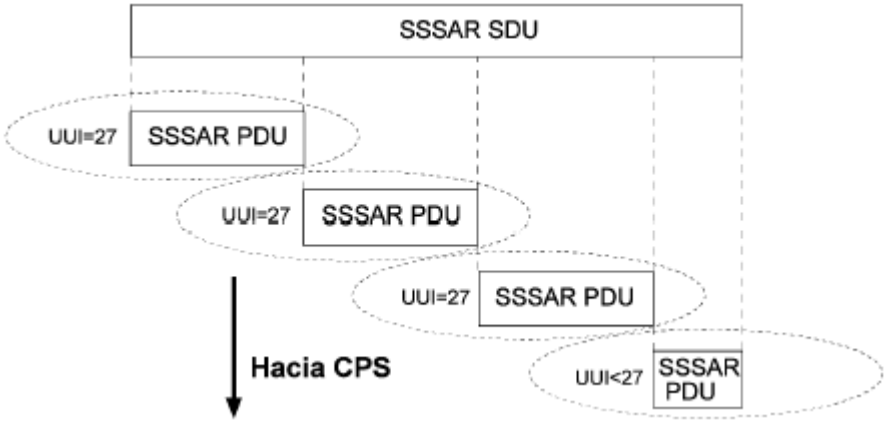


Figura 7. Mecanismo de segmentación y reensamblaje de la subcapa SSSAR.

El paquete pasado como parámetro por el usuario (llamado SSSAR SDU) es descompuesto en múltiples fragmentos cuyo tamaño puede variar hasta 64 octetos. A su vez, cada uno de estos fragmentos es pasado como parámetro a la capa CPS además de información suplementaria (UUI) que indica si el fragmento pasado es el último trozo de una SSSAR SDU o no. El valor 27 significa que se trata de un fragmento intermedio, mientras que valores inferiores a 27 (0..26) indican que se trata de un fragmento final (no serán necesarios más datos para completar la SSSAR SDU). [ITU I.366.1]

En las especificaciones del 3GPP [3G TS 25.926] y [3G TS 25.934], relativas al transporte de señalización e información de usuarios a través de la interfaz Iub, se especifica que sólo el nivel SSSAR será implantado en la red de transporte de UMTS. Dado que SSSAR desempeña únicamente funciones relacionadas con la transferencia

de datos, las funciones de detección de errores y transferencia segura de datos pasan a ser gestionadas exclusivamente por los protocolos MAC y RLC.

3. SEÑALIZACIÓN AAL2: Q.aal2.

Al introducir AAL2 un grado de multiplexación adicional al ofrecido por ATM, se hace necesaria la utilización de un protocolo de señalización que ayude a gestionar las conexiones AAL2 establecidas sobre conexiones del tipo PVC o SVC.

Su principal función será la de establecer y liberar de conexiones AAL2 punto a punto, según sean solicitadas por los usuarios. Por tanto, este protocolo se limitará estrictamente a gestionar conexiones a este nivel: es un nuevo protocolo, y no una extensión de ningún protocolo de señalización ATM.

Desde el punto de vista de la red ATM, la interfaz Iub es una interfaz UNI (User-to-Network Interface). Por tanto, el protocolo de señalización AAL2 reposa sobre los servicios de señalización UNI-SAAL (definidos en las recomendaciones ITU-T Q.2100 (SAAL), Q.2110 (SSCOP) y Q.2130 (SSCF en UNI)).

La primera especificación para señalización AAL2 fue publicada por el ATM-Forum en la recomendación titulada ATM Trunking using AAL2 for Narrowband services que, como su nombre indica, se ocupaba de la señalización AAL2 en la transmisión de servicios de banda estrecha, como la voz. Pero los comités de normalización del 3GPP se acabaron inclinando por la recomendación Q.2630.1, publicada recientemente por la ITU-T. Actualmente se está cerrando la especificación de una mejora sobre la anterior versión, denominada Q.2630.2. Las últimas especificaciones de 3GPP al respecto de la arquitectura de las interfaces Iub/Iur ya señalan a esta última como el protocolo ALCAP de la red de acceso que debe ser utilizado.

Uno de los principales beneficios que aporta el nuevo protocolo de señalización es que permite el establecimiento de conexiones AAL2 independientemente del protocolo utilizado en el establecimiento de los circuitos virtuales ATM sobre los que son transportadas. Esta separación entre protocolos de señalización AAL2 y ATM tienen como consecuencia más reseñable la posibilidad de que la red AAL2 puede poseer su

propio esquema de direccionamiento, permitiendo que múltiples redes AAL2 operen superpuestas sobre una misma red ATM [Ene99].

El protocolo de señalización también proporciona mecanismos para manejar situaciones de congestión en la red de señalización, así como situaciones temporales de indisponibilidad de los enlaces o entidades de señalización. Para ello, tiene la capacidad de reducir las peticiones de conexión o impedir que éstas se dirijan a switches AAL2 congestionados.

Como ya se ha mencionado, la principal funcionalidad del Q.aal2 es el establecimiento y liberación de conexiones AAL2. En la Figura 8 se muestra, de manera esquemática, el intercambio de mensajes, entre los extremos, que forman parte de un proceso de establecimiento de una conexión.

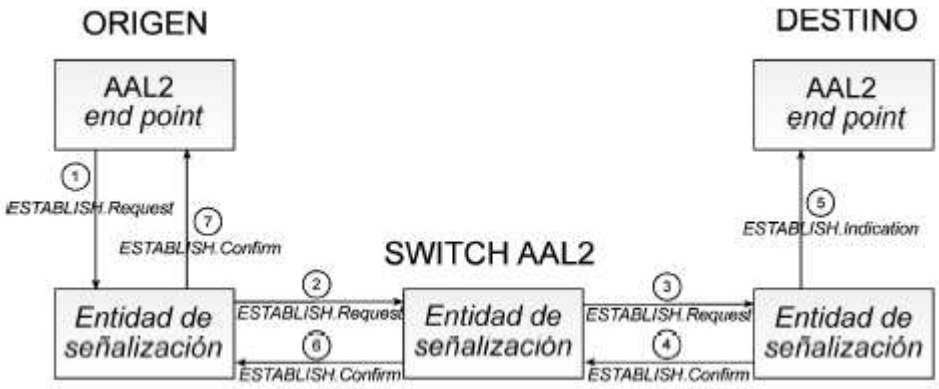


Figura 8. Establecimiento de conexión.

En el intercambio de estos mensajes se produce, en cada una de las entidades de señalización involucradas, la traducción de las primitivas genéricas utilizadas por el protocolo de señalización a nivel AAL2 en otras más específicas del protocolo de transporte de señalización que se emplee.

En el siguiente apartado se estudiará qué implicaciones tiene la utilización de ATM, y los protocolos de señalización asociados, en la red de acceso.

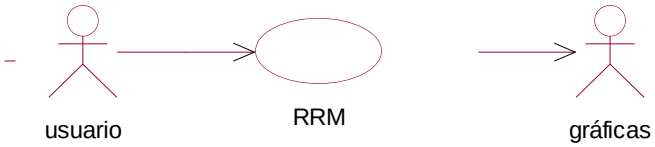
ANEXO 2. MODELAMIENTO DEL SISTEMA EMULACIÓN

1. DIAGRAMAS DE CASOS DE USO

Se describirán las interacciones del sistema con su entorno, identificando los Actores, que representan los diferentes roles desempeñados por los usuarios del sistema, y los Casos de Uso, que corresponden a la funcionalidad que el sistema ofrece a sus usuarios, explicada desde el punto de vista de éstos.

1.1. CASOS DE USO DE ALTO NIVEL

Describen las interacciones muy brevemente. Son utilizados durante las fases iniciales de captura de requerimientos con el fin de obtener rápidamente una visión de la funcionalidad y el grado de complejidad del sistema. Los actores no son solamente humanos, pudiendo ser también otros sistemas con los cuales el sistema en desarrollo interactúa de alguna manera.



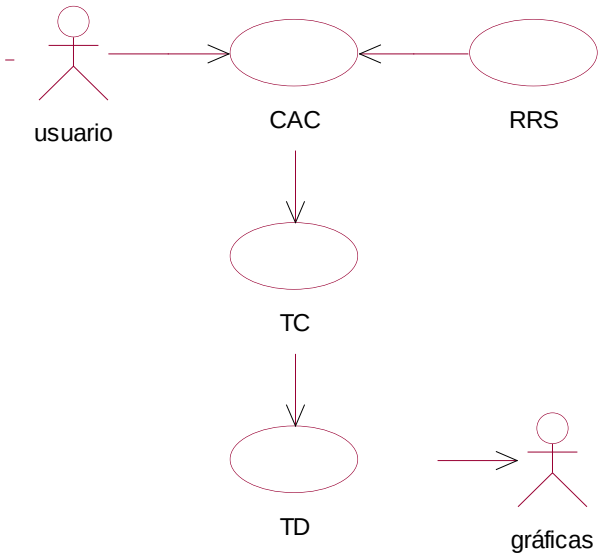
El sistema tiene dos actores, representados por muñecos: usuario y gráficas. El primero de ellos no interactúa directamente con el sistema en algunos casos de uso, pero se le incluye para brindar mayor claridad a la descripción de su funcionalidad. Y finalmente el segundo actor es el resultado de la emulación para los diferentes comportamientos del sistema.

Caso de uso:	**RRM**
Actores:	Usuario (iniciador)
Propósito:	Gestión de los Recursos Radio para el sistema UMTS
Tipo:	Primario

Descripción:	El usuario ingresa (configura) los parámetros con los cuales se analizarán ciertos comportamientos del sistema. El sistema determina que tantos usuarios móviles son admitidos para los diferentes tipos de servicios de UMTS de acuerdo a los parámetros ingresados. Como resultado se obtienen las diferentes gráficas que reflejan el comportamiento de la gestión.

1.2. CASOS DE USO EXTENDIDOS

Se describen las interacciones con mayor detalle que los de alto nivel, enumerando paso a paso los eventos que se presentan durante una ocurrencia típica del caso de uso.



1.2.1. CAC

Información general.

Caso de uso:	**CAC (Control de Admisión de Llamadas)**
Actores:	Usuario (iniciador)
Propósito:	Control la admisión de usuarios basado en la disponibilidad de recursos del sistema
Resumen:	Elección aleatoria de la velocidad del móvil, tipo de servicio solicitado y tasa de transmisión. Según la elección se chequean los parámetros, los cuales deben estar dentro de los valores permitidos de UMTS
Tipo:	Primario
Referencias cruzadas:	Casos de uso: RRS

Precondiciones.

El sistema debe contar con la siguiente información:

- Nivel Máximo de Admisión
- Tamaño paquetes
- Tamaño buffers
- Factor Spreading (ensanchamiento)
- Nivel de Resolución de Congestión
- Nivel de Detección de Congestión

Flujo principal.

- Este caso de uso empieza cuando el usuario ingresa los datos de emulación en la clase *inicio*.
- El sistema inicializa todas las variables establecidas y realiza el llamado a la clase *admisión*.
- De acuerdo al tamaño del paquete, establecido por el usuario, sé inicializan variables que determinan los paquetes transmitidos para cierta tasa de transmisión.
- El sistema elige aleatoriamente el móvil, velocidad para determinado escenario, clase de servicio y tasa de transmisión
- Cuando los parámetros no son congruentes son asignados de nuevo (subflujo S1, reasignación parámetros), de lo contrario se continúa con el proceso (subflujo S2, aceptación parámetros).
- Posteriormente se calcula el valor numérico para el control de admisión de llamadas de cada uno de los tipos de servicios solicitados, si dicho valor no supera el nivel máximo de umbral de admisión se determina la cantidad de paquetes (subflujo S3, nivel menor), de lo contrario se procede a determinar si es desconectado el usuario (subflujo S4, nivel mayor).
- Finalmente el sistema graficará su comportamiento para determinadas condiciones planteadas por el usuario.

Subflujos.

S1. Reasignación parámetros.

- El sistema asigna una nueva tasa de transmisión para el usuario, el cual determina si la acepta (E1) o rechaza (E2).

S2. Aceptación parámetros.

- El sistema continúa con el chequeo del nivel de control de admisión de llamadas.

S3. Nivel menor.

- El sistema determina la cantidad de paquetes asignados al usuario de acuerdo a la tasa de transmisión, los cuales serán almacenados en los buffers de cada uno de los tipos de servicios.
- El sistema realiza el llamado a la clase *congestión*, procedimiento que devuelve una respuesta.

S4. Nivel mayor.

- Inicialización del contador que determinará la cantidad de veces en la cual se desea desconectar a un usuario.
- El sistema realiza el llamado a la clase *desconectar*, procedimiento que devuelve una respuesta.

Flujos de Excepción.

E1. Aceptación de la nueva tasa de transmisión.

- El sistema continúa con el chequeo del nivel de control de admisión de llamadas.

E2. Rechazo de la nueva tasa de transmisión.

- El sistema regresa a escoger aleatoriamente los parámetros necesarios para la emulación.

1.2.2. RRS

Información general.

Caso de uso:	**RRS**
Actores:	No presenta un actor como iniciador
Propósito:	Planificar los Recursos Radios
Resumen:	En este elemento se lleva acabo la estrategia de control de congestión como una de las estrategias que toma gran importancia para garantizar o mantener una QoS aceptable a las conexiones establecidas. También esta encargado de censar cada cierto periodo de tiempo el estado que guarda el canal inalámbrico en función del ruido de fondo.

Tipo:	Primario
Referencias cruzadas:	Casos de uso: CAC

Precondiciones:

- Frecuencia de operación sistema UMTS.
- Ancho de banda.
- Ruido de fondo del canal inalámbrico

Flujo principal.

- Este caso de uso empieza cuando el sistema determina la cantidad de paquetes para cada tipo de servicio solicitado.
- Si la cantidad de paquetes es mayor al tamaño del buffer el sistema debe asignar una nueva tasa de transmisión (S1, cantidad mayor), en el caso contrario se calculará el nivel de ruido de fondo (S2, cantidad menor).
- Una vez realizado el llamado a la clase *ruido*, el sistema debe contar con los valores del ancho de banda y frecuencia de emulación, datos que determinarán las perdidas en la trayectoria según la separación entre la estación móvil y la estación base para los diferentes tipos de escenarios.
- De igual forma el sistema determina todos los parámetros necesarios para calcular el ruido de fondo del canal inalámbrico, por ejemplo, la tasa de transmisión y el nivel de energía para los diferentes tipos de servicio.
- Finalmente se realiza el llamado a las clases *buscar_datos_paq_256*, *buscar_datos_paq_512* y *buscar_datos_paq_1024*, de acuerdo al tamaño de paquetes ingresado inicialmente para el usuario móvil.

Subflujos.

S1. Cantidad mayor.

- El sistema asigna una nueva tasa de transmisión para el usuario, el cual determina si la acepta (E1) o rechaza (E2).

S2. Cantidad menor.

- El sistema continúa con el calculo del nivel de ruido de fondo para el canal inalámbrico, para ello realiza el llamado a la clase *ruido*, la cual retorna una respuesta.

Flujos de Excepción.

E1. Aceptación de la nueva tasa de transmisión.

- El sistema recalcula la cantidad de paquetes asignados al usuario de acuerdo al tipo de servicio solicitado.

E2. Rechazo de la nueva tasa de transmisión.

- El sistema inicializa un contador que determinará la cantidad de veces en la cual se intenta reasignar una nueva tasa de transmisión. Si luego de tres intentos el usuario no es aceptado y el sistema regresa a escoger aleatoriamente los parámetros necesarios para la emulación.

1.2.3. TC

Información general.

Caso de uso:	**TC**
Actores:	No presenta un actor como iniciador
Propósito:	Clasificar el tráfico de acuerdo a los servicios prestados en el sistema UMTS
Resumen:	Clasificará los diferentes tipos de tráfico que son aceptados al RRM. El tráfico es dividido en cuatro tipos de servicios diferentes (conversacional, streaming, interactivo y background). Una vez que el tráfico (paquetes) se clasifica, es enviado a través de buffers al despachador de paquetes
Tipo:	Primario
Referencias cruzadas:	Casos de uso: TD, CAC

Precondiciones.

Ninguna.

Flujo principal.

- Este caso de uso empieza con el procedimiento de abrir el archivo de paquetes para los diferentes tipos de servicio y tamaño de los paquetes.
- Se realiza el llamado a las clases *leer_datos_paq_256*, *leer_datos_paq_512* y *leer_datos_paq_1024*, para cada uno de los tipos de servicio y tamaño de los paquetes.
- El sistema comienza a leer cada uno de los archivos de paquetes, determinando la cantidad de datos según el tipo de servicio a almacenar en cada uno de los buffers.
- Posteriormente se envía dicha cantidad a su respectivo buffers, para lo cual se realiza el llamado a la clase tesis_buffers.

- El sistema analizará si el tiempo de emulación se ha cumplido le indica al caso de uso CAC que no admita más usuarios (S1, tiempo mayor), caso contrario continua con el proceso de almacenamiento (S2, tiempo menor).
- Continuando con el proceso, el sistema determina la eficiencia de la gestión de recursos radios, es decir, condición de resolución de congestión (S3, resolución de congestión) y detección de congestión (S4, detección de congestión), con respecto a un nivel de umbral.

Subflujos.

S1. Tiempo mayor.

- Si el tiempo de emulación es mayor, el sistema inicializa la bandera de admisión de usuarios y a continuación realiza el llamado a la clase *admisión*.
- El proceso de gestión de recursos radio culmina con el despliegue de las graficas, las cuales describen el comportamiento del sistema.

S2. Tiempo menor.

- Cuando es menor el tiempo de emulación, el sistema comienza a almacenar los paquetes en los respectivos buffers de cada tipo de servicio.

S3. Resolución de congestión.

- De acuerdo al nivel de umbral obtenido de sumar la longitud de los buffers en cierto instante, el sistema analiza si es mayor (E1) o menor (E2) con respecto al nivel de umbral de resolución de congestión fijado como condición inicial en la emulación.

S4. Detección de congestión.

- De acuerdo al nivel de umbral obtenido de sumar la longitud de los buffers en cierto instante, el sistema analiza si es mayor (E3) o menor (E4) con respecto al nivel de umbral de detección de congestión fijado como condicional inicial en la emulación.

Flujos de Excepción.

E1. Nivel resolución congestión menor.

- El sistema compara si el nivel obtenido se encuentra entre los niveles de resolución y detección de congestión. Posteriormente realiza el análisis si el nivel de umbral con respecto al nivel de detección de congestión (S4, Detección de congestión).

E2. Nivel resolución congestión mayor.

- El sistema analiza la solución de reducir la carga por medio de la no-aceptación de nuevas solicitudes hasta que la carga tenga un cierto valor permitido y durante un intervalo de tiempo, por lo tanto se realiza un llamado a la clase *despacho* para desalojar cierta cantidad de paquetes almacenados en los buffers. Si el intervalo de tiempo no se cumple, el sistema puede continuar leyendo paquetes para almacenar posteriormente.

E3. Nivel detección congestión menor.

- El sistema al presentar esta condición realiza el llamado a la clase *despacho*, la cual desalojará los paquetes almacenados en los buffers, de acuerdo a un algoritmo de prioridades.

E4. Nivel detección congestión mayor.

- En este caso el sistema continúa leyendo y almacenando paquetes en cada uno de los buffers para los diferentes tipos de servicios solicitados.

1.2.4. TD

Información general.

Caso de uso:	**TD**
Actores:	No presenta un actor como iniciador
Propósito:	Despachar los paquetes almacenados en los buffers
Resumen:	El despacho de paquetes esta considerado bajo un esquema de prioridades en el cual los servicios con características de retardo mas estrictas, en este sentido el servicio conversacional serán los de mayor prioridad, en consecuencia el servicio background serán los de menor prioridad.
Tipo:	Primario
Referencias cruzadas:	Ninguna

Precondiciones.

- Ruido de fondo del canal inalámbrico
- Prioridad de despacho

Flujo principal.

- Este caso de uso empieza con el valor medido para nivel de ruido de fondo en el canal inalámbrico, el cual tendrá la función de limitar el despacho en cada uno de los buffers.
- Finalmente el sistema suma los tamaños de cada uno de los buffers para determinar el nivel de umbral, el cual se emplea para determinar las condiciones de eficiencia del sistema.

Subflujos.

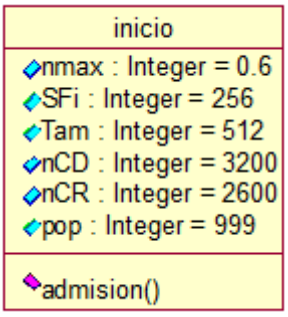
Ninguno.

2. DIAGRAMA DE CLASES

2.1. CLASE INICIO

Ejecutar el sistema de Gestión de Recursos Radio (RRM), la cual presenta como atributos: Captura y redondeo del vector de parámetros de la CPU para obtener el minuto y segundo actual; inicialización de variables públicas para el sistema RRM y además permite configurar las diferentes situaciones de emulación con la variación de los siguientes parámetros:

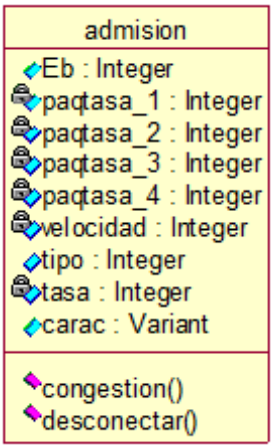
- nmáx. Nivel Umbral Máximo de Admisión
- SFi. Factor de Spreading (ensanchamiento)
- Tam. Tamaño de paquetes
- nCD. Nivel Umbral de Detección de Congestión
- nCR. Nivel Umbral de Resolución de Congestión



Y finalmente presenta como operación la clase *admisión*, en la cual se inicia el proceso de admisión de usuarios con diferentes requerimientos de calidad del servicio. Se debe aclarar que existen muchas variables públicas para todo el sistema que no son representadas en la figura, solo se detallan aquellas que pueden ser modificadas por el usuario.

2.2. CLASE ADMISIÓN

Inicialmente verifica el tamaño del paquete seleccionado y de acuerdo a lo anterior asigna el número de paquetes permitidos para determinada tasa de transmisión, se representa por los atributos privados: paqtasa_1, paqtasa_2, paqtasa_3 y paqtasa_4. También se realiza la elección aleatoria de la velocidad del usuario (atributo privado velocidad), tipo de servicio a solicitar (atributo público tipo) y tasa de transmisión (atributo privado tasa). Continua con la verificación de los parámetros solicitados por el usuario (escenario, clase de servicio y tasa de transmisión), los cuales deben estar dentro los valores permitidos de acuerdo con UMTS. Además presenta los atributos públicos Eb y carac, los cuales representan la energía del móvil y el vector que contiene los parámetros (escenario, clase de servicio y tasa de transmisión) de cada usuario, respectivamente.

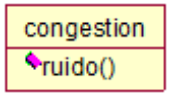


Como operaciones tiene las clases *congestión* y *desconectar*, con sus respectivas relaciones que se describen a continuación. Entre las clases *admisión* y *inicio*, existe una relación de generalización permitiendo representar la herencia de las mismas, con la cual se heredan los atributos y operaciones de la clase padre (*admisión*). De igual

forma las relaciones de asociación con las clases *desconectar* y *congestión*, tienen gran importancia por los roles que permiten expresar el papel que cumplen dichas relaciones; por ejemplo, en cada una de ellas, la referencia se tiene con el valor obtenido por el atributo CaC (variable pública de Control de Admisión de Llamadas). Finalmente la relación de dependencia despliegue con la interfaz gráficas, despliega las diferentes gráficas que representan los comportamientos del sistema para ciertas condiciones.

2.3. CLASE CONGESTION

Clase donde el RRS chequea la carga en los cuatro buffers (uno para cada servicio) y determina si es aceptada la tasa de transmisión del nuevo usuario o si es necesario que este elija otra menor en función a la capacidad disponible, con esto se evita que el buffer se desborde y se pierdan paquetes. Todos los atributos son heredados de la clase *inicio* (relación de generalización) y presenta como operación la clase *ruido*, para chequear el nivel de ruido de fondo para el canal inalámbrico.

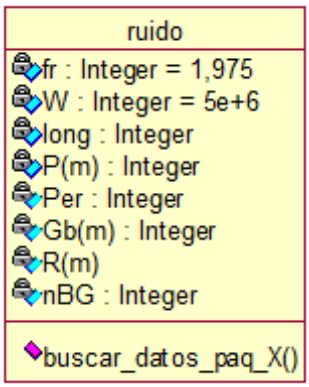


Las relaciones de asociación presentes con las clases *admisión* y *ruido*, tienen como referencia el valor obtenido por el atributo CaC y el nivel de ruido de fondo, respectivamente.

2.4. CLASE RUIDO

Clase que censa cada cierto periodo de tiempo el estado que guarda el canal inalámbrico en función del ruido de fondo que nos indica las condiciones del canal inalámbrico.

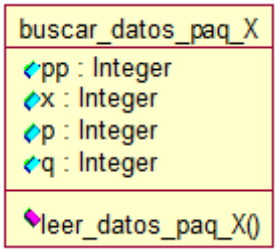
Los atributos privados son la frecuencia de operación del sistema (fr); el ancho de banda para el canal (W); la separación entre la estación móvil y base (long), el nivel de potencia del móvil y la tasa de transmisión [R(m)] para determinado escenario [P(m)]; las pérdidas de la trayectoria (Per), la ganancia obtenida [G(m)] y el nivel de ruido de fondo (nBG). Como atributos públicos se tienen los heredados por la relación de generalización existente con la clase tesis_inicio.



Las operaciones presentes en la clase son: buscar_datos_paq_256, buscar_datos_paq_512 y buscar_datos_paq_1024, las anteriores se han expresado por *buscar_datos_paq_X*, en la cual se solicita la lectura del archivo de paquetes de acuerdo al tipo de servicio solicitado. Por lo tanto la relación de asociación (lectura) con *buscar_datos_paq_X*, arroja un resultado luego de la lectura del archivo.

2.5. CLASE BUSCAR_DATOS_PAQ_X

Clase que ubica los diferentes archivos de paquetes, según el tamaño de los paquetes y tipo de servicio solicitado, con el objeto de ser leídos posteriormente. Los atributos públicos se heredan de la relación de generalización con la clase *admisión* y entre otros que serán empleados por la clase *leer_datos_paq_X*. Presenta como operaciones las clases *leer_datos_paq_256*, *leer_datos_paq_512* y *leer_datos_paq_1024*, las cuales se han representado por *leer_datos_paq_X*, cuya funcionalidad es la lectura del archivo de paquetes.

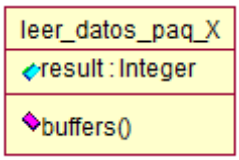


Posee una relación de dependencia (lectura) en la cual el resultado entregado a la clase *ruido* depende de la lectura del archivo de paquetes realizado por *leer_datos_paq_X*, de tal manera que un cambio en el elemento independiente afecta al elemento dependiente. Y finalmente la relación de asociación (archivo) con la clase *ruido*, determina el resultado de la lectura del archivo.

2.6. CLASE LEER_DATOS_PAQUETES_X

Clase que almacena en un arreglo la cabecera del paquete, la cual determina el tipo de servicio solicitado para almacenarlo en su respectivo buffer.

Entre las relaciones existentes tenemos: dependencia (lectura) con la clase *buscar_datos_paq_X*, explicada el anterior item y de asociación (almacenar) con *buffers*, la cual arroja un resultado (atributo privado result) de acuerdo a la cantidad de paquetes leídos en el archivo de paquetes para cada uno de los diferentes tipos de servicio.

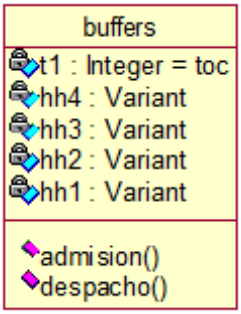


2.7. CLASE BUFFERS

Clase que determina el tamaño de paquetes almacenados en cada uno de los buffers para solicitar su respectivo despacho; además analiza las situaciones de detección de congestión y resolución de congestión.

Algunos de los atributos son heredados mediante la relación de generalización con la clase tesis_inicio y entre los atributos privados tenemos los vectores (hh4, hh3, hh2 y hh1) de servicio conversacional, streaming, interactivo y background, respectivamente. Sus operaciones son las clases tesis_admisión y tesis_despacho, donde la primera tiene como objetivo no admitir más usuarios luego de finalizar el tiempo de simulación

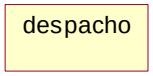
tomado aleatoriamente del vector parámetros de la CPU, y la segunda se encarga del despacho de los buffers de acuerdo a la prioridad fijada en la emulación.



Presenta una relación de dependencia con la clase tesis_despacho, donde el resultado entregado a _datos_2X depende de la ejecución del despacho para la misma. Finalmente la relación de asociación, almacenar, la cual compara el valor obtenido del atributo result de tesis_datos_2X con la cantidad de paquetes almacenados en los cuatro buffers.

2.8. CLASE DESPACHO

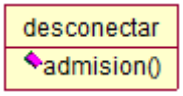
En ella se realiza el despacho de los paquetes almacenados en cada uno de los buffers, inicializándolos en cero.



Todos los atributos son heredados mediante la relación de generalización con la clase *inicio*. Además presenta una relación de dependencia, despacho, donde su resultado es un valor dependiente de *buffers*.

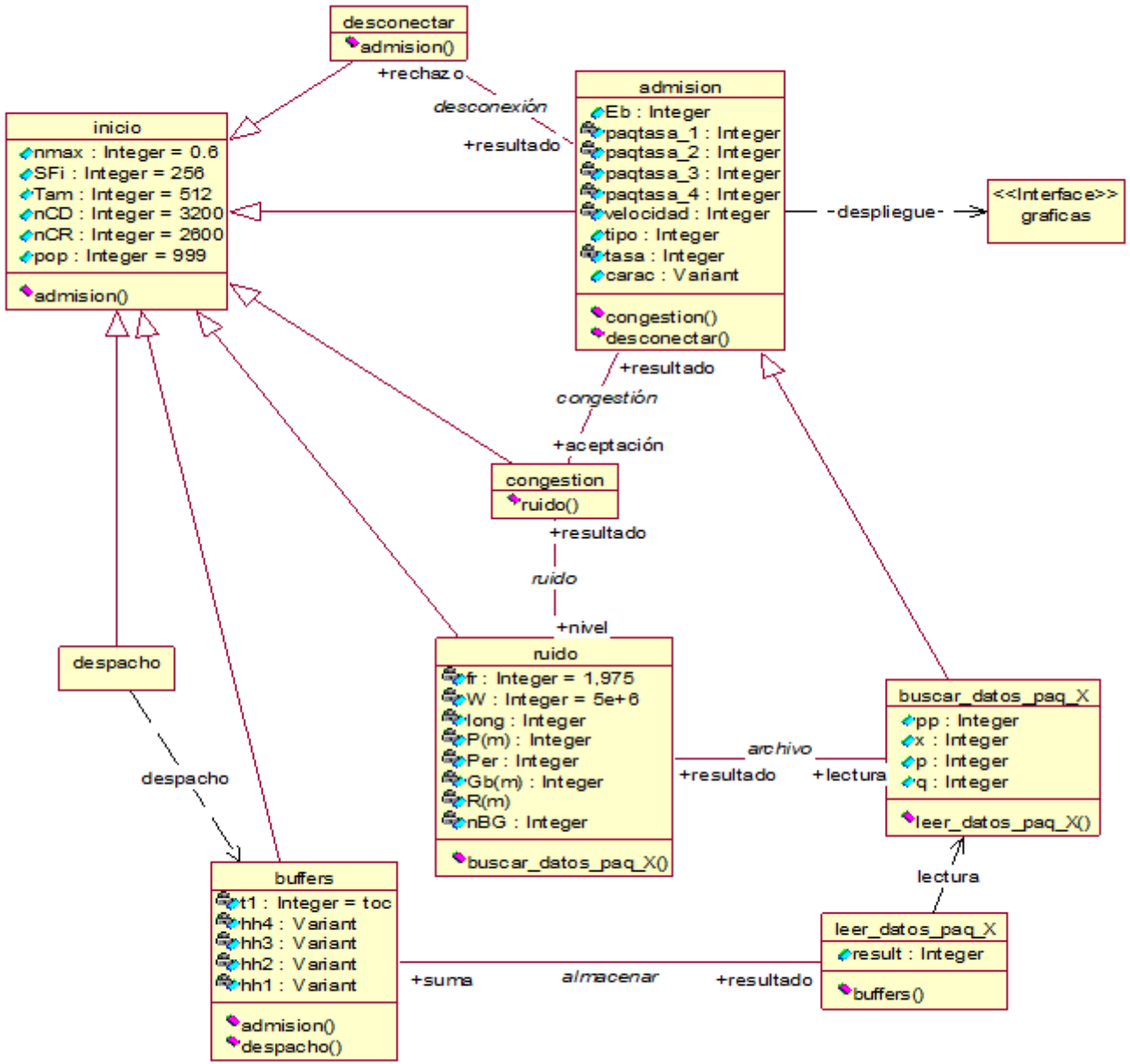
2.9. CLASE DESCONECTAR

Desconecta a los usuarios de acuerdo a una decisión aleatoria pero que depende del control de admisión en el sistema para cada una de las clases de servicio solicitadas.

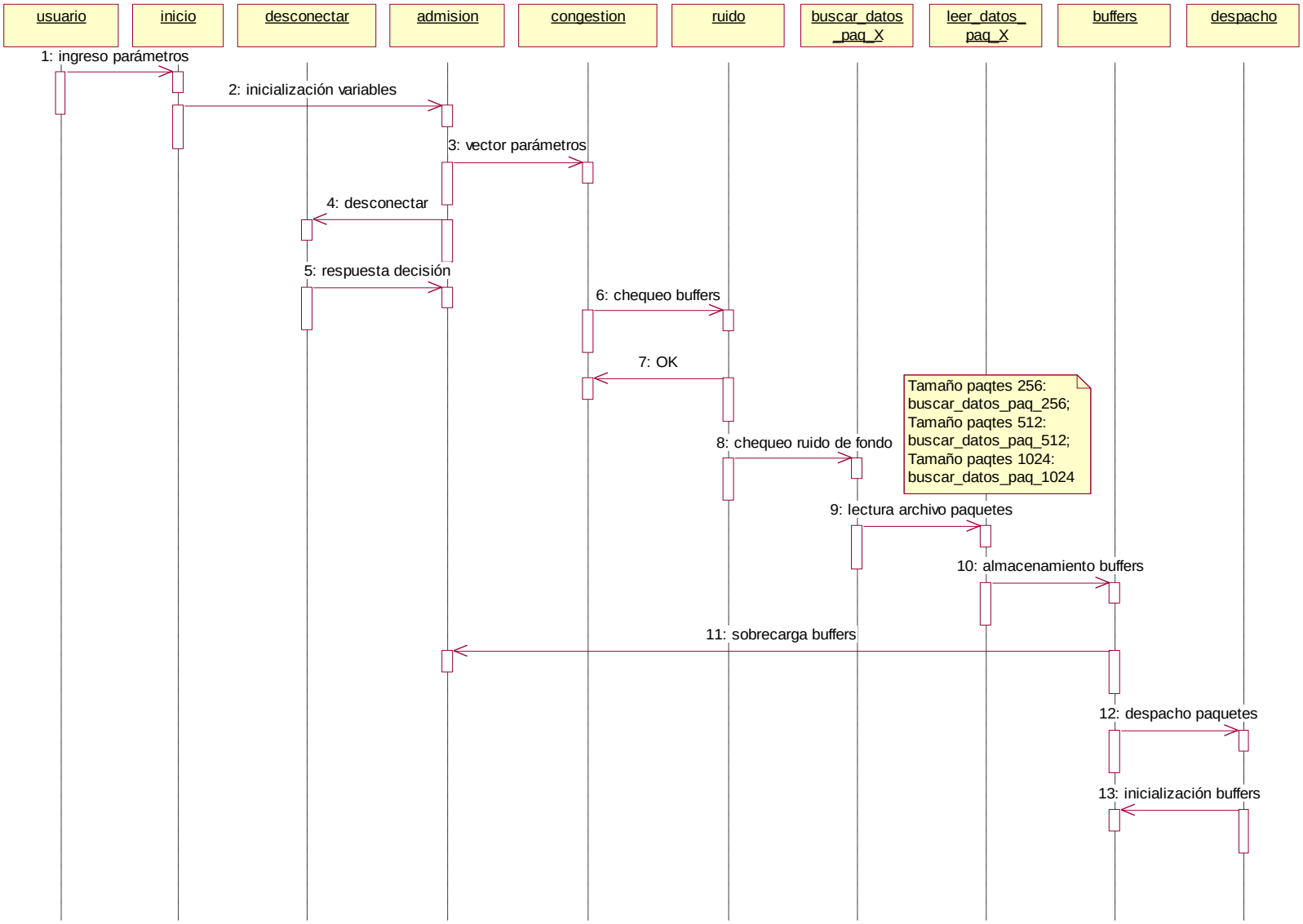


Los atributos empleados para cumplir con el objetivo de la clase son heredados de *inicio*. La única operación es *admisión*, en la cual se decide si un usuario es desconectado o continua siendo admitido.

A continuación se describen todas las relaciones existentes entre cada una de las clases del sistema RRM.

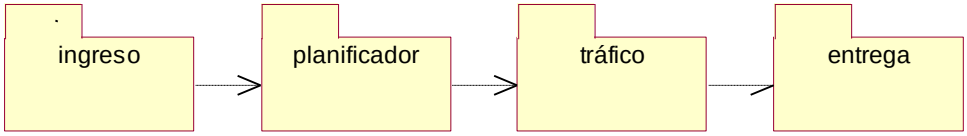

desconectar
admision()
+rechazo
desconexión
+resultado
admision
Eb : Integer
paqtasa_1 : Integer
paqtasa_2 : Integer
paqtasa_3 : Integer
paqtasa_4 : Integer
velocidad : Integer
tipo : Integer
tasa : Integer
carac : Variant
congestion()
desconectar()
-despliegue-
<<Interface>>
graficas
inicio
nmax : Integer = 0.6
SFi : Integer = 256
Tam : Integer = 512
nCD : Integer = 3200
nCR : Integer = 2600
pop : Integer = 999
admision()
+resultado
congestión
+aceptación
congestion
ruido()
+resultado
ruido
+nivel
despacho
despacho
ruido
fr : Integer = 1,975
W : Integer = 5e+6
long : Integer
P(m) : Integer
Per : Integer
Gb(m) : Integer
R(m)
nBG : Integer
buscar_datos_paq_X()
archivo
+resultado
+lectura
buscar_datos_paq_X
pp : Integer
x : Integer
p : Integer
q : Integer
leer_datos_paq_X()
lectura
buffers
t1 : Integer = toc
hh4 : Variant
hh3 : Variant
hh2 : Variant
hh1 : Variant
admision()
despacho()
+suma
almacenar
+resultado
leer_datos_paq_X
result : Integer
buffers()

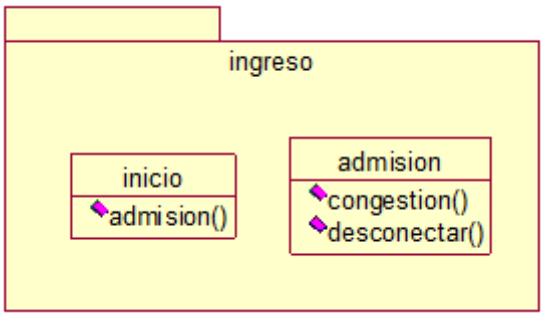
3. DIAGRAMAS DE SECUENCIA

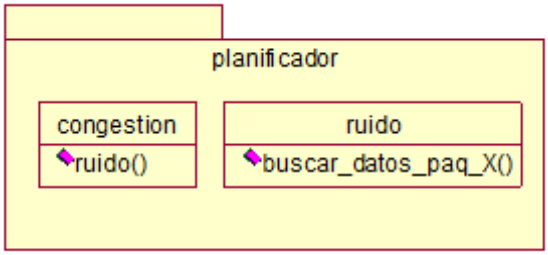


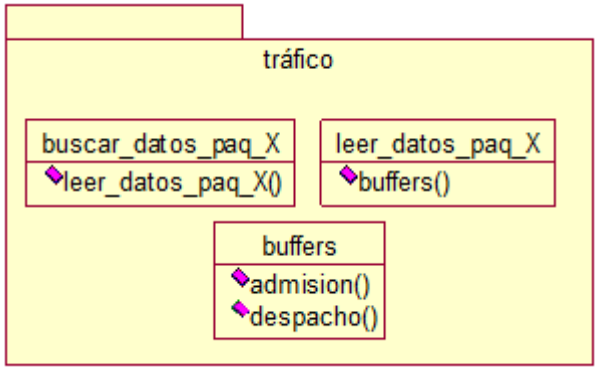
4. DIAGRAMA DE PAQUETES

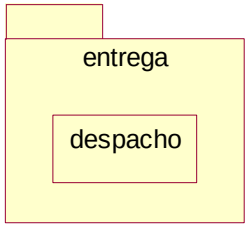
Los paquetes son un mecanismo de estructuración, utilizados para agrupar cualquier tipo de elementos de los modelos que tienen alguna relación semántica, incluyendo otros paquetes.










entrega
despacho

ANEXO 3. CODIFICACIÓN DEL SISTEMA EMULACIÓN

A continuación se describen los archivos desarrollados en Matlab, los cuales tiene extensión *.m.

1. inicio.m

```
% Inicialización de variables para continuar con el control de admisión
% en el sistema RRM
  clear all
  tic
  flag=1;             % Activa bandera
  taz=round(clock);    % Captura y redondea el vector de parámetros del CPU
  min=taz(5)+3         % Asigna a "min" el minuto actual
  sec=taz(6)          % Captura segundos

%Condiciones iniciales de las variables usadas
TS_C=0;
TS_C1=0;
TS_S=0;
TS_S1=0;
TS_I=0;
TS_I1=0;
TS_B=0;
TS_B1=0;
nB4=0;
nB3=0;
nB2=0;
nB1=0;
p_4=0;
p_3=0;
p_2=0;
p_1=0;
n4=0;
n3=0;
n2=0;
n1=0;
n=0;
CaC=0;
kax=0;
kay=0;
kaz=0;
yg=0;
```

```
yk=0;
tfx=0;
tfy=0;
tfz=0;
vas=0;
vez=0;
veces=0;
nivel=0;
usuarios=0;
Tasa_de_bit=0;
porcentaje=0;
Paquetes=0;
desp_nBG=0;
S=0;
rdonBG=0;
desp=0;
lim=0;
yy=0;
ii=0;
tt=0;
gg=0;
hh=0;
kk=0;
S=0;
m=0;
segi=0;
t1=0;
t2=0;
t3=0;
Per=0;
yr=0;
aho=0;
in=0;
clas=0;
desclas=0;
tiposusuarios=0;

nmax=0.6;   % Umbral para el control de admisión de llamadas
SFi=256;    % Factor de Spreading (puede variar de 4 a 256)
Tam=512;    % Tamaño de los paquetes (en Bytes)
nCD=3200;   % Umbral del 80% para 4 buffers de 1000 localidades
nCR=2800;   % Umbral del 70% para 4 buffers de 1000 localidades

pop=999;          % Longitud de los buffers (1000 localidades)
buf_4=[0:1:pop]*0; % Buffer para la Clase de Servicio Conversacional
buf_3=[0:1:pop]*0; % Buffer para la Clase de Servicio Streaming
buf_2=[0:1:pop]*0; % Buffer para la Clase de Servicio Interactiva
buf_1=[0:1:pop]*0; % Buffer para la Clase de Servicio Background

admision;    % Enlace con el programa admision.m
```

2. admisión.m

```
if  Tam == 256;
     paqtasa_1=31;
     paqtasa_2=70;
     paqtasa_3=188;
     paqtasa_4=1000;

     elseif Tam == 512;
          paqtasa_1=16;
          paqtasa_2=35;
          paqtasa_3=94;
          paqtasa_4=500;

          elseif  Tam == 1024;
               paqtasa_1=8;
               paqtasa_2=18;
               paqtasa_3=47;
               paqtasa_4=250;

              end

while flag == 1;
taz=round(clock); % Captura y redondea el vector de parámetros del CPU

yg= rand;        % Se elige aleatoriamente un usuario
yk = yg * 100;
round (yk);

veces=veces+1;
usuarios=usuarios+1;  % Contador de usuarios que solicitan acceso al RRM

tfx=randperm(3)
velocidad=tfx(1) % Se elige aleatoriamente la velocidad del móvil
kax(veces)=velocidad % Vector de velocidades de los móviles

if velocidad == 1; % Velocidad <= 10 Km/hr
  escenario = ['Picocelular']; % Tipo de escenario
  elseif velocidad == 2; % Velocidad <= 120 Km/hr
       escenario = ['Microcelular']; % Tipo de escenario
        elseif velocidad == 3; % Velocidad <= 500 Km/hr
             escenario = ['Macrocelular']; % Tipo de escenario
 end


tfy=randperm(4)
tipo=tfy(2)  % Se elige aleatoriamente un tipo de servicio
kay(veces)=tipo  % Vector de los tipos de servicios solicitados

tfz=randperm(4)
```

```
tasa=tfz(3) % Se elige aleatoriamente una tasa de transmisión
kaz(veces)=tasa % Vector de las tasas de transmisión solicitadas

% Tasas de transmisión implementadas (64, 144, 384 y 2048 kbps)

        if tasa == 1;
           Tasa_de_bit = 64e+3;
            elseif tasa == 2;
                  Tasa_de_bit = 144e+3;
                  elseif tasa == 3;
                        Tasa_de_bit = 384e+3;
                        elseif tasa == 4;
                              Tasa_de_bit = 2048e+3;
         end

  carac =[velocidad, tipo, tasa]; %vector que contienen los parámetros (escenario, clase de servicio y tasa de transmisión) de cada usuario

  % Rutinas para chequear que los parámetros solicitados por el usuario
  % (escenario, clase de servicio y tasa de transmisión); que estén dentro
  % de los valores permitidos de acuerdo con UMTS

if carac(1) == 2;
   if carac(2)== 1 | carac(2)== 2 | carac(2)==3 | carac(2)== 4;
      if carac(3)==4;
         estado=['no permitido'];
         posible=randperm(2);   % Aleatoriamente decide si acepta o rechaza una nueva tasa de transmision
         decision=posible(1);  % Decisión tomada por el usuario
         if decision == 1;
            responde = ['acepto'];
            Tasa_de_bit = 384e+3;  % Nueva tasa de transmisión del usuario que decidió aceptarla
            tasa=3;
            elseif decision == 2;
                  responde = ['no acepto'];
                  Tasa_de_bit = 0e+3;  % El usuario no acepta y termina (no entra al RRM)
             end
         end
    end
end

if carac(1) == 3;
   if carac(2)== 1 | carac(2)== 2 | carac(2)==3 | carac(2)== 4;
      if carac(3)==3;
         estado=['no permitido'];
         posible=randperm(2);   % Aleatoriamente decide si acepta o rechaza una nueva tasa de transmision
         decision=posible(1);  % Decisión tomada por el usuario
         if decision == 1;
            responde = ['acepto'];
```

```
          Tasa_de_bit = 144e+3;  % Nueva tasa de transmisión del usuario que decidió
aceptarla
          tasa=2;
          elseif decision == 2;
                responde = ['no acepto'];
                Tasa_de_bit = 0e+3; % El usuario no acepta y termina (no entra al RRM)
             end
        end

        if carac(3)==4;
          estado=['no permitido'];
          posible=randperm(2);  % Aleatoriamente decide si acepta o rechaza una nueva
tasa de transmision
          decision=posible(1);  % Decisión tomada por el usuario
             if decision == 1;
                responde = ['acepto'];
                Tasa_de_bit = 384e+3;  % Nueva tasa de transmisión del usuario que
decidio aceptarla
                tasa=3;
                     elseif decision == 2;
                           responde = ['no acepto'];
                           Tasa_de_bit = 0e+3; % El usuario no acepta y termina (no entra al
RRM)
             end
          end
     end
 end

carac =[velocidad, tipo, tasa];  % Vector que contiene los parámetros reasignados al
usuario

% Calculo del valor numérico de cada usuario para la clase de servicio solicitada

if tipo == 4 & Tasa_de_bit ~= 0;
   Eb= 2.864; % 4.57 dB
   TS_C =(1+0.6)/(SFi/(0.67*Eb*(1/3))+1); % Valor = 0.00398770322401 para SFi=256
   TS_C1=TS_C1+TS_C; % Suma del mismo servicio
   Clase_de_Servicio = ['Conversacional'];
   elseif tipo == 3  & Tasa_de_bit ~= 0;
        Eb= 2.660; % 4.25 dB
        TS_S  =(1+0.6)/(SFi/(0.57*Eb*(1/3))+1);  % Valor = 0.00287748240254 para
SFi=256
        TS_S1=TS_S1+TS_S; % Suma del mismo servicio
        Clase_de_Servicio = ['Streaming'];
        elseif tipo == 2  & Tasa_de_bit ~= 0;
             Eb= 2.944; % 4.69 dB
             TS_I  =(1+0.6)/(SFi/(0.47*Eb*(1/3))+1);  % Valor = 0.00315252622362 para
SFi=256
             TS_I1=TS_I1+TS_I; % Suma del mismo servicio
             Clase_de_Servicio = ['Interactiva'];
             elseif tipo == 1  & Tasa_de_bit ~= 0;
```

```
                Eb= 2.944; % 4.69 dB
                TS_B  =(1+0.6)/(SFi/(0.37*Eb*(1/3))+1);  %  Valor  =  0.00226611922091
para SFi=256
                TS_B1=TS_B1+TS_B; % Suma del mismo servicio
                Clase_de_Servicio = ['Background'];
        end

         %format long CaC
         CaC=TS_C1+TS_S1+TS_I1+TS_B1; % Suma de todos los servicio

   if CaC <= nmax;   % Compara condicion de admisión
     Paquetes = round (Tasa_de_bit/(Tam*8));    % Cantidad de paquetes asignados al
usuario cuando es aceptado por el CAC
     congestion;  % Enlace con el programa congestion.m
     else
        %CaC
        desconectar; % Enlace con el programa desconectar.m (cuando el umbral "nmax"
es alcanzado)
      end
end

          %Grafica de salida

          algo=length (hh)*0.1;
          gg=(0.1:0.1:algo);

          % Variación de la carga total en función del tiempo
          figure (1)
          plot (gg,hh)
          title ('CARGA TOTAL')
          %axis ([0 max(gg) 0 3500]);
          xlabel ('TIEMPO (segundos)')
          ylabel ('CARGA (Paquetes IP)')
          grid

          % Variación de la carga Background en función del tiempo
          figure (2)
          plot (gg,hh1)
          %axis ([0 max(gg) 0 1500]);
          title ('SERVICIO BACKGROUNG')
          xlabel ('TIEMPO (segundos)')
          ylabel ('CARGA POR CLASE DE SERVICIO')
          grid

          % Variación de la carga Interactiva en función del tiempo
          figure (3)
          plot (gg,hh2)
          %axis ([0 max(gg) 0 1500]);
          title ('SERVICIO INTERACTIVA')
          xlabel ('TIEMPO (segundos)')
          ylabel ('CARGA POR CLASE DE SERVICIO')
```

```
grid

% Variación de la carga Streaming en función del tiempo
figure (4)
plot (gg,hh3)
%axis ([0 max(gg) 0 1500]);
title ('SERVICIO STREAMING')
xlabel ('TIEMPO (segundos)')
ylabel ('CARGA POR CLASE DE SERVICIO')
grid

% Variación de la carga Conversacional en función del tiempo
figure (5)
plot (gg,hh4)
%axis ([0 max(gg) 0 1500]);
title ('SERVICIOS CONVERSACIONAL ')
xlabel ('TIEMPO (segundos)')
ylabel ('CARGA POR CLASE DE SERVICIO')
grid

algos=length (dd)*0.1;
ss=(0.1:0.1:algos);

% Admisión de usuarios (peticiones contra aceptados)
figure (6)
plot (ss,dd,ss,dd1234)
%axis ([0 max(ss) 0 1000]);
title ('PETICIONES Vs ACEPTADOS')
xlabel ('TIEMPO (segundos)')
ylabel ('USUARIOS')
grid


% Admisión de usuarios (por clase de servicio)
figure (7)
plot (ss,dd4,ss,dd3,ss,dd2,ss,dd1)
%axis ([0 max(ss) 0 200]);
title ('USUARIOS ACEPTADOS')
xlabel ('TIEMPO (segundos)')
ylabel ('USUARIOS POR CLASE DE SERVICIO')
grid

break
```

3. congestion.m

```
% Rutinas donde el RRS checa la carga en los cuatro Buffer
% (uno para cada servicio) y determina si es aceptada la
```

```
% tasa de transmisión del nuevo usuario o si es necesario que
% este elija otra menor en función a la capacidad disponible.
% Con esto se evita que el buffer se desborde y se pierdan
% paquetes

if tipo == 4 & Paquetes ~= 0;
  n4=p_4+Paquetes;

   if   n4 <= pop;
        n4;
        nB4=nB4+1;
        ruido; % Enlace con el programa ruido.m
        return
   else

     if carac(3) == 4;
        posible=randperm(2);
        decision=posible(1);
        if decision == 1;
          responde = ['acepto'];
          Tasa_de_bit = 384e+3;
          carac(3) = 3;
          Paquetes = round (Tasa_de_bit/(Tam*8));
          n4=(n4-paqtasa_4)+Paquetes;

          if   n4 <= pop;
               n4;
               nB4=nB4+1;
               ruido; % Enlace con el programa ruido.m
               return
          end

          elseif decision == 2;
                responde = ['no acepto'];
                Tasa_de_bit = 0e+3;
                n4=n4-paqtasa_4;
                TS_C1=TS_C1-TS_C;
                CaC=CaC-TS_C;
               end
            end

      if carac(3) == 3;
         posible=randperm(2);
         decision=posible(1);
         if decision == 1;
           responde = ['acepto'];
           Tasa_de_bit = 144e+3;
           carac(3) = 2;
           Paquetes = round (Tasa_de_bit/(Tam*8));
           n4=(n4-paqtasa_3)+Paquetes;
```

```
            if   n4 <= pop;
                 n4;
                 nB4=nB4+1;
                 ruido; % Enlace con el programa ruido.m
                 return
            end

            elseif decision == 2;
                  responde = ['no acepto'];
                  Tasa_de_bit = 0e+3;
                  n4=n4-paqtasa_3;
                  TS_C1=TS_C1-TS_C;
                  CaC=CaC-TS_C;
                  end
              end

       if carac(3) == 2;
          posible=randperm(2);
          decision=posible(1);
          if decision == 1;
            responde = ['acepto'];
            Tasa_de_bit = 64e+3;
            carac(3) = 1;
            Paquetes = round (Tasa_de_bit/(Tam*8));
            n4=(n4-paqtasa_2)+Paquetes;

            if   n4 <= pop;
                 n4;
                 nB4=nB4+1;
                 ruido; % Enlace con el programa ruido.m
                 return
            end

            elseif decision == 2;
                  responde = ['no acepto'];
                  Tasa_de_bit = 0e+3;
                  n4=n4-paqtasa_2;
                  TS_C1=TS_C1-TS_C;
                  CaC=CaC-TS_C;
                  end
              end

       if  carac(3) == 1;
           Tasa_de_bit = 0e+3;
           n4=n4-paqtasa_1;
           TS_C1=TS_C1-TS_C;
           CaC=CaC-TS_C;
       end
    end
end
```

```
if tipo == 3 & Paquetes ~= 0;
   n3=p_3+Paquetes;

    if   n3 <= pop;
         n3;
         nB3=nB3+1;
         ruido; % Enlace con el programa ruido.m
         return
    else

      if carac(3) == 4;
         posible=randperm(2);
         decision=posible(1);
         if decision == 1;
           responde = ['acepto'];
           Tasa_de_bit = 384e+3;
           carac(3) = 3;
           Paquetes = round (Tasa_de_bit/(Tam*8));
           n3=(n3-paqtasa_4)+Paquetes;

           if   n3 <= pop;
                n3;
                nB3=nB3+1;
                ruido; % Enlace con el programa ruido.m
                return
           end

           elseif decision == 2;
                 responde = ['no acepto'];
                 Tasa_de_bit = 0e+3;
                 n3=n3-paqtasa_4;
                 TS_S1=TS_S1-TS_S;
                 CaC=CaC-TS_S;
                 end
             end

       if carac(3) == 3;
          posible=randperm(2);
          decision=posible(1);
          if decision == 1;
            responde = ['acepto'];
            Tasa_de_bit = 144e+3;
            carac(3) = 2;
            Paquetes = round (Tasa_de_bit/(Tam*8));
            n3=(n3-paqtasa_3)+Paquetes;

            if   n3 <= pop;
                 n3;
                 nB3=nB3+1;
                 ruido; % Enlace con el programa ruido.m
                 return
```

```
            end

            elseif decision == 2;
                  responde = ['no acepto'];
                  Tasa_de_bit = 0e+3;
                  n3=n3-paqtasa_3;
                  TS_S1=TS_S1-TS_S;
                  CaC=CaC-TS_S;
                  end
              end

      if carac(3) == 2;
         posible=randperm(2);
         decision=posible(1);
         if decision == 1;
           responde = ['acepto'];
           Tasa_de_bit = 64e+3;
           carac(3) = 1;
           Paquetes = round (Tasa_de_bit/(Tam*8));
           n3=(n3-paqtasa_2)+Paquetes;

           if   n3 <= pop;
               n3;
               nB3=nB3+1;
               ruido; % Enlace con el programa ruido.m
               return
           end

           elseif decision == 2;
                 responde = ['no acepto'];
                 Tasa_de_bit = 0e+3;
                 n3=n3-paqtasa_2;
                 TS_S1=TS_S1-TS_S;
                 CaC=CaC-TS_S;
                 end
             end

      if  carac(3) == 1;
          Tasa_de_bit = 0e+3;
          n3=n3-paqtasa_1;
          TS_S1=TS_S1-TS_S;
          CaC=CaC-TS_S;
      end
    end
end

if tipo == 2 & Paquetes ~= 0;
   n2=p_2+Paquetes;


   if   n2 <= pop;
```

```
    n2;
    nB2=nB2+1;
    ruido; % Enlace con el programa ruido.m
    return
else

  if carac(3) == 4;
    posible=randperm(2);
    decision=posible(1);
    if decision == 1;
      responde = ['acepto'];
      Tasa_de_bit = 384e+3;
      carac(3) = 3;
      Paquetes = round (Tasa_de_bit/(Tam*8));
      n2=(n2-paqtasa_4)+Paquetes;

      if   n2 <= pop;
           n2;
           nB2=nB2+1;
           ruido; % Enlace con el programa ruido.m
           return
      end

      elseif decision == 2;
            responde = ['no acepto'];
            Tasa_de_bit = 0e+3;
            n2=n2-paqtasa_4;
            TS_I1=TS_I1-TS_I;
            CaC=CaC-TS_I;
            end
        end

   if carac(3) == 3;
     posible=randperm(2);
     decision=posible(1);
     if decision == 1;
       responde = ['acepto'];
       Tasa_de_bit = 144e+3;
       carac(3) = 2;
       Paquetes = round (Tasa_de_bit/(Tam*8));
       n2=(n2-paqtasa_3)+Paquetes;

       if   n2 <= pop;
            n2;
            nB2=nB2+1;
            ruido; % Enlace con el programa ruido.m
            return
       end

       elseif decision == 2;
             responde = ['no acepto'];
```

```
                Tasa_de_bit = 0e+3;
                n2=n2-paqtasa_3;
                TS_I1=TS_I1-TS_I;
                CaC=CaC-TS_I;
                end
            end

      if carac(3) == 2;
        posible=randperm(2);
        decision=posible(1);
        if decision == 1;
          responde = ['acepto'];
          Tasa_de_bit = 64e+3;
          carac(3) = 1;
          Paquetes = round (Tasa_de_bit/(Tam*8));
          n2=(n2-paqtasa_2)+Paquetes;

          if   n2 <= pop;
              n2;
              nB2=nB2+1;
              ruido; % Enlace con el programa ruido.m
              return
          end

          elseif decision == 2;
                responde = ['no acepto'];
                Tasa_de_bit = 0e+3;
                n2=n2-paqtasa_2;
                TS_I1=TS_I1-TS_I;
                CaC=CaC-TS_I;
                end
            end

      if  carac(3) == 1;
          Tasa_de_bit = 0e+3;
          n2=n2-paqtasa_1;
          TS_I1=TS_I1-TS_I;
          CaC=CaC-TS_I;
      end
    end
 end

if tipo == 1 & Paquetes ~= 0;
   n1=p_1+Paquetes;

    if   n1 <= pop;
        n1;
        nB1=nB1+1;
        ruido; % Enlace con el programa ruido.m
        return
    else
```

```
if carac(3) == 4;
   posible=randperm(2);
   decision=posible(1);
   if decision == 1;
      responde = ['acepto'];
      Tasa_de_bit = 384e+3;
      carac(3) = 3;
      Paquetes = round (Tasa_de_bit/(Tam*8));
      n1=(n1-paqtasa_4)+Paquetes;

      if   n1 <= pop;
           n1;
           nB1=nB1+1;
           ruido; % Enlace con el programa ruido.m
           return
      end

      elseif decision == 2;
            responde = ['no acepto'];
            Tasa_de_bit = 0e+3;
            n1=n1-paqtasa_4;
            TS_B1=TS_B1-TS_B;
            CaC=CaC-TS_B;
            end
        end

 if carac(3) == 3;
    posible=randperm(2);
    decision=posible(1);
    if decision == 1;
       responde = ['acepto'];
       Tasa_de_bit = 144e+3;
       carac(3) = 2;
       Paquetes = round (Tasa_de_bit/(Tam*8));
       n1=(n1-paqtasa_3)+Paquetes;

       if   n1 <= pop;
            n1;
            nB1=nB1+1;
            ruido; % Enlace con el programa ruido.m
            return
       end

       elseif decision == 2;
             responde = ['no acepto'];
             Tasa_de_bit = 0e+3;
             n1=n1-paqtasa_3;
             TS_B1=TS_B1-TS_B;
             CaC=CaC-TS_B;
             end
         end
```

```
      if carac(3) == 2;
        posible=randperm(2);
        decision=posible(1);
        if decision == 1;
          responde = ['acepto'];
          Tasa_de_bit = 64e+3;
          carac(3) = 1;
          Paquetes = round (Tasa_de_bit/(Tam*8));
          n1=(n1-paqtasa_2)+Paquetes;

          if   n1 <= pop;
              n1;
              nB1=nB1+1;
              ruido; % Enlace con el programa ruido.m
              return
          end

          elseif decision == 2;
                responde = ['no acepto'];
                Tasa_de_bit = 0e+3;
                n1=n1-paqtasa_2;
                TS_B1=TS_B1-TS_B;
                CaC=CaC-TS_B;
                end
            end

      if  carac(3) == 1;
          Tasa_de_bit = 0e+3;
          n1=n1-paqtasa_1;
          TS_B1=TS_B1-TS_B;
          CaC=CaC-TS_B;
      end
    end
  end
```

4. ruido.m

```
in=in+1;
sumclas(in)=tipo;
clas=length (find (sumclas));

m=m+1;
fr=1.975; % GHz
W=5e+6;   % MHz

if carac(1) == 1;
  P(m)=25.11e-3; % 14dBm
  long=0.1;      % Km
```

```
  elseif carac(1) == 2;
        P(m)=25.11e-3; % 14dBm
        long=1;       % Km
        elseif carac(1) == 3;
              P(m)=125.89e-3; % 21dBm
              long=3;        % Km
end

Per= 142.37+29.74*log10(fr)+50.37*log10(long);  % dB
Gb(m)=10^(-Per/20);  % de dB a numerico

if carac(2) == 4 & nB4 ~= 0;
  Eb(m)= 2.864;  % 4.57 dB
  elseif carac(2) == 3  & nB3 ~= 0;
        Eb(m)= 2.660;  % 4.25 dB
        elseif carac(2) == 2  & nB2 ~= 0;
              Eb(m)= 2.944;  % 4.69 dB
              elseif carac(2) == 1  & nB1 ~= 0;
                    Eb(m)= 2.944;  % 4.69 dB
end

if carac(3) == 4;
  R(m)=2048e+3;  %bps
  elseif carac(3) == 3;
        R(m)=384e+3;  %bps
        elseif carac(3) == 2;
              R(m)=144e+3;  %bps
              elseif carac(3) == 1;
                    R(m)=64e+3;  %bps
end

if m ~= 1;
  S=S+Gb(m-1)*P(m-1);
  end

nBG= ((((Gb(m)*P(m))/R(m))/Eb(m))*W)-(S); %numerico
rdonBG=rdonBG+nBG;
rdodBm=(10*log10(rdonBG))+30;

if  Tam == 256;
   TamBytes=256;
   tesis_Datos_11;
   elseif Tam == 512;
         TamBytes=512;
         tesis_Datos_12;
         elseif  Tam == 1024;
               TamBytes=1024;
               tesis_Datos_13;
               end
```

5. buscar_datos_paq_256

```
if tipo == 4;
      [A,mensaje]= fopen ('C:\matlabR12\work\new\Conversacional_1.txt','r');
      pp=0;
      x=0;
      p=1348;
      q=1350;
      leer_datos_paq_256; % Enlace con el programa leer_datos_paq_256.m

elseif tipo == 3;
      [A,mensaje]= fopen ('C:\matlabR12\work\new\Streaming_1.txt','r');
      pp=0;
      x=0;
      p=1348;
      q=1351;
      leer_datos_paq_256; % Enlace con el programa leer_datos_paq_256.m

elseif tipo == 2;
      [A,mensaje]= fopen ('C:\matlabR12\work\new\Interactiva_1.txt','r');
      pp=0;
      x=0;
      p=1342;
      q=1345;
      leer_datos_paq_256; % Enlace con el programa leer_datos_paq_256.m

elseif tipo == 1;
      [A,mensaje]= fopen ('C:\matlabR12\work\new\Background_1.txt','r');
      pp=0;
      x=0;
      p=1353;
      q=1356;
      leer_datos_paq_256; % Enlace con el programa leer_datos_paq_256.m
  end
```

6. leer_datos_paq_256.m

```
for t=1:Paquetes;
   [F,N]= fread (A,q);
   s= char (F');
   d=[0,0,0,0];

   for ix=p:q;
       df=F(ix);

          if df == 13;
          break;
```

```
    elseif df == 32;
    break;
    elseif df == 48;
    dato=0;
    elseif df == 49;
    dato=1;
    elseif df == 50;
    dato=2;
    elseif df == 51;
    dato=3;
    elseif df == 52;
    dato=4;
    elseif df == 53;
    dato=5;
    elseif df == 54;
    dato=6;
    elseif df == 55;
    dato=7;
    elseif df == 56;
    dato=8;
    elseif df == 57;
    dato=9;
    elseif df == 65;
    dato=10;
    elseif df == 66;
    dato=11;
    elseif df == 67;
    dato=12;
    elseif df == 68;
    dato=13;
    elseif df == 69;
    dato=14;
    elseif df == 70;
    dato=15;
    end
    d(x+1)=dato;
    x=x+1;
    i=i+1;
end

  ra=0;
  rb=0;
  rc=0;
  rd=0;

    if x == 4;
       ra=d(1)*1000;
       rb=d(2)*100;
       rc=d(3)*10;
       rd=d(4)*1;
 elseif x == 3;
```

```
            ra=d(1)*100;
            rb=d(2)*10;
            rc=d(3)*1;
       elseif x == 2;
            ra=d(1)*10;
            rb=d(2)*1;
       elseif x == 1;
            ra=d(1)*1;
          end

      pp=pp+1;
      result=ra+rb+rc+rd;
      buffers; % Enlace con el programa buffers.m

  if tipo == 4;
      if pp < 10;
         z=3726;
         p=0+z;
         q=3+z;
         x=0;
         end
      if pp >= 10;
         z=3727;
         p=0+z;
         q=3+z;
         x=0;
         end
      if pp >= 100;
         z=3728;
         p=0+z;
         q=3+z;
         x=0;
         end
     if pp >= 1000;
         z=3729;
         p=0+z;
         q=3+z;
         x=0;
         end
     if pp >= 10000;
         z=3730;
         p=0+z;
         q=3+z;
         x=0;
         end
    end

 if tipo == 3;
      if pp < 10;
         z=3727;
         p=0+z;
```

```
        q=3+z;
        x=0;
        end
      if pp >= 10;
        z=3728;
        p=0+z;
        q=3+z;
        x=0;
        end
      if pp >= 100;
        z=3729;
        p=0+z;
        q=3+z;
        x=0;
        end
     if pp >= 1000;
        z=3730;
        p=0+z;
        q=3+z;
        x=0;
        end
     if pp >= 10000;
        z=3731;
        p=0+z;
        q=3+z;
        x=0;
        end
    end

if tipo == 2;
      if pp < 10;
        z=3721;
        p=0+z;
        q=3+z;
        x=0;
        end
      if pp >= 10;
        z=3722;
        p=0+z;
        q=3+z;
        x=0;
        end
      if pp >= 100;
        z=3723;
        p=0+z;
        q=3+z;
        x=0;
        end
     if pp >= 1000;
        z=3724;
        p=0+z;
```

```
        q=3+z;
        x=0;
        end
    if pp >= 10000;
        z=3725;
        p=0+z;
        q=3+z;
        x=0;
       end
  end

if tipo == 1;
    if pp < 10;
        z=3732;
        p=0+z;
        q=3+z;
        x=0;
        end
    if pp >= 10;
        z=3733;
        p=0+z;
        q=3+z;
        x=0;
        end
    if pp >= 100;
        z=3734;
        p=0+z;
        q=3+z;
        x=0;
        end
    if pp >= 1000;
        z=3735;
        p=0+z;
        q=3+z;
        x=0;
        end
    if pp >= 10000;
        z=3736;
        p=0+z;
        q=3+z;
        x=0;
        end
    end
end
```

7. buffers.m

```
if taz(5) == min;
```

```
  if taz(6) > sec;
    flag=0;
    admision; % Enlace con el programa admision.m
  end
end

t1=toc;
if t1 >= 0.1;
  ii=ii+1;
  hh(ii)=n;
  hh4(ii)=p_4;  % Vector de servicio Conversacional
  hh3(ii)=p_3;  % Vector de servicio Streaming
  hh2(ii)=p_2;  % Vector de servicio Interactivo
  hh1(ii)=p_1;  % Vector de servicio Background
  dd(ii)=usuarios;  % Vector de peticiones de usuarios
  dd4(ii)=nB4;  % Vector de usuarios conversacionales
  dd3(ii)=nB3;  % Vector de usuarios Streaming
  dd2(ii)=nB2;  % Vector de usuarios Interactivos
  dd1(ii)=nB1;  % Vector de usuarios Background
  dd1234(ii)=nB1+nB2+nB3+nB4; % Vector que contiene a todos los usuarios
  kk=kk+1;
  tic
  if kk == 30;
    kk=0;
    desp_nBG=t1*30;
    n;
    despacho; % Enlace con el programa despacho.m
    yy;
    n;
    rdonBG=0;
    S=0;
    m=0;
    end
  end

   if result == 12;
     if p_4 > 0;
       for i=p_4:-1:1;
         buf_4(i+1)=buf_4(i);
       end
     end
         buf_4(1)= result;

   elseif result == 13;
        if p_3 > 0;
          for i=p_3:-1:1;
              buf_3(i+1)= buf_3(i);
          end
        end
              buf_3(1)= result;
```

```
    elseif result == 2;
          if p_2 > 0;
            for i=p_2:-1:1;
                buf_2(i+1)= buf_2(i);
            end
          end
                buf_2(1)= result;

    elseif result == 21;
          if p_1 > 0;
            for i=p_1:-1:1;
                buf_1(i+1)= buf_1(i);
            end
          end
                buf_1(1)= result;
      end

p_4= length (find (buf_4));
p_3= length (find (buf_3));
p_2= length (find (buf_2));
p_1= length (find (buf_1));
n=p_1+p_2+p_3+p_4;
porcentaje=(n*100)/((pop*4)+4);

if n >= nCR;
  nivel=1;
  elseif n < nCR & nivel == 1 & lim == 1;
       nivel=2;
     end

    if n <= nCR;
      Carga_del_Sistema = sprintf('Menor a nCR: %6.3f\n',porcentaje);
      if nivel == 2 & aho == 1;
        vez=vez+1;
        if vez == 1;
          segCR=clock;
          minCR=segCR(5);
          seguCR=segCR(6);
          t3=seguCR+1;
          despacho; % Enlace con el programa despacho.m
          yy;
          n;
          if t3 >= 60;
            minCR=minCR+1;
            t3=t3-60;
            end
          end
          if vez > 1;
            CRseg=clock;
            CRmin=CRseg(5);
            CRsegu=CRseg(6);
```

```
        if  CRsegu >= t3 & minCR == CRmin;
            CRmin;
            CRsegu;
            nivel=0;
            vez=0;
            lim=0;
            aho=0;
            n;
          end
      end
   end

elseif nCR < n & n < nCD;
      Carga_del_Sistema = sprintf('Entre nCR y nCD: %6.3f\n',porcentaje);
      CDsegu=0;
      CDmin=0;
      vas=0;

elseif n >= nCD;
      Carga_del_Sistema = sprintf('Mayor a nCD: %6.3f\n',porcentaje);
      vas=vas+1;
      if vas == 1;
        segCD=clock;
        minCD=segCD(5);
        seguCD=segCD(6);
        t2=seguCD+3;
        lim=1;
        if t2 >= 60;
          minCD=minCD+1;
          t2=t2-60;
          end
        end
        if vas > 1;
          CDseg=clock;
          CDmin=CDseg(5);
          CDsegu=CDseg(6);
          if CDsegu >= t2 & minCD == CDmin;
            CDmin;
            CDsegu;
            vas=0;
            aho=1;
            n;
            while n >= nCR;
                 despacho; % Enlace con el programa despacho.m
                 yy;
                 n;
              end
            rdonBG=0;
            S=0;
            m=0;
          end
```

```
        end
      end
```

8. despacho.m

```
desp=(rdodBm*100)/(-106.98);
yy=round(desp);

for ff=1:yy;

  for i=p_4:-1:p_4-3;
     if  i > 0
        buf_4(i)=0;
     end
  end

  for i=p_3:-1:p_3-2;
     if i > 0
       buf_3(i)=0;
     end
  end

  for i=p_2:-1:p_2-1;
     if i > 0
       buf_2(i)=0;
     end
  end

  i=p_1;
  if i > 0
    buf_1(i)=0;
  end

p_4= length (find (buf_4));
p_3= length (find (buf_3));
p_2= length (find (buf_2));
p_1= length (find (buf_1));
n=p_1+p_2+p_3+p_4;
end
```

9. desconectar

```
    usuarios;
    tiposusuarios=nB4+nB3+nB2+nB1;
```

```
      if  tipo == 4;
         TS_C1=TS_C1-TS_C;
         CaC=CaC-TS_C;
         elseif tipo == 3;
               TS_S1=TS_S1-TS_S;
               CaC=CaC-TS_S;
               elseif tipo == 2;
                     TS_I1=TS_I1-TS_I;
                     CaC=CaC-TS_I;
                      elseif tipo == 1;
                           TS_B1=TS_B1-TS_B;
                           CaC=CaC-TS_B;

                        end
       CaC
       usuarios=usuarios-1;
       veces=veces-1;

for zz=1:clas;
   reales=isreal(sumclas(zz));
   if reales == 1;
     clastipo=sumclas(zz);
     desconectar=randperm(2);   % Aleatoriamente decide si se desconecta o sigue
conectado
      elige=desconectar(1);     % Decision tomada por el usuario
      if elige == 1;
        responde = ['Desconectar'];
        usuarios=usuarios-1;
        sumclas(zz)=sumclas(zz)+1i;
        if  clastipo == 4;
           TS_C1=TS_C1-TS_C;
           CaC=CaC-TS_C;
           nB4=nB4-1;
           elseif clastipo == 3;
                 TS_S1=TS_S1-TS_S;
                 CaC=CaC-TS_S;
                 nB3=nB3-1;
                 elseif clastipo == 2;
                       TS_I1=TS_I1-TS_I;
                       CaC=CaC-TS_I;
                       nB2=nB2-1;
                       elseif clastipo == 1;
                             TS_B1=TS_B1-TS_B;
                             CaC=CaC-TS_B;
                             nB1=nB1-1;
                        end

       elseif elige == 2;
             responde = ['No Desconectar'];
             end
               else
```

```
        responde = ['Ya Esta Desconectado'];
      end
end
tiposusuarios=nB4+nB3+nB2+nB1;
usuarios=tiposusuarios;
admision; % Enlace con el programa admision.m
```

ANEXO 4. MANUAL DE USUARIO

1. GUIA DE INSTALACIÓN.

Para poder ejecutar el tutorial demostrativo se debe tener instalado el siguiente software:

- ✓ Matlab 6.0 Release 12.
- ✓ Macromedia Flash MX.
- ✓ Macromedia Flash Player 6.

Posteriormente se deben ubicar los archivos correspondientes al tutorial en las siguientes carpetas:

- ✓ Archivos Matlab (extensión .m) en la carpeta C:\matlabR12\WORK.
- ✓ Archivos Macromedia Ejecutables (extensión .exe) en la carpeta C:\Archivos de programa\Macromedia\Flash MX\Players\Fscommand.
- ✓ Archivos Macromedia Shockwave Flash Object (extensión .swf) en la carpeta C:\UMTS
- ✓ Archivos por lotes MS-DOS (extensión .bat) en la carpeta C:\Archivos de programa\Macromedia\Flash MX\Players\Fscommand.
- ✓ Archivos de texto (extensión .txt) en la carpeta C:\matlabR12\WORK.

Una vez realizado el anterior procedimiento se procede a ejecutar el tutorial demostrativo.

2. PROCEDIMIENTO DE EJECUCIÓN.

La ejecución del tutorial es muy sencilla, sólo basta con dar doble clic al archivo portada.swf ubicado en la carpeta C:\Archivos de programa\Macromedia\Flash MX\Players\Fscommand y a continuación disfrutar por cada uno de los ítems descrito en el mismo.

3. NAVEGACIÓN.

Una vez el usuario ubique, seleccione y ejecute el archivo portada.swf, la pantalla de navegación desplegada se muestra en la Figura 1.

Figura 1. Portada Tutorial

Finalmente el tutorial queda listo para iniciar la exposición en los diferentes temas incluidos. Para tal acción es necesario hacer clic en el botón con la literatura “Diseño de un emulador RRM para requerimientos de QoS en una red UMTS”. La Figura 2 representa lo mencionado.

Figura 2.Botón inicio tutorial

Posteriormente se listan siete (7) botones que le permiten al usuario escoger el tema por el cual desea iniciar la navegación del tutorial. Ellos son: Infraestructura, UTRAN, Calidad de Servicio, Diseño, Emulación, Proyecto Arrows y Enlaces. Lo anterior se representa en la Figura 3.

Figura 3. Menu principal

De acuerdo al botón de selección que el usuario escoja se presentará la pantalla con el tema relacionado. En ella encontrará un menú en la parte derecha con tres (3) opciones que le permiten retornar al menú anterior, en este caso el menú principal, o al glosario de términos empleados en el tutorial y finalmente el botón de salir de la aplicación. Ver Figura 4.

Figura 4. Herramientas tema seleccionado

Como funcionalidades adicionales se encuentran botones de desplazamiento a los siguientes y anteriores temas. Como punto adicional y de acuerdo al tema seleccionado existen botones de presión que permiten observar gráficas. Ver Figura 4 y Figura5..

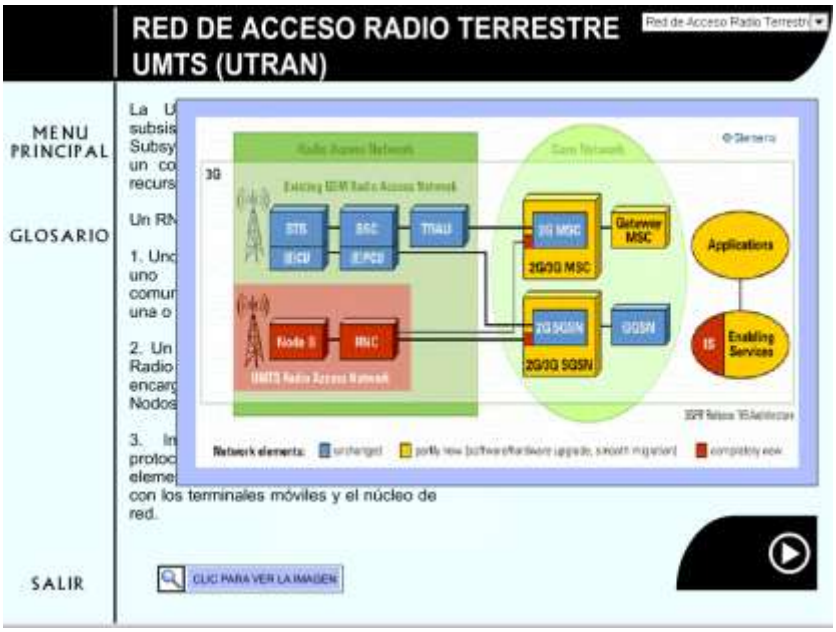


Figura 5. Botones de presión.

De igual forma en dicha pantalla el usuario puede desplazarse a los siguientes temas del tema escogido, por medio del menú de despliegue vertical, ubicado en la parte superior derecha de la misma, lo anterior se representa en la Figura 6.

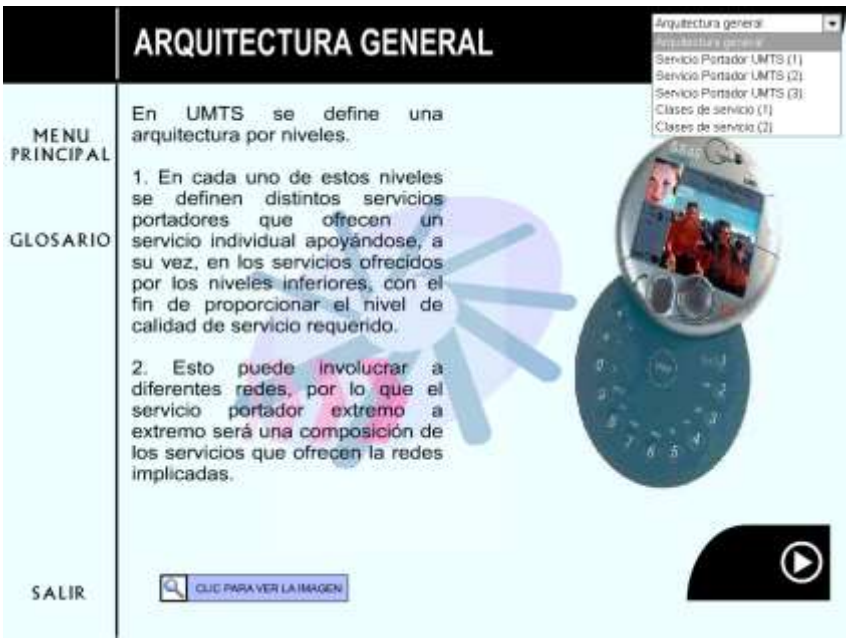


Figura 6. Menú de despliegue vertical.

Si el usuario presiona el botón de Glosario se abrirá una ventana como la representada en la Figura 7.

Figura 7. Glosario.

En la anterior pantalla se pide que el usuario presione alguna de las letras ubicadas en la parte superior, si para tal letra existen palabras asociadas cambia el fondo del dibujo de la pantalla por el glosario encontrado. Si el número de palabras encontradas son bastantes se le brinda la opción de cursores de desplazamiento ubicados a cada costado del dibujo. En la figura 8 se encuentra demostrado lo anterior.

Figura 8. Resultado y cursores de desplazamiento.

En caso de que la búsqueda sea negativo el tutorial desplegará el respectivo mensaje de palabra no encontrada (Figura 9).

Figura 9. Palabra no encontrada.

Otra opción que el usuario puede seleccionar es el botón salir, permitiéndole abandonar el tutorial.

El manejo de la navegación es similar en todos los temas a tratar en la pantalla del menú principal (Figura 3), excepto en Enlaces, el cual ofrece la opción de consultar ciertos sitios web, siempre y cuando se tenga una conexión constante a Internet. La figura 10 representa lo anterior.

Figura 10. Enlaces bibliográficos

Finalmente el principal botón del menú principal es Emulación, el cual le indica al usuario las condiciones iniciales que debe configurar para ejecutar la emulación, ellos la configuración del nivel umbral de admisión, el tiempo de emulación, los umbrales de detección de congestión y umbral de resolución de congestión. Una vez tenido en cuenta lo anterior existe un botón (“Ejecutar”) de ejecución al programa Matlab. Ver Figura 11.

Figura 11. Pantalla emulación

Ya teniendo el programa Matlab abierto, el usuario puede escribir en el prompt de la consola de comandos la palabra “inicio” y presionar “Enter”, momento que la emulación inicio su ejecución, actividad que depende del tiempo de duración de la misma. Para mayor entendimiento ver la Figura 13.

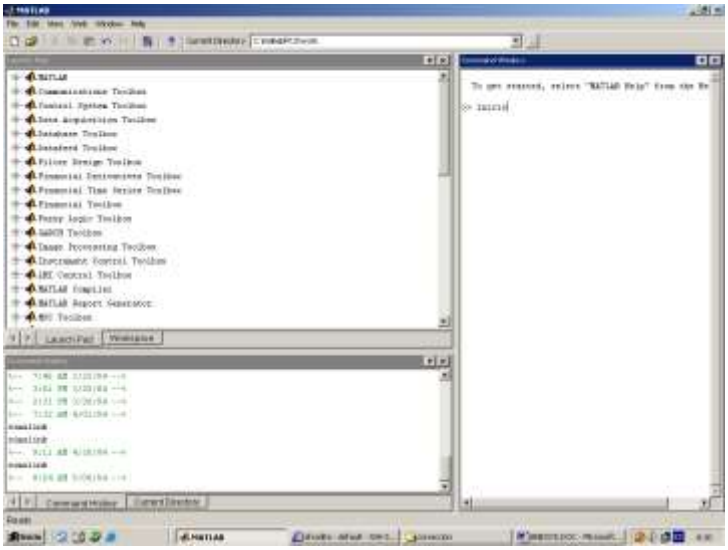


Figura 13. Entorno Matlab

Finalmente una vez transcurrido el tiempo de emulación se desplegarán las gráficas que indicarán cómo fue el comportamiento del sistema de gestión de recursos radio implementado para el presente trabajo de grado. De aquí en adelante es el usuario quien configura los parámetros para analizar el sistema.