DISEÑO Y ESPECIFICACION EN ESTELLE DE UN SISTEMA DE MENSAJERIA PARA LA UIS

BENJAMIN AUGUSTO PICO MERCHAN

Proyecto de Grado presentado como requisito parcial para optar al título de Magister en Informática.

Director

Doctor HERNAN PORRAS DIAZ

BUCARAMANGA
UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE CIENCIAS FISICO-MECANICAS
POSTGRADO EN INFORMATICA
1992

Nota de Aceptación

_____

_____

_____

_____
Presidente del Jurado

_____
Jurado

_____
Jurado

Bucaramanga, Julio 31 de 1992

AGRADECIMIENTOS

El autor expresa su agradecimiento a:

Dr. HERNAN PORRAS D., Director de este Proyecto.

Dr. JORGE H. RAMON S., Jefe de Servicios Académicos.

A las personas que laboran en la Sección de Sistematización y Procesamiento de Datos.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

INTRODUCCION

La Mensajería Electrónica se refiere al intercambio de información, mediante almacenamiento y reenvío, entre entidades utilizando terminales y procesos de mensajería.

Las tecnologías utilizadas para intercambiar correo han evolucionado considerablemente desde los comienzos mismos de la civilización empezando con la tabla de arcilla y las hojas de papiro hasta los últimos años en que surgieron los sistemas de mensajería computarizada. La evolución de estos últimos ha sido rápida desde su primitiva implantación como simples programas de utilidad sobre máquinas aisladas. Posteriormente, el desarrollo de redes de computadores de alcance local, regional, nacional e internacional, favoreció la expansión del correo electrónico como un servicio de valor agregado a la red en tal forma que hoy en día las redes de computadores frecuentemente ofrecen entre sus servicios a los usuarios, Sistemas distribuídos de correo. No obstante, esos

sistemas de mensajería pueden diferir entre sí en muchos aspectos así sea por el simple hecho de que provienen de diferentes fabricantes, es decir, se trata de "sistemas cerrados" no sujetos a ninguna normalización. Esta incompatibilidad puso de manifiesto la necesidad de normas internacionales que facilitasen su interconexión y la formación de una red de mensajería de alcance universal.

La aprobación formal por la asamblea plenaria del CCITT (Comité Consultivo Internacional de Telefonía y Telegrafía), en 1984 de la serie de recomendaciones X.400 definiendo los servicios de mensajería, marcó un hito fundamental que posibilita la implantación internacional del correo electrónico, culminando el camino trazado por el Telégrafo, el Telex y el Teletex.

Con base en la serie de recomendaciones X.400 del CCITT, la investigación que estamos presentando diseñó y especificó formalmente un sistema electrónico de mensajería que está siendo implementado sobre la red de computadores de alcance local instalada actualmente en el campus de la Universidad Industrial de Santander.

Durante las etapas de diseño y especificación se utilizaron las herramientas de la Ingeniería de Protocolos conocidas

como Técnicas de Descripción Formal (Formal Descriptión Techniques, FDT), concretamente el lenguaje de especificación formal ESTL (Extended State Transitión Language), más conocido como ESTELLE, desarrollado por la ISO (International Organization of Standardization) y recientemente llevado a la condición de norma internacional (ISO/TC 97/SC 21/WG 1).

Además de dotar a la Universidad con el diseño y la especificación de un servicio informático distribuído como el mencionado anteriormente basado en la aplicación de un conjunto de normas internacionales, se pretende introducir el estudio y la utilización de los conceptos y métodos de la Ingeniería de Software en el desarrollo de proyectos informáticos de comunicaciones.

Este proyecto, junto con los de implementación, forma parte de la línea de investigación en comunicaciones y sistemas distribuídos dirigida por los Doctores Hernán Porras Díaz y Jorge H. Ramón Suárez.

La implementación de este trabajo se está llevando a cabo mediante la realización de proyectos de grado por parte de algunos estudiantes de pre-grado como requisito parcial para optar al título de Ingeniero de Sistemas de la UIS.

En estos momentos, se están realizando los proyectos de grado titulados "Implementación del Módulo Despachador de Mensajes para el Servicio de Mensajería de la Red Local de la Universidad Industrial de Santander" realizado por Carlos Javier Duarte D., e "Implementación del Módulo Gestor de Asociaciones del Agente de Transferencia de Mensajes para el Servicio de Mensajería de la Red Local de la Universidad Industrial de Santander" realizado por Luz Adriana Valdés G. (1991).

También se encuentra en proceso de realización el proyecto de grado titulado "Diseño e Implementación del Agente Usuario de Correo Electrónico para el Servicio de Mensajería de la Red Local de la Universidad Industrial de Santander" elaborado por Olga Lucía Gómez F. y Rosa Aydee Granados M.

1. PRESENTACION

1.1 ANTECEDENTES

La necesidad del intercambio de información entre los hombres ha estado presente desde el comienzo de la civilización. Esta necesidad de comunicación hizo que se desarrollaran las primeras herramientas orientadas a satisfacerla tales como el lenguaje, la escritura y el uso de los diversos elementos para llevarla a cabo. Para ampliar el alcance de estos mensajes se emplearon sucesivamente las diferentes modalidades de transporte que el hombre ha utilizado y perfeccionado.

Luego, con el desarrollo de las sociedades y su creciente complejidad, aumentó el volumen de la información a transmitir y las distancias entre transmisores y receptores de los mensajes, lo cual trajo como consecuencia la necesidad de crear entidades encargadas de la recolección, distribución y entrega de los mensajes escritos, que valiéndose de la infraestructura de transporte disponible

en cada época, dieron origen a lo que hoy se denomina correo.

Actualmente con el avance de la tecnología, se han involucrado al correo nuevos medios de transmisión y de almacenamiento de mensajes. Es así como con el surgimiento de la informática, el computador empieza a jugar un papel importante en la administración de la información, y al asociarlo con los recursos técnicos disponibles para su interconexión, surge la disciplina que hoy se conoce como Telemática.

Todo esto produjo el surgimiento de redes de computadores de alcance tanto local como amplio y la consiguiente proliferación de sistemas de correo electrónico incompatibles entre sí ofrecidos por diversas compañias de telecomunicaciones y PTTs (Post, Telegraph and Telephone administration) interesadas en ofrecerlo como un servicio estándar a compañías y suscriptores individuales. Para evitar este caos, en 1984 el CCITT definió una serie de protocolos para lo que llamó MHS (Message Handling Systems), Sistemas de Gestión de Mensajes en su serie de recomendaciones X.400, que posteriormente fueron tomadas como base para la definición de uno de los protocolos del nivel de aplicación del modelo OSI (Open Systems

Interconection) de la ISO (International Organization for Standardization). Este protocolo del nivel de aplicación se conoce con el nombre de MOTIS (Message-Oriented Text Interchange Systems) en la terminología ISO/OSI y define completamente la aplicación de correo electrónico.

Específicamente, la serie de recomendaciones X.400 del CCITT pretende alcanzar la cobertura universal de los servicios de mensajería electrónica mediante su normalización. Estas normas se adaptan a la filosofía de diseño implícita en el Modelo de Referencia para la Interconexión de Sistemas Abiertos, OSI, de la ISO, que estructura los sistemas de comunicaciones en siete niveles: físico, enlace, red, transporte, sesión, presentación y aplicación. En este último nivel se sitúa el servicio de Correo Electrónico.

La instalación en el campus de la Universidad Industrial de Santander de una red de computadores de área local favoreció la posibilidad de diseñar e implementar servicios informáticos de valor agregado, que luego de una etapa de prueba y verificación en el entorno universitario, puedan ser posteriormente ofrecidos a otras Instituciones.

1.2 OBJETIVOS DEL PROYECTO

1.2.1 Objetivos Generales

- Introducir las técnicas de la Ingeniería de Protocolos como herramientas indispensables en el diseño, especificación y realización de sistemas informáticos distribuídos siguiendo las normas y recomendaciones de las organizacioanes internaciones de normalización.

- Aplicar las Técnicas de Descripción Formal al diseño y especificación de un Sistema de Tratamiento de Mensajes como servicio de valor agregado de una red de computadores.

1.2.2 OBJETIVOS ESPECIFICOS

- Definir un Entorno de Tratamiento de mensajes basado en la red de computadores de área local instalada en la UIS y en las necesidades de intercambio de información existentes en esta Institución, de acuerdo a la recomendación X.400 del CCITT.

- Seleccionar y definir los elementos de servicio que debe proporcionar dicho entorno a sus usuarios con base en la

recomendación X.401 del CCITT.

- Diseñar el Sistema de Tratamiento de Mensajes (MHS) para soportar el entorno anteriormente mencionado de acuerdo a las recomendaciones X.400, X.401, X.411 y X.420 del CCITT.

- Especificar el sistema MHS y sus protocolos de comunicación asociados, utilizando la técnica de descripción formal conocida como ESTELLE de acuerdo a la norma ISO/TC 97/SC 21 de la ISO.

1.3 ALCANCES DEL PROYECTO

El presente trabajo se centra en el diseño y presentación de la arquitectura de los módulos principales que conforman el llamado AGENTE DE TRANSFERENCIA DE MENSAJES, según la terminología de la serie de recomendaciones X.400 del CCITT. Específicamente, se hace énfasis en la especificación formal de los módulos llamados DESPACHADOR DE MENSAJES y GESTOR DE ASOCIACIONES, teniendo en cuenta que el primero de ellos realiza las principales funciones activas de un Agente de Transferencia de Mensajes y el segundo es el responsable de sus funciones de conexión, como se verá en los capítulos respectivos.

La etapa de implementación sobre la red local de la Universidad, será llevada a cabo mediante proyectos de grado de estudiantes de Ingeniería de Sistemas, quienes utilizando los entornos de desarrollo de sistemas provistos por los sistemas operativos UNIX y PRIMOS de los nodos de la red, producirán la codificación en lenguaje C a partir de las especificaciones en ESTELLE. Se espera en el futuro contar con un entorno de desarrollo más automatizado que, por ejemplo, genere directamente la codificación en algún lenguaje de programación como Pascal o C, tomando como entrada especificaciones codificadas en ESTELLE. Este entorno deberá contar además con herramientas que permitan validar la especificación, no sólo desde el punto de vista de la sintaxis, sino también su consistencia.

1.4 CONTENIDO

En este primer capítulo se han incluído aspectos básicos como: antecedentes, objetivos y alcances del proyecto, para que el lector tenga un conocimiento general sobre el tema tratado y se oriente sobre el contenido de este Informe.

Las bases teóricas fundamentales sobre las Técnicas de Descripción Formal se exponen en el capítulo 2 donde se dan

las definiciones básicas, sus características y su clasificación. La parte central de este capítulo está enfocada a la presentación del lenguaje de especificación ESTELLE, mostrando sus características más relevantes.

El capítulo 3 muestra en forma general los Sistemas de Gestión de Mensajes y presenta la serie de recomendaciones X.400 del CCITT, el modelo del Sistema de Mensajería allí propuesto, su descripción funcional y por niveles.

En el capítulo 4 se presenta el diseño propuesto para el sistema de mensajería electrónica de la UIS, sus alcances, los elementos de servicio considerados y la estructura de la especificación formal.

Por ultimo, el capítulo 5 hace un resumen de los resultados obtenidos y expone las Conclusiones que se pueden derivar y las líneas en que puede continuar esta investigación.

En el Anexo 1 se incluye la codificación fuente ESTELLE de la especificación, principal objetivo de este Proyecto.

2. TECNICAS DE DESCRIPCION FORMAL

2.1 DEFINICIONES Y GENERALIDADES

Al igual que todo proyecto de desarrollo de software, el diseño y realización de software de comunicaciones tiene los mismos problemas, aunque ciertos aspectos hacen de este un caso especial. El principal problema encontrado al plantear un diseño es su ambiguedad: dos realizadores pueden interpretar de manera diferente una misma norma o recomendación, haciendo que sus versiones no sean compatibles.

Ambos se ajustan a lo que ellos creen dice la recomendación, y sin embargo sus implementaciones no pueden comunicarse.

Si no se encontrasen mecanismos para solucionar o minimizar este problema, toda la concepción subyacente en los sistemas abiertos perdería así todo sentido.

Las principales organizaciones internacionales de normalización, como el CCITT y la ISO, se propusieron diseñar métodos con algún respaldo matemático en lo posible, que les permitiera expresar sus normas y recomendaciones sin ambiguedad. Estos métodos son conocidos como Técnicas de Descripción Formal o TDF.

Las TDF son herramientas para el diseño, análisis y especificación de sistemas informáticos, especialmente sistemas distribuídos; son métodos formales para definir el comportamiento de un sistema sin tener que recurrir a un lenguaje natural como el español. Con ellas se logra la generación de descripciones consistentes, completas, precisas y no ambiguas de los sistemas al iniciar su diseño. Posteriormente permiten el análisis y verificación rigurosas del diseño, después de su realización.

Las TDF además, se utilizan como un eficaz paso intermedio hacia la realización de un protocolo de comunicaciones ya que permiten describir este tipo de sistemas sin tener que ocuparse de detalles de bajo nivel que estas técnicas dejan relegados para su tratamiento durante refinamientos posteriores del problema. La importancia de esto último se hace evidente si tenemos en cuenta que los protocolos de comunicaciones forman el sistema nervioso de las redes de

teleproceso y por lo tanto son responsables de que los componentes del sistema trabajen en forma armoniosa.

2.2 CARACTERISTICAS DE UNA TDF

- Expresiva. Una TDF debe poder definir tanto las especificaciones de los protocolos como las definiciones de los servicios que prestan las entidades que se comunican por medio de esos protocolos.

- Bien Definida. Una TDF debe poseer un modelo formal apropiado para la verificación de las especificaciones y de las definiciones. El mismo modelo debe soportar la validación y las pruebas de conformidad de las realizaciones.

- Bien Estructurada. Una TDF debe ofrecer posibilidades para que la estructuración de la descripción sea de tal manera que incremente la legibilidad, la flexibilidad, el entendimiento, el análisis y el mantenimiento del sistema.

- Abstracto. Significa que una TDF debe poseer los siguientes dos aspectos de abstracción: independencia total de los métodos de realización, para que la técnica no coloque restricciones a los realizadores; y supresión de

detalles no relevantes respecto al contexto global. Estos dos aspectos de abstracción reducen la complejidad de las descripciones.

2.3 CLASIFICACION

Las técnicas que se han utilizado para especificar y diseñar sistemas informáticos pueden agruparse en tres grandes categorías:

- Modelos de Transición donde se emplean técnicas tales como las máquinas de estado finito y las Redes de Petri.

- Modelos orientados a lenguajes en los cuales se utilizan lenguajes formales y ciertos lenguajes de programación como PL/1 o Pascal con algunas extensiones.

- Modelos híbridos entre lenguajes y uso de técnicas de máquinas de estado finito.

Las principales FDT's son SDL, Estelle, LOTOS y FAPL.

SDL (Specificatión Definition Language) fue desarrollado por el CCITT y propuesto como la recomendación X.250, mientras que Estelle y LOTOS fueron desarrollados por la

ISO. FAPL (Formal And Protocol Language) fue desarrollado por la IBM y utilizado para describir formalmente su arquitectura de comunicaciones SNA. Es un lenguaje procedimental de alto nivel derivado del PL/1 que puede adaptarse a la concurrencia y al modelamiento de máquinas de estado finito.

Por su parte, LOTOS está basado en un modelo lógico de ordenamiento temporal de eventos que modela el intercambio de mensajes en la comunicación de procesos mediante técnicas de ordenamiento temporal.

En el subcapítulo siguiente se presentan los conceptos generales sobre Estelle, que es la TDF utilizada para la especificación de este proyecto, subcapítulo extractado básicamente del tutorial sobre Estelle publicado por la ISO como addendum a la norma respectiva [ISO88a].

2.4 ESTELLE

El lenguaje de especificación ESTL (Extended State Transition Language), mas conocido como Estelle, es una técnica de descripción formal basada en un modelo de máquinas de estado finito extendidas, modelos que se

conocen como autómatas extendidos. Fué desarrollado por la ISO como resultado delos trabajos de su Comité Técnico 97, Subcomité 16, Grupo de trabajo 1 (TC97/SC16WG1), subgrupo B y es actualmente la norma ISO 9074 [ISO88b].

Los autómatas en que se basa Estelle frecuentemente se pueden representar como grafos compuestos por nodos, ramas y etiquetas. Los nodos representan los estados y las ramas representan las transiciones de un estado a otro. Las etiquetas sobre las ramas que conectan los nodos señalan una entrada o evento que inicia la transición.
También suelen representarse los autómatas con tablas formadas por los nombres de los estados, las transiciones, los criterios para la transición de un estado a otro y las acciones que se toman cuando se dispare la transición. Tanto grafos como tablas, son ayudas que frecuentemente se utilizan para el diseño, antes de hacer la especificación formal en Estelle.

2.4.1 Descomposición Modular e Instancias de Módulos

En Estelle, una especificación está formada por un conjunto de componentes que se comunican entre sí. Cada componente es una instancia de un módulo definido en la especificación Estelle mediante una definición de módulo, llamándose

*instancias de **módulos** a estos componentes. Por comodidad, se utilizará en adelante el término **módulo** para referirse a una instancia de módulo, mientras no haya confusión.*

El comportamiento de un módulo lo define un conjunto de transiciones (de un modelo de transición de estados extendido) que el módulo puede ejecutar, mientras que la estructura interna se especifica por medio de la definición de submódulos (módulos hijos) del módulo y sus interconexiones.

*Atendiendo a su comportamiento, el módulo puede ser **activo** o **inactivo**. Si presenta por lo menos una transición en su parte de definición de transiciones, el módulo es activo. En caso contrario, es inactivo.*

2.4.2 Interfaz de Comunicaciones

La interfaz de comunicaciones del módulo, para interactuar con su medio-ambiente y con los módulos hijos, se define utilizando tres conceptos:

- *Puntos de interacción.*
- *Canales*
- *Interacciones o mensajes*

*Los **puntos de interacción** son un conjunto de puntos de acceso para entrada/salida que el módulo puede tener. Hay dos clases de puntos de interacción: **externos** e **internos**. Existe un **canal** asociado a cada punto de interacción.*
*Cada canal, a su vez, define dos conjuntos de **interacciones**. Estos dos conjuntos están compuestos por las transiciones que pueden transmitirse y recibirse, respectivamente, a través del punto de interacción respectivo. Las interaccioanes son eventos abstractos (mensajes) intercambiados con los módulos hijos (a través de los puntos de interacción internos), y con el medio-ambiente del módulo a través de los puntos de interacción externos.*

Informalmente, un módulo puede representarse gráficamente como una caja (rectángulo) posiblemente con puntos en el borde (los puntos de interacción externos) y dentro (los puntos de interacción internos).
Puede adicionarse el nombre del módulo, su atributo clase (Véase 2.4) los nombres de los puntos de interacción y sus interacciones asociadas (entrantes o salientes).

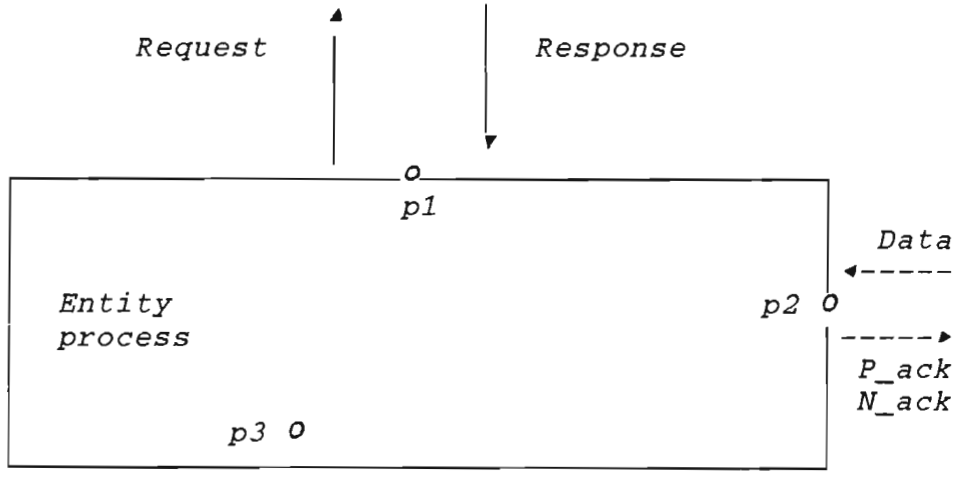


FIGURA 1. Representación gráfica de un módulo

2.4.3 ESTRUCTURACION MODULAR

En Estelle, los módulos que componen una especificación forman una estructura jerárquica en árbol de definiciones de módulos, por cuanto la definición de un módulo puede incluír definiciones de otros módulos.

La estructura jerárquica en árbol de módulos puede representarse gráficamente como en la figura 2. Los módulos se representan con cajas. La relación padre-hijo se representa mediante una línea o anidando las cajas.

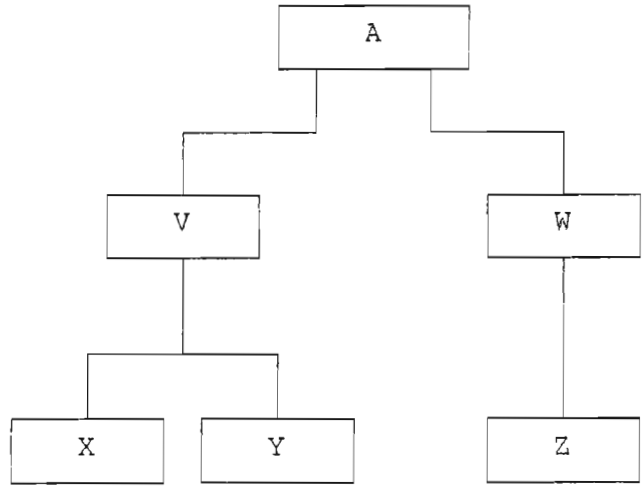
La raíz del árbol jerárquico o la caja grande que encierra a las otras, es el módulo especificación (principal) y representa toda la especificación. Siempre existe una y solo una instancia del módulo especificación.

La estructura jerárquica en árbol de los módulos constituye un patrón para cualquier jerarquía de instancias de módulos. La posición jerárquica de una instancia de un módulo corresponde a la posición de la definición del módulo en dicho patrón. Por definición el módulo especificación corresponde a una instancia de módulo. A cualquier otro módulo le puede corresponder cualquier número de instancias y este número puede cambiar dinámicamente.

La estructura jerárquica en árbol de las instancias de los módulos que corresponde a la estructura jerárquica en árbol de los módulos puede describirse como en la figura 2.

Los hijos de un mismo padre se llaman hermanos, por ejemplo los módulos V y W en la figura 2. La relación transitiva entre módulos de una jerarquía se llama ancestro y descendiente.

a)



b)

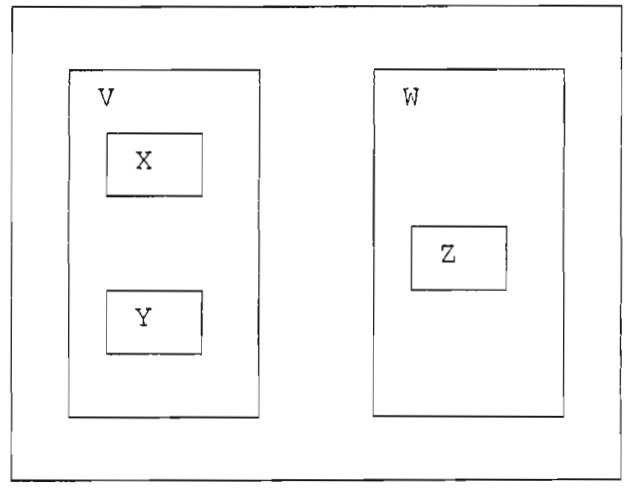
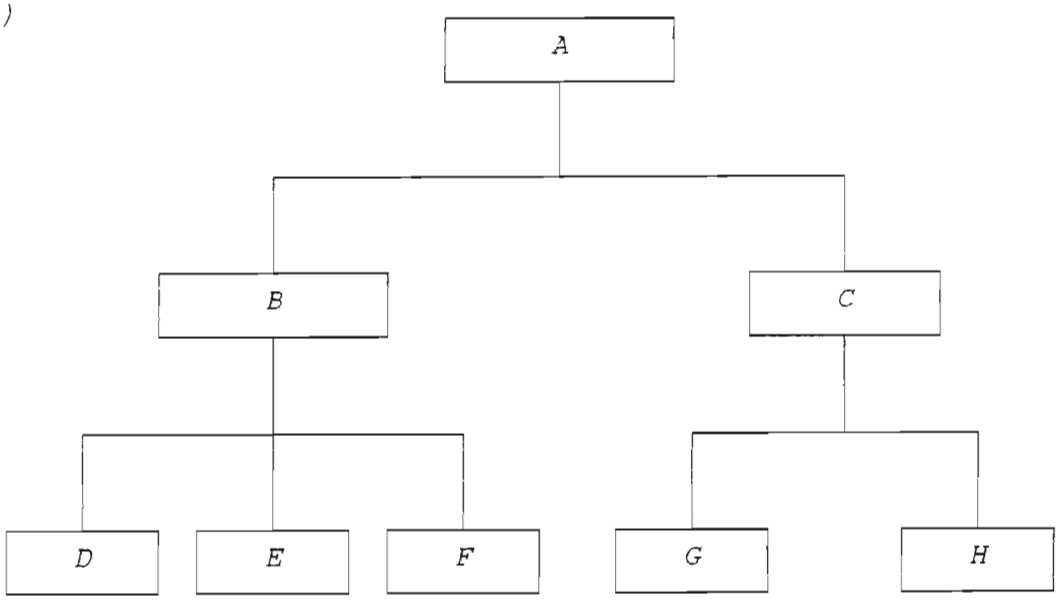
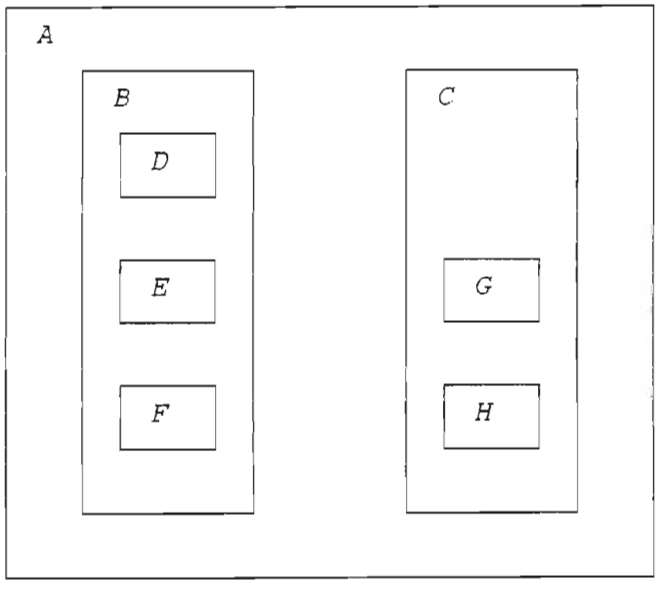


FIGURA 2 Representación gráfica de jerarquía de módulos

c)



d)



Continuación FIGURA 2 Representación gráfica de jerarquía de módulos

2.4.4 COMUNICACION

Las instancias de módulos dentro de una jerarquía pueden comunicarse intercambiando mensajes (interacciones) o compartiendo (en forma restringida) variables.

Una instancia de un módulo puede enviar una interacción a otra instancia de un módulo a través de una línea de comunicación previamente establecida entre dos puntos de interacción. La interacción recibida por una instancia de un módulo en un punto de interacción se añade a una cola FIFO ilimitada que está asociada a ese punto de interacción. La cola FIFO pertenece, ya sea a un solo punto de interacción identificándose como ***individual queue****, o puede ser compartida con otro(s) punto(s) de interacción del módulo llamándose en este caso,* ***common queue****.*

Las operaciones ***connect*** *y* ***attach*** *se utilizan en Estelle para especificar a qué módulos les está permitido intercambiar interacciones mediante las líneas de comunicación establecidas entre los puntos de interacción de los módulos.*

Una línea de comunicación entre dos puntos de interacción está compuesta de exactamente un segmento connect y cero o

más segmentos attach. Gráficamente, cada segmento de comunicación connect o attach puede representarse informalmente mediante segmentos de línea que unan los puntos de interacción de los módulos (Ver figura 3).

Se dice que dos puntos de interacción están ***atados*** *cuando un punto de interacción externo de un módulo está ligado a un punto de interacción externo de su módulo padre. En la figura 3 los siguientes pares de puntos de interacción están atados: (1,3), (2,8), (18,17), (9,11) y (10,13).*

Dos puntos de interacción están ***conectados*** *si: ambos son puntos de interacción externos de módulos hermanos (e.g., (1,9), (2,10), (4,7) y (5,6) en la figura 3), o uno es un punto de interacción interno de un módulo y el otro es un punto de interacción externo de uno de sus módulos hijos (e.g., (12,14) en la figura 3), o ambos son puntos de interacción internos o externos del mismo módulo (e.g., (18,19) y (15,16) en la figura 3).*

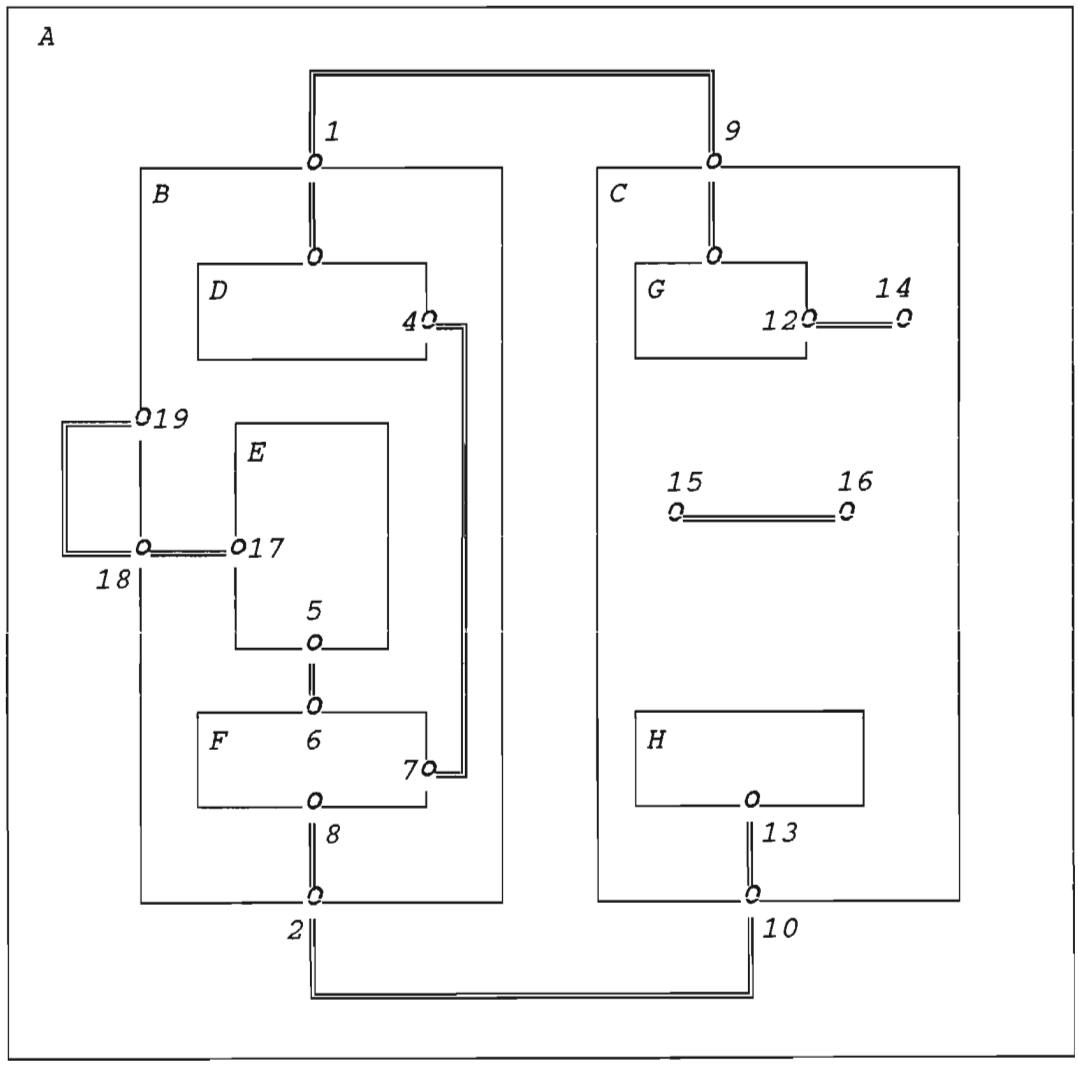


FIGURA 3 Instancias de módulos y líneas de comunicación

Debe notarse que la definición de un punto de interacción no determina cómo debe ligarse dicho punto de interacción. Dos instancias del mismo módulo pueden tener sus puntos de interacción ligados en forma diferente.

Así, en la figura 5, las instancias de módulo E1 y E2 pueden ser instancias del mismo módulo pero sus correspondientes puntos de interacción están ligados en forma diferente.

Las líneas de comunicación permiten que dos instancias de módulos cuyos puntos de interacción son puntos terminales de la línea, puedan comunicarse intercambiando mensajes en ambas direcciones a través de los puntos de interacción ligados. Por ejemplo, los puntos de interacción 8 y 13, o, 3 y 11 son puntos de interacción terminales de las líneas entre los módulos F y H, y, D y G, respectivamente en la figura 3. Los puntos de interacción 1, 9, 2 y 10 no son puntos terminales. Se debe notar que un punto terminal puede en un momento dado estar ligado a más de un punto de interacción terminal.

Si un módulo envía una interacción a través de un punto de interacción que es un punto terminal de una línea de comunicación, entonces esta interacción llega directamente

al otro punto terminal de la línea y se deposita en una cola ilimitada FIFO en el módulo receptor. Si un módulo envía una interacción a través de un punto de interacción que no es punto terminal de la línea de comunicación, entonces la interacción se descarta. Por lo tanto, en una especificación Estelle solo es posible la comunicación extremo a extremo entre los puntos de comunicación de los módulos.

Sin embargo, pueden existir simultáneamente varias líneas de comunicación entre los puntos de interacción de una instancia de un módulo dado y los puntos de interacción de otras instancias de módulos. En esta forma se especifica la comunicación irradiada (broadcast communication) enviándola simultáneamente en una transición a través de los puntos de interacción p[1], p[2], p[3] a los módulos A1, A2 y A3. En una especificación Estelle estos tres puntos de interacción pueden declararse como elementos de un arreglo.

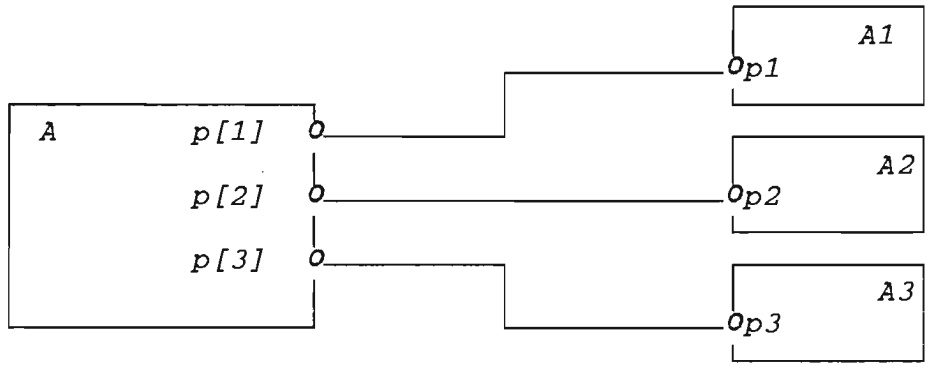


FIGURA 4 Comunicación irradiada

Ciertas variables pueden ser compartidas entre un módulo y su módulo padre. Estas variables deben declararse en el módulo hijo como variables exportadas y esta es la única forma en que se pueden compartir variables. Es imposible el acceso simultáneo a estas variables, tanto por el módulo como por su padre, porque la ejecución de las acciones del padre siempre tienen prioridad sobre las acciones de los hijos.

Sin embargo, el compartir variables no es la única forma de comunicación entre un módulo padre y su hijo. También pueden comunicarse intercambiando mensajes lo cual puede verse gráficamente en las líneas de comunicación entre los puntos de interacción (12,14) y (17,19) en la figura 3.

2.4.5 Paralelismo y no Determinismo

La forma en que las instancias de módulos se comportan respecto a otras depende estrictamente de la forma en que estén anidadas y de sus atributos.

Un módulo puede tener una de las siguientes clases de atributos:

- *systemprocess,*
- *systemactivity,*
- *process,*
- *activity*

  o puede no tener atributos.

Los módulos systemprocess o systemactivity son llamados módulos sistema.
Todas las instancias de un módulo tienen el mismo atributo definido para el módulo del cual son instancias.

Dentro de una jerarquía de módulos se deben observar los siguientes cinco principios sobre atributos:

a) Cada módulo activo, es decir, aquel cuya definición incluye al menos una transición, debe tener atributo.

b) Los módulos sistema no deben estar anidados dentro de un módulo con atributo.

c) Todo módulo cuyo atributo sea process o activity debe estar anidado dentro de un módulo sistema.

d) Los módulos con atributo process o systemprocess sólo pueden subestructurarse en módulos con atributo process o activity.

e) Los módulos con atributo activity o systemactivity sólo pueden subestructurarse en módulos con atributo activity.

No obstante, los módulos inactivos pueden tener atributos. También se debe notar que todos los módulos que incluyan un módulo sistema son inactivos y sin atributos, y que dentro de la jerarquía, estos son los únicos módulos sin atributos.

Los atributos systemprocess y systemactivity se utilizan para identificar sistemas comunicantes separados dentro de la especificación. Mas concretamente, la misma especificación puede tener uno de estos atributos y en este caso la especificación describe un sistema. Cada sistema

es un subárbol de módulos con raíces en el módulo sistema. El número de instancias de sistema dentro de una especificación es siempre fijo.

Esto se muestra en la figura 5, donde se asumen las siguientes convenciones para claridad en su presentación:

- Los módulos sistema o raíces del sistema y sus líneas de comunicación se muestran en líneas gruesas indicando que forman la arquitectura estática del sistema.

- Los módulos que encierran módulos sistema se representan con líneas punteadas.

- El resto de los módulos y líneas de comunicación se representan con líneas normales.

En la figura 5, el módulo A es el módulo especificación sin atributos, o módulo principal. Tiene dos hijos B y C ambos módulos sistema con atributos systemprocess y systemactivity, respectivamente. El módulo B tiene tres hijos: D y F con atributos process, y E con atributo activity. El módulo C tiene dos hijos G y H ambos con atributo activity. El módulo E tiene dos hijos con atributo activity. En la especificación de este ejemplo se

identifican dos sistemas. Cada sistema es un subárbol de instancias de módulos con raíces en el módulo sistema respectivo.

La definición del comportamiento de la especificación depende en buena medida de los atributos de los módulos.

Dentro de un sistema, se puede especificar uno de los dos posible comportamientos siguientes para las instancias del módulo de acuerdo al atributo asignado al módulo sistema:

- ejecución sincrónica paralela, con el atributo systemprocess ó

- ejecución no-determinística, con el atributo systemactivity.

Una instancia de módulo con su atributo actúa como un administrador-supervisor de las instancias de sus hijos. Dado que todos los módulos ascendentes de un módulo sistema no tienen atributos, se deduce que los módulos sistema no tienen ningún supervisor y no se tiene ningún medio de control de su respectivo comportamiento. Esto significa que un sistema se ejecuta en forma asincrónica paralela respecto a cualquier otro.

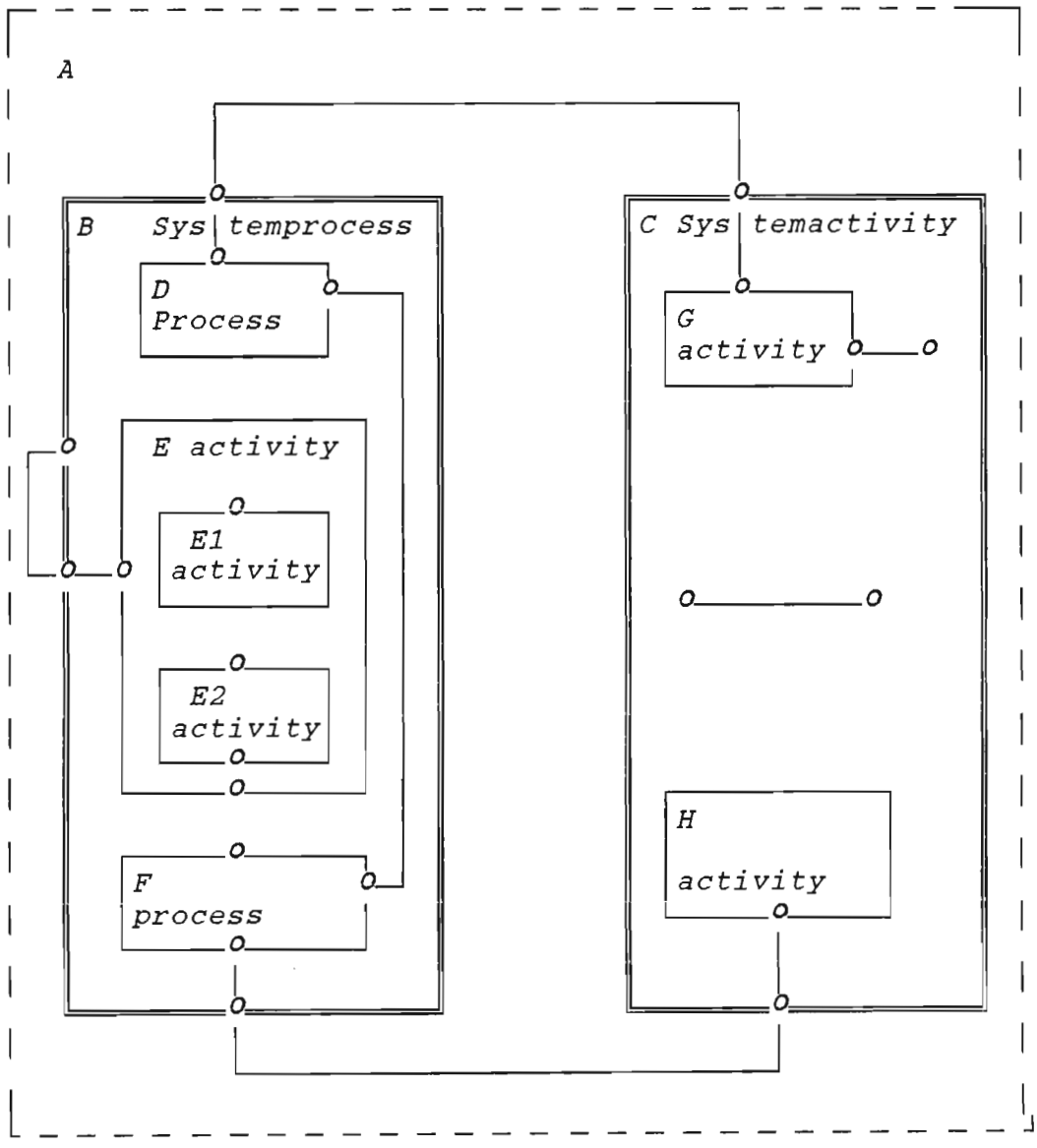


FIGURA 5 Instancias de módulos con atributos y conexiones

2.4.6 Dinamismo en un Sistema

Los módulos que agrupan módulos sistema son siempre inactivos ya que no tienen ninguna transición y por lo tanto no presentan ningún medio para cambiar dinámicamente la configuración del sistema. De lo anterior se deduce que las instancias de un sistema y sus interconexiones (conexiones de sus puntos de interacción) una vez creadas cuando se ejecuta la parte de inicialización de la especificación, quedan fijas por siempre, son invariantes. Sin embargo, debido a que dentro de la especificación pueden definirse diferentes formas de inicialización, es posible crear diferentes estructuras invariantes, estáticas.

Contrastando con lo anterior, las acciones (transiciones) de una instancia de un módulo activo dentro de un sistema pueden incluír instrucciones para crear y destruír sus hijos o para crear y destruír conexiones entre puntos de interacción de sus hijos o entre los puntos de interacción de la instancia del módulo y sus hijos. Por lo tanto, la estructura interna de cada sistema y las conexiones entre los puntos de interacción de sus submódulos pueden variar, es decir, ambas son dinámicas.

En la figura 5, por ejemplo, una acción de una instancia del módulo E puede crear o destruír las instancias de sus módulos hijos E1 y E2, o destruír la conexión entre E1 y E2 y la conexión entre F y E2 (comparar la figura 5 con la figura 3).

Finalmente, nótese que la posición jerárquica de cada instancia corresponde a la posición respectiva de la definición del módulo en la especificación Estelle, aunque el número de instancias de un módulo específico puede cambiar en la estructura dinámica de la especificación.

2.4.7 Comportamiento Interno

El comportamiento dinámico interno de un módulo Estelle está caracterizado en términos de un modelo de transición de estados extendidos no determinístico, que define un conjunto de estados, un subconjunto de estados iniciales y una relación próximo estado.

En general, un estado extendido es una estructura compleja compuesta de varios componentes tales como: valor del estado de control, valor de las variables, el estado de la estructura interna del módulo (instancias de submódulos, conexiones entre puntos de interacción, etc.), y los

contenidos de las colas FIFO asociadas a los puntos de interacción.

Los estados iniciales de una instancia de un módulo se definen en la parte de inicialización de la definición del módulo.

La relación próximo estado de una instancia de un módulo se define mediante el conjunto de transiciones declaradas en la parte de transiciones de la definición del módulo. Cada definición de transición contiene las condiciones necesarias para habilitar la ejecución de la transición, y una acción a realizar cuando ésta se ejecute. Una acción puede cambiar el estado de la instancia del módulo y puede enviar interacciones al entorno del módulo. Se utilizan instrucciones compuestas del lenguaje de programación Pascal para definir las acciones de las transiciones.

Una vez que ha empezado la ejecución de la transición, no se puede interrumpir, y conceptualmente, no se pueden observar resultados intermedios. Se dice entonces, que la ejecución de una transición por una instancia de un módulo es una operación atómica.

El comportamiento global de una especificación está basado

en una semántica operacional especialmente definida y que aparece descrita en el documento de la ISO que propone a Estelle como norma internacional. [ISO88b].

En esta semántica operacional se describe la forma en que se generan las secuencias de transiciones dependiendo del atributo de los módulos y sus anidamientos.

3. SISTEMAS DE GESTION DE MENSAJES

En este capítulo se hace una descripción de los Sistemas de Gestión de Mensajes empezando con un breve recuento histórico sobre sus inicios y evolución que desemboca en la publicación por parte del CCITT de la serie de recomendaciones X.400. A continuación se presentan estas recomendaciones dando una breve descripción de ellas que se tomaron como base para la elaboración de esta tesis. Continúa este capítulo presentando el modelo del sistema de mensajería según se describe en las recomendaciones X.400 del CCITT que incluye su descripción funcional y por niveles.

3.1 RECUENTO HISTORICO

Durante toda la historia de la civilización, el hombre ha necesitado transferir información de un lugar a otro. Mejorar esta transferencia ha implicado siempre la búsqueda de soluciones al problema de salvar la distancia y de mantener el registro de la información.

Esto nos muestra que en toda transferencia de información intervienen dos elementos indispensables: el medio de transporte y el medio de almacenamiento. Hasta mediados del siglo XIX, el único medio de transporte disponible para la información fué el hombre mismo, ya fuese a pie, a caballo, en carreta, etc. y el único medio de almacenamiento fue la escritura física (sobre tabletas de arcilla, papiro, papel, etc.) de la información.

Cuando la información que se transfiere está escrita, recibe el nombre de ***correo*** *que puede ser una carta, un periódico, un recibo, etc. Así, el servicio comercial de reenvío de correo a caballo aparece en la Roma Antigua, utilizando su extensa red de vías. El servicio postal se establece en Inglaterra durante el siglo XVI, y más tarde se extiende sobre Europa y Norteamérica, cuando surge como un servicio respaldado por el Estado, ofreciéndose desde finales del siglo XVIII un servicio postal normalizado cubriendo diversos países y grandes distancias.*

A medida que estas sociedades progresan, aumenta la necesidad de comunicaciones más económicas, rápidas y eficientes. Esto conduce a tratar de utilizar la electricidad, recién descubierta entonces, como medio para alcanzar esos objetivos y así, hacia 1830 se hace la

primera demostración de telecomunicación eléctrica permitiendo que en 1844, en Norteamérica comiencen las comunicaciones telegráficas entre ciudades. Durante la segunda mitad del siglo XIX las líneas telegráficas se extienden por casi todo el mundo, atravesando los mares por medio de cables submarinos, el primero de los cuales es tendido hacia 1866, convirtiéndose muy pronto el telégrafo en un servicio Universal, con una transmisión punto a punto en tiempo real, sin ninguna inteligencia actuando sobre el sistema de transmisión, excepto la humana empleada en la codificación y decodificación del mensaje depositado por el usuario. Con el telégrafo surge el Correo Electrónico y se modificó la naturaleza de las comunicaciones, pues el mensaje que hasta ahora era transportado físicamente sobre su medio de almacenamiento original, fué convertido (codificado) en impulsos eléctricos, transportado sobre cables y al final convertido (decodificado) a su forma escrita inicial.

El siguiente avance importante en telecomunicaciones fué el desarrollo del teléfono, hacia 1876, un servicio de comunicaciones bidireccional y directo ya que no requiere de codificación y decodificación humanas.

Luego se desarrollaron las redes de teleimpresoras como el

Télex, donde existe autoconmutación, pero la inteligencia y el control siguieron dependiendo de la decisión humana. En los últimos años surgieron otras formas de comunicación electrónica punto a punto tales como el ***Teletex*** *y el* ***Facsímil*** *que significaron mejora y diversidad en la calidad del servicio de comunicación. Hacia 1960, buscando mejorar la eficiencia en el transporte de los mensajes, se empieza a utilizar el computador en los sistemas telegráficos y de télex para la conmutación electrónica de mensajes.*

Todos los sistemas de interconexión electrónica mencionados hasta aquí, son conocidos como sistemas de comunicación directa porque siempre se requiere establecer un circuito entre dos terminales cuando se desea transferir un mensaje, ya que el terminal emisor no podrá enviar el mensaje si el terminal receptor está en comunicación con otro, o se encuentra desconectado

Esa desventaja se supera con el ***buzón electrónico,*** *que aparece en los años 1970 como resultado de la utilización del computador y su red de terminales como herramienta para la comunicación. Aquí, el usuario prepara su mensaje sobre una terminal conectada al computador central donde se almacena. El mensaje así depositado no se entrega*

directamente al terminal receptor sino que se almacena para ser enviado al usuario receptor quien lo podrá recibir en el momento que lo desee. Este modelo se llevó a las redes de computadores, permitiendo la comunicación de usuarios registrados en diferentes computadores interconectados entre sí, implantándose por primera vez en la red Arpanet, la red de computadores del Ministerio de Defensa de los Estados Unidos y se llamó Sistema de Mensajería basado en Computador-CBMS (Computer Based Message System).

Buscando normalizar el desarrollo de las redes de computadores y de los distintos protocolos necesarios para sus comunicaciones, en 1978 la Organización Internacional para la Normalización-ISO, a través de su Comité Técnico 97, establece un Modelo de Referencia para la Interconexión de Sistemas Abiertos [ISO84] que proporciona una arquitectura por niveles para el desarrollo de las normas de los sistemas de información distribuídos.

Con estas bases, y como ejemplo de protocolos y servicios del nivel de aplicación en el modelo OSI, la ISO produjo una norma referente al intercambio de mensajes textuales, conocida con el título de Sistema de Intercambio de Texto Orientado a Mensajes (Message Oriented Text Interchange System-MOTIS) [ISO85]. Sin embargo, fué el grupo de

trabajo 6.5 de la IFIP (International Federation of Information Processing) el primer grupo que desarrolló un modelo de Sistema de Gestión de Mensajes, buscando impulsar un servicio más cómodo, eficiente y estandarizado de correo electrónico con ámbito internacional.

Este modelo fué adoptado posteriormente, con algunas modificaciones, por el subgrupo VII del Comité Consultivo Internacional de Telefonía y Telegrafía, CCITT que en su reunión plenaria de Octubre de 1984 aprobó el conjunto de recomendaciones X.400 sobre los Sistemas de Gestión de Mensajes.

3.2 DEFINICIONES Y CONCEPTOS

Utilizaremos como definiciones del término ***Correo Electrónico****, las que se presentan en [POR87], donde se le define como:*

"La transferencia electrónica de información en forma de mensajes a través de un sistema de telecomunicación intermedio desde una parte originadora identificada a una o más partes destinatarias identificadas".

Allí también se lo define como:

"El nombre genérico para la comunicación no interactiva de

mensajes tipo texto, datos, imagen o voz entre un originador y un(os) destinatario(s) utilizando enlaces de telecomunicaciones".

Por su parte, en la terminología ISO/OSI se utiliza ***Servicio de Transferencia de Mensajes*** *para referirse al término Correo Electrónico puesto que desde el punto de vista del nivel de aplicación del Modelo de Referencia y teniendo en cuenta las definiciones anteriormente mencionadas, la información que se transfiere debe estar en unidades autocontenidas (mensajes).*
Los medios para distribuir el correo se pueden clasificar en varias categorías teniendo en cuenta si el servicio es total o parcialmente electrónico y si es o no un servicio normalizado. Según esto, podemos clasificarlos en:

- Sistemas de Mensajería Electrónica Convencional: Donde la transmisión y la entrega del mensaje es totalmente electrónica. En esta categoría aparece el Télex y el Facsímil Analógico que son sistemas de transferencia de mensajes sin almacenamiento intermedio entre emisor y receptor para transmitir mensajes tipo texto, en el caso del Télex, e información gráfica con el Facsímil Analógico. Obviamente, a esta categoría pertenecen los sistemas de conmutación de mensajes con almacenamiento y reenvío, cuya

aplicación principal son los Sistemas de Gestión de Mensajes según se encuentran definidos en las Recomendaciones X.400 del CCITT y que serán presentados con más detalle en el apartado siguiente.

- Sistemas Mixtos: Comprende aquellos sistemas como el Telegrama y el Facsímil donde la transmisión que ofrece el proveedor del servicio es electrónica, pero los mensajes son depositados por el originador y entregados al receptor en copia de papel.

- Sistemas Telemáticos. La aplicación de los computadores en las telecomunicaciones amplía las posibilidades de desarrollo de nuevos servicios que se pueden crear con base en estas nuevas tecnologías. Así, varios PTTs nacionales ofrecen varios de estos servicios, lo cual produce un conjunto de definiciones de nuevos servicios llamados telemáticos.

Uno de los primeros de este tipo es el Teletex, desarrollado por el PTT alemán como un servicio internacional para el intercambio automático de correspondencia basado en un esquema memoria a memoria utilizando una red de telecomunicaciones para el intercambio de texto, básicamente.

Para la transferencia de información graáfica extremo a extremo, se desarrolló otro servicio telemático llamado Facsímil utilizando técnicas digitales para la transmisión y la compresión de los datos.

En Francia se desarrolló el Videotex interactivo basado en un computador central y terminales sencillos no inteligentes para los usuarios abonados. Se trata de un sistema de distribución de información tipo datos, texto y gráfico al cual se accede a través de procedimientos de consulta de la información. Este sistema ofrece adicionalmente el servicio de intercambio de mensajes a través de almacenamientos y reenvío y con operaciones de depósito y recuperación.

- Sistemas No Normalizados. Esta categoría comprende todos los sistemas de correo electrónico diseñados y ofrecidos por los fabricantes de computadores y equipos de oficina o casas de software cuyos productos de mensajería no se acogen a las normas internacionales al respecto. Aquí se sitúan los sistemas CBMS mencionados anteriormente y los sistemas de correo que operan sobre las redes de computadores personales y de procesadores de texto conectados por medio de redes públicas o privadas. También los sistemas de correo de voz que son similares a los CBMS

pero que almacenan mensajes orales en lugar de datos.

3.3 RECOMENDACIONES X.400 DEL CCITT

El diseño del sistema de correo electrónico presentado en este trabajo se basa en las normas señaladas por el CCITT, compiladas en la serie de recomendaciones X.400 para los sistemas de mensajería electrónica.

Durante su reunión plenaria de Octubre de 1984 el CCITT aprobó las recomendaciones X.400, las cuales pretenden normalizar una aplicación particular de los sistemas distribuídos, el correo electrónico, y por lo tanto, según el modelo de referencia para la interconexión de los sistemas abiertos propuesto por la ISO, corresponden a su nivel séptimo, el nivel de aplicación.

La serie de ocho recomendaciones aspira a ser una base sobre la cual desarrollar la interconexión internacional de los sistemas de mensajería. Las recomendaciones suministran un modelo de referencia para mensajería interpersonal global de documentos de oficina, mensajes personales, télex, facsímil, gráficos y voz, con base en su almacenamiento y reenvío a través de un sistema de relevos cooperantes. [POR87].

Según el CCITT, las redes de mensajería deben proporcionar al menos dos servicios: la transferencia de mensajes y la mensajería interpersonal. La serie de recomendaciones X.400 definen estos servicios, la arquitectura de redes, la estructura de los protocolos y las opciones para la implementación de la interconexión de redes de mensajería.

3.3.1 Recomendación X.400

Define los servicios de mensajería que los PTTs públicos o privados deben proporcionar a los usuarios para intercambiar mensajes sobre la base de almacenamiento y reenvío. El servicio de mensajería interpersonal debe ofrecer la comunicación interpersonal, incluyendo la comunicación con los servicios telemáticos y télex aprobados por el CCITT. El servicio de transferencia de mensajes ofrece la transferencia de mensajes independientes de la aplicación.

Se define el Sistema de Gestión de Mensajes, MHS (Message Handling System). Este modelo se compone de elementos funcionales diferentes que proporcionan los servicios ofrecidos por medio de la cooperación en su trabajo.

Las otras recomendaciones de la serie toman como referencia

a este modelo el cual permite la posibilidad de diversas configuraciones físicas y administrativas.

En esta recomendación se presenta la descripción detallada de los elementos del servicio de mensajería, la arquitectura por niveles propuesta, según el modelo de referencia OSI de la ISO [CCI84a].

3.3.2 Recomendación X.401

Define los elementos de servicio básicos y las facilidades de usuario opcionales del servicio de mensajería interpersonal, y los elementos del servicio de transmisión de mensajes.

Dentro del MHS se especifican elementos del servicio inherentes llamados ***servicios de mensajería interpersonal y de transferencia de mensajes básicos.***

También se especifican otros elementos de servicio llamados en la recomendación ***Facilidades de Usuario Opcionales.*** *Estos pueden ser seleccionados por el usuario para cada mensaje depositado o por recibir, o por un período de tiempo especificado.*

Dado que un MHS debe poder instalarse sobre cualquier red física, los servicios de mensajería interpersonal y de

transferencia de mensajes aquí especificados, pueden ser ofrecidos por separado o en combinación con otros servicios de comunicación de datos y servicios telemáticos existentes ya normalizados [CCI84b].

3.3.3 Recomendación X.408

Define las reglas para la conversión de un tipo de información codificada en otro. Así, se especifican los algoritmos que debe utilizar el MHS en cada caso para hacer las conversiones entre los diferentes tipos de información codificada que se pueden emplear [CCI84c].

Los tipos de información codificada que se pueden utilizar en un MHS y su respectiva recomendación del CCITT, son:

- *Télex definido en la recomendación F.1.*
- *Texto IA5 definido en la recomendación V.3.*
- *Teletex definido en las recomendaciones F.200, S60 y S61.*
- *Facsímil definido en las recomendaciones T.4 y T30.*
- *Los formatos de Intercambio de texto definidos en la recomendación S.*
- *Videotex definido en la recomendación S.100.*
- *Documentos de formato simple definidos en la recomendación X.420.*

- *Voz.*

3.3.4 Recomendación X.409

Especifica la sintaxis de transferencia de presentación que utilizan los protocolos del MHS y el protocolo de Intercambio de Documentos de los servicios telemáticos. Para representar el intercambio de información entre entidades de aplicación, se utiliza la sintaxis de transferencia de presentación aquí definida.

En la recomendación también se especifica una notación normalizada para describir estructuras de datos complejas a partir de la cual se puede derivar la secuencia de bits de transmisión. Se define que cada elemento de información tiene asociado un tipo y un valor. El tipo es una clase de información, y el valor es la instanciación de la clase de información.

La codificación del valor de un cierto tipo es definida en la recomendación como una secuencia de bytes que forman tres campos: identificador, longitud y contenido. A su vez el contenido puede llevar la codificación del valor de otro tipo, por lo que está permitida la recursividad [CCI84d].

3.3.5 Recomendación X.410

Se especifica un protocolo general para invocar una operación distante y para informar el resultado de una solicitud recibida. Además especifica cómo el nivel de aplicación puede utilizar los servicios de los niveles de presentación y de sesión para la transferencia fiable de mensajes entre redes.

Inicialmente se describen en esta recomendación las operaciones distantes que se requieren para diseñar protocolos interactivos del nivel de aplicación, la notación y las unidades de datos de protocolos requeridos.

Se define el Servidor de Transferencia Fiable, RTS (Reliable Transfer Server) que es la parte de la entidad de aplicación responsable por transferir las unidades de datos del protocolo de aplicación, de forma fiable. También define las primitivas de servicio utilizadas. Finalmente indica los protocolos de los niveles inferiores del modelo de referencia aplicables para el funcionamiento del MHS [CCI84e].

3.3.6 Recomendación X.411

Define el servicio del subnivel de transferencia de mensajes. También especifica los elementos constitutivos de los protocolos del subnivel de transferencia de mensajes.

El servicio se describe mediante un conjunto de primitivas de servicio para:

- El establecimiento y la liberación de un diálogo entre el usuario y el subnivel.

- La modificación por el usuario, de los valores de sus parámetros de registro mantenidos por el subnivel.

- El depósito de un mensaje por parte del usuario, para ser transferido a uno o más usuarios destinatarios.

- La entrega de un mensaje a un usuario determinado.

- Determinar si un mensaje podría o no entregarse a uno o más usuarios destinatarios, si se depositara.

- La notificación de entrega en el caso de que el usuario

origen solicite confirmación explícita de la entrega del mensaje.

- La notificación de no entrega en el caso de que un mensaje no sea entregado satisfactoriamente al usuario deseado.

- El control temporal por el usuario, de los tipos y longitudes de mensaje que el subnivel le puede enviar.

Cuando los servicios ofrecidos no pueden ser soportados por funciones situadas en un solo nodo, la recomendación define el protocolo de transferencia de mensajes utilizado para lograr la cooperación de los nodos de la red de mensajería.

La recomendación describe finalmente el protocolo de depósito y entrega cuya función es permitir que un usuario que no se encuentra comunicado directamente con un nodo de la red de transferencia de mensajes pueda acceder a los servicios del subnivel de transferencia de mensajes [CCI84f].

3.3.7 Recomendación X.420

Especifica el contenido del protocolo que rige el servicio

de mensajería interpersonal, el cual básicamente consiste de:

- La definición de cada uno de los elementos del protocolo con su sintaxis y semántica.

- Las operaciones relacionadas con el intercambio de estos elementos.

- Las reglas que se deben seguir para utilizar los servicios del subnivel de transferencia de mensajes.

También define la representación utilizada para transmitir documentos de formato sencillo. Aquí se tiene en cuenta que las características físicas de los equipos utilizados por los usuarios pueden diferir entre sí, y por lo tanto la información en el dispositivo de salida del remitente puede aparecer en forma diferente, incluso ilegible, en el dispositivo de salida del destinatario. Este problema se soluciona permitiendo que el remitente especifique la estructura lógica de un documento pero no su estructura de presentación detallada. Asi, la responsabilidad de las decisiones de presentación corresponde al subnivel de mensajería interpersonal [CCI84g].

3.3.8 Recomendación X.430

Para permitir el intercambio de mensajes entre el mundo de usuarios teletex y X.400, en esta recomendación se especifica el protocolo de acceso para terminales teletex.

En la recomendación se definen:

- El modelo de acceso a teletex.

- Las acciones del protocolo utilizado por los terminales teletex.

- El funcionamiento de la unidad de acceso teletex.

- Los procedimientos de control.

- Los procedimientos de recuperación de errores.

- La codificación y formalización de la información de control.

3.4 MODELO DEL SISTEMA DE MENSAJERIA DEL CCITT

Básicamente, un sistema de correo electrónico está

compuesto de un originador, un receptor y un medio de transmisión. Su comportamiento y sus componentes se pueden describir así, según [POR87]:

- Existe una persona A que desea enviar un mensaje a una persona B.

- A se acerca a un terminal de mensajería, por ejemplo un computador personal, interactúa con éste a través de algún software instalado en el equipo terminal y solicita el envío del mensaje a B.
Para enviar un mensaje debe especificarse el nombre y la dirección del destino (receptor), el mendaje y un nombre y una dirección remitente (originador), tal como en un sistema de correo normal.

- La administración institucional que ofrece el servicio de mensajería, con las indicaciones del receptor dadas por el emisor, hace llegar el mensaje al terminal de mensajería señalado en la dirección destino, utilizando para ello un sistema de telecomunicaciones.

- Finalmente, B interactúa con el equipo terminal receptor para obtener la información enviada por A.

Los usuarios del modelo pueden ser personas o procesos (aplicaciones de computador). Siempre que un usuario envía un mensaje se le llama ***originador*** *y cuando lo recibe se denomina* ***receptor.***

El conjunto de procesos de aplicación que ayudan al usuario en la preparación, almacenamiento, envío y recepción de mensajes, se denomina ***Agente Usuario, UA*** *(User Agent).*

El usuario del MHS solamente ve su Agente Usuario quien actúa en representación suya frente al ***Sistema de Transferencia de Mensajes, MTS*** *(Message Transfer System), la red que efectúa la transmisión de los mensajes.*

En la figura 6 se observa una representación gráfica del modelo de mensajería recomendado por el CCITT.

Debido a que existe diversidad de intereses dentro de los usuarios potenciales, habrá varias clases de usuarios y el UA debe adaptarse a las solicitudes de los diferentes usuarios del MHS.

El ***UA*** *soporta únicamente una clase de usuario y las solicitudes de servicio de esta clase de usuarios, determina en gran parte su funcionalidad.*

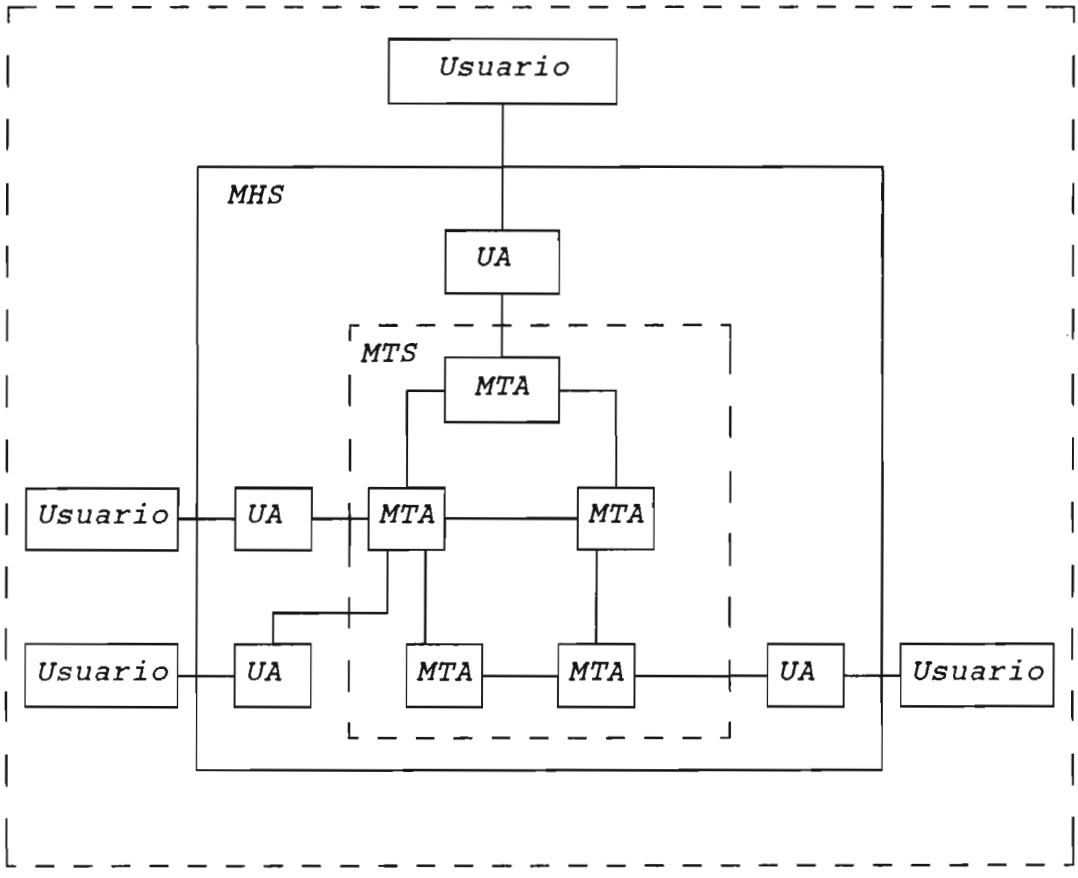


FIGURA 6. Modelo de mensajería según el CCITT

*Según el tipo de contenido de los mensajes que pueden manejar, los UA se agrupan en clases. Los UA que pertenecen a una clase dada se llaman **UAs cooperantes** debido a que colaboran entre sí para mejorar la comunicación entre sus respectivos usuarios.*
Un usario es asignado a un único UA, sin embargo varios usuarios pueden utilizar el mismo UA.

3.4.1 Sistema de Transferencia de Mensajes

*El MTS es una red de transmisión de mensajes cuyos nodos reciben el nombre de **Agentes de Transferencia de Mensajes, MTA** (Message Transfer Agent).*
Cada UA interactúa con un MTA que lo asiste y le recibe o transfiere un mensaje (Ver figura 6).
*El mensaje depositado al MTS por un usario, consiste de un **sobre** de depósito y un **contenido**. El sobre contiene la información que necesitan los MTAs para transferir el mensaje, y la especificación de los elementos de servicio solicitados al depositar el mensaje. El contenido es la información que el UA originador desea entregar a uno o mas UAs receptores y es transparente a los MTAs.*

El MTA que recibe el mensaje en primera instancia del UA originador, selecciona otro MTA con base en la información

de direccionamiento definida en el sobre de depósito. El mensaje es transferido al siguiente MTA en un ***sobre de remisión*** *con el contenido del mensaje. El sobre de remisión contiene información relativa a la operación de remisión y los elementos de servicio solicitados por el UA originador.*

Este proceso se repite las veces que sea necesario hasta que el mensaje llegue al MTA que da servicio al UA receptor. Este MTA destino transfiere (entrega) el contenido del mensaje al UA receptor, junto con el ***sobre de entrega*** *que contiene información relativa a la entrega del mensaje.*

Como en el correo convencional, sólo cuando el mensaje es entregado por el UA al usuario destino comienza la responsabilidad de éste. El depósito y la entrega, transfieren la responsabilidad de un mensaje desde el UA hasta el MTS y desde el MTS al UA.

La diferencia fundamental entre este modelo de mensajería y los demás sistemas de comunicación directa existentes, consiste en que en el modelo del CCITT el mensaje puede ser almacenado en MTAs intermedios para su posterior reenvío.

3.4.2 Estructura de un Mensaje

Como se vió en el apartado anterior, para un mejor manejo del mensaje, este se compone de sobre y contenido. A su vez el contenido comprende ***cabecera y cuerpo.*** *En el sobre se registra la información que el MTS necesita para enrutar el mensaje y su información puede ser alterada por los MTAs.*

3.4.2.1 Inscripciones en el Sobre

En las recomendaciones X.400 y X.401 el CCITT define el siguiente conjunto de campos para el sobre de un mensaje:

- Nombre del remitente del mensaje,

- Nombre(s) del(los) destinatario(s) del mensaje.
Los dos anteriores son de caracter obligatorio y los que siguen son opcionales:

- Prioridad: solicitada para la transferencia del mensaje,

- Destinatario alternativo permitido: en caso de no estar registrado el destinatario original,

- Fecha de entrega diferida: antes de la cual el mensaje no debe entregarse,

- Petición de informe de entrega del mensaje, y

- Supresión de aviso de no entrega del mensaje.

3.4.2.2 Inscripciones de la Cabecera

El cuerpo del mensaje va acompañado por un encabezamiento que consta de las siguientes inscripciones:

- Identificación del mensaje en el contexto del UA remitente,

- Referencias cruzadas de los identificadores de los mensajes con los que el presente está relacionado,

- Mensajes caducados: identificadores de los mensajes que el actual deja sin efecto,

- Indicador de auto-reenvío del mensaje por parte del UA del destinatario,

- Asunto: tema general del mensaje,

- Importancia: grado de urgencia con que se requiere la atención del destinatario,

- Sensibilidad: grado de intimidad del mensaje,

- Fuentes: lista de descriptores de los usuarios involucrados en la elaboración del mensaje,

- Destinatarios principales,

- Destinatarios con copia oculta: quienes reciben reservadamente una copia del mensaje,

- Petición de réplica: cuando se solicita una respuesta para este mensaje,

- Destinatarios de réplica,

- Fecha de réplica: límite en que se desea la réplica,

- Réplica a: identificador del mensaje al que se contesta,

- Petición de informe de recibo: solicitud al UA destinatario de un informe sobre la suerte del mensaje.

El cuerpo del mensaje debe ser especificado por el remitente usando una estructura lógica, es decir, dividiendo el documento en párrafos.

3.4.3 Arquitectura por Niveles

Siguiendo los lineamientos del modelo de referencia OSI de la ISO, las entidades y protocolos del MHS se ubican en el nivel de aplicación, como se muestra en la figura 7.

El modelo MHS del CCITT estructura el nivel de aplicación en dos subniveles:

- El nivel de Agente Usuario, UAL (User Agent Layer).

- El nivel de Transferencia de Mensajes, MTL (Message Transfer Layer).

En cada subnivel existirán las entidades que soportan la funcionalidad del nivel y por lo tanto los protocolos que permiten el diálogo entre estas entidades. Ver figura 8.

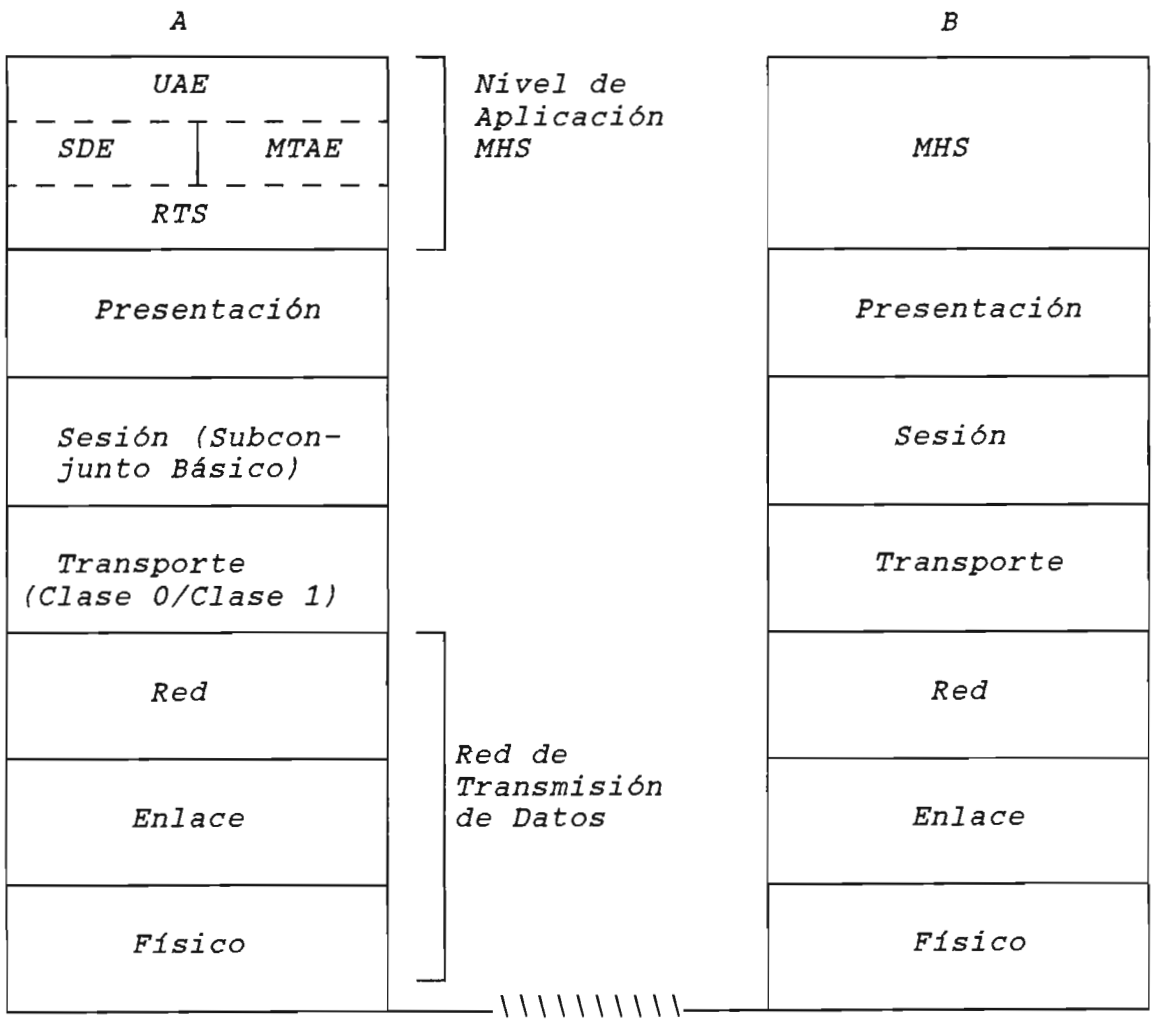


FIGURA 7. Arquitectura general del MHS según el CCITT

3.4.3.1 Nivel de Agente Usuario

El UAL aloja la Entidad Agente Usuario, UAE (User Agent Entity) en la cual se localizan todos aquellos aspectos de la funcionalidad del UA que se relacionan con la representación del contenido del mensaje, y también todos los aspectos del UAL necesarios para la cooperación de una clase específica del UAs.

Los usuarios del UAL pueden utilizar los servicios proporcionados por el MTL, además de los servicios ofrecidos por la cooperación de UAs.
Las normas y reglas que gobiernan el diálogo que permite la cooperación de UAEs constituyen el ***Protocolo Convenido, Pc.*** *Cada posible protocolo Pc está relacionado con una cierta clase de UAs cooperantes.*
En estos protocolos Pc están definidas la sintaxis y la semántica del contenido del mensaje que se puede transferir entre la clase de UAs.

Los servicios de mensajería interpersonal son soportados por una clase de UAs definida en las recomendaciones X.400. Para permitir la cooperación en esta clase de UAs, se utiliza una instanciación específica del protocolo Pc llamada ***Protocolo de Mensajería Interpersonal*** *definido en la recomendación X.420 del CCITT como el protocolo P2.*

Esta recomendación define los elementos del protocolo, las operaciones necesarias para el intercambio de estos elementos y las reglas que debe seguir una UAE para utilizar los servicios del MTL. El UA de mensajería interpersonal se construye con base en el protocolo P2 donde se definen la estructura del mensaje y la información de control del encabezamiento del contenido y su posible composición heterogénea (mezcla de multimedios: texto, voz, gráficos, etc.).

3.4.3.2 Nivel de Transferencia de Mensajes

El MTL describe la funcionalidad del MTA y proporciona el servicio del MTS.
El MTL contiene dos tipos de entidades:

- Entidad Agente de Transferencia de Mensajes, MTAE (Message Transfer Agent Entity) que proporciona las funciones requeridas para soportar los servicios del MTL, por medio de su cooperación con otras MTAEs.

- Entidad de Depósito y Entrega, SDE (Submission Delivery Entity) que hace accesible los servicios del MTL a aquellos UAs remotos no situados en el mismo lugar que la MTAE.
En el MTL existen dos protocolos: el de Transferencia de

Mensajes y el de Depósito y Entrega.

Algunos de los servicios proporcionados por el MTL se suministran por medio de funciones situadas en una sola MTAE. Pero para suministrar otros servicios, tales como la transmisión de mensajes, es necesaria la cooperación de dos o más MTAEs. Esta cooperación se consigue a través del ***Protocolo de Transferencia de Mensajes*** *también llamado protocolo P1 según las recomendaciones X.400.*

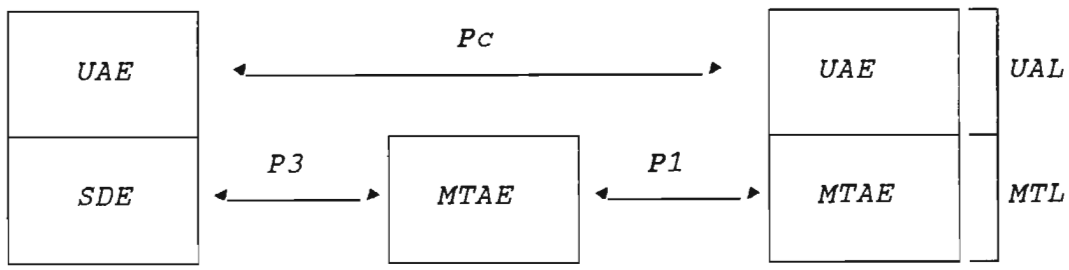


FIGURA 8. Entidades y Protocolos en el MHS

Los elementos o formatos de P1 se llaman Unidades de Datos del Protocolo de Mensajes, MPDUs (Message Protocol Data Units).

Hay tres tipos de MPDUs:

*- La **MPDU de usuario** que contiene la información del sobre y el contenido depositado por el usuario y transparente a través del MTS. El sobre contiene el identificador de la MPDU, el originador, la lista de destinos y la ruta seguida por el mensaje. En otras palabras, este tipo de MPDU se utiliza para transportar mensajes depositados por el usuario del nivel.*

*- La **sonda**, una MPDU que contiene solamente la información del sobre. Se utiliza para determinar si una MPDU de usuario puede ser entregada a un determinado destino.*

*- El **informe de entrega**. Se utiliza para determinar el éxito o fracaso en la entrega de una MPDU de usuario o de sonda sobre el MTS.*

Contiene la información del sobre y un contenido específico.

*La sonda y el informe de entrega se llaman **MPDUs de servicio** porque son utilizadas para transportar información sobre los mensajes cursados entre las MTAEs.*

La SDE para proporcionar el servicio de transferencia de mensajes a aquellas UAs remotos no situados en el mismo lugar que la MTAE, interactúa con otra MTAE para permitir el acceso al MTS. Esta interacción está regida por el ***Protocolo de Depósito y Entrega*** *o protocolo P3, según X.400.*

3.4.3.3. Relación con otros niveles del Modelo OSI

Las entidades del MTL no utilizan ningún servicio específico del nivel de presentación. Debido a esto las MPDUs son transmitidas por medio de un módulo situado en el nivel de aplicación, encargado de gestionar las primitivas de sesión dentro del MTA. Este módulo se llama ***Servidor de Transferencia Fiable, RTS*** *(Reliable Transfer Server) y ofrece un servicio mucho más reducido y sencillo orientado a gestionar las primitivas de sesión para evitar la pérdida parcial o total de una MPDU.*

El objetivo de esto es separar las funciones necesarias para asegurar la transmisión fiable de un mensaje entre dos MTAs vecinos, de aquellas funciones de encaminamiento de mensajes. Continuando con otros niveles mas bajos del modelo de referencia OSI de la ISO, el CCITT establece lo siguiente:

- Para dialogar con el nivel de sesión, el RTS utiliza un

subconjunto del servicio ofrecido por aquel y definido en la recomendación X.215 del CCITT que se conoce como Subconjunto Básico, BAS (Basic Activity Subset) del servicio de sesión definido por la ISO.

- En el nivel de transporte se utiliza al menos la clase cero y a lo más la clase 1 del servicio de transporte definido en la recomendación X.214 del CCITT.

- Se puede ofrecer el MHS sobre cualquier tipo de red siempre y cuando se permita la intercomunicación con otras redes siendo posible la utilización de una amplia variedad de protocolos para los niveles de red, de enlace y físico.

3.4.4 Descripción Funcional del MTA

El MTS puede ofrecer sus servicios debido a la colaboración de sus MTAs componentes y esta colaboración se logra a través de ciertas funciones que son realizadas por cada una de las respectivas entidades cooperantes, las MTAEs. Las principales funciones activas de una MTAE son: el enrutamiento, la generación de informes de entrega y la conversión de contenido.

3.4.4.1 Enrutamiento

Se permite a un usuario por medio de su UAE, especificar que su mensaje depositado se entregue a mas de un usuario destino servidos por diferentes UAEs destinatarios. Para poder ofrecer este servicio, la entrega de un mensaje dirigido a múltiples destinatarios, se necesita la participación de uno o mas MTAEs. Se pueden presentar las siguientes posibilidades:

- Los UAs destinatarios están registrados (servidos) por el mismo MTA que el UA originador.

- Los UAs destinatarios de un mensaje están registrados en un mismo MTA pero éste es diferente del MTA en el cual está registrado el UA originador.

- Los UAs destinatarios de un mensaje están registrados en varios MTAs.

Así, un mensaje puede ser transferido a través del MTS por diversos trayectos. Un MTAE que debe remitir un mensaje a destinatarios con diferentes trayectos, crea una copia del mensaje por cada uno de los trayectos y las remite a los MTAEs siguientes en los respectivos trayectos. Las operaciones de copia y bifurcación de los mensajes se repiten hasta que una

copia haya llegado a cada uno de sus MTAEs destino final, en donde el mensaje podrá entregarse a los UAs destinatarios servidos por este MTA.

En cada uno de los trayectos seguidos por el mensaje depositado, las MTAEs involucradas en la transferencia del mensaje son responsables de la entrega del mensaje a los UAs destinatarios servidos por estas, y de la retransmisión si es necesario.

*Una **ruta** es la información que describe el trayecto que ha de seguir un mensaje al recorrer el MTS desde el UA origen al UA destino.*
El MTS utiliza las direcciones origen y destino de los mensajes como una base para la elección de las rutas, porque dada una dirección destino, el MTS puede determinar una ruta hacia el UA denotado por esta dirección.

3.4.4.2 Generación de Informes de Entrega

El usuario del MTS puede solicitar que se devuelva al UA origen una notificación del éxito o fracaso de un mensaje depositado. La notificación se relaciona con el mensaje depositado mediante un identificador de mensaje.

Como se vió en el apartado 3.4.2.2, el protocolo de transferencia de mensajes facilita este servicio remitiendo MPDUs de servicio entre las MTAEs denominadas ***Informes de Entrega.***

Un informe de entrega positivo se genera cuando la MTAE ha entregado correctamente una copia del mensaje depositado a la UAE destinataria.

Se genera un informe de entrega negativo, cuando la MTAE determina que no es posible entregar una copia del mensaje a un determinado destinatario local, o no puede retransmitir dicho mensaje. Así, el informe de entrega negativo ocurre cuando no se puede realizar la entrega del mensaje a la UAE destinataria o, la transferencia del mensaje a la MTAE siguiente en la trayectoria.

Cuando un mensaje es remitido a múltiples destinos, la notificación afirmativa o negativa de la entrega puede referirse a cualquiera o a todos los UAs destinatarios. Por lo que los informes de entrega generados por diferentes MTAEs se transfieren independientemente, aunque se encaminen a través de una misma MTAE intermedia.

3.4.4.3 Conversión de Contenido

El MTS ofrece dos tipos de conversión de tipo de información

codificada: la conversión implícita y la conversión explícita.

En la ***conversión implícita*** *el servicio no lo solicita ni el UA origen, ni el UA destino. La UAE origen se limita a indicar los tipos de información codificada del mensaje en el momento del depósito. Si las capacidades de tipo de información codificada de la UAE destino permiten más de un tipo de conversión, se efectúa la más apropiada.*

La ***conversión explícita*** *permite a un UA origen pedir que el MTS realice una conversión especificada, como cuando un originador desea que su mensaje sea entregado a un destinatario en cierto tipo de codificación deseado.*

El protocolo de transferencia de mensajes tiene parámetros para determinar los tipos de información codificada y las conversiones necesarias podrán efectuarse a menos que estén explícitamente prohibidas por algunas UAEs.

Antes de efectuar una conversión la MTAE debe conocer los tipos de información codificada que puede recibir la UAE destino. Esto lo puede saber de tres formas diferentes:

- A través de información registrada por la UAE cuando se asocia a un MTA.

- A través del nombre del MTA que da servicio a la UAE destinataria.

- A través del nombre destinatario suministrado por el usuario remitente.

Cuando se entrega un mensaje después que se ha realizado la conversión, se informa al UA destino los tipos de información codificada original así como los tipos actuales de información codificada del mensaje.

4. ARQUITECTURA DEL SISTEMA DE MENSAJERIA DE LA UIS

Este capítulo presenta la descripción del sistema de gestión de mensajes diseñado y propuesto para operar sobre la red de computadores de alcance local de la UIS.

Inicialmente se describe el entorno computacional utilizado como base para diseñar, y en futuros desarrollos de este trabajo, implementar el sistema propuesto en esta tesis. Luego se fijan las limitaciones del diseño respecto a las recomendaciones del CCITT, se presentan los servicios que ofrecerá la red de mensajería a los usuarios de la Universidad y se describe el MTA propuesto, sus principales módulos y funciones. Finaliza el capítulo con la presentación de la especificación del MTA, sus módulos y entidades principales, sus autómatas, constantes, variables y transiciones propuestas.

4.1 ALCANCES DEL DISEÑO

Como se dijo en el apartado (3.4.3.3), un sistema MHS puede ser

implementado sobre cualquier tipo de red de computadores, siempre y cuando el sistema de interconexión ofrezca en todos sus sistemas los protocolos de comunicación requeridos en los niveles de sesión y transporte.

Teniendo en cuenta lo anterior y dado que en la infraestructura de redes de la Universidad no se cuenta aún con este soporte de comunicaciones en esos niveles, se hace necesario fijar límites muy precisos al sistema propuesto a fin de cumplir en la medida de lo posible, con lo dispuesto en la serie de recomendaciones X.400.

4.1.1 Red local de la Universidad

La red de computadores de alcance local actualmente instalada en la UIS, es una red conocida normalmente como ***red Ethernet*** *porque cumple con la norma especificada en la recomendación IEEE 802.3 que normaliza redes de área local, con topología bus compartido y método de acceso al medio conocido como CSMA/CD o Método de Acceso Múltiple por detección de portadora con detección de colisiones (Carrier Sense Multiple Access/Collision Detection).*

Su medio físico de transporte es un cable coaxial grueso de cobre que permite una rata de transferencia máxima de 10

megabits por segundo y tiene una longitud de 500 metros que recorren la parte central del campus de la Universidad, desde el laboratorio de microcomputación "Luis E. Arias" hasta el edificio del Postgrado en Informática.
Sobre este medio se pueden conectar cada cinco metros hosts (computadores) o terminales, utilizando para cada caso un ***Transceiver*** *y un cable transceiver.*

El transceiver es el dispositivo que permite la conexión física a nivel eléctrico con el cable coaxial y su componente básico es una aguja de metal que hace contacto con el núcleo conductor del cable.

Por su parte, el cable transceiver permite conectar el controlador Ethernet del host con el transceiver. Normalmente, las terminales o equipos no inteligentes, no se conectan uno a uno directamente a la red. En su lugar, se conecta un dispositivo llamado ***terminal server*** *que permite conectar hasta ocho equipos asincrónicos como terminales, impresoras, plotters, etc. Este equipo posee en ROM los protocolos para brindar acceso a los servicios de la red y solo necesita un transceiver y un cable transceiver para conectarse a ella.*

*Sobre esta infraestructura de red opera un conjunto de protocolos conocidos como TCP/IP (****Transport*** *Control*

Protocol/Internet Protocol) desarrollados por el Departamento de Defensa de los Estados Unidos y que son un estandard de hecho debido a su amplia utilización y aceptación. Normalmente, son los protocolos mas utilizados sobre redes locales Ethernet con computadores grandes tales como minis o mainframes.

Por el contrario, es difícil encontrar todavía en el mercado productos comerciales que implementen los protocolos de transporte y sesión de acuerdo a las normas internacionales de la ISO, para redes locales tipo Ethernet como la instalada en la Universidad.

TCP/IP ofrece en todos los hosts conectados a la red, los siguientes servicios básicos:

- Login remoto: acceso a cualquier computador multiusuario de la red desde terminales no conectadas a él.

- Transferencia bidireccional de archivos: intercambio de información en archivos hacia y desde cualquier host en la red que puede operar con cualquier tipo de sistema operativo.

- Servidor de terminales en la red: control y conexión de dispositivos no inteligentes conectados a la red por medio de

un terminal server.

Actualmente la red local de la UIS tiene conectados los siguientes nodos:

- Computador PR1ME EXL 316: multiusuario con sistema operativo UNIX System V release 3.

- Computador PR1ME 4050: multiusuario con sistema operativo PRIMOS revisión 22.

- Estación de trabajo para diseño APOLLO DN 3000: permite acceso a su propia red APOLLO TOKEN RING de cinco estaciones de diseño con sistema operativo DOMAIN/OS que ofrece adicionalmente los sistemas operativos UNIX System V release 3, UNIX Aegis, UNIX BSD 4.3 y emulación MS-DOS.

- Dos terminal servers con capacidad de ocho puertos asincrónicos cada uno.

Adicionalmente, por medio del computador PR1ME 4050 que opera en este caso como gateway, se conecta el computador PR1ME 9655, multiusuario con sistema operativo PRIMOS revisión 21. Estos dos computadores están comunicados por una red punto a punto controlada por un conjunto de protocolos propietarios de la

firma PR1ME llamados PRIMENET, operando sobre un enlace sincrónico a 9600 baudios sobre cable telefónico común y modems a la salida del respectivo puerto sincrónico del controlador inteligente de comunicaciones en cada computador.

Se espera en el curso del presente año conectar a la red un minicomputador TEXAS INSTRUMENTS modelo 1300, una red APOLLO TOKEN RING de dos estaciones de trabajo APOLLO DN 3500 y una estación IBM RS/6000.

4.1.2 Alcances de la Arquitectura Propuesta

Dadas las características mencionadas en el apartado anterior, se decidió no diseñar un RTS "real" según la recomendación X.410, sino utilizar uno "virtual", es decir, solo las interacciones descritas en dicha recomendación a través de un Gestor de Asociaciones.

Con esas interacciones, y utilizando la interfaz a TCP/IP provista por el sistema operativo de cada host (también llamada "librería de Sockets de TCP/IP"), el Gestor de Asociaciones puede comunicar el MTA respectivo con los niveles inferiores.

Con relación al UAL, se establece que todos los UAs son co-residentes con alguno de los MTA de la red MTS y por lo tanto

no se diseñó la entidad de Depósito y Entrega, SDE. Según se deduce del apartado (3.4.3.2) la SDE no es necesaria para el funcionamiento del nivel de transferencia de mensajes, MTL y solo se necesita en caso de existir UAEs remotos. El CCITT hizo esta previsión para el caso de existir en la red computadores o dispositivos limitados a poder servir únicamente como UA, por sus limitadas capacidades de proceso y/o almacenamiento.

4.2 ELEMENTOS DE SERVICIO OFRECIDOS

Para poder ofrecer un servicio a la vez útil y práctico, en lo referente a mensajería electrónica, el sistema que se propone debe tener las mismas prestaciones de cualquier buen sistema de correo, junto con características y facilidades que exploten a fondo las potencialidades que se obtienen con el uso de computadores y su interconexión.

Así, se propone y se decide que el presente sistema de mensajería ofrezca a sus futuros usuarios los siguientes elementos de servicio, según se definen en las recomendaciones X.400 y X.401 del CCITT:

- Gestión de Acceso: permite a un UA y a un MTA establecer accesos entre sí y gestionar información asociada con el

establecimiento del acceso tales como identificar y validar la identidad del otro. Proporciona una capacidad para que el UA especifique su nombre y mantenga la seguridad del acceso. Cuando se logra la seguridad de acceso a través de contraseñas, éstas pueden actualizarse periódicamente.

- Indicación de sello de tiempo de entrega: permite al MTS indicar a un UA destinatario la fecha y hora en la que el MTS entregó un mensaje.

- Identificación de mensaje: permite al MTS proporcionar a un UA un identificador exclusivo para cada mensaje depositado en el MTS, o entregado por éste. Los UAs y el MTS utilizan éste identificador para hacer referencia a un mensaje previamente depositado en relación con servicios tales como Notificación de entrega y no entrega. - Notificación de no entrega: permite al MTS notificar a un UA de origen que un mensaje no fue entregado al UA (o UAs) destinatario especificado. El motivo por el cual no se entregó el mensaje se incluye como parte de la notificación. En el caso de un mensaje multidestino, la notificación de no entrega puede referirse a cualquiera o a todos los UAs destinatarios a los cuales el mensaje no puede entregarse.

- Indicación de sello de tiempo de depósito: permite al MTS

indicar a un UA de origen y al UA destino la fecha y hora en que se depositó un mensaje en el MTS.

- Destinatario alternativo autorizado: permite a un UA de origen especificar que el mensaje depositado puede entregarse a otro destinatario que se indique.

- Entrega diferida: permite a un UA de origen ordenar al MTS que un mensaje en curso de depósito no se entregue antes de una fecha y hora determinada. La entrega podrá efectuarse lo más próxima posible a la fecha y hora especificadas, pero no antes.

- Cancelación de entrega diferida: permite que un UA originador ordene al MTS que cancele un mensaje de entrega diferida depositado satisfactoriamente con anterioridad. Es posible que la tentativa de cancelación no siempre tenga éxito. Los posibles motivos de fallo son: expiración del plazo de entrega diferida, o que el mensaje ha sido retransmitido ya dentro del MTS.

- Notificación de entrega: permite a un UA de origen solicitar que se devuelva al UA de origen una notificación explícita cuando un mensaje depositado ha sido correctamente entregado a un UA destino.

- Revelación de otros destinatarios: permite a un UA de origen, cuando deposite un mensaje con múltiples destinos, ordenar al MTS que revele los nombres de todos los demás destinatarios a cada UA destinatario cuando le entregue el mensaje. Los nombres revelados serán los suministrados por el UA de origen.

- Selección de grado de entrega: permite a un UA de origen solicitar que la transferencia por conducto del MTS se efectúe con carácter "urgente" o "no urgente", en lugar de "normal". Los períodos de tiempo definidos para las transferencias urgente y no urgente son respectivamente más corto y más largo que el definido para la transferencia normal.

- Entrega a múltiples destinos: permite a un UA de origen especificar que el mensaje depositado se entregue a más de un UA destinatarios. El servicio no implica la entrega simultánea a todos los UAs especificados.

- Prevención de notificación de no entrega: permite a un UA de origen ordenar al MTS que no devuelva una notificación de no entrega al UA de origen cuando considere que el mensaje depositado es inentregable.

- Devolución de contenido: permite que un UA de origen pida

que el contenido de un mensaje depositado se devuelva con cualquier notificación de no entrega.

- Sonda: permite a un UA averiguar si un mensaje particular podrá entregarse, antes de depositarlo efectivamente. El MTS proporciona la información de depósito y genera notificaciones de entrega y/o no entrega para indicar si podrá entregarse un mensaje con la misma información de depósito a los UAs destinatarios especificados.

- Retención para entrega: permite a un UA destinatario solicitar que el MTS retenga sus mensajes, para que sean entregados y se devuelvan las notificaciones en un momento posterior. El UA indica al MTS cuando estará indisponible para aceptar la entrega de mensajes y notificaciones, y también, cuando volverá a estar disponible.

- Destinatario de copias ciego: permite al originador proporcionar los nombres de uno o más usuarios adicionales que son destinatarios deseados del mensaje que se envía. Estos nombres no se revelan a los destinatarios primarios ni a los destinatarios de copias.

4.3 ESTRUCTURA MODULAR DEL MTA

Antes de continuar con la presentación del diseño del MTA, es necesario ubicar su entorno de funcionamiento. Para esto, se empieza con la definición de la arquitectura del MTA sobre la cual se elaboró el presente trabajo.

Una MTAE que esté en capacidad de ejecutar el protocolo de transferencia de mensajes P1, se puede descomponer en los siguientes módulos, según se ilustra en la figura 9:

- Despachador de mensajes.
- Archivador de mensajes con entrega diferida.
- Gestor de asociaciones.
- Registrador estadístico de datos.
- Parte del gestor del sistema.

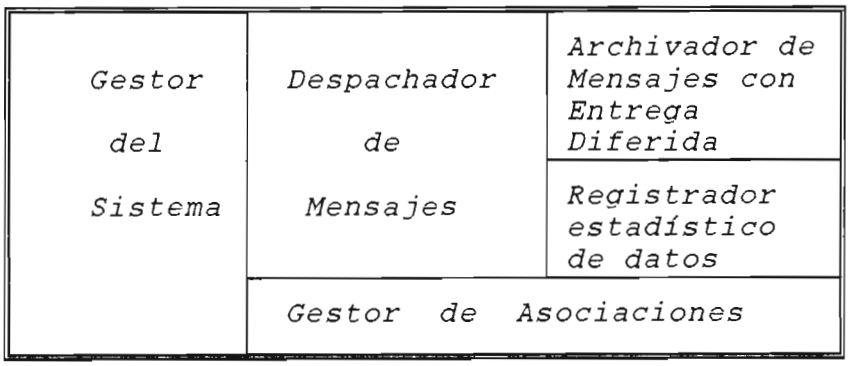


FIGURA 9. Modelo del MTA propuesto

El MTA modelo interactúa a través de su módulo Gestor de Asociaciones con un RTS "virtual" para solicitar los servicios de los niveles inferiores del modelo de referencia.

Estos módulos interactúan entre sí y con otros elementos de su entorno, por medio de interfaces cuya definición determina en buena medida, la funcionalidad de los módulos conectados.

En la figura 10 se ilustran las interrelaciones entre los componentes del MTA y de estos con su entorno.

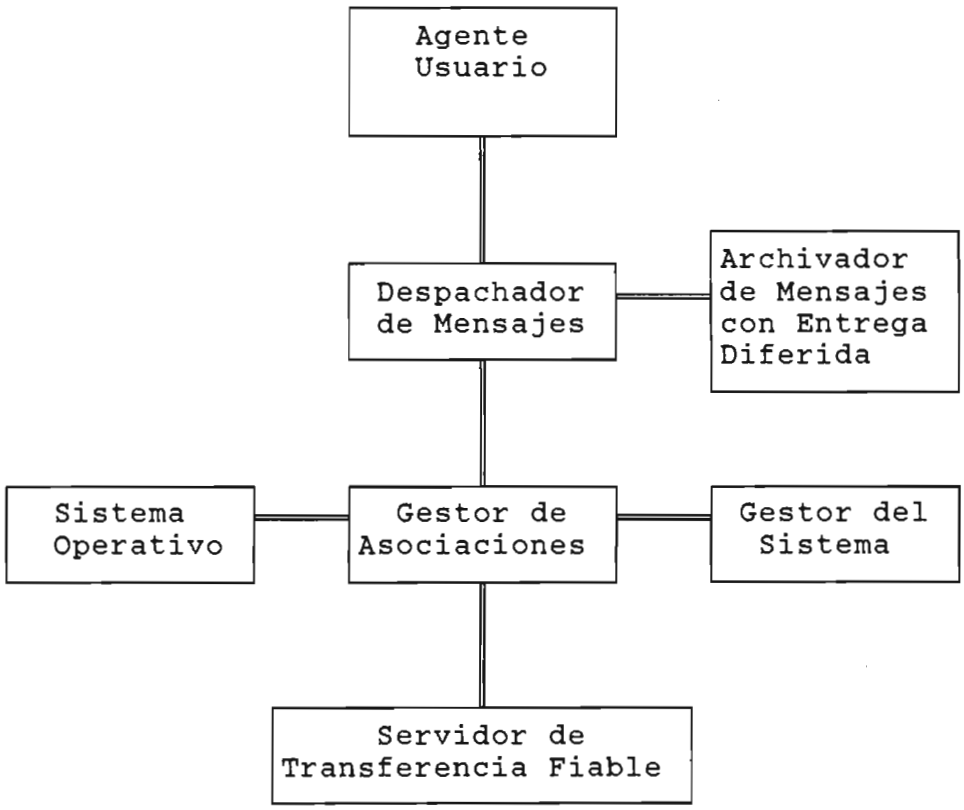


FIGURA 10. Interfaces en el MTA

Todas las primitivas de servicio definidas en la recomendación X.411 del CCITT, definen completamente la interfaz de servicio entre la MTAE y la UAE (Ver Tabla 1).

Aquí, los servicios del MTL son ofrecidos por una MTAE a alguna UAE que actúa como usuario (user) de estos servicios; el proveedor (provider) de estos servicios es la MTAE.

En los apartados siguientes se describen estos módulos siguiendo la metodología propuesta por el CCITT y la ISO. Siguiendo estas directrices, se define el servicio de cada módulo a través de primitivas (interacciones) de servicio abstractas intercambiadas a través de los puntos de acceso al servicio, situados en la interfaz entre los módulos. El uso de estas primitivas por parte de uno de los módulos especifica su condición de usuario o proveedor (servidor) del servicio.

4.3.1 Despachador de Mensajes

Todos los mensajes que llegan al MTA son procesados por el Despachador de Mensajes que se convierte así en un distribuidor central y va delegando el trabajo a realizar en los restantes módulos del MTA. Según la definición de

TABLA 1. Primitivas de servicio del MTL

Servicios	*Primitivas*
Establecimiento de acceso por usuario	*Petición de Logon* *Confirmación de Logon*
Establecimiento de acceso por el MTL	*Indicación de Logon* *Respuesta de Logon*
Terminación de acceso	*Petición de Logoff* *Confirmación de Logoff*
Registro	*Petición de Registro* *Confirmación de Registro*
Retención para entrega	*Petición de Control* *Confirmación de Control*
Indicación de restricción	*Indicación de Control* *Respuesta de Control*
Sonda	*Petición de Sonda* *Confirmación de Sonda*
Depósito de mensajes	*Petición de Depósito* *Confirmación de Depósito*
Entrega de mensajes	*Indicación de Entrega*
Notificación de mensajes	*Indicación de Notificación*
Cancelación de entrega diferida	*Petición de Cancelación* *Confirmación de Cancelación*
Modificación de contraseña (UAL)	*Petición de Cambio* *Confirmación de Cambio*
Modificación de contraseña (MTL)	*Indicación de Cambio*

[POR87], el Despachador de Mensajes es una máquina lógica capacitada para procesar el protocolo de transferencia de mensajes.

El Despachador de Mensajes es la entidad dentro del MTA que realiza las funciones de enrutamiento descritas en el apartado (3.4.4.1).

Se activa cuando recibe una solicitud de Depósito, una petición de Sonda o una petición de Cancelación a través de su interfaz con el UAL.

También se activa cuando detecta una indicación o una confirmación de Transferencia a través de la interfaz con el Gestor de Asociaciones; o cuando aparece una indicación de Expiración a través del interfaz con el Archivador de Mensajes con entrega diferida.

Cuando el Despachador de Mensajes recibe las anteriores primitivas que lo activan, realiza las siguientes acciones:

4.3.1.1 Petición de Depósito

Cuando una UAE desea enviar un mensaje a uno o más destinatarios, emite una solicitud de servicio a su MTAE

asistente; es decir emite la primitiva "petición de Depósito".

Cuando el despachador de Mensajes recibe esta petición, primero se validan los parámetros presentes en la solicitud. Si la validación es correcta, se envía al UAL una confirmación positiva de la petición solicitada. En caso contrario se envía al UAL una confirmación de Depósito negativa indicando la razón de rechazo de la petición.

Si la petición de Depósito solicita el servicio de entrega diferida, se registra toda la información necesaria en el módulo Archivador de Mensajes con Entrega Diferida y el mensaje se olvida allí hasta que se reciba una indicación de Expiración a través de la interfaz con el Archivador de Mensajes con Entrega Diferida.

Si no se solicita entrega diferida, se ejecuta el algoritmo de enrutamiento que retorna el conjunto de trayectorias por donde debe transferirse el mensaje. Se envía una indicación de Entrega a los UAs destinos locales, si existen, y una petición de Transferencia al Gestor de Asociaciones por cada una de las trayectorias seleccionadas que permitan hacer llegar el mensaje a los UAs destinatarios remotos. Por cada copia del mensaje entregada a una UAE local se genera un informe de Entrega

positivo y se generan informes de Entrega negativos, cuando no es posible entregar el mensaje al UA local o continuar con la remisión.

4.3.1.2 Petición de Sonda

Cuando recibe esta primitiva desde la interfaz con el UAL, el Despachador de Mensajes realiza las mismas acciones señaladas en el apartado anterior para una petición de Depósito, salvo que se genera una confirmación de Sonda y no hay entregas locales.

4.3.1.3 Confirmación de Transferencia

Si la confirmación señala que ha tenido éxito la Transferencia de un mensaje a otro MTAE por parte del Gestor de Asociaciones (ver apartado 4.3.1.1), el Despachador de Mensajes elimina la información que tenga registrada del mensaje confirmado.

Si la confirmación es negativa, se ejecuta de nuevo el algoritmo de encaminamiento para cada uno de los destinos asignados a la trayectoria que no ha tenido éxito en la Transferencia. Si el algoritmo de encaminamiento encuentra al menos una ruta nueva, el Despachador de Mensajes envía

el mensaje por la nueva trayectoria generando una nueva petición de Transferencia al Gestor de Asociaciones y actualizando la información de rastreo de este mensaje.
Si el algoritmo de encaminamiento no encuentra una trayectoria alternativa y el mensaje es un informe de entrega, el Despachador de Mensajes descarta este último. Pero si el mensaje es de usuario o de sonda, se genera un informe de entrega con la información de rastreo apropiada, se ejecuta el algoritmo de encaminamiento para encontrar la trayectoria del originador del mensaje. Si el originador es local, se le envía una indicación de Notificación. Si el originador es remoto, se envía un informe de entrega, por la trayectoria seleccionada, a través de una petición de Transferencia al Gestor de Asociaciones.

4.3.1.4 Indicación de Transferencia

Cuando el Despachador de Mensajes recibe del Gestor de Asociaciones esta primitiva, decodifica la MPDU recibida del MTA vecino y si es inválida la única acción que toma es la de registrar el evento para control estadístico.

Si la MPDU recibida es válida, ocurre lo siguiente:

- Para las MPDUs de usuario o de sonda se ejecuta el

algoritno de encaminamiento para obtener las trayectorias que debe seguir el mensaje.

Si existen destinatarios locales, las MPDUs se entregan a través de una indicación de Entrega.
Si la MPDU corresponde a un mensaje en tránsito, se envía por cada una de las trayectorias utilizando una petición de Tranferencia al Gestor de Asociaciones.

- Para las MPDUs de informes de entrega, se ejecuta el algoritmo de encaminamiento para encontrar la trayectoria del originador del mensaje. Si el originador es local, se le envía una indicación de Notificación a través del interfaz con el UAL. En caso contrario, se reenvía el informe de entrega utilizando una petición de Transferencia al Gestor de Asociaciones.

4.3.1.5 Indicación de Expiración

Con esta primitiva el Archivador de Mensajes con Entrega Diferida le señala al Despachador de Mensajes que el período de tiempo especificado por el originador para la entrega diferida de un mensaje, ha finalizado.

El Despachador de Mensajes solicita al Archivador los

mensajes expirados y para cada uno de ellos realiza las mismas acciones indicadas anteriormente en la primitiva petición de Depósito cuando se trata de un mensaje sin entrega diferida (apartado 4.3.1.1)

4.3.1.6 Petición de Cancelación

Cuando una UAE desea que se detenga la entrega a una UAE local o la transferencia a otra MTAE, de un mensaje con entrega diferida depositado anteriormente, la UAE originadora solicita este elemento de servicio enviando al Despachador de Mensajes una petición de Cancelación.

Cuando recibe esta petición de Cancelación, el Despachador de Mensajes solicita este servicio al módulo Archivador de Mensajes con Entrega Diferida quien le confirma el éxito o fracaso de la operación.

El Despachador de Mensajes notifica a la UAE el resultado de la tentativa de cancelación, a través de una confirmación de Cancelación enviada por su interfaz con el UAL.

4.3.2 Gestor de Asociaciones

La conexión entre el MTA local y los MTAs vecinos es la responsabilidad total del Gestor de Asociaciones. Para esto, está encargado de establecer, controlar y liberar las asociaciones proporcionadas por el RTS.

Se puede definir una asociación como una conexión o canal lógico entre dos MTAs, con ciertas características.

Para su funcionamiento el Gestor de Asociaciones se ayuda de la información de los MTAs vecinos registrada en el MTA local.

Este módulo se activa cuando recibe una petición de Transferencia a través de la interfaz con el Despachador de Mensajes, o cuando recibe una indicación de Apertura a través de la interfaz con el RTS. Por lo tanto, es el encargado de solicitar los servicios a los niveles inferiores.

4.3.2.1 Funciones de Optimización de Asociaciones

El módulo entidad Gestor de Asociaciones realiza funciones de utilización óptima de las asociaciones tales como:

- *Establecimiento de asociaciones, de conformidad con acuerdos bilaterales que tratan sobre:*

  . *Número máximo de asociaciones que pueden existir simultáneamente*

  . *Modo de diálogo a utilizar en la asociación: unidireccional o bidireccional alternado.*

  . *Tipo de asociación: permanente, establecida y liberada.*

  . *Quien tiene la responsabilidad para el establecimiento de la asociación.*

- *Rechazo del establecimiento de asociaciones.*

- *Transferencia de mensajes según prioridad, a través del RTS.*

- *Decisión de establecer o aceptar nuevas asociaciones con un mismo MTA.*

- *Recepción de mensajes desde el RTS.*

- *Gestión del turno (para transmitir o no).*

- *Liberación de las asociaciones cuando han sido abiertas.*

- Recuperación de asociaciones bloqueadas.

4.3.2.2 Interfaz con el Despachador de Mensajes

Esta interfaz se describe como el conjunto de primitivas de servicio que el Gestor de Asociaciones (servidor) ofrece al Despachador de Mensajes (usuario).
Según la ISO y el CCITT, las primitivas de servicio tienen por objeto especificar solamente aquellos detalles de la interacción entre las entidades que son aspectos del servicio ofrecido. Se debe tener en cuenta que las primitivas de servicio no especifican, ni restringen la realización de las entidades.

El Gestor de Asociaciones ofrece al Despachador de Mensajes un único servicio: la transferencia de mensajes a un MTA vecino. Este servicio tiene asociadas tres primitivas:

- Petición de Transferencia, que tiene como parámetros: la referencia del MTA vecino y la referencia del mensaje asignada cuando el Despachador de Mensajes lo recibió.

- Indicación de Transferencia cuyos parámetros son: la referencia del MTA emisor y el mensaje transmitido.

- Confirmación local de Transferencia con los parámetros: la indicación del éxito o fracaso de la petición de Transferencia, la referencia del mensaje citado en la petición, y el motivo del fracaso si procede.

4.3.2.3 Interfaz con el Gestor del Sistema

En esta interfaz, el Gestor de Asociaciones ofrece al Gestor del Sistema un único servicio: El acceso directo a operaciones ejecutadas por el Gestor de Asociaciones.

Este servicio ofrecido tiene asociada la primitiva petición de Operación, que tiene como parámetros: el tipo de operación solicitada, el número de parámetros necesarios para la operación, y los parámetros (si existen).

4.3.2.4 Interfaz con el RTS

El Gestor de Asociaciones utiliza los servicios ofrecidos por el RTS para poder a su vez suministrar sus servicios de manejo de asociaciones al conjunto del MTA.

Los servicios ofrecidos por el RTS están definidos en la Recomendación X.410 y se muestran resumidos en la siguiente tabla:

TABLA 2. Primitivas de Servicio del RTS

SERVICIOS	*PRIMITIVAS*
Establecimiento de una Asociación.	*Petición de Apertura.* *Indicación de Apertura.* *Respuesta de Apertura.* *Confirmación de Apertura.*
Liberación de una Asociación	*Petición de cierre.* *Indicación de cierre.* *Respuesta de cierre.* *Confirmación de cierre.*
Solicitud de Intercambio de turno	*Petición de turno.* *Indicación de turno.*
Intercambio de turno	*Petición de dar turno.* *Indicación de dar turno.*
Transferencia fiable de Mensajes	*Petición de Transferencia* *Indicación de Transferencia*
Indicación de fallo	*Indicación de Excepción*

Junto a estas primitivas de servicio, en este interfaz se incluye un conjunto de primitivas locales definidas en [POR87], para:

- *Control de flujo de datos:*
  - *. Crédito por parte del usuario.*
  - *. Crédito por parte del proveedor.*

- *Evitar bloqueo de asociaciones:*
  - *. Aborto de asociación por el usuario.*
  - *. Aborto de asociación por el proveedor.*

4.3.2.5 Interfaz con el Sistema Operativo

El Gestor de Asociaciones interactúa con el sistema operativo para sus tareas de gestión del tiempo. Para esto utiliza tres servicios del sistema:

- *Reloj, que tiene asociadas dos primitivas:*
  - *. Petición Disparar reloj, que tiene como parámetros: la referencia de la asociación que solicita el servicio, y el plazo de vencimiento del reloj.*
  - *. Confirmación de vencimiento del plazo de tiempo solicitado, que tiene como parámetro la referencia de la asociación que solicitó el servicio.*

- Parar reloj, que tiene asociada una primitiva: petición de Detener reloj. Esta tiene como único parámetro la referencia de la Asociación que ha solicitado previamente el servicio de reloj.

- Período de tiempo, con una única primitiva asociada: indicación de Expiración de período de tiempo, sin parámetros y es permanentemente periódica durante el funcionamiento del Gestor de Asociaciones.

4.3.3 Archivador de Mensajes con entrega diferida

Este módulo simple se encarga únicamente de guardar los mensajes que han solicitado el elemento de servicio entrega diferida y que le han sido entregados por el Despachador de Mensajes.

Realiza las operaciones siguientes:

- Registro de mensajes: la información referente a una entrega diferida se almacena en una estructura de datos que puede contener, entre otros, los siguientes campos: el identificador del mensaje, el tiempo de llegada, el tiempo de entrega diferida, nombre del originador, el contenido, etc.

- Cancelación de entregas: verifica la existencia del del mensaje en la estructura de datos anteriormente mencionada, y si el mensaje se encuentra registrado, lo borra de la estructura. Confirma el fracaso o éxito de esta operación al Despachador de Mensajes.

- Indicación de Expiración: Señala al Despachador de Mensajes la existencia de mensajes cuyo tiempo de entrega se ha cumplido.

- Retorno de mensajes con tiempo de entrega expirado: devuelve al Despachador de Mensajes los mensajes cuyo tiempo de entrega diferido ha expirado.

4.3.4 Registrador Estadístico de Datos

Aunque este módulo no es necesario para el funcionamiento del MTS, siguiendo lo expuesto en [POR87], un MHS debería llevar un control sobre los mensajes que utilizan los servicios del MTS y sobre el comportamiento de este último. Para esto, sería recomendable diseñar un módulo Registrador estadístico de datos encargado de recolectar datos, posiblemente elaborar estadísticas y registrar la historia para una buena gestión del sistema.

Existen dos tipos de información que debería ser registrada:

- La historia de todos los mensajes procesados por el MTA, que se puede utilizar para detectar errores de funcionamiento del MTS y para solucionar las quejas de los usuarios del sistema. Por lo tanto con este módulo se debería proporcionar a un administrador del MTA una forma cómoda para consultar la información almacenada.

- Las acciones y eventos de los protocolos que soportan el servicio, que permitiría detectar el comportamiento del MTS en cuanto a pérdida de mensajes, bucles, bloqueos, etc. y serviría para que el administrador del MTA tratara de recuperar el sistema ante estos posibles fallos.

Este módulo ofrece su servicio de registro a otros módulos del MTA, así:

4.3.4.1 Interfaz con el Despachador de Mensajes

Al Despachador de Mensajes, el Registrador estadístico .de datos ofrece los siguientes servicios:

- Registro de Depósito: Con esta operación se registra el

éxito o fracaso de la petición de depósito de un UA local.

- Registro de Entrega: esta operación registra la entrega de un mensaje a un UAE por parte de la MTAE que la sirve.

- Registro de Envío: con esta operación se registra el éxito o fracaso de la transferencia de un mensaje a otro MTA.

4.3.4.2 Interfaz con el Gestor de Asociaciones

En cuanto al Gestor de Asociaciones, el módulo ofrece los servicios de registro mostrados a continuación y que pretenden ayudar en la evaluación del comportamiento del protocolo.

- Registro de Apertura: con esta operación se registra el éxito o fracaso en la solicitud de apertura de una asociación.

- Registro de cierre: registra el cierre de una asociación.

- Registro de Solicitud de Turno: con esta operación se registra la demanda del turno por el MTA remoto en una

asociación bidireccional alternada.

- Registro de Sesión de Turno: esta operación registra el cambio de turno en una asociación bidireccional alternada.

- Registro de Excepción: registra el fracaso del RTS al intentar transmitir un mensaje.

4.3.5 Gestor del Sistema

En este módulo podrían estar concentradas las funciones de control del nodo MTS. La administración del sistema se propone distribuída entre los diferentes niveles que comprenden la arquitectura del MHS. Debe poseer una interfaz para el administrador del MHS que le permita un acceso fácil y cómodo al sistema.

En lo relacionado con el MTA diseñado, la gestión se encuentra distribuída entre los módulos Despachador de Mensajes y Gestor de Asociaciones.

4.4 ESPECIFICACION

Enlos apartados siguientes se describen los aspectos sobresalientes de la especificación formal del MTS. El

lenguaje de especificación utilizado fue el ESTELLE que es una norma internacional de la ISO para diseño y especificación de sistemas distribuídos.

El texto completo de la especificación formal se encuentra en el Anexo 1 de este informe.

El nivel mas externo de la especificación lo constituye la declaración de la entidad MTA como una ***'especificación'****. Se declaran a continuación todas las constantes y tipos de datos que serán considerados* ***'globales'*** *en el curso de toda la especificación. Luego aparecen los* ***'puntos de interacción'*** *que corresponden a los puntos extremos de las conexiones entre los diferentes módulos que forman el MTA, y de éste con su entorno. Para estas definiciones se utilizan las declaraciones de* ***'tipo de canal'*** *a cada uno de los cuales se les asigna un nombre y dos posibles modos de uso: user y provider. El módulo que interactúe a través de este canal, deberá estar definido mas adelante únicamente como servidor, o como usuario respecto a los servicios que se prestan a través de este canal.*

Se completa la declaración del canal con la lista de las primitivas disponibles en este canal, agrupadas entre las que puede ejecutar el usuario y las que ejecuta el

servidor; cada primitiva con su lista de parámetros.

Se continúa con la declaración del comportamiento externo de los módulos que componen la especificación. Básicamente se declara cuales de los canales utiliza el módulo y qué función desempeña respecto a esta interfaz: usuario o proveedor de servicios.

El comportamiento interno del módulo se especifica mediante una declaración "Body" para ese módulo. Se declaran los tipos de datos y variables que serán locales durante el funcionamiento del módulo. El módulo funciona como un autómata extendido, por lo tanto se declara un tipo especial de variable, STATE para los estados del autómata.

El comportamiento del autómata se modela partiendo de un estado inicial a partir del cual pasará a los demás estados definidos dependiendo de eventos que pueden sobrevenir del exterior (interacciones de otros módulos o del entorno), o debido a la actividad misma del autómata. Esto se especifica mediante el conjunto de transiciones que constituyen la parte "ejecutable" de toda la especificación.

Durante una transición, se ejecutan procedimientos,

funciones y otro tipo de operaciones que se traducirán en código ejecutable de máquina, cuando se realice la implementación de la especificación en algún lenguaje de computador.

4.4.1 Despachador de Mensajes

La Entidad Despachador de Mensajes interacciona con el Gestor de Asociaciones para realizar, junto con éste, la mayor parte de la funcionalidad del MTA.

A continuación se describen con algún detalle los componentes que lo describen en el código de la especificación. Para una mejor comprensión del autómata que modela su comportamiento, se recomienda estudiar el diagrama de estados de la figura 11.

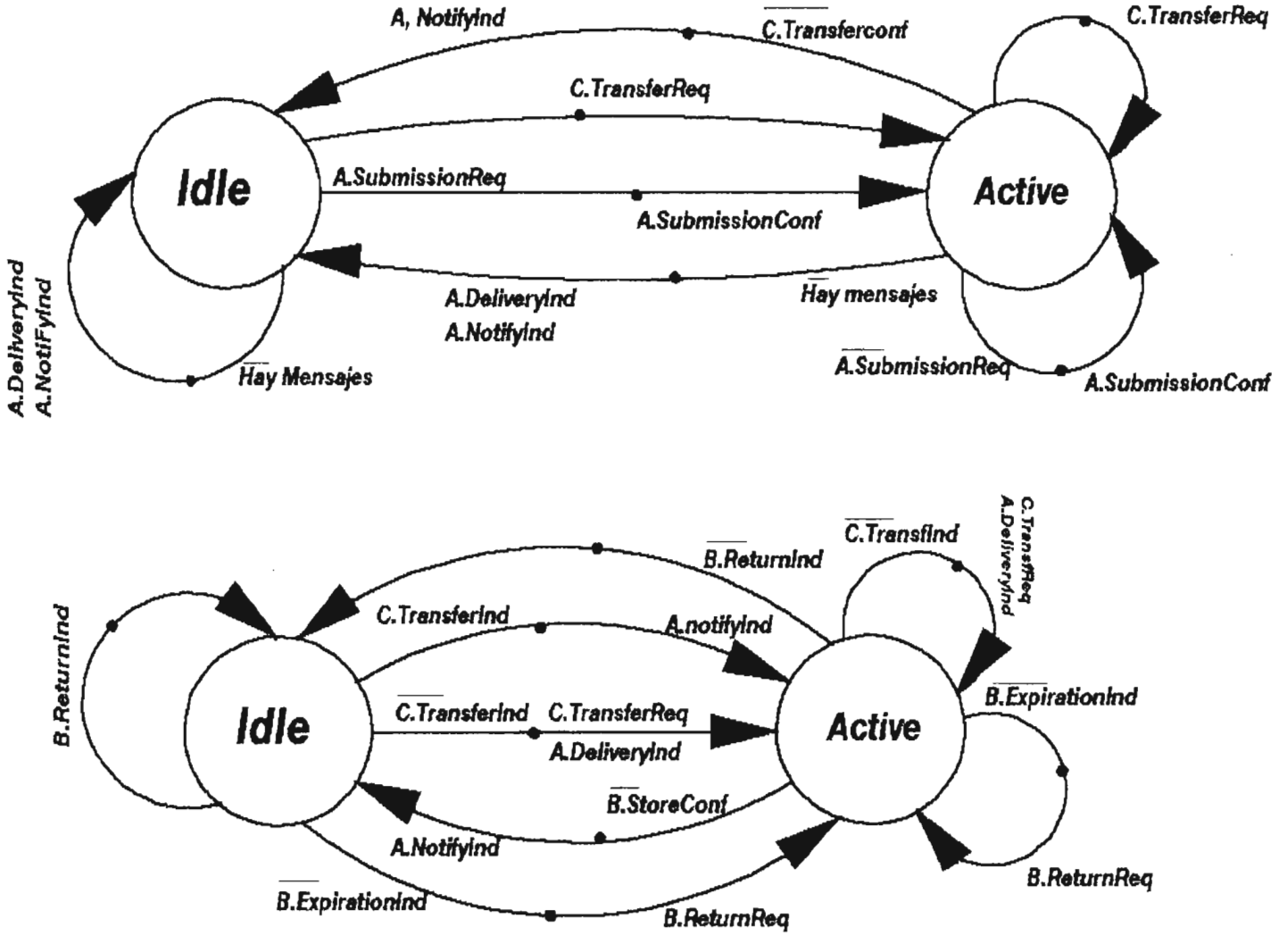


FIGURA 11. Diagrama de estados de Despachador de Mensajes

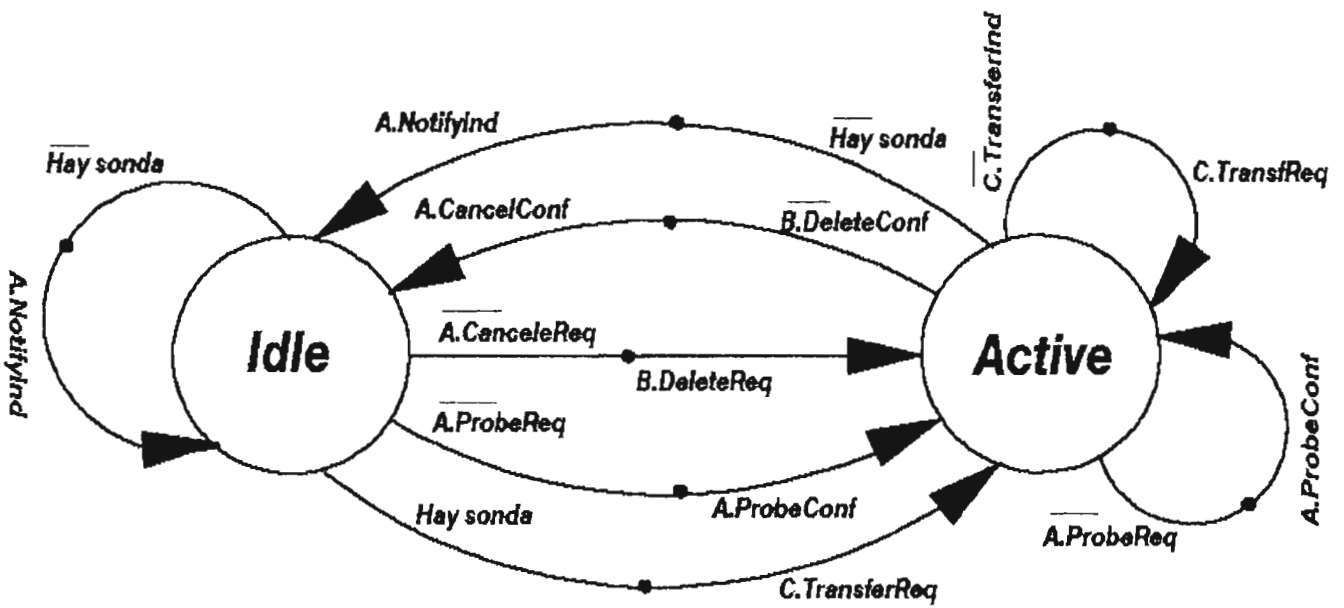


FIGURA 11. Diagra de estados del Despachador de Mensajes (continuación)

4.4.1.1 Canales

Los canales que definen el comportamiento externo del Despachador de Mensajes son:

- UAE-DM: interfaz con la entidad UA, donde actúa como proveedor de servicios a dicha entidad. Canal A.

- DM-AMED: interfaz con el Archivador de Mensajes con entrega diferida. Actúa como usuario. Canal B.

- DM-GA: interfaz con el Gestor de Asociaciones, a través del cual recibe sus servicios (usuario). Canal C.

4.4.1.2 Estados

El autómata que define el comportamiento de la entidad tiene dos estados en los que realiza todas sus funciones:

- Idle: Estado inicial del autómata. Si está en este estado, no se realiza ninguna actividad, se encuentra en reposo.

- Active: Estado del autómata al que se llega cuando ocurren eventos que requieren la realización de alguna

actividad. En este estado se realiza la funcionalidad de la Entidad.

Partiendo del estado inicial, el autómata va pasando de un estado al otro dependiendo de la ocurrencia de ciertos eventos tales como la recepción de alguna primitiva por uno de los canales, o el cumplimiento de ciertas condiciones determinadas por las variables de estado de la entidad. Esta dinámica se muestra gráficamente en la figura 11. Aquí, los círculos representan los estados y los arcos representan las transiciones que hacen pasar el autómata de un estado al otro. El evento que dispara la transición aparece subrayado.

4.4.1.3 Procedimientos y Funciones

Durante su actividad, la entidad utiliza procedimientos y funciones para:

- Controlar la información que mantiene sobre los usuarios.

- Acceder a la fecha y hora del sistema.

- Administrar la información sobre el MTA local.

- Controlar el manejo de las variables globales de la especificación.

- Crear las MPDUs.

- Manejar las colas de MPDUs.

- Manejo de las estructuras de datos con información de destinatarios.

4.4.1.4 Transiciones

En la fase de iniciación, se parte del estado "Idle" y se colocan en cero (0) o "FALSE" todas las variables globales y de estado.

Se conmuta al estado activo si:

- Llega una petición de Depósito por el canal A. Se envía una confirmación de Depósito por el canal A, o una petición de Almacenamiento por el canal B.

- Existe algún mensaje o sonda para transmitir a otro MTA. Por el canal C se envía una petición de Transferencia.

- Llega una indicación de Transferencia por el canal C: se envía una indicación de Depósito por el canal A, o una petición de Transferencia por el C, o una indicación de Notificación por el A.

- Se recibe una indicación de Expiración por el canal B: se envía una petición de REtorno por el mismo canal.

- Llega una petición de Cancelación por el canal A: se envía una petición de Borrar por el canal B.

- Por el canal A se recibe una petición de Sonda: se devuelve una confirmación de Sonda.

El autómata pasará del estado activo a reposo si:

- Existen mensajes o sondas para entregar a usuarios locales. Por el canal A se envía una indicación de Notificación.

- Por el canal C llega una confirmación de Transferencia: se elimina toda la información referente al mensaje transmitido y se envía una indicación de Notificación por el canal A.

- Se recibe una confirmación de Almacenamiento por el canal B. Se transmite por el canal A una indicación de Notificación.

- Llega una indicación de Retorno por el canal B. Se prepara el mensaje para entregarlo a la UAE receptora.

- Llega una confirmación de Borrar por el canal B: se envía por el canal A una confirmación de Cancelación.

4.4.2 GESTOR DE ASOCIACIONES

La Entidad Gestor de Asociaciones es un único módulo sin refinamiento, donde cada asociación establecida o no, se controla a través de un registro con la información necesaria para la gestión, dentro de un mapa de asociaciones. Este diseño se tomó según se describe en [POR87].

El mapa de asociaciones es una lista que contiene por cada asociación los campos mostrados en la Tabla 3.

A continuación se describe con algún detalle el comportamiento de la entidad. Su presentación se hace

TABLA 3. Mapa de Asociaciones

Campo	Descripción
AssociationState	*Indica el estado de la asociación.*
Association	*Tipo de asociación*
MTAid	*Identificador del MTA con el que se establece la asociación*
Password	*Palabra de acceso del MTA*
Address	*Dirección del RTS del MTA con el que se estableció asociación*
Dialogue	*Modo de diálogo de la asociación*
NumberOpenTries	*Número de intentos de establecimiento de una asociación con el mismo MTA, válido sólo en asociaciones tipo permanentes*
DidIOpen	*Indica si se estableció la asociación y por tanto si se tiene la responsabilidad de cerrarla*
HaveIturn	*Señala si se tiene el turno parcial para transmitir*

Continuación TABLA 3. Mapa de Asociaciones

Campo	Descripción
RequestedTurn	Indica si el MTA con el que se ha establecido la asociación ha solicitado turno
RequestPriority	Señala la prioridad máxima que espera transmitir el MTA que ha solicitado turno
UserReference	Referencia del último mensaje depositado al RTS, necesario para el Despachador de Mensajes
UserSubscript	Referencia del último mensaje deposito al RTS, necesario para el Despachador de Mensajes
SubmissionPriority	Prioridad del último mensaje depositado al RTS
MPDULastContentLength	Longitud del último mensaje depositado al RTS
SubmissionLastTime	Tiempo de depósito del último mensaje al RTS
DeliveryLastTime	Tiempo de recibo del último mensaje desde el RTS
MPDULastStoredTime	Tiempo de almacenamiento del último mensaje depositado al RTS

siguiendo las partes en que se divide la especificación en ESTELLE.

4.4.2.1 Estados

El autómata que define el comportamiento de la entidad tiene dos estados con los que realiza todas sus funciones:

- Active: Estado inicial del autómata y desde donde realiza la funcionalidad de la entidad. Cambia de estado recibe una petición del gestor del sistema con un evento de parada.

- Idle: Estado en el cual la entidad se encuentra en reposo, es decir no ejecuta ninguna tarea. Regresa al estado anterior cuando recibe una petición del gestor del sistema con un evento de ejecución.

4.4.2.2 Variables

La entidad utiliza variables globales para la identificación de sus propiedades locales y una variable de contexto, el mapa de asociaciones desde donde se controla, entre otros datos, el estado actual de cada una de las asociaciones.

Cada asociación puede tener alguno de los estados descritos en la tabla 4.

El comportamiento del autómata en un momento dado dependerá del estado de la asociación establecida, y si se la considera en la etapa de Transmisión o de Recepción. Esto se ve en la figura 12 que muestra el diagrama de estados de una asociación en la etapa de Transmisión y en la figura 13 se observa el diagrama de estados en la etapa de Recepción.

4.4.2.3 Procedimientos y funciones

Para especificar la parte de ejecución, en el módulo Gestor de Asociaciones se utilizan procedimientos y funciones para;

- Manejo de las propiedades de los MTAs con los cuales pueden establecerse asociaciones.

- Operaciones sobre la información registrada de los acuerdos bilaterales que permiten, entre otras, validar el establecimiento de asociaciones.

- Control de los recursos del MTA.

TABLA 4. Estados de una Asociación

Estado	*Descripción*
notused	*Estado inicial de la asociación, indica que la asociación está libre*
opening	*Señala que esta asociación ha solicitado su establecimiento y espera la confirmación de éste*
busy	*Asociación que ha depositado un mensaje al RTS y espera la confirmación de éste*
closing	*Estado que indica que la asociación ha solicitado su cierre y está esperando confirmación del mismo*
repose	*Estado en el que ni se transfieren ni se reciben mensajes*
waitingcall	*Asociación por la que el RTS puede depositar una indicación de establecimiento de asociación*
prereceiver	*Asociación que está lista para recibir un mensaje del RTS pero no tiene crédito para hacerlo*
receiver	*Asociación que puede recibir mensajes desde el RTS*

- Administración de los MTAs con los cuales se ha intentado establecer asociaciones y no se les ha aceptado.

- Invocación de las operaciones ofrecidas por el Registrador estadístico de datos, necesarias para las estadísticas e historia del comportamiento de los protocolos del MTA.

- Tomar decisiones de tiempo sobre:

. El intervalo de tiempo que una asociación estará en estado de reposo.
. El intervalo dentro del cual se volverá a intentar establecer una asociación con un MTA que previamente rechazó el establecimiento de ésta.
. Si es o no tiempo de intentar establecer una asociación con un determinado MTA.
. El intervalo de tiempo de que dispone el RTS para transmitir un determinado mensaje.

- Manejo de la librería para el formato físico de datos de usuario. Se utilizan para codificar y decodificar la información de los MTAs que permiten la validación del establecimiento de las asociaciones.

- Codificar y almacenar (encolar) las primitivas que pueden intercambiarse por cada uno de los canales del módulo.

- Controlar las estructuras de datos donde los mensajes esperan para ser transferidos al RTS. Existen varios tipos de operaciones sobre esta estructura de datos, tales como: almacenar, sacar, borrar, compactar, total de mensajes por destinatario, prioridad máxima del mensaje existente por destino, etc.

- Selección de asociaciones de acuerdo al cumplimiento de cierto criterio.

- Establecimiento de asociaciones. Se encargan de iniciar una asociación cuando está libre, aceptar y validar peticiones de establecimiento de asociaciones y establecer asociaciones.

- Decisión de utilización de asociaciones: cuando una asociación queda libre, se debe decidir si se establece una asociación, según los mensajes que esperan ser transmitidos; o si se establece más de una asociación con un mismo destino, según el número de mensajes conprioridad máxima que haya para este destino y el número de mensajes sin asociación establecida.

4.4.2.4 Transiciones

Partiendo de la fase de iniciación, se limpia el mapa de asociaciones colocando las asociaciones en el estado libre.

A continuación se selecciona la última asociación del mapa y se emite por ella una primitiva al RTS para que éste último pueda comunicarse con el Gestor de Asociaciones.

De acuerdo con los procedimientos y funciones que manejan la información de los MTAs con los que puede existir comunicación, se seleccionan aquellos con los que debe establecerse asociaciones permanentes y a cada uno se le asigna una asociación de las libres del mapa, para tratar de establecer comunicación.

El estado del autómata de la entidad se coloca en activo. Los parámetros globales de la entidad son inicializados en esta parte, lo mismo que las variables globales de la entidad. Estos valores permanecen estáticos y no hay forma de cambiarlos durante la vida de una asociación.

Se recomienda tener presente los diagramas de estados de las figuras 12 y 13 al seguir el funcionamiento de las transiciones

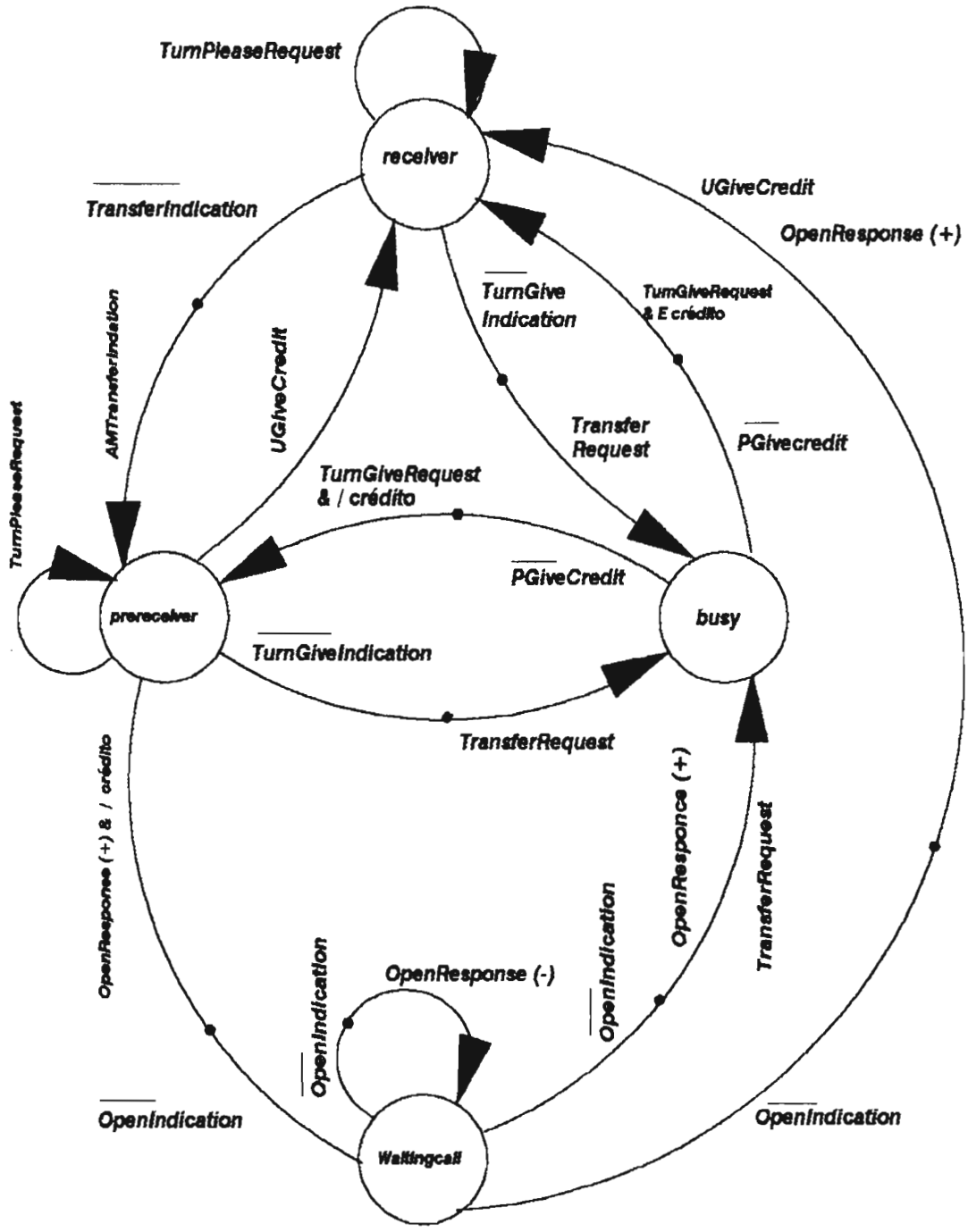


FIGURA 12. Diagrama de una Asociación: Recepción

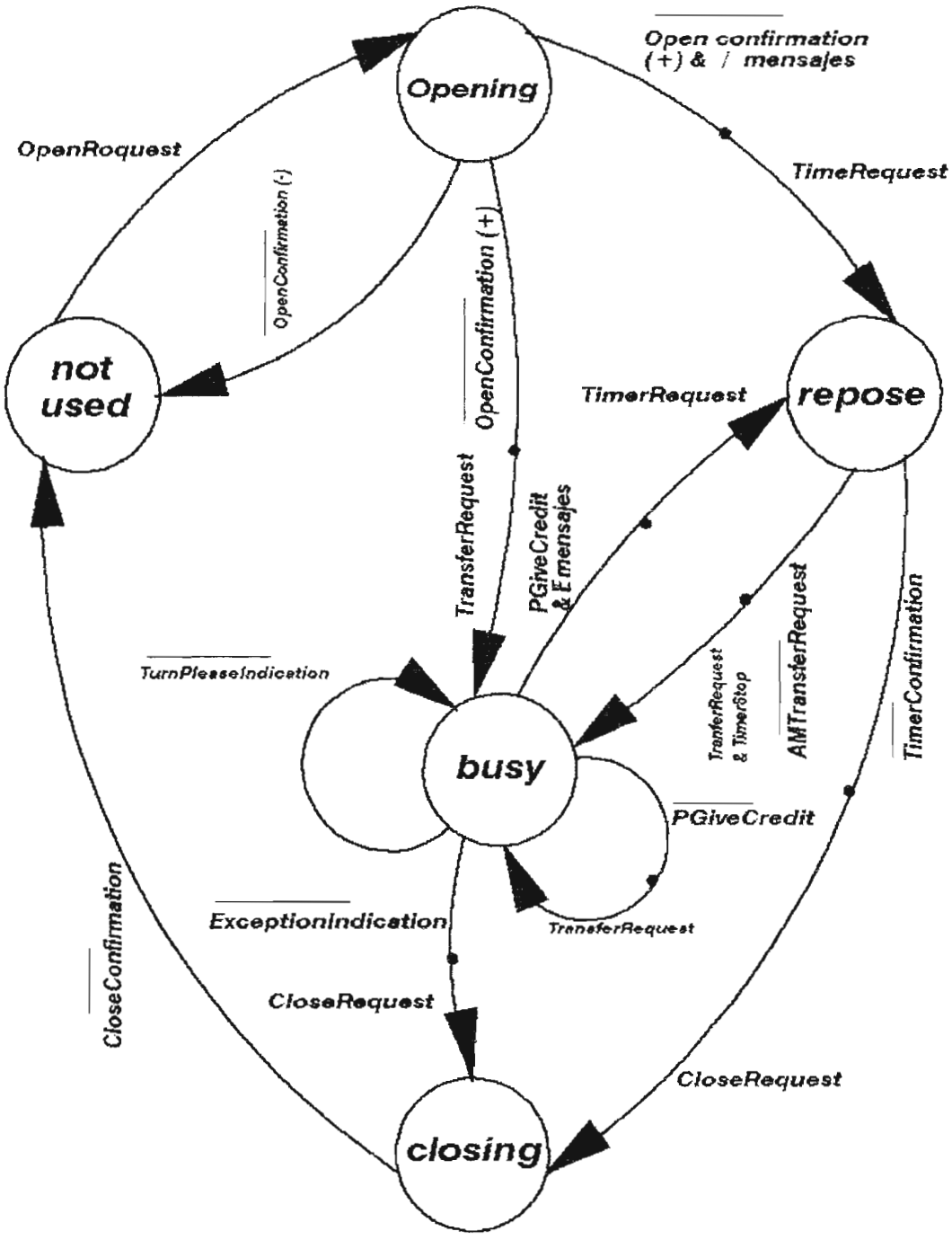


FIGURA 13. Diagrama de una Asociación: Transmisión.

4.4.2.4.1 Establecimiento de asociación desde el iniciador

Con estas transiciones se controlan el establecimiento de una asociación iniciada cuando el Despachador de Mensajes solicita transferencia de un mensaje.

Cuando se desea establecer una asociación se consultan, a través de las funciones de manejo de las propiedades de los MTAs, los acuerdos bilaterales para solicitar el turno inicial y según se tenga o no mensajes para transmitir se construye y envía una solicitud de apertura de asociación. También se coloca la asociación en estado abierto, y se actualiza en el mapa de asociaciones el campo de tiempo de depósito al tiempo actual, para controlar el bloqueo de la asociación.

Si se recibe una confirmación de apertura aceptada, se pasa la asociación al estado de transmisión o recepción, según se tenga o no el turno inicial para transmitir.

Una asociación que reciba una confirmación de apertura rechazada se coloca en estado libre. Si se trata de una asociación de tipo temporal, el MTA destino es almacenado con un período de tiempo de castigo, después del cual puede intentar de nuevo establecer la asociación. Para el caso

de las asociaciones permanentes, el procedimiento consiste en disparar un reloj con un plazo que depende del número de intentos de apertura de la asociación; vencido el plazo se intenga abrir de nuevo la asociación.

El único caso en el que el Gestor de Asociaciones se ve imposibilitado para establecer una asociación, ocurre cuando recibe una confirmación de Rechazo de apertura debido a que el MTA respondedor no puede validar al MTA iniciador. En esta situación el Gestor de Asociaciones devuelve al Despachador de Mensajes los mensajes destinados a este MTA, a través de una confirmación de transferencia rechazada.

4.4.2.4.2 Establecimiento de asociación desde el Respondedor

Para establecer asociaciones con el Gestor de Asociaciones, el RTS tiene en cada instante como máximo, un punto de asociación para esto.

Cuando se recibe una indicación de apertura de asociación, lo primero que hace el Gestor de Asociaciones es validar el MTA iniciador. Esto le permite encontrar, si existe, una de las posibles causas por las cuales no se puede aceptar

una asociación. Los otros dos motivos para rechazar una asociación son el modo de diálogo no aceptado por la entidad local, y el no tener recursos (a criterio del Gestor de Asociaciones) para dar servicio a esta asociación, por lo que responde como ocupado.

Si se tiene el turno para transmitir y hay mensajes para este MTA originador, se pasa esta asociación al estado de transmisión. En caso contrario se devuelve el turno del originador.

Cada vez que se recibe del RTS una indicación de apertura de asociación, éste queda sin poder comunicarse con el Gestor de Asociaciones.
Por esta razón, el Gestor de Asociaciones debe decidir si adjudica una asociación al RTS entre la asociaciones libres, o se queda temporalmente incomunicado de éste.

4.4.2.4.3 Envío de Mensaje

Cuando llega un mensaje desde el Despachador de Mensajes, pueden suceder alguna de las siguientes situaciones:

- Existe una asociación establecida con el MTA destino en estado de reposo, por lo que se utiliza esta asociación

para depositar el mensaje al RTS.

- Existe una asociación establecida con el MTA destino pero se decide establecer otra asociación con el mismo MTA. El mensaje se almacena y espera el establecimiento de la nueva asociación.

- Existe una asociación establecida con el MTA destino, no se decide abrir nueva asociación y se tiene el turno para transmitir. El mensaje se almacena.

- Existe una asociación establecida con el MTA destino, no se decide abrir nueva asociación y no se tiene el turno para transmitir. El mensaje se almacena y se solicita el turno con la máxima prioridad de los mensajes que esperan ser transferidos a esta MTA.

- No existe asociación establecida con el MTA destino, existe asociación libre, se tiene responsabilidad para abrir, es tiempo de apertura y no se ha intentado previamente abrir asociación con este MTA destino (habiendo sido rechazada). El mensaje se almacena y espera que se establezca la asociación.

- No existe asociación establecida con el MTA destino, no

existe asociación libre, pero existe alguna asociación establecida con otro MTA en estado de reposo y se tiene responsabilidad para abrir, es tiempo de apertura y no se ha intentado previamente establecer asociación con este MTA destino. El mensaje se almacena y se cierra la asociación que estuviera con el tiempo de reposo más próximo a cumplirse.

En cualquier otra situación se almacena el mensaje y se espera que el Gestor de Asociaciones tome algún tipo de decisión cuando queden asociaciones libres.

Como puede notarse, los mensajes se van almacenando en una estructura de datos por prioridades. Existen tres tipos de prioridades para los mensajes en tránsito: Urgente, Normal y No urgente.

Cuando se tiene el turno para transmitir, se copia de la estructura de datos el mensaje de mayor prioridad en orden de llegada y se deposita a través de una petición de transferencia de datos al RTS, actualizando el tiempo de depósito del último mensaje y el estado de la asociación se coloca ocupado.

Si vence el plazo máximo dado para recibir algún tipo de

activacción del RTS, sin obtener ninguna información, se recupera el mensaje depositado y se aborta la asociación.

Cuando el RTS por cualquier motivo no puede transmitir el mensaje depositado en el tiempo dado, devuelve el mensaje dentro de una indicación de excepción, por lo que el Gestor de Asociaciones recupera el mensaje y cierra la asociación si tiene responsabilidad para hacerlo, en caso contrario cede el turno.

Normalmente el RTS, para control de flujo local, envía un crédito al Gestor de Asociaciones por cada mensaje ya recibido en el otro extremo (es una especie de confirmación local). Cuando el Gestor de Asociaciones recibe crédito, confirma al Despachador de Mensajes el envío del mensaje, borra este mensaje de la estructura de datos y toma una decisión de acuerdo a la información registrada en el mapa de asociaciones y a la estructura de datos de los mensajes en espera de ser transmitidos.
La decisión puede ser transmitir un nuevo mensaje, ceder el turno o pasar a estado de reposo.

4.4.2.4.4 Recepción de Mensajes

Una asociación acepta mensajes enviados por el RTS a través

de una indicación de transferencia sólo cuando se encuentra en estado de recepción. El mensaje recibido es pasado directamente al Despachador de Mensajes a través de una indicación de transferencia y además se libera el crédito o recurso del MTA utilizado para aceptar este mensaje. La asociación inicialmente transita al estado de pre-recepción y solicita un recurso a las funciones que controlan la gestión de los recursos del MTA. Si éste es otorgado la asociación envía crédito al RTS y vuelve al estado de recepción, en caso contrario se mantiene en el estado de pre-recepción.

Por otro lado si el Gestor de Asociaciones receptor tiene mensajes para transmitir o si ha establecido la asociación, solicita el turno con la prioridad mínima que espera recibir. Obviamente si el modo de diálogo de la asociación es bidireccional alternado.

4.4.2.4.5 Cierre de Asociación

La fase de cierre de asociación se puede establecer en forma normal únicamente desde el Gestor de Asociaciones que abrió la asociación y con la asociación en estado de reposo.

Pero en algunos casos puede obligarse a cerrar las asociaciones de forma fuera de lo normal.

Este es el caso en que el gestor del sistema puede desear parar el Gestor de Asociaciones por cualquier motivo válido, sin necesidad de parar todo el sistema.

De esta forma todas las asociaciones que no se encuentren en estado de reposo deben cerrar la asociación a través de una solicitud de aborto al RTS.

Igualmente para evitar los bloqueos especialmente en la entidad que no tiene la responsabilidad de cerrar las asociaciones, se puede cerrar anormalmente en casos tales como:

- Caída del sistema que abrió la asociación.

- Pérdida del cambio de turno.

El proceso de cierre de asociación consiste en parar el reloj del estado de reposo si aún no se ha disparado, emitir la solicitud de cierre y colocar la asociación en estado de cierre. Luego de recibir la confirmación se debe liberar la asociación y llamar el procedimiento que se

encarga de tomar la decisión cuando quedan asociaciones libres.

Cuando no se ha abierto la asociación, no se puede rechazar la indicación de cierre emitida por el originador, sólo queda liberar la asociación, responder al cierre de asociación y llamar al procedimiento que gestiona las asociaciones libres.

4.4.2.4.6 Gestión de turno

Si la asociación es bidireccional alternada, el Gestor de Asociaciones sin turno puede solicitar el turno indicando la prioridad máxima de los mensajes que espera transmitir.

El turno se solicita cuando:

- Llega una solicitud de transferencia de mensajes desde el Despachador de Mensajes y la asociación está en estado de pre-recepción o de recepción.

- Cuando se acepta una solicitud de apertura de asociación, no se tiene turno inicial y existen almacenados mensajes para el MTA iniciador.

- *Cuando se tiene la responsabilidad de cerrar la asociación, independiente de si hay o no mensajes esperando ser transmitidos al MTA del otro extremo.*

Cuando se recibe una indicación de solicitud de turno, el Gestor de Asociaciones que posee el turno puede estar en estado ocupado o en estado de reposo. Si está en estado ocupado el Gestor de Asociaciones actualiza los campos solicitud de turno y prioridad máxima solicitada del registro de la asociación en el mapa de asociaciones.

Y toma la decisión de ceder o no el turno cuando recibe la confirmación local de la transferencia del último mensaje depositado.

Si el Gestor de Asociaciones se encuentra en estado de reposo, detiene el reloj de reposo y cede el turno al Gestor de Asociaciones solicitante, colocando la asociación en estado de recepción o de pre-recepción según se tenga o no recursos para recibir mensajes.

Cuando se recibe el turno y existen mensajes para transmitir por esta asociación, se transita al estado ocupado siguiendo las pautas señaladas anteriormente, liberando el recurso de control de flujo local si se

encuentra en estado de recepción.

Si se recibe el turno y no existen mensajes para transmitir el Gestor de Asociaciones transita a estado de reposo si estableció la asociación y cede el turno en caso contrario. Si transita al estado de reposo y el tipo de asociación es temporal dispara el reloj para controlar el plazo, pues vencido éste debe cerrar la asociación.

4.4.2.4.7 Control de tiempo

Para gestionar el tiempo dentro de la entidad, existe una indicación periódica del reloj del sistema que permite controlar todos los plazos a verificar dentro de la entidad.

Cada indicación del reloj del sistema permite:

- Compactar la estructura de datos donde se almacenan los mensajes que esperan ser transferidos.

- Devolver al Despachador de Mensajes, si existen, los mensajes que han vencido un plazo de espera límite sin que hayan podido ser transferidos.

- Verificar si existe alguna asociación bloqueada, por caída del sistema o por pérdida del cambio de turno.

- Sacar de la estructura de datos de espera de los MTAs con los que se ha intentado establecer asociación, con resultado negativo, aquellos que han cumplido su plazo en la estructura de datos.

4.4.2.4.8 Parada y Arranque

Existe una interacción que permite al gestor del sistema interactuar con las entidades del nivel de aplicación. Esta interacción tiene los siguientes argumentos: el tipo de evento, el número de parámetros que necesita el evento para efectuarse y la lista de parámetros.

El gestor del nivel a través de la interacción anterior puede activar dos eventos: parada (stop) y arranque (run). Ninguno de ellos tiene parámetros por lo que la lista de parámetros es vacía.

Con el evento de parada el Gestor de Asociaciones detiene completamente su ejecución. Y luego de liberar las asociaciones y estructuras de datos que administra, se coloca en estado inactivo.

Para sacar al Gestor de Asociaciones del estado inactivo, el gestor del sistema solicita al primero la operación de rearranque, colocando el evento de la solicitud de operación en ejecución. Esto obliga a que el Gestor de Asociaciones se comporte tal como si estuviera en la fase de iniciación.

5. CONCLUSIONES Y OBSERVACIONES

Se presentan en este Capítulo las conclusiones que se pueden derivar de las actividades de estudio e investigación realizadas para elaborar esta tesis, así como algunos comentarios sobre sus aportes y futuros desarrollos.

Se reseñan inicialmente las observaciones que se derivan de las fases de investigación teórica para modelar el Agente de Transferencia de Mensajes propuesto. Luego se señalan los aspectos que se consideran más significativos derivados de la experiencia adquirida al diseñar y especificar formalmente el Agente de Transferencia de Mensajes. Se finaliza señalando las acciones que deberían seguirse para continuar la investigación aquí propuesta.

5.1 FASES DE DISEÑO Y ESPECIFICACION

Se pretende resumir aquí las conclusiones obtenidas durante el trabajo que se realizó en las etapas de especificación

formal y definición de la arquitectura del MTA. Se espera sean tenidas en cuenta no solo para los trabajos que se emprendan al implementar este diseño, sino en otras tesis e investigaciones de la Universidad en el área de la Telemática.

Mas que aportes originales, se ratifican afirmaciones señaladas en otros estudios en ésta área y que sirvieron como experiencia previa en la búsqueda de los objetivos propuestos, principalmente las aportadas por el Director de esta Tesis.

Se resltan las siguientes observaciones:

- No se pueden emprender desarrollos en Ingeniería de Software sin utilizar las Técnicas de Descripción Formal. Antes de emprender cualquier diseño, se deben definir claramente todos los requisitos funciones solicitados.

- De acuerdo a la disponibilidad de recursos, se deben limitar en forma práctica los alcances del diseño. Se debe tener en cuenta que las normas o recomendaciones internacionales en que se base un diseño, normalmente abarcan un expectro muy amplio de posibles realizaciones.

- La era de los sistemas abiertos ya está aquí. Todo desarrollo en Ingeniería de Software debe basarse en normas y recomendaciones internacionales, si se pretende que sea un aporte más allá del simple ejercicio teórico o docente.

- La filosofía de diseño implícita en el modelo de referencia OSI de la ISO, presupone el empleo de grupos de trabajo donde se distribuyan las tareas de diseño y especificación de los diversos módulos en que se descompone cualquier modelo. Naturalmente, esta "modularización" del trabajo debe reflejarse en las fases de implementación.

5.2 SOBRE EL MTA DISEÑADO

Concluído el diseño propuesto, se pueden consignar las siguientes observacionees:

- La arquitectura del Agente de Transferencia de Mensajes presentada, no es la única solución posible. Por el contrario, este modelo no considera todos los servicios que puede prestar, según las Recomendaciones X.400.

- Naturalmente, este diseño no considera todas las variantes posibles previstas en las recomendaciones y en sus actualizaciones.

- Aunque el lenguaje ESTELLE no es la única ni la mejor Técnica de Descripción Formal, en la fase de especificación cumplió los objetivos que se buscaban y se considera que facilitará considerablemente las fases de realización futuras, debido a su proximidad a Pascal. Se le recomienda como la herramienta para los cursos iniciales de diseño y especificación de Sistemas Distribuídos.

- Las Entidades Despachador de Mensajes y Gestor de Asociaciones no alteran su funcionamiento ante las diferentes arquitecturas posibles del Agente de Transferencia de Mensajes.

- Sería muy recomendable contar con una herramienta de traducción automática de ESTELLE a un lenguaje de programación para disminuír el tiempo de desarrollo en las fases de implementación y aumentar la fiabilidad del código generado.

- Una vez adquirida la suficiente experiencia en ESTELLE, se recomienda emprender el estudio y utilización de otra Técnica de Descripción Formal como lo es LOTOS, que ya es una norma internacional de la ISO.

5.3 PARA CONTINUAR EN ESTA LINEA

El presente informe prevee la continuación de algunas tareas que complementen y amplíen el estudio realizado, teniendo en cuenta el objetivo mas general de extender el conocimiento y la experiencia sobre los Sistemas Distribuídos.

- Dotar a los sistemas conectados a la red local de la Universidad con los protocolos de comunicaciones que cumplan con las normas del modelo OSI/ISO.

- Concretamente, instalar protocolos que cumplan con el Subconjunto Básico del servicio de sesión definido por la ISO, y un protocolo a nivel de transporte que utilice al menos la Clase cero según se define en la Recomendación X.214 del CCITT.

- Realizado lo anterior, proceder a diseñar, especificar y realizar el módulo RTS.

- Considerar la posibilidad de entidades UA no conectadas directamente a la red local, para dar servicio a la comunidad universitaria desde fuera del campus universitario vía modem y línea telefónica. Para esto se

debe diseñar, especificar y realizar la entidad de Depósito y Entrega y su protocolo asociado.

BIBLIOGRAFIA

ALABAU M, Antonio, RIERA G, Juan. Teleinformática y Redes de computadores. Segunda Edición, Barcelona, Boixareu, 1984. [ALA84]

CCITT. Recommendation X.400. Message Handling Systems: System Model-Service Elements. Torremolinos, 1984. [CCI84a]

-------- Recommendation X.401. Message Handling Systems: Basic Servie Elements and Optional User Facilities. Torremolinos, 1984. [CCI84b]

-------- Recommendation X.408. Message Handling Sustems: Encoded Information Type Conversion Rules. Torremolinos, 1984. [CCI84c]

-------- Recommendation X.409. Message Handling Systems: Presentation Transfer Syntax and Notation. Torremolinos, 1984. [CCI84d]

-------- Recommendation X.410. Message Handling Systems: Remote Operations and Reliable Transfer Server. Torremolinos, 1984. [CCI84e]

-------- Recommendation X.411. Message Handling Systems: Message Transfer Layer. Torremolinos, 1984. [CCI84f]

-------- Recommendation X.420. Message Handling Systems: Interpersonal Messaging User Agent Layer. Torremolinos, 1984. [CCI84g]

-------- Recommendation X.430. Message Handling Systems: Access Protocol for Teletex Terminals. Torremolinos, 1984. [CCI84h]

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. Information Systems Processing - Open Systems Interconnection Basic Reference Model. IS7498, 1984. [ISO84]

-------- TC97d/SC21-FDT Subgroup B. ESTELLE: A formal Description Technique Based on an Extended State Transition Model. Second DP 9074. ISO, Sep., 1986. [ISO86]

NEMZOW, Martín. Keeping the Link. Ethernet Installation and Management. McGraw-Hill Book Company, 1988. [NEM88]

PORRAS, Hernán. Arquitectura de la gestión de mensajes de los sistemas distribuídos. Tesina, Master en Informática, Universidad Politécnica de Madrid, 1986. [POR86]

------- Gestión Optima de Asociaciones y Algoritmos de Decisión asociados en una red de mensajería según las recomendaciones X.400 del CCITT. Tesis Doctoral, Universidad Politécnica de Madrid, 1987. [POR87]

ROSE, Marshall. The Open Book. A practical perspective on OSI. Prentice Hall, 1990. [ROS90]

ANEXO 1.

CODIFICACION DE LA ESPECIFICACION EN ESTELLE

```
SPECIFICATION MTA;

CONST
  max = ANY INTEGER;
  num_dest = ANY INTEGER;
  totalUsuarios = ANY INTEGER;
  TamMaxMensaje = ANY INTEGER;
  TamMaxCola = ANY INTEGER;
  TamMaxComentario = ANY INTEGER;
  EsteMta = ANY STRING;
  TamMaxCola = ANY INTEGER;

TYPE
  nombre_O/D = STRING[max];
  datos = STRING[TamMaxMensaje];
  clase_prdad = (urgente,no_urgente,normal);
  fecha_hora = ....;
  comentario = STRING[TamMaxComentario];
  nombres_O/D = ARRAY[1..num_dest] OF nombre_O/D;
  cadena_caracteres = ARRAY[1..max,carac];
  dir_red_transp = ....;
  nombre_mta = STRING[max];
  tipo_operacion = (run,stop);
  parametros = ARRAY[1..Numero_parametros];
  tipo_asociacion = (monologo,bidirecAlter);
  tiene_turno = (iniciador,respondedor);
  disp_nueva_asoc = (aceptada,rechazada);
  periodo_tiempo = ....;
  Clase_mpdu = (Umpdu,DRmpdu,Pmpdu);

  RefMpdu = RECORD
              RefMens : INTEGER;
              TipoMpdu : Clase_mpdu
            END;
```

```
TipoUmpdu = RECORD
              MpduId : RECORD
                         GDId : EsteMta;
                         Sec : INTEGER
                       END;
              Originador : nombre_O/D;
              UAContId : INTEGER;
              DefDeliv : fecha_hora
              Prioridad : clase_prdad;
              DiscRec : BOOLEAN;
              AlRecAllow : BOOLEAN;
              ContRetReq : BOOLEAN;
              CountryN : ....;
              AdDomN : ....;
              BilatInfo : ....;
              RecInfo : ARRAY[1..num_dest] OF RecipientInfo;
              TraceInfo : ARRAY[1..2] OF InfRastreo;
              Content : datos
            END;

TipoDRmpdu = RECORD
               MpduId : RECORD
                          GDId : EsteMta;
                          Sec: INTEGER
                        END;
               Originador: nombre_O/D;
               TraceInfo: ARRAY[1..2] OF InfRastreo;
               OrigMdpuId: RECORD
                             GDId: EsteMta;
                             Sec: INTEGER
                           END;
               InterTraceInfo:ARRAY[1..2] OF InfRastreo;
               UAContId: INTEGER;
               RepRecInfo:ARRAY[1..num_dest] OF InfDestCom;
               RetUmpduCont: datos
             END;
```

```
Tipo Pmpdu = RECORD
               MpduId: RECORD
                         GDId: EsteMta;
                         Sec: INTEGER
                       END;
               Originador: nombre_O/D;
               UAContId: INTEGER;
               TraceInfo: ARRAY[1..2] OF InfRastreo;
               DiscRec: BOOLEAN;
               AlRecAllow: BOOLEAN;
               ContRetReq: BOOLEAN;
               ContentLength: INTEGER;
               CountryN:....;
               AdDomN:....;
               BilatInfo:....;
               RecInfo: ARRAY[1..num_dest] OF RecipientInfo
   END;

RecipientInfo = RECORD
                  Recep:nombre_O/D;
                  ExtId: INTEGER;
                  FlagResp = (Abierto,Cerrado);
                  PeticionInf = (Basico,Confirmado,InvesYconf);
                  PetInfUs = (NoInf,Basico,Confirmado)
                END;

InfRastreo = RECORD
               GDId: EsteMta;
               Tllegada: fecha_hora;
               TDif: fecha_hora
               Accion: INTEGER;
               PGDomN:....
             END;

InfDestCom = RECORD
               Recep: nombre_O/D;
               ExtId: INTEGER;
               FlagResp=(Abierto,Cerrado);
               PeticionInf=(Basico,Confirmado,InvesYConf);
               PetInfUs=(NoInf,Basico,Confirmado);
               Tllegada: fecha_hora;
               TEntrega: fecha_hora;
               TipoUA: ClaseUA;
               RazonNoEntrega: ReasonCode
             END;
```

```
ClaseUA=(Publico,Privado);
ReasonCode=(TransferFailure,UnableToTransfer,ConversionNotPerformed);

us_mta= RECORD
           usuario: nombre_O/D;
           mta: nombre_mta
        END;

Usuarios= ARRAY[1..totalUsuarios] OF us_mta;
```

```
CHANNEL UAE_DM (User,Provider);

   BY User:

      SUBMISSION.Request(Destinatarios: nombres_O/D;
                         Originador: nombre_O/D;
                         Contenido: datos;
                         Supresion_NNE: BOOLEAN;
                         Prioridad: clase_prdad;
                         T_E_Diferida: fecha_hora;
                         Notificacion_E: BOOLEAN;
                         Revelacion_Destin: BOOLEAN;
                         Destin_Alter: BOOLEAN;
                         Devolucion_Contenido: BOOLEAN;
                         Id_Contenido_AU: INTEGER);

      PROBE.Request(Destinatarios: nombres_O/D;
                    Originador: nombre_O/D;
                    Destin_Alter: BOOLEAN;
                    Long_contenido: INTEGER;
                    Id_Contenido_AU: INTEGER);

      CANCEL.Request(IdSucesoDep: INTEGER);

   BY Provider:

      SUBMISSION.Confirmation(Indicacion_exito:BOOLEAN;
                              Sello_deposito: fecha_hora;
                              Id_suceso_deposito: INTEGER;
                              Motivo_fracaso: comentario;
                              Id_Contenido_AU: INTEGER);

      PROBE.Confirmation(Indicacion_exito: BOOLEAN;
                         Tiempo_sonda: fecha_hora;
                         Id_suceso_sonda: INTEGER;
                         Motivo_fracaso: comentario;
                         Id_Contenido_AU: INTEGER);

      CANCEL.Confirmation(Indicacion_exito: BOOLEAN;
                          Motivo_fracaso:comentario);
```

```
DELIVERY.Indication(Originador: nombre_O/D;
                    Destinatario: nombre_O/D;
                    Otros_destinatarios: nombres_O/D;
                    Contenido: datos;
                    Tiempo_entrega: fecha_hora;
                    Tiempo_deposito: fecha_hora;
                    Prioridad: clase_prdad;
                    Destinatario_deseado: nombre_O/D;
                    Id_suceso_entrega: INTEGER);

NOTIFY.Indication(Tipo_notificacion: BOOLEAN;
                  Id_suceso_deposito_sonda: INTEGER;
                  Destinatario: nombres_O/D;
                  Motivo_no_entrega: comentario;
                  Tiempo_entrega: fecha_hora;
                  Destinatario_deseado: nombre_O/D;
                  Contenido_devuelto: datos;
                  Id_Contenido_AU: INTEGER);
```

```
CHANNEL UA_GS(User,Provider);

   BY User:

      LOGON.Request(Nombre: nombre_O/D;
                    Contrasena: cadena_caracteres);

      LOGON.Response(Indicacion_exito: BOOLEAN;
                     Motivo_fracaso: comentario);

      LOGOFF.Request;

      REGISTER.Request(Long_max: INTEGER;
                       Fijaciones_cont_supl:
                       Direccion: dir_red_transp;
                       Nombre_UA: nombre_O/D);

      CONTROL.Request(Long_max: INTEGER;
                      Prioridad_mas_baja: clase_prdad;
                      Mensajes: BOOLEAN;
                      Notificaciones: BOOLEAN);

      CONTROL.Response(Mens_retenidos: BOOLEAN);

      (UAL)CHANGE-PASSWORD.Request(Contrasena_antigua: cadena_caracteres;
                                   Contrasena_nueva: cadena_caracteres);

   BY Provider:

      LOGON.Confirmation(Indicacion_exito: BOOLEAN;
                         Motivo_fracaso: comentario;
                         Mens_en_espera:

      LOGON.Indication(Nombre_MTA: nombre_mta;
                       Mens_en_espera:
                       Contrasena: cadena_caracteres);

      LOGOFF.Confirmation;

      REGISTER.Confirmation(Indicacion_exito: BOOLEAN;
                            Motivo_fracaso: comentario);

      CONTROL.Confirmation(Indicacion_exito: BOOLEAN;
                           Motivo_fracaso: comentario;
                           Mens_retenidos: BOOLEAN);
```

```
      CONTROL.Indication(Long_max: INTEGER;
                         Prdad_mes: clase_prdad;
                         Sondas: BOOLEAN;
                         Mensajes: BOOLEAN);

      (UAL)CHANGE-PASSWORD.Confirmation(Indication_exito: BOOLEAN;
                                        Motivo_fracaso: comentario);

      (MTL)CHANGE-PASSWORD.Indication(Contrasena_antigua: cadena_caracteres;
                                      Contrasena_nueva: cadena_caracteres);


CHANNEL DM_AMED (User,Provider);

   BY User:

      STORE.Request(Umpdu : TipoUmpdu);

      RETURN.Request(MensSolicitado : INTEGER);

      DELETE.Request(IdMensaje : INTEGER);


   BY Provider:

      STORE.Confirmation(Indicacion_exito: BOOLEAN;
                         Motivo_fracaso: comentario;
                         Id_Contenido_AU : INTEGER);

      EXPIRATION.Indication(MensExp : INTEGER);

      RETURN.Indication(Umpdu : TipoUmpdu);

      DELETE.Confirmation(Indicacion_exito: BOOLEAN;
                          Motivo_fracaso: comentario;
                          IdMensaje: INTEGER);
```

```
CHANNEL DM_GA (User,Provider);

   BY User:

      TRANSFER.Request(Nombre_MTA: nombre_mta;
                       RefMensaje: RefMpdu;
                       Destinatario: nombre_O/D;
                       Umpdu: TipoUmpdu;
                       DRmpdu: TipoDRmpdu;
                       Pmpdu: TipoPmpdu);

   BY Provider:

      TRANSFER.Indication(RefMensaje : RefMpdu;
                          Destinatario: nombre_O/D;
                          Umpdu: TipoUmpdu;
                          DRmpdu: TipoDRmpdu;
                          Pmpdu: TipoPmpdu);

      TRANSFER.Confirmation(Indicacion_exito: BOOLEAN;
                            RefMensaje: RefMpdu;
                            Destinatario: nombre_O/D;
                            Motivo_fracaso: comentario);


CHANNEL GS_GA (User,Provider);

   BY User:

      OPERATION.Request(Operacion_solicitada: tipo_operacion;
                        Numero_parametros: INTEGER;
                        Lista_parametros: parametros);

   BY Provider:

      OPERATION.Indication(Indicacion_exito: BOOLEAN);
```

```
CHANNEL GA_RTS (user,Provider);

   BY User:

      OPEN.Request(Dir_respondedor:      ;
                   Modo_dialogo: tipo_asociacion;
                   Turno_inicial: tiene_turno);

      OPEN.Response(Disposicion: disp_nueva_asoc;
                    Motivo_rechazo: comentario);

   BY Provider:

      OPEN.Indication(Dir_iniciador     ;
                      Modo_dialogo: tipo_asociacion;
                      Turno_inicial: tiene_turno);

      OPEN.Confirmation(Disposicion: disp_nueva_asoc;
                        Motivo_rechazo: comentario);


CHANNEL GA_SO (User,Provider);

   BY User:

      TIMER.Request(Id_asociacion: INTEGER;
                    Plazo: periodo_tiempo);

      TIMERSTOP.Request(Id_asociacion: INTEGER);

   BY Provider:

      TIMER.Confirmation(Id_asociacion: INTEGER);

      TIMER.Indication;
```

```
MODULE DespachadorMensajes PROCESS
          (A : UAE_DM (Provider) COMMON QUEUE;
           B : DM_AMED (User) COMMON QUEUE;
           C : DM_GA (User) COMMON QUEUE );


BODY Dm FOR DespachadorMensajes;

  TYPE
    TipoIdSDS = RECORD
                  MpduId : INTEGER;
                  DepSonda : INTEGER
                END;

  VAR
     UMPDU: TipoUmpdu;
     DRMPDU: TipoDRmpdu;
     PMPDU: TipoPmpdu;
     lm: INTEGER;
     rm: INTEGER;
     DestLocM: nombres_O/D;
     DestRemM: nombres_O/D;
     tdm: INTEGER;
     ls : INTEGER;
     rs : INTEGER;
     DestLocS : nombres_O/D;
     DestRemS : nombres_O/D;
     tds : INTEGER;
     HayMensaje : BOOLEAN;
     HaySonda : BOOLEAN;
     ColaUmpdu : ARRAY[1..TanMaxCola] OF TipoUmpdu;
     ColaPmpdu : ARRAY[1..TamMaxCola] OF TipoPmpdu;
     IdSD : ARRAY[1..TamMaxCola] OF TipoIdSDS;
     IdSS : ARRAY[1..TamMaxCola] OF TipoIdSDS;
     UsuariosMTS: Usuarios;
     NumMpdu: INTEGER;

  STATE Idle,Active;

  STATESET
     Cualquiera = [Idle,Active];
```

```
FUNCTION CheckUsuario(x : nombre_O/D) : BOOLEAN;
{Chequea si el usuario 'x' existe en la tabla 'UsuariosMTS' de
 usuarios validos}
VAR
  i : INTEGER;
BEGIN
  CheckUsuario := FALSE;
  FOR i := 1 TO totalUsuarios DO
     IF x = UsuariosMTS[i].usuario THEN
       BEGIN
         CheckUsuario := TRUE;
         i := totalUsuarios + 1
       END
END;

FUNCTION FechaHoraSist : fecha_hora;
{ Extrae fecha y hora del sistema }
PRIMITIVE;

PROCEDURE BuscarMTA(x : nombre_O/D;
                    VAR y : nombre_mta);
{ Retorna nombre del MTA al cual pertenece el usuario x, buscandolo en el
  arreglo global UsuariosMTS .}
VAR
  i : INTEGER;
BEGIN
  FOR i := 1 TO totalUsuarios DO
     IF x = UsuariosMTS[i].usuario THEN
       BEGIN
         y := UsuariosMTS[i].mta;
         i := totalUsuarios + 1
       END
END;
```

```
PROCEDURE Validar(Originador : nombre_O/D;
                  Destinatarios : nombres_O/D;
                  VAR Indicacion_exito :  BOOLEAN;
                  VAR Motivo_fracaso :  comentario);
{Verifica si Origninador y Destinatarios exiten en la tabla de
  usuarios validos 'UsuariosMTS'}
VAR
  i : INTEGER;
BEGIN
  Indicacion_exito := TRUE;
  Motivo_fracaso := " ";
  IF NOT CheckUsuario(Originador) THEN
     BEGIN
       Indicacion_exito := FALSE;
       Motivo_fracaso := `Nombre de originador invalido'
     END
  ELSE
     BEGIN
       FOR i := 1 to TotalDest(Destinatarios) DO
        BEGIN
           IF NOT CheckUsuario(Destinatarios[i])THEN
           BEGIN
             Indicacion_exito := FALSE;
             Motivo_fracaso := `Nombre de destinatario invalido'
           END
         END
     END
END;


FUNCTION TotalDest(Destinatarios : nombres_O/D) : INTEGER;
{Calcula cuantos destinatarios hay en la tabla 'Destinatarios'}
VAR
  i : INTEGER;
BEGIN
  TotalDest := 0;
  FOR i := 1 TO num_dest DO
     IF Destinatarios[i] <> " " THEN TotalDest := TotalDest + 1;
END;
```

```
FUNCTION DestinatarioLocal(x : nombre_O/D) : BOOLEAN;
{Determina si usuario 'x' es un usuario localmente
 registrado en esta MTA. Se busca 'x' en la tabla 'UsEstemta'}
VAR
  i : INTEGER;
BEGIN
  DestinatarioLocal := FALSE;
  FOR i := 1 TO totalUsuarios DO
     IF x = UsuariosMTS[i].usuario THEN
       BEGIN
         IF UsuariosMTS[i].mta = EsteMta THEN
           BEGIN
             DestinatarioLocal := TRUE;
             i := totalUsuarios + 1
           END
       END
END;

PROCEDURE OtrosDest(x : nombres_O/D;
                    z : nombre_O/D;
                    VAR y : nombres_O/D);
{Crea el arreglo 'y' con todos los destinatarios presentes en el arreglo
 'x' ,excepto el destinatario z .}
VAR
  i : INTEGER;
  J : INTEGER;
BEGIN
  J := 1;
  FOR i := 1 TO num_dest DO
     IF x[i] <> z THEN
       BEGIN
         y[J] := x[i];
         J := J + 1
       END
END;
```

```
PROCEDURE SeparaLocRem(x : nombres_0/D;
                       VAR dl : nombres_0/D;
                       VAR m : INTEGER;
                       VAR dr : nombres_0/D;
                       VAR n : INTEGER);
{Dada la tabla de destinatarios 'x', calcula m, numero de
 destinatarios locales y los almacena en dl, calcula n,
 numero de destinatarios remotos y los almacena en dr}
VAR
  i : INTEGER;
BEGIN
   m := 0;
   n := 0;
   FOR i := 1 TO TotalDest(x) DO
     BEGIN
         IF DestinatarioLocal(x[i]) THEN
            BEGIN
               m := m + 1;
               dl[m] := x[i]
            END
         ELSE
            BEGIN
               n := n + 1;
               dr[n] := x[i]
            END
      END
 END;
```

```
PROCEDURE BorraGlobales(VAR l: INTEGER;
                        VAR DestLoc: nombres_O/D;
                        VAR r: INTEGER;
                        VAR DestRem: nombres_O/D;
                        VAR td: INTEGER);
{Elimina el contenido de todas estas variables}
PRIMITIVE;
```

```
PROCEDURE CreaUmpdu(NumMpdu : INTEGER;
                    Destinatarios : nombres_O/D;
                    Originador : nombre_O/D;
                    Contenido : datos;
                    Supresion_NNE : BOOLEAN;
                    Prioridad : clase_prdad;
                    T_E_diferida : fecha_hora;
                    Notificacion_E : BOOLEAN;
                    Revelacion_Destin : BOOLEAN;
                    Destin_Alter : BOOLEAN;
                    Devolucion_Contenido : BOOLEAN;
                    Id_Contenido_AU : INTEGER;
                    VAR Umpdu : TipoUmpdu);
VAR
 t,i: INTEGER;
BEGIN
   Umpdu.MpduId.Sec := NumMpdu;
   Umpdu.Originador := Originador;
   Umpdu.UAContId := Id_Contenido_AU;
   Umpdu.DefDeliv := T_E_Diferida;
   Umpdu.Prioridad := Prioridad;
   Umpdu.DiscRec := Revelacion_Destin;
   Umpdu.AltRecAllow := Destin_Alter;
   Umpdu.ContRetReq := Devolucion_Contenido;
   t := TotalDest(Destinatarios);
```

```
    FOR i := 1 TO t DO
      BEGIN
        Umpdu.RecInfo[i].Recep := Destinatarios[i];
        Umpdu.RecInfo[i].ExtId := i;
        Umpdu.RecInfo[i].FlagResp := Abierto;
        Umpdu.RecInfo[i].PeticionInf := Basico;
        IF Supresion_NNE THEN Umpdu.RecInfo[i].PetInfUs := NoInf
        ELSE
          BEGIN
            IF Notificacion_E THEN
              BEGIN
                Umpdu.RecInfo[i].PetInfUs := Confirmado;
                Umpdu.RecInfo[i].PeticionInf := Confirmado
              END
            ELSE
              BEGIN
                Umpdu.RecInfo[i].PetInfUs := Basico;
                Umpdu.RecInfo[i].PeticionInf := Basico
              END
          END
      END;
    Umpdu.TraceInfo[1].GDId := EsteMta;
    Umpdu.TraceInfo[1].Tllegada := FechaHoraSist;
    Umpdu.TraceInfo[1].TDif := T_E_Diferida;
    Umpdu.TraceInfo[1].Accion := 0;
    Umpdu.Content := Contenido
END;
```

```
PROCEDURE EncolarUmpdu(Umpdu : TipoUmpdu;
                       VAR ColaUmpdu : ARRAY[1..TamMaxCola] OF TipoUmpdu);
{Almacena Umpdu en el arreglo global ColaUmpdu.}
VAR
  i : INTEGER;
BEGIN
  FOR i := 1 TO TamMaxCola DO
     BEGIN
       IF ColaUmpdu[i] = " " THEN
         BEGIN
           ColaUmpdu[i] := Umpdu;
           i := TamMaxCola + 1
         END
     END
END;
```

```
PROCEDURE SacarUmpdu(ColaUmpdu : ARRAY[1..TamMaxCola] OF TipoUmpdu;
                     VAR UMPDU : TipoUmpdu);
{Extrae una variable global UMPDU de la cola y compacta la cola.
 La extraccion se efectua de acuerdo al campo que indica la prioridad
 de la Umpdu. Solo se consideran las Umpdu que no sean para entrega
 diferida. }
VAR
  i : INTEGER;
  p : clase_prdad;
  j : INTEGER;
  s : INTEGER;
  ColaT : ARRAY[1..TamMaxCola] OF TipoUmpdu;
BEGIN
  j := 0;
  FOR i := 1 TO TamMaxCola DO
     BEGIN
       IF ColaUmpdu[i].DefDeliv = 0 THEN
         BEGIN
           j := j + 1;
           ColaT[j] := ColaUmpdu[i]
         END
     END;
  IF ColaT[2] = " " THEN UMPDU := ColaT[1]
  ELSE
    BEGIN
      p := ColaT[1].Prioridad;
      j := 1;
      FOR i := 2 TO TamMaxCola DO
         BEGIN
           IF ColaT[i].Prioridad > p THEN
             BEGIN
               p := ColaT[i].Prioridad;
               j := i
             END
         END;
      UMPDU := ColaT[j]
    END;
  s := UMPDU.MpduId.Sec;
  FOR i := 1 TO TamMaxCola DO
     BEGIN
       IF ColaUmpdu[i].MpduId.Sec = s THEN
          BEGIN
            j := i;
            i := TamMaxCola + 1
          END
     END;
  CompactaColaUmpdu(j,ColaUmpdu)
END;
```

```
PROCEDURE CompactaColaUmpdu(j : INTEGER;
                            VAR ColaUmpdu : ARRAY[1..TamMaxCola]
                                            OF TipoUmpdu);
{Compacta el arreglo global ColaUmpdu.}
VAR
  m : INTEGER;
  i : INTEGER;
BEGIN
  m := TamMaxCola - 1;
  FOR i := j TO m DO ColaUmpdu[i] := ColaUmpdu[i + 1];
  ColaUmpdu[m + 1] := " "
END;
```

```
PROCEDURE ExtraeUmpdu(ColaUmpdu : ARRAY[1..TamMaxCola] OF TipoUmpdu;
                      IdContUA : INTEGER;
                      VAR Umpdu : TipoUmpdu);
{Extrae de la cola , la Umpdu que corresponde al Identificador de
 Contenido de UA dado. Compacta la cola .}
VAR
  i : INTEGER;
  J : INTEGER;
BEGIN
  FOR i := 1 TO TamMaxCola DO
     BEGIN
       IF ColaUmpdu[i].UAContId = IdContUA THEN
         BEGIN
           Umpdu := ColaUmpdu[i];
           J := i;
           i := TamMaxCola + 1
         END
     END;
  CompactaColaUmpdu(J,ColaUmpdu)
END;
```

```
PROCEDURE CreaDests(n : nombre_O/D;
                    VAR Dests : nombres_O/D);
{Coloca el nombre de un destinatario como primero y unico elemento
 del arreglo de nombres Dests.}
VAR
  i : INTEGER;
BEGIN
  Dests[1] := n;
  FOR i := 2 TO num_dest DO
     Dests[i] := " "
END;
```

```
PROCEDURE Receptores(UMPDU:TipoUmpdu;
                     VAR Dests: nombres_O/D);
{Crea el arreglo Dests que contiene los nombres de los destinatarios
 del mensaje depositado}
VAR
  i : INTEGER;
BEGIN
  FOR i := 1 TO num_dest DO
    BEGIN
      Dests[i] := " ";
      IF UMPDU.RecInfo[i].Recep <> " " THEN
         Dests[i] := UMPDU.RecInfo[i].Recep
    END
END;


PROCEDURE BorraDRmpdu(VAR DRMPDU:TipoDRmpdu);
{Elimina el contenido de la variable DRMPDU}
PRIMITIVE;


PROCEDURE BorraPmpdu(VAR PMPDU: TipoPmpdu);
{Elimina el contenido de la variable PMPDU}
PRIMITIVE;


PROCEDURE BorraUmpdu(VAR UMPDU: TipoUmpdu);
{Elimina el contenido de la variable UMPDU)
PRIMITIVE;
```

```
PROCEDURE GuardaIdSDep(IdSDep : INTEGER;
                       MpduId : INTEGER;
                       VAR IdSD : ARRAY[1..TamMaxCola] OF TipoIdSDS);
{Almacena el Identificador del Suceso Deposito que genero una Umpdu y
 el Identificador de la Umpdu , en el arreglo global IdSD .}
VAR
  i : INTEGER;
BEGIN
  FOR i := 1 TO TamMaxCola DO
     IF IdSD[i].MpduId = 0 THEN
       BEGIN
         IdSD[i].MpduId := MpduId;
         IdSD[i].DepSonda := IdSDep;
         i := TamMaxCola + 1
       END
END;


PROCEDURE SacarIdSDep(IdSD : ARRAY[1..TamMaxCola] OF TipoIdSDS;
                      MpduId : INTEGER;
                      VAR IdSDep : INTEGER);
{Extrae el Identificador del Suceso Deposito que genero una Umpdu
 dada , desde el arreglo global IdSD .}
VAR
  i : INTEGER;
BEGIN
  FOR i := 1 TO TamMaxCola DO
     IF IdSD[i].MpduId = MpduId THEN
       BEGIN
         IdSDep := IdSD[i].DepSonda;
         i := TamMaxCola + 1
       END
END;
```

```
PROCEDURE SacarIdSSonda(IdSS : ARRAY[1..TamMaxCola] OF TipoIdSDS;
                        MpduId : INTEGER;
                        VAR IdSSonda : INTEGER);
{Extrae el Identificador del Suceso Sonda que genero una Pmpdu
 dada , desde el arreglo global IdSS .}
VAR
  i : INTEGER;
BEGIN
  FOR i := 1 TO TamMaxCola DO
     IF IdSS[i].MpduId = MpduId THEN
       BEGIN
         IdSSonda := IdSS[i].DepSonda;
         i := TamMaxCola + 1
       END
END;


PROCEDURE BorraIdSDep(MpduId : INTEGER;
                      VAR IdSD : ARRAY[1..TamMaxCola] OF TipoIdSDS);
{Elimina la entrada correspondiente a una Umpdu dada , del arreglo
 global IdSD y lo compacta .}
VAR
  m , j , i : INTEGER;
BEGIN
  m := 0;
  FOR i := 1 TO TamMaxCola DO
     IF IdSD[i].MpduId <> 0 THEN m := m + 1;
  FOR i := 1 TO m DO
     BEGIN
       IF IdSD[i].MpduId = MpduId THEN
         BEGIN
           IdSD[i].MpduId := 0;
           IdSD[i].DepSonda := 0;
           j := i;
           i := TamMaxCola + 1
         END
     END;
  IF j <= (m - 1) THEN
    BEGIN
      FOR i := j TO (m - 1) DO
         IdSD[i] := IdSD[i + 1]
    END;
  IdSD[m].MpduId := 0;
  IdSD[m].DepSonda := 0
END;
```

```
PROCEDURE BorraIdSSonda(MpduId : INTEGER;
                        VAR IdSS : ARRAY[1..TamMaxCola] OF TipoIdSDS);
{Elimina la entrada correspondiente a una Pmpdu dada , del arreglo
 global IdSS y lo compacta .}
VAR
  m , j , i : INTEGER;
BEGIN
  m := 0;
  FOR i := 1 TO TamMaxCola DO
     IF IdSS[i].MpduId <> 0 THEN m := m + 1;
  FOR i := 1 TO m DO
     BEGIN
       IF IdSS[i].MpduId = MpduId THEN
         BEGIN
           IdSS[i].MpduId := 0;
           IdSS[i].DepSonda := 0;
           j := i;
           i := TamMaxCola + 1
         END
     END;
  IF j <= (m - 1) THEN
    BEGIN
      FOR i := j TO (m - 1) DO
         IdSS[i] := IdSS[i + 1]
    END;
  IdSS[m].MpduId := 0;
  IdSS[m].DepSonda := 0
END;
```

```
PROCEDURE CreaDRmpduXUmpdu(NumMpdu : INTEGER;
                           Umpdu : TipoUmpdu;
                           i : INTEGER;
                           VAR DRMPDU: TipoDRmpdu);
{Crea DRMPDU a partir de los campos de una Umpdu, para reportar informacion
 sobre una entrega al destinatario i .La informacion sobre el destinatario
 reportado se guarda en RepRecInfo[1] .}
BEGIN
  DRMPDU.MpduId.Sec := NumMpdu + 1;
  DRMPDU.Originador := Umpdu.Originador;
  DRMPDU.TraceInfo[1].GDId := EsteMta;
  DRMPDU.TraceInfo[1].Tllegada := FechaHoraSist;
  DRMPDU.TraceInfo[1].TDif := Umpdu.DefDeliv;
  DRMPDU.TraceInfo[1].Accion := 0;
  DRMPDU.OrigMpduId := Umpdu.MmpduId;
  DRMPDU.InterTraceInfo[1] := Umpdu.TraceInfo[1];
  DRMPDU.InterTraceInfo[2] := Umpdu.TraceInfo[2];
  DRMPDU.UAContId := Umpdu.UAContId;
  DRMPDU.RepRecInfo[1].Recep := Umpdu.RecInfo[i].Recep;
  DRMPDU.RepRecInfo[1].ExtId := Umpdu.RecInfo[i].ExtId;
  DRMPDU.RepRecInfo[1].FlagResp := Umpdu.RecInfo[i].FlagResp;
  DRMPDU.RepRecInfo[1].PeticionInf := Umpdu.RecInfo[i].PeticionInf;
  DRMPDU.RepRecInfo[1].PetInfUs := Umpdu.RecInf[i].PetInfUs;
  DRMPDU.RepRecInfo[1].Tllegada := FechaHoraSist;
  DRMPDU.RepRecInfo[1].RazonNoEntrega := " "
  DRMPDU.RepRecInfo[1].TipoUA := Privaado;
  IF Umpdu.RecInfo[1].PetInfUs = Basico AND Umpdu.ContRetReq
                           THEN DRMPDU.RetUmpduCont := Umpdu.Content
                           ELSE DRMPDU.RetUmpduCont := " "
END;
```

```
PROCEDURE ExtIdDestUmpdu(Destinatario : nombre_O/D;
                         Umpdu : TipoUmpdu;
                         VAR d : INTEGER);
{Encuentra el Identificador de Extension para un destinatario dado,
 buscandolo en el campo RecInfo de Umpdu}
VAR
  i : INTEGER;
BEGIN
  FOR i := 1 TO num_dest DO
     IF Umpdu.RecInfo[i].Recep = Destinatario THEN
       BEGIN
         d := Umpdu.RecInfo[i].ExtId;
         i := num_dest + 1
       END
END;
```

```
PROCEDURE CreaPmpdu(NumMpdu: INTEGER;
                    Destinatarios: nombres_O/D;
                    Originador: nombre_O/D;
                    Destin_Alter: BOOLEAN;
                    Long_contenido: INTEGER;
                    Id_Contenido_AU: INTEGER;
                    VAR Pmpdu: TipoPmpdu);
{Crea la variable Pmpdu con base a los parametros suministrados
 en la primitiva PROBE.Request.}
VAR
  i: INTEGER;
  t: INTEGER;
BEGIN
  Pmpdu.MpduId.Sec := NumMpdu;
  Pmpdu.Originador := Originador;
  Pmpdu.UAContId := Id_Contenido_AU;
  Pmpdu.TraceInfo[1].Tllegada := FechaHoraSist;
  Pmpdu.TraceInfo[1].TDif := 0;
  Pmpdu.TraceInfo[1].Accion := 0;
  Pmpdu.DiscRec := FALSE;
  IF Destin_Alter THEN Pmpdu.AlRecAllow := TRUE
                  ELSE Pmpdu.AlRecAllow := FALSE;
  Pmpdu.ContRetReq := FALSE;
  Pmpdu.ContLength := Long_contenido;
  t := TotalDest(Destinatarios);
  FOR i := 1 TO t DO
     BEGIN
       Pmpdu.RecInfo[i].Recp := Destinatarios[i];
       Pmpdu.RecInfo[i].ExtId := i;
       Pmpdu.RecInfo[i].FlagResp := Abierto;
       Pmpdu.RecInfo[i].PeticionInf := Basico;
       Pmpdu.RecInfo[i].PetInfUs := NoInf
     END
END;
```

```
PROCEDURE EncolarPmpdu(Pmpdu : TipoPmpdu;
                       VAR ColaPmpdu : ARRAY[1..TamMaxCola] of TipoPmpdu);
{Almacena Pmpdu en el arreglo global ColaPmpdu.}
VAR
  i : INTEGER;
BEGIN
  FOR i := 1 TO TamMaxCola DO
     BEGIN
       IF ColaPmpdu[i] = " " THEN
         BEGIN
           ColaPmpdu[i] := Pmpdu;
           i := TamMaxCola + 1
         END
     END
END;


PROCEDURE SacarPmpdu(ColaPmpdu : ARRAY[1..TamMaxCola] OF TipoPmpdu;
                     VAR PMPDU : TipoPmpdu);
{Extrae una variable global PMPDU de la cola y compacta la cola.
 La extraccion se efectua segun la politica Fifo.}
VAR
  i : INTEGER;
  m : INTEGER;
BEGIN
  PMPDU := ColaPmpdu[1];
  m := TamMaxCola - 1;
  FOR i := 1 TO m DO ColaPmpdu[i] := ColaPmpdu[i + 1];
  ColaPmpdu[m + 1] := " "
END;


PROCEDURE GuardaIdSSonda(IdSSonda : INTEGER;
                         MpduId : INTEGER;
                         VAR IdSS : ARRAY[1..TamMaxCola] OF TipoIdSDS);
{Almacena el Identificador del Suceso Sonda que genero una Pmpdu y
 el Identificador de la Pmpdu , en el arreglo global IdSS .}
VAR
  i : INTEGER;
BEGIN
  FOR i := 1 TO TamMaxCola DO
     IF IdSS[i].MpduId = 0 THEN
       BEGIN
         IdSS[i].MpduId := MpduId;
         IdSS[i].DepSonda := IdSSonda;
         i := TamMaxCola + 1
       END
END;
```

```
PROCEDURE RecepPmpdu(PMPDU:TipoPmpdu;
                     VAR Dests: nombres_O/D)
{Crea el arreglo Dests que contiene los nombres de los destinatarios
 de la sonda depositada .}
VAR
  i : INTEGER;
BEGIN
  FOR i := 1 TO num_dest DO
    BEGIN
      Dests[i] := " ";
      IF PMPDU.RecInfo[i].Recep <> " " THEN
         Dests[i] := PMPDU.RecInfo[i].Recep
    END
END;


PROCEDURE CreaDRmpduXPmpdu(NumMpdu : INTEGER;
                           Pmpdu : TipoPmpdu;
                           i : INTEGER;
                           VAR DRMPDU: TipoDRmpdu);
{Crea la variable global DRMPDU a partir de los campos de una Pmpdu, para
 reportar informacion sobre una hipotetica entrega al destinatario i. La
 informacion sobre el destinatario reportado se guarda en RepRecInfo[1] .}
BEGIN
  DRMPDU.MpduId.Sec := NumMpdu + 1;
  DRMPDU.Originador := Pmpdu.Originador;
  DRMPDU.TraceInfo[1].GDId := EsteMta;
  DRMPDU.TraceInfo[1].Tllegada := FechaHoraSist;
  DRMPDU.TraceInfo[1].TDif := 0;
  DRMPDU.TraceInfo[1].Accion := 0;
  DRMPDU.OrigMpduId := Pmpdu.MmpduId;
  DRMPDU.InterTraceInfo[1] := Pmpdu.TraceInfo[1];
  DRMPDU.InterTraceInfo[2] := Pmpdu.TraceInfo[2];
  DRMPDU.UAContId := Pmpdu.UAContId;
  DRMPDU.RepRecInfo[1].Recep := Pmpdu.RecInfo[i].Recep;
  DRMPDU.RepRecInfo[1].ExtId := Pmpdu.RecInfo[i].ExtId;
  DRMPDU.RepRecInfo[1].FlagResp := Pmpdu.RecInfo[i].FlagResp;
  DRMPDU.RepRecInfo[1].PeticionInf := Pmpdu.RecInfo[i].PeticionInf;
  DRMPDU.RepRecInfo[1].PetInfUs := Pmpdu.RecInf[i].PetInfUs;
  DRMPDU.RepRecInfo[1].Tllegada := FechaHoraSist;
  DRMPDU.RepRecInfo[1].RazonNoEntrega := " ";
  DRMPDU.RepRecInfo[1].TipoUA := Privaado;
  DRMPDU.RetUmpduCont := " "
END;
```

```
PROCEDURE ExtIdDestPmpdu(Destinatario : nombre_O/D;
                         Pmpdu : TipoPmpdu;
                         VAR d : INTEGER);
{Encuentra el Identificador de Extension para un destinatario dado,
 buscandolo en el campo RecInfo de Pmpdu}
VAR
  i : INTEGER;
BEGIN
  FOR i := 1 TO num_dest DO
     IF Pmpdu.RecInfo[i].Recep = Destinatario THEN
       BEGIN
         d := Pmpdu.RecInfo[i].ExtId;
         i := num_dest + 1
       END
END;


INITIALIZE
    TO Idle
    BEGIN
       Id_suceso_deposito := 0;
       Id_suceso_entrega := 0;
       Id_suceso_sonda := 0;
       NumMpdu := 0;
       lm := 0;
       rm := 0;
       tdm := 0;
       ls := 0;
       rs := 0;
       tds := 0;
       HayMensaje := FALSE;
       HaySonda := FALSE;
    END;
```

```
TRANS

{Trans #1. Peticion de deposito por un usuario}
WHEN A.SUBMISSION.Request
FROM Cualquiera TO Active
VAR
  Umpdu : TipoUmpdu;
  IdSDep : INTEGER;
BEGIN
  Validar(Originador,Destinatarios,Indicacion_exito,Motivo_fracaso);
  IF Indicacion_exito THEN
    BEGIN
      Sello_deposito := FechaHoraSist;
      Id_suceso_deposito := Id_suceso_deposito + 1;
      IdSDep := Id_suceso_deposito;
      NumMpdu := NumMpdu + 1;
      CreaUmpdu(NumMpdu,Destinatarios,Originador,Contenido,
                Supresion_NNE,Prioridad,T_E_Diferida,Notificacion_E,
                Revelacion_Destin,Destin_Alter,Devolucion_Contenido,
                Id_Contenido_AU,Umpdu);
      EncolarUmpdu(Umpdu,ColaUmpdu);
      GuardaIdSDep(IdSDep,Umpdu.MpduId.Sec,IdSD);
      IF Umpdu.DefDeliv = 0 THEN HayMensaje := TRUE
      ELSE
        BEGIN
          HayMensaje := FALSE;
          OUTPUT B.STORE.Request(Umpdu)
        END
    END
  ELSE
    BEGIN
      Sello_deposito := 0;
      IdSDep := 0;
      HayMensaje := FALSE
    END;
  OUTPUT A.SUBMISSION.Confirmation(Indicacion_exito,Sello_deposito,
         IdSDep,Motivo_fracaso,Id_Contenido_AU)
END;
```

```
{Trans #2. Extrae una Umpdu de la cola, creandose la variable global
           UMPDU y separa los destinatarios locales de los remotos en
           arreglos separados. }
FROM Cualquiera TO SAME
PROVIDED (lm =  0) AND (rm =  0) AND HayMensaje
VAR
  Dests : nombres_D/D;
BEGIN
  SacarUmpdu(ColaUmpdu,UMPDU);
  Receptores(UMPDU,Dests);
  tdm := TotalDest(Dests);
  SeparaLocRem(Dests,DestLocM,lm,DestRemM,rm);
END;
```

```
{Trans #3. Entrega de mensajes a los destinatarios locales y Notificacion
           de entrega al Originador si esto ultimo se solicito.}
FROM Cualquiera TO Idle
PROVIDED HayMensaje AND (lm > 0)
VAR
  Dests : nombres_O/D;
  IdSDep : INTEGER;
  i : INTEGER;
BEGIN
  Receptores(UMPDU,Dests);
  IF UMPDU.DiscRec THEN OtrosDest(Dests,DestLocM[1],Otros_destinatarios)
                   ELSE Otros_destinatarios := " ";
  Tiempo_entrega := FechaHoraSist;
  FOR i := 1 TO lm DO
     BEGIN
       IF DestLocM[lm] <> UMPDU.RecInfo[1].Recep
                     THEN Destinatario_deseado := UMPDU.RecInfo[1].Recep
                     ELSE Destinatario_deseado := " ";
       Id_suceso_entrega := Id_suceso_entrega + 1;
       OUTPUT A.DELIVERY.Indication(UMPDU.Originador,DestLocM[lm],
                           Otros_destinatarios,UMPDU.Content,Tiempo_entrega,
                           UMPDU.TraceInfo[1].Tllegada,UMPDU.Prioridad,
                           Destinatario_deseado,Id_suceso_entrega)
     END;
  IF UMPDU.RecInfo[1].PeticionInf = Confirmado THEN
    BEGIN
      Tipo_notificacion := TRUE;
      SacarIdSDep(IdSD,UMPDU.MpduId.Sec,IdSDep);
      Id_suceso_deposito_sonda := IdSDep;
      Motivo_no_entrega := " ";
      Destinatario_deseado := " ";
      IF UMPDU.ContRetReq THEN Contenido_devuelto := UMPDU.Content
                          ELSE Contenido_devuelto := " ";
      Id_contenido_AU := UMPDU.UAContId;
      FOR i := 1 TO lm DO
         BEGIN
           CreaDests(DestLoc[lm],Dests);
           OUTPUT A.NOTIFY.Indication(Tipo_notificacion,
                             Id_suceso_deposito_sonda,Dests,
                             Motivo_no_entrega,Tiempo_entrega,
                             Destinatario_deseado,Contenido_devuelto,
                             Id_contenido_AU)
         END
    END;
  tdm := tdm - lm;
  lm := 0;
  IF tdm = 0 THEN
    BEGIN
      BorraGlobales(lm,DestLocM,rm,DestRemM,tdm);
```

```
        BorraIdSDep(UMPDU.MpduId.Sec,IdSD);
        BorraUmpdu(UMPDU);
        HayMensaje := FALSE
    END
END;
```

```
{Trans #4. Transferencia de mensajes a los destinatarios remotos.}
FROM Cualquiera TO Active
PROVIDED HayMensaje AND (rm > 0)
VAR
  Nombre_MTA:nombre_mta;
  i : INTEGER;
  IdSDep :INTEGER;
BEGIN
  SacarIdSDep(IdSD,UMPDU.MpduId.Sec,IdSDep);
  RefMensaje.RefMens := IdSDep;
  RefMensaje.TipoMpdu := Umpdu;
  FOR i := 1 TO rm DO
     BEGIN
       BuscarMTA(DestRemM[rm],Nombre_MTA);
       OUTPUT C.TRANSFER.Request(Nombre_MTA,RefMensaje,
                                 DestRemM[rm],UMPDU,DRMPDU,PMPDU)
     END;
  IF (tdm - rm) = 0 THEN HayMensaje := FALSE;
  rm := 0;
END;
```

```
{Trans #5. Llega Confirmacion de una solicitud previa de Transferencia.}
WHEN C.TRANSFER.Confirmation
FROM Active TO Idle
PROVIDED Indicacion_exito
BEGIN
  IF RefMensaje.Tipo = Umpdu THEN
    BEGIN
      tdm := tdm - 1;
      IF tdm = 0 THEN
        BEGIN
          HayMensaje := FALSE;
          BorraGlobales(lm,DestLocM,rm,DestRemM,tdm);
          BorraIdSDep(UMPDU.MpduId.Sec,IdSD);
          BorraUmpdu(UMPDU)
        END
    END;
  IF RefMensaje.TipoMpdu = Pmpdu THEN
    BEGIN
      tds := tds - 1;
      IF tds = 0 THEN
        BEGIN
          HaySonda = FALSE;
          BorraGlobales(ls,DestLocS,rs,DestRemS,tds);
          BorraIdSSonda(PMPDU.MpduId.Sec,IdSS);
          BorraPmpdu(PMPDU)
        END
    END;
  IF RefMensaje.TipoMpdu = DRmpdu THEN BorraDRmpdu(DRMPDU)
END;

WHEN C.TRANSFER.Confirmation
FROM Active TO Idle
PROVIDED NOT Indicacion_exito
VAR
  d : INTEGER;
  Dests : nombres_O/D;
  IdSDep : INTEGER;
  IdSSonda : INTEGER;
BEGIN
  IF RefMensaje.TipoMpdu = DRmpdu THEN BorraDRmpdu(DRMPDU)
  ELSE
     BEGIN
       IF RefMensaje.TipoMpdu = Umpdu THEN
         BEGIN
           ExtIdDestUmpdu(Destinatario,UMPDU,d);
           IF UMPDU.RecInfoId].PetInfUs <> NoInf THEN
             BEGIN
               Tipo_notificacion := FALSE;
               Tiempo_entrega := 0;
```

```
            :  :arIdSDep(IdSD,UMPDU.MpduId.Sec,IdSDep);
               suceso_deposito_sonda := IdSDep;
            |  ;ivo_no_entrega := Motivo_fracaso;
               UMPDU.ContRetReq THEN Contenido_devuelto := UMPDU.Content
                                ELSE Contenido_devuelto := " ";
            IF d <> 1 THEN Destinatario_deseado := UMPDU.RecInfo[1].Recep
                      ELSE Destinatario_deseado := " ";
            Id_contenido_AU := UMPDU.UAContId;
            CreaDests(Destinatario,Dests);
            OUTPUT A.NOTIFY.Indication(Tipo_notificacion,
                     Id_suceso_deposito_sonda,Dests,
                     Motivo_no_entrega,Tiempo_entrega,
                     Destinatario_deseado,Contenido_devuelto,
                     Id_contenido_AU)
          END;
        tdm := tdm - 1;
        IF tdm = 0 THEN
          BEGIN
            BorraGlobales(lm,DestLocM,rm,DestRemM,tdm);
            BorraIdSDep(UMPDU.MpduId.Sec,IdSD);
            BorraUmpdu(UMPDU)
          END
      END
    ELSE
      BEGIN
        Tipo_notificacion := FALSE;
        Tiempo_entrega := 0;
        Motivo_no_entrega := Motivo_fracaso;
        SacarIdSSonda(IdSS,PMPDU.MpduId.Sec,IdSSonda);
        Id_suceso_deposito_sonda := IdSSonda;
        IF d <> 1 THEN Destinatario_deseado := PMPDU.RecInfo[1].Recep
                  ELSE Destinatario_deseado := " ";
        Contenido_devuelto := " ";
        Id_contenido_AU := PMPDU.UAContId
        CreaDests(Destinatario,Dests);
        OUTPUT A.NOTIFY.Indication(Tipo_notificacion,
                 Id_suceso_deposito_sonda,Dests,
                 Motivo_no_entrega,Tiempo_entrega,
                 Destinatario_deseado,Contenido_devuelto,
                 Id_contenido_AU);
        tds := tds - 1;
        IF tds = 0 THEN
          BEGIN
            BorraGlobales(ls,DestLocS,rs,DestRemS,tds);
            BorraIdSSonda(PMPDU.MpduId.Sec,IdSS);
            BorraPmpdu(PMPDU)
          END
      END
  END
END;
```

```
{Trans #6 :Llega Umpdu desde el Gestor de Asociaciones}
WHEN C.TRANSFER.Indication
FROM Cualquiera TO Active
PROVIDED RefMensaje.TipoMpdu = Umpdu
VAR
  Dests : nombres_O/D;
  d : INTEGER;
  Nombre_MTA : nombre_mta;
BEGIN
  Umpdu.TraceInfo[2].GDId := EsteMta;
  Umpdu.TraceInfo[2].Tllegada := FechaHoraSist;
  Umpdu.TraceInfo[2].Accion := 0;
  Receptores(Umpdu,Dests);
  IF Umpdu.DiscRec THEN OtrosDest(Dests,Destinatario,Otros_destinatarios)
                   ELSE Otros_destinatarios := " ";
  Tiempo_entrega := FechaHoraSist;
  IF Destinatario <> Umpdu.RecInfo[1].Recep
          THEN Destinatario_deseado := Umpdu.RecInfo[1].Recep
          ELSE Destinatario_deseado := " ";
  Id_suceso_entrega := Id_suceso_entrega + 1;
  OUTPUT A.DELIVERY.Indication(Umpdu.Originador,Destinatario,
                               Otros_destinatarios,Umpdu.Content,
                               Tiempo_entrega,Umpdu.TraceInfo[1].Tllegada,
                               Umpdu.Prioridad,Destinatario_deseado,
                               Id_suceso_entrega);
  ExtIdDestUmpdu(Destinatario,Umpdu,d);
  IF Umpdu.RecInfo[d].PetInfUs = Confirmado THEN
    BEGIN
      NumMpdu := NumMpdu + 1;
      DRMPDU.RepRecInfo[1].TEntrega := Tiempo_entrega;
      CreaDRmpduXUmpdu(NumMpdu,Umpdu,d,DRMPDU);
      BuscarMTA(Umpdu.Originador,Nombre_MTA);
      RefMensaje.RefMens := RefMensaje.RefMens;
      RefMensaje.TipoMpdu := DRmpdu;
      OUTPUT C.TRANSFER.Request(Nombre_MTA,RefMensaje,
                        Umpdu.Originador,UMPDU,DRMPDU,PMPDU)
    END
END;
```

```
{Trans #7 : Llega DRmpdu desde el Gestor de Asociaciones.}
WHEN C.TRANSFER.Indication
FROM Cualquiera TO Active
PROVIDED RefMensaje.TipoMpdu = DRmpdu
VAR
  d: INTEGER;
  Dests : nombres_O/D;
BEGIN
  DRmpdu.TraceInfo[2].GDId := EsteMta;
  DRmpdu.TraceInfo[2].Tllegada := FechaHoraSist;
  DRmpdu.TraceInfo[2].TDif := 0;
  DRmpdu.TraceInfo[2].Accion := 0;
  Id_suceso_deposito_sonda := RefMensaje.RefMens;
  Motivo_no_entrega := DRmpdu.RepRecInfo[1].RazonNoEntrega;
  Tipo_notificacion := TRUE;
  Tiempo_entrega := DRmpdu.RepRecInfo[1].TEntrega;
  Destinatario_deseado := " ";
  Contenido_devuelto := DRmpdu.RetUmpduCont;
  CreaDests(DRmpdu.RepRecInfo[1].Recep,Dests);
  Id_Contenido_AU := DRmpdu.UAContId;
  OUTPUT A.NOTIFY.Indication(Tipo_notificacion,Id_suceso_deposito_sonda,
                             Dests,Motivo_no_entrega,Tiempo_entrega,
                             Destinatario_deseado,Contenido_devuelto,
                             Id_Contenido_AU)
END;
```

```
{Trans #8. Llega Confirmacion de Almacenamiento para entrega diferida}
WHEN B.STORE.Confirmation
FROM Active TO Idle
VAR
  Umpdu : TipoUmpdu;
  Dests : nombres_O/D;
  IdSDep : INTEGER;
BEGIN
  ExtraeUmpdu(ColaUmpdu,Id_contenido_AU,Umpdu);
  IF NOT Indicacion_exito THEN
    BEGIN
      IF Umpdu.RecInfo[1].PetInfUs <> NoInf THEN
        BEGIN
          Tipo_notificacion := FALSE;
          Receptores(Umpdu,Dests);
          Tiempo_entrega := 0;
          SacarIdSDep(IdSD,Umpdu.MpduId.Sec,IdSDep);
          Id_suceso_deposito_sonda := IdSDep;
          Motivo_no_entrega := Motivo_fracaso;
          IF Umpdu.ContRetReq THEN Contenido_devuelto := UMPDU.Content
                              ELSE Contenido_devuelto := " ";
          Destinatario_deseado := Umpdu.RecInfo[1].Recep;
          OUTPUT A.NOTIFY.Indication(Tipo_notificacion,
                   Id_suceso_deposito_sonda,Dests,Motivo_no_entrega,
                   Tiempo_entrega,Destinatario_deseado,
                   Contenido_devuelto,Id_contenido_AU);
        END
    END;
  BorraIdSDep(Umpdu.MpduId.Sec,IdSD)
END;
```

```
{Trans #9.  El Archivador de Mensajes con Entrega Diferida indica
            al Despachador de Mensajes que el periodo de tiempo
            especificado para la entrega diferida  de un mensaje,
            ha finalizado.}
WHEN B.EXPIRATION.Indication
FROM Cualquiera TO Active
BEGIN
  MensSolicitado := MensExp;
  OUTPUT B.RETURN.Request(MensSolicitado)
END;


{Trans #10.  El Archivador de Mensajes con Entrega Diferida retorna
            al Despachador de Mensajes un mensaje cuyo periodo de
            tiempo especificado para la entrega diferida ha finalizado.
            Los mensajes son entregados a la Entidad Agente Usuario
            receptora por las transiciones #2 , #3 y #4. }
WHEN B.RETURN.Indication
FROM Cualquiera TO Idle
VAR
  Umpdu : TipoUmpdu;
BEGIN
  HayMensaje := TRUE;
  EncolarUmpdu(Umpdu,ColaUmpdu);
END;


{Trans #11.  Solicitud de Cancelacion de entrega o transferencia de un
            mensaje con Entrega Diferida depositado anteriormente}
WHEN A.CANCEL.Request
FROM Cualquiera TO Active
BEGIN
  IdMensaje := Id_Suceso_deposito;
  OUTPUT B.DELETE.Request(IdMensaje)
END;


{Trans #12.El Archivador de Mensajes con Entrega Diferida confir-
            ma la eliminacion de un mensaje}
WHEN B.DELETE.Confirmation
FROM Active TO Idle
BEGIN
  OUTPUT A.CANCEL.Confirmation(Indicacion_exito,
                               Motivo_fracaso)
END;
```

```
{Trans #13. Peticion de sonda por un usuario.}
WHEN A.PROBE.Request
FROM Cualquiera TO Active
VAR
  Pmpdu := TipoPmpdu;
  IdSSonda := INTEGER;
BEGIN
  Validar(Originador,Destinatarios,Indicacion_exito,Motivo_fracaso);
  IF Indicacion_exito THEN
    BEGIN
      Tiempo_sonda := FechaHoraSist;
      HaySonda := TRUE;
      Id_suceso_sonda := Id_suceso_sonda + 1;
      NumMpdu := NumMpdu + 1;
      CreaPmpdu(NumMpdu,Destinatarios,Originador,Destin_Alter,
                Long_contenido,Id_contenido_AU,Pmpdu);
      EncolarPmpdu(Pmpdu,ColaPmpdu);
      IdSSonda := Id_suceso_sonda;
      GuardaIdSSonda(IdSSonda,Pmpdu.MpduId.Sec,IdSS)
    END
  ELSE
    BEGIN
      Tiempo_sonda := 0;
      IdSSonda := 0;
      HaySonda := FALSE
    END;
  OUTPUT A.PROBE.Confirmation(Indicacion_exito,tiempo_sonda,
                              IdSSonda,Motivo_fracaso,
                              Id_contenido_AU)
END;


{Trans #14. Se extrae una Pmpdu de la cola creandose la variable global
            PMPDU y se separan los destinatarios locales de los
            remotos en arreglos separados .}
PROVIDED (ls = 0) AND (rs = 0) AND HaySonda
VAR
  Dests : nombres_O/D;
BEGIN
  SacarPmpdu(ColaPmpdu,PMPDU);
  RecepPmpdu(PMPDU,Dests);
  tds := TotalDest(Dests);
  SeparaLocRem(Dests,DestLocS,ls,DestRemS,rs)
END;
```

```
{Trans #15. Notificaciones de entrega al originador como resultado de una
            sonda con destinatarios locales .}
FROM Cualquiera TO Idle
PROVIDED HaySonda AND (ls > 0)
VAR
  Dests : nombres_O/D;
  IdSSonda : INTEGER;
  i : INTEGER;
BEGIN
  Tipo_notificacion := TRUE;
  SacarIdSSonda(IdSS,PMPDU.MpduId.Sec,IdSSonda);
  Id_suceso_deposito_sonda := IdSSonda;
  Motivo_no_entrega := " ";
  Tiempo_entrega := FechaHoraSist;
  Destinatario_deseado := " ";
  Contenido_devuelto := " ";
  Id_contenido_AU := PMPDU.UAContId;
  FOR i := 1 TO ls DO
     BEGIN
       CreaDests(DestLocS[ls],Dests);
       OUTPUT A.NOTIFY.Indication(Tipo_notificacion,
                        Id_suceso_deposito_sonda,Dests,
                        Motivo_no_entrega,Tiempo_entrega,
                        Destinatario_deseado,Contenido_devuelto,
                        Id_contenido_AU)
     END;
  tds := tds - ls;
  ls := 0;
  IF tds = 0 THEN
    BEGIN
      BorraGlobales(ls,DestLocS,rs,DestRemS,tds);
      BorraIdSSonda(PMPDU.MpduId.Sec,IdSS);
      BorraPmpdu(PMPDU);
      HaySonda := FALSE
    END
END;
```

```
{Trans #16. Transferencia de sondas buscando a los destinatarios remotos.}
FROM Cualquiera TO Active
PROVIDED HaySonda AND (rs > 0)
VAR
  Nombre_MTA:nombre_mta;
  i : INTEGER;
  IdSSonda : INTEGER;
BEGIN
   SacarIdSSonda(IdSS,PMPDU.MpduId.Sec,IdSSonda);
   RefMensaje.RefMens := IdSSonda;
   RefMensaje.TipoMpdu := Pmpdu;
   FOR i := 1 TO rs DO
      BEGIN
        BuscarMTA(DestRemS[rs],Nombre_MTA);
        OUTPUT C.TRANSFER.Request(Nombre_MTA,RefMensaje,
                                  DestRemS[rs],UMPDU,DRMPDU,PMPDU)
      END;
   IF (tds - rs) = 0 THEN HaySonda := FALSE;
   rs := 0
END;
```

```
{Trans #17 :Llega Pmpdu desde el Gestor de Asociaciones}
WHEN C.TRANSFER.Indication
FROM Cualquiera TO Active
PROVIDED RefMensaje.TipoMpdu = Pmpdu
VAR
  d:INTEGER;
  Nombre_MTA : nombre_mta;
BEGIN
  Pmpdu.TraceInfo[2].GDId := EsteMta;
  Pmpdu.TraceInfo[2].Tllegada := FechaHoraSist;
  Pmpdu.TraceInfo[2].Accion := 0;
  ExtIdDestPmpdu(Destinatario,Pmpdu,d);
  NumMpdu := NumMpdu + 1;
  DRMPDU.RepRecInfo[1].TEntrega := FechaHoraSist;
  CreaDRmpduXPmpdu(NumMpdu,Pmpdu,d,DRMPDU);
  BuscarMTA(Pmpdu.Originador,Nombre_MTA);
  RefMensaje.RefMens := RefMensaje.RefMens;
  RefMensaje.TipoMpdu := DRmpdu;
  OUTPUT C.TRANSFER.Request(Nombre_MTA,RefMensaje,
                    Pmpdu.Originador,UMPDU,DRMPDU,PMPDU)
END;
```

```
SPECIFICATION ModuloAM  ;

CONST
   max = ANY INTEGER ;
   MaxNPuntosInteracRTS = ANY INTEGER ; {Numero maximo de puntos de interaccion
                                         en la interface con el RTS}


TYPE
   Clase_mpdu = (Umpdu,DRmpdu,Pmpdu) ;
   RefMpdu = RECORD
               RefMens : INTEGER ;
               TipoMpdu : Clase_mpdu
             END ;

   TipoMPDU = RECORD
                CASE Clase_mpdu OF
                  Umpdu : ( UMPDU : TIPOUmpdu ) ;
                  DRmpdu : ( DRMPDU : TipoDRmpdu ) ;
                  Pmpdu : ( PMPDU : TipoPmpdu )
               END ;

   TipProtocoloAplicacion = P1 ;
   TipoDireccSesion = ... ;
   TipoModoDialogo = (Monologo, BidirecAlter) ;
   TTurnoInicial = (Iniciador, Respondedor) ;
   DatosU_RTS = ... ;
   TipoDisposicion = (Aceptado, Rechazado) ;
   RazonRechazoGA-GA = (AMtoAMUnacceptableDialogueMode,
                        AMtoAMAuthenticationFailure, AMtoAMBusy) ;
   { Posibles motivos de rechazo de una asociacion por un MTA remoto }
   ClasePrdad = (Urgente, Normal, NoUrgente) ;
   FechaHora = INTEGER ;  { Se sugiere el formato  aammddhhmmss  }
   RazonRechazoGA-DM = (AMtoMDMTAUnrecognized,
                        AMtoMDAuthenticationFailure,
                        AMtoMDUnacceptableProperties,
                        AMtoMDBusy) ;
   { Posibles motivos de rechazo local de una asociacion }
   nombre_mta = STRING[max] ;
   Index = 1 .. MaxNPuntosInteracRTS ;
```

```
{    Definicion de los Canales del Gestor de Asociasiones  }

CHANNEL DM_GA (User, Provider) ;

   BY User :

      TRANSFER.Request (Nombre_MTA : nombre_mta ;
                        MessageReference : RefMpdu ;
                        Destinatario : Nombre_O/D ;
                        Umpdu : TipoUMPDU ;
                        DRmpdu : TipoDRmpdu ;
                        Pmpdu : TipoPmpdu );

   BY Provider :

      TRANSFER.Indication (Nombre_MTA : nombre_mta ;
                           RefMensaje : RefMpdu ;
                           Destinatario : Nombre_O/D ;
                           Umpdu : TipoUmpdu ;
                           DRmpdu : TipoDRmpdu ;
                           Pmpdu : TipoPmpdu ) ;

      TRANSFER.Confirmation ( Indicacion_exito : BOOLEAN ;
                              MessageReference : RefMpdu ;
                              Destinatario : nombre_O/D ;
                              MotivoRechazo : { optional } RazonRechazoGA-DM) ;
```

```
CHANNEL GA_RTS (User, Provider) ;

   BY User :

      WAITING_Call (K : Index) ;

      OPEN.Request (K : Index ;
                    DireccRespondedor : TipoDireccSesion ;
                    ModoDialogo : TipoModoDialogo ;
                    TurnoInicial : TTurnoInicial ;
                    ProtocoloAplic : TipProtocoloAplicacion ;
                    DatosUsuario : { optional } DatosU_RTS) ;

      OPEN.Response (K : Index ;
                     Disposicion : TipoDisposicion ;
                     DatosUsuario : { optional } DatosU_RTS ;
                     MotivoRechazo : { optional } RazonRechazoGA-GA) ;

      CLOSE.Request (K : Index) ;

      CLOSE.Response (K : Index) ;

      TURN_PLEASE.Request (K : Index ;
                           Prioridad : ClasePrdad) ;

      TURN_GIVE.Request (K : Index) ;

      TRANSFER.Request (K : Index ;
                        MPDU : TipoMPDU ;
                        HoraTransfer : INTEGER) ;

      USER_Credit (K : Index) ;


   BY Provider :

      OPEN.Indication (K : Index ;
                       DirIniciador : TipoDireccSesion ;
                       ModoDialogo : TipoModoDialogo ;
                       TurnoInicial : TTurnoInicial ;
                       ProtocoloAplic : TipProtocoloAplicacion ;
                       DatosUsuario :  { optional } DatosU_RTS) ;

      OPEN.Confirmation (K : Index ;
                         Disposicion : TipoDisposicion ;
                         DatosUsuario : { optional } DatosU_RTS ;
                         MotivoRechazo : { optional } RazonRechazoGA-GA) ;
```

```
      CLOSE.Indication (K : Index) ;

      CLOSE.Confirmation (K : Index) ;

      TURN_PLEASE.Indication (K : Index ;
                              Prioridad : ClasePrdad) ;

      TURN_GIVE.Indication (K : Index) ;

      TRANSFER.Indication (K : Index ;
                           MPDU : TipoMPDU ) ;

      EXCEPTION.Indication (K : Index ;
                            MPDU : TipoMPDU ) ;

      ABORT_Association (K : Index) ;

      PROVIDER_Credit (K : Index) ;

CHANNEL GA_SO (User, Provider) ;

   BY User :

      TIMER.Request (K : Index ;
                     Time : INTEGER) ;

      TIMER_Stop (K : Index) ;

   BY Provider :

      TIMER.Confirmation (K : Index) ;

      TIMER_CHECK.Indication ;
```

```
MODULE GestorAsociaciones PROCESS
       (C : DM_GA (Provider) COMMON QUEUE ;
        D : GA_RTS (User) COMMON QUEUE ;
        E : GA_SO (User) COMMON QUEUE) ;

BODY Ga FOR GestorAsociaciones ;

CONST
   NumMaxIntentosOpen = ... ;

TYPE
   TipEstadoAsociacion = (free, opening, busy, closing, repose,
                          waitingcall, prereceiver, receiver) ;
   TipoAsociacion = (permanent, temporal) ;
   TipoPassw = ... ;
   TipMapaAsociacion = RECORD
                          AssociationState : TipEstadoAsociacion ;
                          Association : TipoAsociacion ;
                          Name : nombre_mta ;
                          Password : TipoPassw ;
                          Address : TipoDireccSesion ;
                          Dialogue : TipoModoDialogo ;
                          NumberOpenTries : { optional } INTEGER ;
                          DidIopen : BOOLEAN ;
                          HaveIturn : BOOLEAN ;
                          RequestedTurn : BOOLEAN ;
                          RequestedPriority : ClasePrdad ;
                          MessageReference : RefMpdu ;
                          SubmissionPriority : ClasePrdad ;
                          MPDULastContentLength : INTEGER ;
                          SubmissionLastTime : FechaHora ;
                          DeliveryLastTime : FechaHora ;
                          MPDULastStoredTime : FechaHora
                       END ;
   TipMapaAsociaciones = ARRAY [1 .. MaxNPuntosInteracRTS] OF TipMapaAsociacion ;
   TipoIdentifMPDU = ... ;
   TipoHora = 0..23 ;


VAR
   NomEsteMTA : nombre_mta ;
   PasswEsteMTA : TipoPassw ;
   DirecSesionEsteMTA : TipoDireccSesion ;
   N : INTEGER ;    { Numero de recursos del MTA local.}
   MapaAsociaciones : TipMapaAsociaciones ;
   STATE : (EstadoUnico) ;
```

```
PROCEDURE AtributosEsteMTA ( VAR NomEsteMTA : nombre_mta ;
                             VAR PasswEsteMTA : TipoPassw ;
                             VAR DirecSesionEsteMTA : TipoDireccSesion ) ;
{ Retorna las propiedades del mta local. }
        PRIMITIVE ;


FUNCTION HayPropiedadesMTA ( Nombre_MTA : nombre_mta ;
                           VAR PasswMTA : { optional } TipoPassw ;
                           VAR DireccSesion : TipoDireccSesion ) : BOOLEAN ;
{ Retorna TRUE y las propiedades del MTA remoto si se encuentran
  registradas. }
        PRIMITIVE ;


FUNCTION BusqueNomMTAtemporal ( VAR Nombre_MTA : nombre_mta )
                                          : BOOLEAN ;
{ Retorna TRUE y el nombre del MTA para el cual existe una asociacion
  temporal. }
        PRIMITIVE ;


FUNCTION BusqueNomMTApermanent ( VAR Nombre_MTA : nombre_mta )
                                          : BOOLEAN ;
{ Retorna TRUE y el nombre del MTA para el que exista una asociacion
  permanente. }
        PRIMITIVE ;
```

```
FUNCTION HayAcuerdoBilateral ( Nombre_MTA : nombre_mta ;
                               VAR NumeroAsociacion : INTEGER ;
                               VAR ModoDialogo : TipoModoDialogo ;
                               VAR ResponsabilidadAbrir : BOOLEAN ;
                               VAR HoraInicialAbrir : { optional } TipoHora ;
                               VAR HoraFinalAbrir : { optional } TipoHora )
                               : BOOLEAN ;
{ Retorna TRUE si los acuerdos bilaterales se establecieron correctamente.
  El numero de asociaciones se refiere solamente a asociaciones temporales.
  Retorna los valores Default si no se establecen acuerdos.
  Los valores Default son :
  NumeroAsociacion := 1
  ModoDialogo := BidirecAlter
  ResponsabilidadAbrir := TRUE
  HoraInicialAbrir := 0
  UltimaHoraAbrir := 23  }
        PRIMITIVE ;


FUNCTION TieneResponsabilidadAbrir ( Nombre_MTA : nombre_mta ) : BOOLEAN ;
{ Retorna TRUE si se tiene la responsabilidad de abrir asociaciones con
  el MTA remoto. }
VAR
   NumeroAsociacion : INTEGER ;
   ModoDialogo : TipoModoDialogo ;
   ResponsabilidadAbrir : BOOLEAN ;
   HoraInicial : TipoHora ;
   HoraFinal : TipoHora ;
BEGIN
   IF HayAcuerdoBilateral (Nombre_MTA, NumeroAsociacion, ModoDialogo,
                           ResponsabilidadAbrir, HoraInicial, HoraFinal)
   THEN
      TieneResponsabilidadAbrir := ResponsabilidadAbrir
   ELSE
      TieneResponsabilidadAbrir := FALSE
END ;


PROCEDURE InicialiceRecursos ;
{ Inicializa la varable Recursos disponibles.}
        PRIMITIVE ;
```

```
FUNCTION ExisteRecursos : BOOLEAN ;
{ Retorna TRUE si hay recursos disponibles en este MTA. Tambien decrementa
  el numero de recursos disponibles. }
BEGIN
   IF N > 0  THEN
      BEGIN
         N := N - 1 ;
         ExisteRecursos := TRUE
      END
   ELSE
      ExisteRecursos := FALSE
END;



PROCEDURE DevuelveRecursos ;
{ Retorna un recurso disponible. Incrementa la variable Recursos.}
BEGIN
   N := N + 1
END;



PROCEDURE InicialiceColaOpening ;
{ Inicializa la estructura de datos que contendra MTAs remotos que esperan
  abrir asociaciones (opening). }
        PRIMITIVE ;



PROCEDURE EncolarParaAbrir ( Nombre_MTA : nombre_mta ;
                             Tiempo : FechaHora )
{ Coloca el nombre del MTA y el tiempo de rechazo de la apertura al final de
  estructura de datos del MTA remoto no abierto .}
                              PRIMITIVE  ;



FUNCTION HayMTAesperandoAbrir ( Nombre_MTA : nombre_mta ) : BOOLEAN ;
{ Retorna TRUE si el MTA remoto esta esperando abrir una asociacion. }
        PRIMITIVE ;
```

```
PROCEDURE RegistreOpen ( DirIniciador : TipoDireccSesion ;
                         DireccRespondedor : TipoDireccSesion ;
                         Disposicion : TipoDisposicion ;
                         MotivoRechazo : { optional } RazonRechazoGA-GA ;
                         ModoDialogo : TipoModoDialogo ;
                         TurnoInicial : TTurnoInicial ;
                         ProtocoloAplic : TipProtocoloAplicacion ) ;
{ Con  esta operacion se registra el exito o fracaso de la apertura de una
  asociacion del RTS hacia/desde un MTA vecino. Se graba en un archivo con
  propositos estadisticos.}
PRIMITIVE ;


PROCEDURE RegistreTurnGive ( DirIniciador : TipoDireccSesion ;
                             DireccRespondedor : TipoDireccSesion ) ;
{ Con esta operacion se registra o graba el intercambio de turno debido a una
  asociacion abierta con el RTS. }
                                  PRIMITIVE  ;


PROCEDURE RegistreTurnPlease ( DirIniciador : TipoDireccSesion ;
                               DireccRespondedor : TipoDireccSesion ;
                               Prioridad : ClasePrdad ) ;
{ Esta operacion registra la solicitud de turno durante una asociacion
  abierta del RTS. }
        PRIMITIVE ;


FUNCTION FechaHoraSist : FechaHora ;
{ Retorna Fecha y Hora del sistema en el momento de su llamada. }
        PRIMITIVE ;


FUNCTION TimeInHours ( Time : FechaHora ) : TipoHora
        PRIMITIVE ;
```

```
FUNCTION EsMomentoDeAbrir ( Nombre_MTA : nombre_mta ) : BOOLEAN ;
{ Retorna TRUE si el MTA remoto esta habilitado para abrir en el momento
  de la llamada. }
VAR
   NumeroAsociacion : INTEGER ;
   ModoDialogo : TipoModoDialogo ;
   ResponsabilidadAbrir : BOOLEAN ;
   HoraInicial : TipoHora ;
   HoraFinal : TipoHora ;
   HoraActual : TipoHora ;
BEGIN
   IF HayAcuerdoBilateral (Nombre_MTA, NumeroAsociacion, ModoDialogo,
                           ResponsabilidadAbrir, HoraInicial, HoraFinal)
   THEN
     BEGIN
        HoraActual := TimeInHours (FechaHoraSist) ;
        IF HoraFinal > HoraInicial THEN
           EsMomentoDeAbrir := ((HoraActual > HoraInicial) AND
                                (HoraActual < HoraFinal))
        ELSE
           EsMomentoDeAbrir := ((HoraActual > HoraInicial) OR
                                (HoraActual < HoraFinal))
     END
   ELSE
     EsMomentoDeAbrir := FALSE
END ;



FUNCTION PeriodoTiempoAsSinAbrir ( K : Index ;
                                   Map : TipMapaAsociaciones )
                                      : INTEGER ;
{ Retorna el periodo de tiempo durante el cual la associacion permanente
  permanecera sin abrir. De acuerdo al numero de tentativas de apertura. }
CONST
   TiempoLimiteEspera = ANY INTEGER ;
VAR
   v : INTEGER
BEGIN
   CASE Map[K].NumberOpenTries OF
        1 : v := TiempoLimiteEspera ;
        2 : v := 1.5 * TiempoLimiteEspera ;
        3 : v := 2 * TiempoLimiteEspera ;
        OTHERWISE : v := 2.5 * TiempoLimiteEspera
   END;
   PeriodoTiempoAsSinAbrir := v
END ;
```

```
FUNCTION TiempoTransfRTS ( MPDU : TipoMPDU ) : INTEGER ;
{ Retorna el periodo de tiempo dentro del cual el RTS debe transferir
  satisfactoriamente la MPDU. Se sugiere que sea igual a la longitud de
  la MPDU. }
        PRIMITIVE ;



FUNCTION PrioridadMPDU ( MPDU : TipoMPDU ) : ClasePrdad ;
{ Retorna la prioridad de la MPDU dada. }
        PRIMITIVE ;



FUNCTION TamanoMPDU ( MPDU : TipoMPDU ) : INTEGER ;
{ Retorna la longitud en Bytes del contenido de la MPDU dada. }
        PRIMITIVE ;



PROCEDURE EncodeDatosUsuario ( NomEsteMTA : nombre_mta ;
                               PasswEsteMTA : TipoPassw ;
                               VAR DatosUsuario : DatosU_RTS ) ;
{ Encodifica DatosUsuario para enviar la primitiva OPEN.Request. }
        PRIMITIVE ;



FUNCTION DecodeDatosUsuario ( VAR Nombre_MTA : nombre_mta ;
                              VAR Password : TipoPassw ;
                              DatosUsuario : DatosU_RTS)
                            : BOOLEAN ;
{ Si existe DatosUsuario, retorna TRUE y lo decodifica para validarlo. }
        PRIMITIVE ;
```

```
PROCEDURE InicialiceColasRTS ;
{ Inicializa las estructuras de datos donde se van a almacenar las MPDUs
  que esperan transferencia al RTS. Hay una estructura de datos para cada
  una de las 3 prioridades. }
        PRIMITIVE ;


PROCEDURE CrearMPDU (TipoMpdu : Clase_mpdu ;
                     Umpdu : TipoUmpdu ;
                     DRmpdu : TipoDRmpdu ;
                     Pmpdu : TipoPmpdu ;
                     VAR MPDU : TipoMPDU ) ;
{ Genera la MPDU con base a los parametros dados.
        PRIMITIVE ;


PROCEDURE EncolarParaRTS ( Prioridad : ClasePrdad ;
                           HoraDeposito : FechaHora ;
                           Nombre_MTA : nombre_mta ;
                           MPDU : TipoMPDU ;
                           MessageReference : RefMpdu ) ;
{ Coloca la MPDU ,el MTA receptor, la hora de deposito, y la referencia
  de mensaje al final de la estructura de datos apropiada, de acuerdo a la
  prioridad. }
        PRIMITIVE ;


PROCEDURE SacarDeColaRTS ( Nombre_MTA : nombre_mta ;
                           VAR HoraDeposito : FechaHora ;
                           VAR MPDU : TipoMPDU ;
                           VAR MessageReference : RefMpdu ) ;
{ Dado un MTA, retorna la MPDU, su hora de deposito y referencia de mensaje
  desde la cola de mayor prioridad que almacena MPDUs para transferencia
  al RTS. }
        PRIMITIVE ;
```

```
FUNCTION HayMPDUsEsperando ( Nombre_MTA : nombre_mta ) : BOOLEAN ;
{ Indica si hay MPDUs para transferir a un MTA dado.
  Retorna TRUE si hay alguna MPDU esperando transferencia al MTA en la
  estructura de datos de mensajes a enviar al RTS. En caso contrario
  retorna FALSE.}
        PRIMITIVE ;


FUNCTION HayMPDUsSinAsocAbierta ( Map : TipMapaAsociaciones ;
                                        VAR Nombre_MTA : { optional } nombre_mta
                                        : BOOLEAN ;
{ Retorna TRUE si hay alguna MPDU esperando transferencia al MTA sin
  asociacion Opening y se tiene la responsabilidad de abrir y no esta en la
  cola Opening. Adicionalmente, selecciona el MTA con la responsabilidad para
  abrir y que no este en la cola Opening y que tenga la MPDU con la hora de
  deposito mas antigua. }
        PRIMITIVE ;


FUNCTION PrdadMaxEsperando ( Nombre_MTA : nombre_mta ) : ClasePrdad ;
{ Retorna la prioridad mayor entre las MPDUs que estan esperando
  transferencia al MTA. }
        PRIMITIVE ;


FUNCTION CuantasMPDUesperando ( Nombre_MTA : nombre_mta ) : INTEGER ;
{ Retorna cuantas MPDUs hay esperando transferencia al MTA remoto. }
        PRIMITIVE ;
```

```
FUNCTION HayAsociacion ( Nombre_MTA : nombre_mta ;
                         Map : TipMapaAsociaciones ) : BOOLEAN ;
{ Retorna TRUE si existe alguna asociacion abierta con el MTA remoto. }
VAR
   Sw : BOOLEAN ;
   K : Index ;
BEGIN
   Sw := FALSE ;
   K := 1 ;
   WHILE ((K <= MaxNPuntosInteracRTS) AND NOT Sw) DO
      BEGIN
         IF ((Map[K].Name = Nombre_MTA) AND
             ((Map[K].AssociationState <> free)  AND
              (Map[K].Association = temporal)) AND
             (Map[K].AssociationState <> waitingcall) AND
             (Map[K].AssociationState <> closing)) THEN
            Sw := TRUE
         ELSE
            K := K + 1
       END ;
   HayAsociacion := Sw
END ;


FUNCTION HayAsociacionFREE ( Map : TipMapaAsociaciones ) : BOOLEAN ;
{ Retorna TRUE si existe alguna asociacion abierta en estado Free. }
VAR
   Sw : BOOLEAN ;
   K : Index ;
BEGIN
   Sw := FALSE ;
   K := 1 ;
   WHILE ((K <= MaxNPuntosInteracRTS) AND NOT Sw) DO
      BEGIN
         IF ((Map[K].AssociationState = free)
              AND
              (Map[K].Association = temporal)) THEN
            Sw := TRUE
         ELSE
            K := K + 1
      END ;
   HayAsociacionFREE := Sw
END ;
```

```
FUNCTION HayAsociacionPermanenteFREE ( Nombre_MTA : nombre_mta ;
                                       Map : TipMapaAsociaciones ) : BOOLEAN ;
{ Retorna TRUE si existe alguna asociacion permanente en estado Free
  con el MTA remoto. }
VAR
   Sw : BOOLEAN ;
   K : Index ;
BEGIN
   Sw := FALSE ;
   K := 1 ;
   WHILE ((K <= MaxNPuntosInteracRTS) AND NOT Sw) DO
      BEGIN
         IF ((Map[K].Name = Nombre_MTA)
             AND
             (Map[K].AssociationState = free)
             AND
             (Map[K].Association = permanent)) THEN
            Sw := TRUE
         ELSE
            K := K + 1
      END ;
   HayAsociacionPermanenteFREE := Sw
END ;


FUNCTION HayAsociacionOpening ( DatosUsuario : DatosU_RTS ;
                                K : Index ;
                                Map : TipMapaAsociaciones ) : BOOLEAN ;
{ Retorna TRUE si la asociacion K esta en estado 'opening' con el MTA remoto
  transferido en DatosUsuario. }
VAR
   Nombre_MTA : nombre_mta ;
   Password : TipoPassw ;
BEGIN
   IF DecodeDatosUsuario (Nombre_MTA, Password, DatosUsuario) THEN
      HayAsociacionOpening:= (Map[K].Name = Nombre_MTA) AND
                             (Map[K].Password = Password) AND
                                (Map[K].AssociationState = opening)
   ELSE
      HayAsociacionOpening := (Map[K].Name = Nombre_MTA) AND
                               (Map[K].AssociationState = opening)
END ;
```

```
FUNCTION HayAsocPRERECEIVER ( Nombre_MTA : nombre_mta ;
                              Map : TipMapaAsociaciones )
                            : BOOLEAN ;
{ Retorna TRUE si existe alguna asociacion abierta en estado Prereceiver
  con el MTA remoto con modo de dialogo Bidireccional alternado. }
VAR
   Sw : BOOLEAN ;
   K : Index ;
BEGIN
   Sw := FALSE ;
   K := 1 ;
   WHILE ((K <= MaxNPuntosInteracRTS) AND NOT Sw) DO
      BEGIN
         IF ((Map[K].Name = Nombre_MTA)
             AND
             (Map[K].AssociationState = prereceiver)
             AND
             (Map[K].Dialogue = BidirecAlter)) THEN
            Sw := TRUE
         ELSE
            K := K + 1
      END ;
   HayAsocPRERECEIVER := Sw
END ;


FUNCTION HayEstadoPRERECEIVER ( Map : TipMapaAsociaciones ) : BOOLEAN ;
{ Retorna TRUE si hay alguna asociacion abierta en estado Prereceiver. }
VAR
   Sw : BOOLEAN ;
   K : Index ;
BEGIN
   Sw := FALSE ;
   K := 1 ;
   WHILE ((K <= MaxNPuntosInteracRTS) AND NOT Sw) DO
      BEGIN
         IF Map[K].AssociationState = prereceiver THEN
            Sw := TRUE
         ELSE
            K := K + 1
      END ;
   HayEstadoPRERECEIVER := Sw
END ;
```

```
FUNCTION HayAsociacionRECEIVER ( Nombre_MTA : nombre_mta ;
                                 Map : TipMapaAsociaciones ) : BOOLEAN ;
{ Retorna TRUE si hay alguna asociacion abierta en estado Receiver con el
  MTA remoto con modo de dialogo Bidireccional Alternado. }
VAR
   Sw : BOOLEAN ;
   K : Index ;
BEGIN
   Sw := FALSE ;
   K := 1 ;
   WHILE ((K <= MaxNPuntosInteracRTS) AND NOT Sw) DO
      BEGIN
         IF ((Map[K].Name = Nombre_MTA)
             AND
             (Map[K].AssociationState = receiver)
             AND
             (Map[K].Dialogue = BidirecAlter)) THEN
            Sw := TRUE
         ELSE
            K := K + 1
      END ;
   HayAsociacionRECEIVER := Sw
END ;



FUNCTION HayAsociacionREPOSE ( Nombre_MTA : nombre_mta ;
                               Map : TipMapaAsociaciones )
                             : BOOLEAN ;
{ Retorna TRUE si existe alguna asociacion abierta en estado Repose con
  el MTA remoto. }
VAR
   K : Index ;
   Sw : BOOLEAN ;
BEGIN
   K := 1 ;
   Sw := FALSE ;
   WHILE ((K <= MaxNPuntosInteracRTS) AND NOT Sw) DO
      BEGIN
         IF ((Map[K].Name = Nombre_MTA)
             AND
             (Map[K].AssociationState = repose)) THEN
            Sw := TRUE
         ELSE
            K := K + 1
      END ;
   HayAsociacionREPOSE := Sw
END ;
```

```
FUNCTION HayEstadoREPOSE ( Map : TipMapaAsociaciones ) : BOOLEAN ;
{ Retorna TRUE si hay alguna asociacion abierta en estado 'repose'. }
VAR
   Sw : BOOLEAN ;
   K : Index ;
BEGIN
   Sw := FALSE ;
   K := 1 ;
   WHILE ((K <= MaxNPuntosInteracRTS) AND NOT Sw) DO
      BEGIN
         IF (Map[K].AssociationState = repose) THEN
            Sw := TRUE
         ELSE
            K := K + 1
      END ;
   HayEstadoREPOSE := Sw
END ;
```

```
FUNCTION HayAsociacionWAITINGCALL ( Map : TipMapaAsociaciones ) : BOOLEAN ;
{ Retorna TRUE  si existe alguna asociacion en estado Waitingcall. }
VAR
   Sw : BOOLEAN ;
   K : Index ;
BEGIN
   Sw := FALSE ;
   K := 1 ;
   WHILE ((K <= MaxNPuntosInteracRTS) AND NOT Sw) DO
      BEGIN
         IF (Map[K].AssociationState = waitingcall) THEN
            Sw := TRUE
         ELSE
            K := K + 1
      END ;
   HayAsociacionWAITINGCALL := Sw
END ;
```

```
FUNCTION NumeroAsociaciones ( Nombre_MTA : nombre_mta ;
                              Map : TipMapaAsociaciones ) : INTEGER ;
{ Retorna la cantidad de asociaciones temporales abiertas con el MTA
  remoto. }
VAR
   K : Index ;
   I : INTEGER ;
BEGIN
   I := 0 ;
   K := 1 ;
   WHILE (K <= MaxNPuntosInteracRTS) DO
      BEGIN
         K := K + 1 ;
         IF ((Map[K].Name = Nombre_MTA) AND (Map[K].Association = temporal)) THEN
            I := I + 1
      END ;
   NumeroAsociaciones := I
END ;



FUNCTION SeleccioneAsocParaCerrar ( Map : TipMapaAsociaciones ) : Index ;
{ Selecciona la asociacion abierta en estado Repose sin estado permanente
  que tenga el mas antiguo tiempo de deposito del ultimo mensaje al RTS. }
VAR
   K : Index ;
   LastTime : FechaHora ;
BEGIN
   K := 1 ;
   LastTime := 0 ;
   WHILE (K <= MaxNPuntosInteracRTS) DO
      BEGIN
         IF ((Map[K].AssociationState = repose) AND
             (Map[K].SubmissionLastTime < LastTime)) THEN
         BEGIN
            SeleccioneAsocParaCerrar := K ;
            LastTime := Map[K].SubmissionLastTime
         END ;
         K := K + 1
      END
END ;
```

```
FUNCTION SeleccioneAsociacionFREE ( Map : TipMapaAsociaciones ) : Index ;
{ Retorna una asociacion en estado Free para abrir. }
VAR
   Sw : BOOLEAN ;
   K : Index ;
BEGIN
   Sw := TRUE ;
   K := 1 ;
   WHILE ((K <= MaxNPuntosInteracRTS) AND Sw) DO
      BEGIN
         IF ((Map[K].AssociationState = free) AND
             (Map[K].Association = temporal)) THEN
            BEGIN
               Sw := FALSE ;
               SeleccioneAsociacionFREE := K
            END ;
         K := K + 1
      END
END ;



FUNCTION SeleccioneAsocRECEIVER ( Nombre_MTA : nombre_mta ;
                                  Map : TipMapaAsociaciones ) : Index ;
{ Selecciona la asociacion abierta en estado Receiver con el MTA remoto que
  tenga la menor prioridad solicitada. }
VAR
   K : Index ;
   Prioridad : ClasePrdad ;
BEGIN
   K := 1 ;
   Prioridad := NoUrgente ;
   WHILE (K <= MaxNPuntosInteracRTS) DO
      BEGIN
         IF ((Map[K].Name = Nombre_MTA) AND
             (Map[K].AssociationState = Receiver) AND
             (Map[K].Dialogue = BidirecAlter) AND
             (Map[K].RequestedPriority > Priorit)) THEN
         BEGIN
            SeleccioneAsocRECEIVER := K ;
            Prioridad := Map[K].RequestedPriority
         END ;
         K := K + 1 ;
      END
END ;
```

```
FUNCTION SeleccioneAsocREPOSE ( Nombre_MTA : nombre_mta ;
                                Map : TipMapaAsociaciones ) : Index ;
{ Selecciona la asociacion abierta en estado Repose con el MTA remoto que
  tenga el mas antiguo tiempo de ultimo deposito. }
VAR
   K : Index ;
   LastTime : FechaHora ;
BEGIN
   K := K + 1 ;
   LastTime := 99999999 ;
   WHILE (K <= MaxNPuntosInteracRTS) DO
      BEGIN
         IF ((Map[K].Name = Nombre_MTA) AND
             (Map[K].AssociationState = Repose) AND
             (Map[K].SubmissionLastTime < LastTime)) THEN
         BEGIN
            SeleccioneAsocREPOSE := K ;
            LastTime := Map[K].SubmissionLastTime
         END ;
         K := K + 1
      END
END ;



PROCEDURE ActualiceMapaAs ( K : Index ;
                            Nombre_MTA : nombre_mta ;
                            PasswordNuevo : TipoPassw ;
                            ModoDialogoNuevo : TipoModoDialogo ;
                            DireccSesion : TipoDireccSesion ;
                            ResponsabilidadNueva : BOOLEAN ;
                            VAR Map : TipMapaAsociaciones ) ;
{ Actualiza el mapa de asociaciones. }
BEGIN
   WITH Map[K] DO
          BEGIN
             Dialogue := ModoDialogoNuevo ;
             DidIopen := ResponsabilidadNueva ;
             Name := Nombre_MTA ;
             Password := PasswordNuevo ;
             Address := DireccSesion
          END
END ;
```

```
PROCEDURE EnviarCloseReq ( K : Index ;
                           Map : TipMapaAsociaciones ) ;
{ Envia  la primitiva CLOSE.Request y actualiza el mapa de asociaciones. }
BEGIN
   OUTPUT D.CLOSE.Request (K) ;
   Map[K].AssociationState := closing
END ;


PROCEDURE EnvieUSER_Credit ( K : Index ;
                             VAR Map : TipMapaAsociaciones ) ;
{ Envia la primitiva USER_Credit al RTS para la asociacion k y coloca el
  estado de la asociacion a Receiver. }
BEGIN
    OUTPUT D.USER_Credit (K) ;
    Map[K].AssociationState := receiver
END ;
```

```
PROCEDURE EnviarOpenReq ( K : Index ;
                          VAR Map : TipMapaAsociaciones ;
                          DireccRespondedor : TipoDireccSesion ;
                          ModoDialogo : TipoModoDialogo ;
                          Nombre_MTA : nombre_mta ;
                          NomEsteMTA : nombre_mta ;
                          PasswEsteMTA : TipoPassw ) ;
{ Envia  la primitive OPEN.Request al RTS para la asociacion K y actualiza
  el mapa de asociaciones. }
VAR
   TurnoInicial : TTurnoInicial ;
   DatosUsuario : { optional } DatosU_RTS ;
BEGIN
   IF ((ModoDialogo = Monologo) OR
       HayMPDUsEsperando (Nombre_MTA)) THEN
      BEGIN
         Map[K].HaveIturn := TRUE ;
         TurnoInicial := Iniciador
      END
   ELSE
      BEGIN
         Map[K].HaveIturn := FALSE ;
         TurnoInicial := Respondedor
      END ;
   EncodeDatosUsuario (NomEsteMTA, PasswEsteMTA, DatosUsuario) ;
   OUTPUT D.OPEN.Request (K, DireccRespondedor, ModoDialogo,
                          TurnoInicial, P1, DatosUsuario );
   Map[K].AssociationState := opening
END ;
```

```
PROCEDURE EnviarOpenResp ( K : Index ;
                           VAR Map : TipMapaAsociaciones ;
                           Nombre_MTA : nombre_mta ;
                           DirIniciador : TipoDireccSesion ;
                           ModoDialogo : TipoModoDialogo ;
                           TurnoInicial : TTurnoInicial ;
                           Disposicion : TipoDisposicion ;
                           MotivoRechazo : { optional } RazonRechazoGA-GA ;
                           NomEsteMTA : nombre_mta ;
                           PasswEsteMTA : TipoPassw ;
                           DireccRespondedor : TipoDireccSesion ) ;
{ Envia la primitiva OPEN.Response al RTS para la asociacion K y actualiza
  el mapa de asociaciones. }
VAR
   DireccSesion : TipoDireccSesion ;
   Password : TipoPassw ;
   DatosUsuario : { optional } DatosU_RTS ;
BEGIN
   IF Disposicion = Aceptado THEN
      BEGIN
         EncodeDatosUsuario (NomEsteMTA, PasswEsteMTA, DatosUsuario) ;
         IF HayPropiedadesMTA (Nombre_MTA, Password, DireccSesion) THEN
            ActualiceMapaAs (K, Nombre_MTA, Password, ModoDialogo,
                             DirIniciador, FALSE, Map)
      END
   ELSE
      DatosUsuario := { undefined } NIL ;
   OUTPUT D.OPEN.Response (K, Disposicion, DatosUsuario, MotivoRechazo ) ;
   RegistreOpen (DirIniciador, DireccRespondedor, Disposicion, MotivoRechazo,
                 ModoDialogo, TurnoInicial, P1)
END ;
```

```
PROCEDURE EnviaTransfReqRTS ( MPDU : TipoMPDU ;
                              Message : RefMpdu ;
                              HoraDeposito : FechaHora ;
                              K : Index ;
                              VAR Map : TipMapaAsociaciones ) ;
{ Envia la primitiva TRANSFER.Request al RTS para la asociacion K y
  actualiza el mapa de asociaciones. }
BEGIN
   WITH Map[K] DO
     BEGIN
        AssociationState := busy ;
        MessageReference := Message ;
        SubmissionPriority := PrioridadMPDU (MPDU) ;
        MPDULastContentLength := TamanoMPDU (MPDU) ;
        SubmissionLastTime := FechaHoraSist ;
        MPDULastStoredTime := HoraDeposito
     END ;
   OUTPUT D.TRANSFER.Request (K, MPDU, TiempoTransfRTS (MPDU))
END ;



PROCEDURE EnviaTURN_GIVEreq ( K : Index ;
                              Var Map : TipMapaAsociaciones ;
                              DirIniciador : TipoDireccSesion ) ;
{ Envia la primitiva TURN_GIVE.Request al RTS para la asociacion K actualiza el
  mapa de asociaciones .}
BEGIN
   Map[K].AssociationState := prereceiver ;
   Map[K].HaveITurn := FALSE ;
   Map[K].RequestedTurn := FALSE ;
   Map[K].RequestedPriority := NoUrgente ;
   OUTPUT D.TURN_GIVE.Request (K) ;
   RegistreTurnGive (DirIniciador, Map[K].Address)
END ;



PROCEDURE EnviarTurnPleaseReq ( K : Index ;
                                Map : TipMapaAsociaciones ;
                                Prioridad : ClasePrdad ;
                                DirIniciador : TipoDireccSesion ) ;
{ Envia la primitiva TURN_PLEASE.Request al RTS para la asociacion K .}
BEGIN
   OUTPUT D.TURN_PLEASE.Request (K,Prioridad) ;
   RegistreTurnPlease (DirIniciador, Map[K].Address, Prioridad)
END ;
```

```
PROCEDURE EnviarWAITING_Call ( K : Index ;
                               VAR Map : TipMapaAsociaciones ) ;
{ Envia  la primitiva WAITING_CALL al RTS para la asociacion K y actualiza
  el mapa de asociaciones. }
BEGIN
   OUTPUT D.WAITING_Call (K) ;
   Map[K].AssociationState := waitingcall
END ;



PROCEDURE CerrarAsociacion ( VAR Map : TipMapaAsociaciones ) ;
{ Selecciona una asociacion en estado Repose para cerrarla. }
VAR
   K : Index ;
BEGIN
   K := SeleccioneAsocParaCerrar (Map) ;
   EnviarCloseReq (K, Map) ;
   OUTPUT E.TIMER_Stop (K)
END ;



PROCEDURE AsociacionREPOSEenviaMPDU ( Map : TipMapaAsociaciones ;
                                      MPDU : TipoMPDU ;
                                      Nombre_MTA : nombre_mta ;
                                      MessageReference : RefMpdu ) ;
{ Selecciona una asociacion en estado Repose enviando la MPDU al RTS. }
VAR
   K : Index ;
BEGIN
   K := SeleccioneAsocREPOSE (Nombre_MTA, Map) ;
   OUTPUT E.TIMER_Stop (K) ;
   EnviaTransfReqRTS (MPDU, MessageReference, FechaHoraSist, K, Map)
END ;
```

```
PROCEDURE EnviaMPDUaMD ( K : Index ;
                         Map : TipMapaAsociaciones ;
                         Razon : RazonRechazoGA-DM ) ;
{ Retira la MPDU de la estructura de datos donde se almacenan las MPDUs
  que esperan transferencia al RTS y envia la MPDU al Despachador de
  Mensajes (MD). }
VAR
   HoraDeposito : FechaHora ;
   MPDU : TipoMPDU ;
   MessageReference : RefMpdu ;
BEGIN
   SacarDeColaRTS (Map[K].Name, HoraDeposito, MPDU, MessageReference) ;
   OUTPUT C.TRANSFER.Confirmation ( Rechazado, MessageReference,
                                    Razon )
END ;


PROCEDURE DepositeMPDUenRTS (K : Index ;
                             Map : TipMapaAsociaciones) ;
{ Remueve la MPDU de la estructura de datos donde se almacenan las MPDUs
  que esperan transferencia al RTS y se invoca al procedimiento que
  deposita la MPDU en el RTS. }
VAR
   HoraDeposito : FechaHora ;
   MPDU : TipoMPDU ;
   MessageReference : RefMpdu ;
BEGIN
    SacarDeColaRTS (Map[K].Name, HoraDeposito, MPDU, MessageReference)
    EnviaTransfReqRTS (MPDU, MessageReference, HoraDeposito, K, Map)
END ;


PROCEDURE PrepareWaitingCall ( VAR Map : TipMapaAsociaciones ) ;
{ Selecciona una asociacion que este en estado Free y envia la primitiva
  WAITING_CALL al RTS .}
VAR
   N : Index ;
BEGIN
   N := SeleccioneAsociacionFREE (Map) ;
   EnviarWAITING_Call (N, Map)
END ;
```

```
PROCEDURE InicialiceAsociacion ( K : Index ;
                                 VAR Map : TipMapaAsociaciones ) ;
{ Inicialice la asociacion K. }
BEGIN
   WITH Map[K] DO
        BEGIN
           AssociationState := free ;
           Association := temporal ;
           RequestedTurn := FALSE ;
           RequestedPriority := NoUrgente
        END
END ;


PROCEDURE InicialiceAsociaciones ( Map : TipMapaAsociaciones ) ;
{ Inicializa todas las asociaciones. }
VAR
   K : Index ;
BEGIN
   K := 1 ;
   WHILE (K <= MaxNPuntosInteracRTS) DO
      BEGIN
         InicialiceAsociacion (K, Map) ;
         K := K + 1
      END
END ;
```

```
FUNCTION AsociacionValida ( DatosUsuario : DatosU_RTS ;
                            ModoDialogoNuevo : TipoModoDialogo ;
                            Map : TipMapaAsociaciones ) : BOOLEAN ;
{ Retorna TRUE si es valido el MTA reomoto que solicita el establecimiento
  de una asociacion. }
VAR
   Nombre_MTA : nombre_mta ;
   PasswordNuevo, Password : TipoPassw ;
   DireccSesion: TipoDireccSesion ;
   NumeroAsociacion : INTEGER ;
   ModoDialogo : TipoModoDialogo ;
   ResponsabilidadAbrir : BOOLEAN ;
   HoraInicial : TipoHora ;
   HoraFinal : TipoHora ;
   Sw : BOOLEAN ;
BEGIN
   IF HayPropiedadesMTA (Nombre_MTA, Password, DireccSesion) THEN
      BEGIN
         IF HayAcuerdoBilateral (Nombre_MTA, NumeroAsociacion,
                                 ModoDialogo, ResponsabilidadAbrir,
                                 HoraInicial, HoraFinal) THEN
            IF (NumeroAsociacion = NumeroAsociaciones (Nombre_MTA, Map)) THEN
               Sw := FALSE
            ELSE
               Sw := TRUE
         ELSE
            Sw := TRUE
      END
   ELSE
      Sw := FALSE ;
   IF (Sw AND DecodeDatosUsuario (Nombre_MTA, PasswordNuevo, DatosUsuario)) THEN
      IF PasswordNuevo <> Password THEN
         Sw := FALSE ;
   AsociacionValida := Sw
END ;
```

```
FUNCTION AverigueMotivoRechazo ( DatosUsuario : DatosU_RTS ;
                                 ModoDialogoNuevo : TipoModoDialogo ;
                                 Map : TipMapaAsociaciones )
                               : RazonRechazoGA-GA ;
{ Retorna el motivo de rechazo por el cual el establecimiento de una
  asociacion no es valido .}
VAR
   MotivoRechazo : RazonRechazoGA-GA ;
   Nombre_MTA : nombre_mta ;
   PasswordNuevo, Password : TipoPassw ;
   DireccSesion: TipoDireccSesion ;
   NumeroAsociacion : INTEGER ;
   ModoDialogo : TipoModoDialogo ;
   ResponsabilidadAbrir : BOOLEAN ;
   HoraInicial : TipoHora ;
   HoraFinal : TipoHora ;
BEGIN
   IF HayPropiedadesMTA (Nombre_MTA, Password, DireccSesion) THEN
      BEGIN
         IF HayAcuerdoBilateral (Nombre_MTA, NumeroAsociacion,
                                 ModoDialogo, ResponsabilidadAbrir,
                                 HoraInicial, HoraFinal) THEN
            IF (NumeroAsociacion = NumeroAsociaciones (Nombre_MTA, Map)) THEN
               MotivoRechazo := AMtoAMBusy
      END
   ELSE
      MotivoRechazo := AMtoAMAuthenticationFailure ;
   IF ((MotivoRechazo = AMtoAMBusy) AND
       DecodeDatosUsuario (Nombre_MTA,PasswordNuevo, DatosUsuario)) THEN
      IF PasswordNuevo <> Password THEN
         MotivoRechazo := AMtoAMAuthenticationFailure ;
   AverigueMotivoRechazo := MotivoRechazo
END ;



FUNCTION DecisionAbrirAsociacion ( VAR Nombre_MTA : nombre_mta ;
                                       Map : TipMapaAsociaciones )
                                     : BOOLEAN ;
{ Decide de acuerdo a las estructuras de datos de las MPDUs si abre una
  nueva asociacion. Si decide abrir (osea, si retorna TRUE), retorna
  tambien el nombre del MTA respondedor. }
BEGIN
   IF HayMPDUsSinAsocAbierta (Map, Nombre_MTA) THEN
      DecisionAbrirAsociacion := TRUE
   ELSE
      DecisionAbrirAsociacion := FALSE
END ;
```

```
FUNCTION DecideAbrirNuevaAsociacion ( NumMPDUsEsperando : INTEGER ;
                                      NumAsociacionesAbiertas : INTEGER ;
                                      PrioridadMayor : ClasePrdad )
                                          : BOOLEAN ;
{ Decide si abre una nueva asociacion con base al numero de MPDUs esperando
  transferencia, al numero de asociaciones Opening y a la prioridad mas
  grande de las MPDUs que esperan transferencia. }
        PRIMITIVE ;


FUNCTION AbrirAsociacionNueva ( Nombre_MTA : nombre_mta ;
                                Map : TipMapaAsociaciones ) : BOOLEAN ;
{ Retorna TRUE si no hay ninguna asociacion permanente en estado Free con
  este MTA y hay alguna asociacion temporal Free y hay MPDUs esperando
  transferencia al MTA y este MTA no esta esperando abrir asociaciones y
  se tiene la responsabilidad de abrir asociaciones y el numero de
  asociaciones acordado es mayor que el numero de asociaciones a abrir
  (opening) con este MTA y se decide abrir una nueva aunque exista ya una
  asociacion abierta con este MTA. }
VAR
   NumeroAsociacion : INTEGER ;
   ModoDialogo : TipoModoDialogo ;
   Responsabilidad : BOOLEAN ;
   HoraInicial : TipoHora ;
BEGIN
    IF HayAsociacionPermanenteFREE (Nombre_MTA, Map) THEN
       AbrirAsociacionNueva := FALSE
    ELSE
       IF (HayAsociacionFREE (Map)
           AND
           HayMPDUsEsperando (Nombre_MTA)
           AND
           NOT HayMTAesperandoAbrir (Nombre_MTA)
           AND
           TieneResponsabilidadAbrir (Nombre_MTA)
           AND
           EsMomentoDeAbrir (Nombre_MTA)) THEN
          BEGIN
             IF HayAcuerdoBilateral (Nombre_MTA, NumeroAsociacion,
                                     ModoDialogo, Responsability,
                                     HoraInicial) THEN
                IF NumeroAsociaciones (Nombre_MTA, Map) = NumeroAsociacion THEN
                   AbrirAsociacionNueva := FALSE
                ELSE
                   IF DecideAbrirNuevaAsociacion (CuantasMPDUesperando (Nombre_MTA),
                                                  NumeroAsociaciones (Nombre_MTA, Map),
                                                  PrdadMaxEsperando (Nombre_MTA)) THEN
                      AbrirAsociacionNueva := TRUE
                   ELSE
                      AbrirAsociacionNueva := FALSE
          END
       ELSE
          AbrirAsociacionNueva := FALSE
END ;
```

```
FUNCTION IntentarAbrirAsNuevamente ( K : Index ;
                                     VAR Map : TipMapaAsociaciones ;
                                     MotivoRechazo : RazonRechazoGA-GA )
                                   : BOOLEAN ;
{ Retorna TRUE si se decide intentar abrir nuevamente la asociacion
  luego de recibir una OPEN.Confirmation negativa. }
VAR
  IntentarNuevamente : BOOLEAN ;
BEGIN
   CASE MotivoRechazo OF
      AMtoAMAuthenticationFailure :
                              IntentarNuevamente := FALSE ;
      AMtoAMBusy :
                  IntentarNuevamente := TRUE ;
      AMtoAMUnacceptableModoDialogo :
                                 IntentarNuevamente := FALSE ;
   END ;
   IF (IntentarNuevamente AND (Map[K].Association = permanent)) THEN
      IF Map[K].NumberOpenTries = NumMaxIntentosOpen THEN
         IntentarNuevamente := FALSE ;
   IntentarAbrirAsNuevamente := IntentarNuevamente
END ;



FUNCTION IndiqueRazonMD ( MotivoRechazo : RazonRechazoGA-GA )
                        : RazonRechazoGA-DM ;
{ Retorna el motivo que origino la decision de rechazar la comunicacion entre
  el Gestor de Asociaciones y el Despachador de Mensajes. }
BEGIN
   CASE MotivoRechazo OF
      AMtoAMAuthenticationFailure :
                  IndiqueRazonMD := AMtoMDAuthenticationFailure ;
      AMtoAMBusy :
                  IndiqueRazonMD := AMtoMDBusy ;
      AMtoAMUnacceptableModoDialogo :
                  IndiqueRazonMD := AMtoMDUnacceptableProperties
   END
END ;
```

```
PROCEDURE AbrirAsociacion ( Nombre_MTA : nombre_mta ;
                            K : Index ;
                            VAR Map : TipMapaAsociaciones ;
                            NomEsteMTA : nombre_mta ;
                            PasswEsteMTA : TipoPassw ) ;
{ Solicita el establecimiento de la asociacion K con el MTA remoto. }
VAR
   Password : TipoPassw ;
   DireccSesion : TipoDireccSesion ;
   NumeroAsociacion : INTEGER ;
   ModoDialogo : TipoModoDialogo ;
   ResponsabilidadAbrir : BOOLEAN ;
   HoraInicial : TipoHora ;
   HoraFinal : TipoHora ;
   HoraDeposito : FechaHora ;
   MessageReference : RefMpdu ;
   MPDU : TipoMPDU ;
BEGIN
   IF HayPropiedadesMTA (Nombre_MTA, Password, DireccSesion) THEN
      BEGIN
         IF HayAcuerdoBilateral (Nombre_MTA, NumeroAsociacion, ModoDialogo,
                                 ResponsabilidadAbrir, HoraInicial,
                                 HoraFinal) THEN
            BEGIN
               EnviarOpenReq (K, Map, DireccSesion, ModoDialogo, Nombre_MTA,
                              NomEsteMTA, PasswEsteMTA) ;
               ActualiceMapaAs (K, Nombre_MTA, Password, ModoDialogo,
                                DireccSesion, TRUE, Map)
            END
      END
   ELSE
      BEGIN
         WHILE HayMPDUsEsperando (Nombre_MTA) DO
            BEGIN
               SacarDeColaRTS (Nombre_MTA, HoraDeposito, MPDU,
                               MessageReference ) ;
               Indicacion_exito := FALSE ;
               Destinatario := NIL ;
               OUTPUT C.TRANSFER.Confirmation ( Indicacion_exito,
                                                Destinatario,
                                                Messagereference,
                                                AMtoMDMTAUnrecognized )
            END ;
         IF Map[K].Association = temporal THEN
            DespuesDeLiberarAsoc (K, Map, NomEsteMTA, PasswEsteMTA)
      END
END ;
```

```
PROCEDURE AbrirAsociacionPermanente ( VAR Map : TipMapaAsociaciones ;
                                      NomEsteMTA : nombre_mta ;
                                      PasswEsteMTA : TipoPassw ) ;
{ Solicita el establecimiento de asociaciones permanentes con responsabilidad
  para abrir. }
VAR
   Nombre_MTA : nombre_mta ;
   K : Index ;
BEGIN
   K := 1 ;
   WHILE BusqueNomMTApermanent (Nombre_MTA) DO
      IF EsMomentoDeAbrir (Nombre_MTA) THEN
         BEGIN
            Map[K].Association := permanent ;
            AbrirAsociacion (Nombre_MTA, K, Map, NomEsteMTA, PasswEsteMTA) ;
            K:= K + 1
         END
      ELSE
         BEGIN
            Map[K].Association := permanent ;
            Map[K].AssociationState := opening ;
            OUTPUT E.TIMER.Request (K, PeriodoTiempoAsSinAbrir (K, Map) )
         END
END ;


PROCEDURE DespuesDeLiberarAsoc ( K : Index ;
                                 VAR Map : TipMapaAsociaciones ;
                                 NomEsteMTA : nombre_mta ;
                                 PasswEsteMTA : TipoPassw ) ;
{ Ejecuta las acciones indicadas despues de liberar una asociacion. }
BEGIN
   InicialiceAsociacion (K, Map) ;
   IF DecisionAbrirAsociacion (Nombre_MTA, Map)
   THEN
      AbrirAsociacion (Nombre_MTA, K, Map, NomEsteMTA, PasswEsteMTA)
   ELSE
      IF NOT HayAsociacionWAITINGCALL (Map) THEN
         EnviarWAITING_Call (K, Map)
END ;
```

```
INITIALIZE TO EstadoUnico
   BEGIN
      InicialiceRecursos;
      InicialiceColasRTS ;
      InicialiceColaOpening ;
      AtributosEsteMTA (NomEsteMTA, PasswEsteMTA, DirecSesionEsteMTA) ;
      InicialiceAsociaciones (MapaAsociaciones) ;
      AbrirAsociacionPermanente (MapaAsociaciones, NomEsteMTA, PasswEsteMTA) ;
      EnviarWAITING_Call (MaxNPuntosInteracRTS, MapaAsociaciones)
   END ;
```

TRANS

```
WHEN C.TRANSFER.Request
FROM EstadoUnico TO SAME
{ Existen asociaciones establecidas con el MTA remoto y en estado de
  reposo. Se transfiere la MPDU. }
  PROVIDED HayAsociacionREPOSE (Nombre_MTA, MapaAsociaciones)
  BEGIN
     CrearMPDU (MessageReference.TipoMpdu, Umpdu, DRmpdu, Pmpdu, MPDU) ;
     EncolarParaRTS (PrioridadMPDU (MPDU), FechaHoraSist, Nombre_MTA, MPDU,
                     MessageReference) ;
     AsociacionREPOSEenviaMPDU (MapaAsociaciones, MPDU, Nombre_MTA,
                                MessageReference)
  END  ;


{ Hay asociaciones establecidas con el MTA remoto, ninguna en estado de
  reposo pero hay algunas asociaciones temporales en estado libre (free).
  Se decide abrir una nueva asociacion y encolar la MPDU . }
  PROVIDED (NOT HayAsociacionREPOSE (Nombre_MTA, MapaAsociaciones)
            AND
            HayAsociacion (Nombre_MTA, MapaAsociaciones)
            AND
            AbrirAsociacionNueva (Nombre_MTA, MapaAsociaciones))
  BEGIN
     CrearMPDU (MessageReference.TipoMpdu, Umpdu, DRmpdu, Pmpdu, MPDU) ;
     EncolarParaRTS (PrioridadMPDU (MPDU), FechaHoraSist, Nombre_MTA, MPDU,
                     MessageReference) ;
     AbrirAsociacion (Nombre_MTA, SeleccioneAsociacionFREE (MapaAsociaciones),
                      MapaAsociaciones, NomEsteMTA, PasswEsteMTA)
  END ;
```

```
{ Hay asociaciones establecidas con el MTA remoto, pero ninguna en estado de
  reposo y se decide no abrir una nueva asociacion. Se encola la MPDU y se
  solicita el turno si hay algunas asociaciones en estado Receiver o
  Prereceiver . }
  PROVIDED (NOT HayAsociacionREPOSE (Nombre_MTA, MapaAsociaciones)
            AND
            HayAsociacion (Nombre_MTA, MapaAsociaciones)
            AND
            NOT AbrirAsociacionNueva (Nombre_MTA, MapaAsociaciones))
  BEGIN
     CrearMPDU (MessageReference.TipoMpdu, Umpdu, DRmpdu, Pmpdu, MPDU) ;
     EncolarParaRTS (PrioridadMPDU(MPDU), FechaHoraSist, Nombre_MTA, MPDU,
                     MessageReference) ;
     IF HayAsociacionRECEIVER (Nombre_MTA, MapaAsociaciones) THEN
        EnviarTurnPleaseReq (SeleccioneAsocRECEIVER (Nombre_MTA,
                                                     MapaAsociaciones),
                             MapaAsociaciones, PrdadMaxEsperando (Nombre_MTA),
                             DirecSesionEsteMTA)
     ELSE
        IF HayAsocPRERECEIVER (Nombre_MTA, MapaAsociaciones) THEN
           EnviarTurnPleaseReq (SelectPrereceiverAssociation (Nombre_MTA,
                                                              MapaAsociaciones),
                                MapaAsociaciones,
                                PrdadMaxEsperando (Nombre_MTA),
                                DirecSesionEsteMTA)
  END ;
```

```
{ No hay asociaciones establecidas con el MTA remoto ni MPDU esperando ser
  transferidas pero hay algunas asociaciones temporales en estado Free y se
  tiene la responsabilidad de apertura. Se encola la MPDU y se decide abrir
  una asociacion si el MTA remoto no esta en la cola para apertura y si es
  valido el momento (hora) de apertura. }
  PROVIDED (NOT HayAsociacion (Nombre_MTA, MapaAsociaciones)
            AND
            HayAsociacionFREE (MapaAsociaciones)
            AND
            TieneResponsabilidadAbrir (Nombre_MTA)
            AND
            NOT HayMPDUsEsperando (Nombre_MTA))
  BEGIN
     CrearMPDU (MessageReference.TipoMpdu, Umpdu, DRmpdu, Pmpdu, MPDU) ;
     EncolarParaRTS (PrioridadMPDU (MPDU), FechaHoraSist, Nombre_MTA, MPDU,
                     MessageReference) ;
     IF NOT HayMTAesperandoAbrir (Nombre_MTA)
        AND
        EsMomentoDeAbrir (Nombre_MTA)
        THEN
        AbrirAsociacion (Nombre_MTA, SeleccioneAsociacionFREE (MapaAsociaciones),
                         MapaAsociaciones, NomEsteMTA, PasswEsteMTA)
  END ;
```

```
{ No hay asociaciones establecidas con el MTA remoto ni asociaciones
  temporales en estado Free ni MPDUs esperando transferencia, pero hay algunas
  asociaciones en estado Repose y se tiene la responsabilidad para apertura.
  Se encola la MPDU y se decide cerrar una asociacion en estado Repose si el
  MTA remoto no esta en la cola para apertura y si es valido el momento (hora)
  de apertura. }
  PROVIDED (NOT HayAsociacion (Nombre_MTA, MapaAsociaciones)
            AND
            NOT HayAsociacionFREE (MapaAsociaciones)
            AND
            HayEstadoREPOSE (MapaAsociaciones)
            AND
            TieneResponsabilidadAbrir (Nombre_MTA)
            AND
            NOT HayMPDUsEsperando (Nombre_MTA))
  BEGIN
     CrearMPDU (MessageReference.TipoMpdu, Umpdu, DRmpdu, Pmpdu, MPDU) ;
     EncolarParaRTS (PrioridadMPDU (MPDU), FechaHoraSist, Nombre_MTA, MPDU,
                     MessageReference) ;
     IF NOT HayMTAesperandoAbrir (Nombre_MTA)
        AND
        EsMomentoDeAbrir (Nombre_MTA)
     THEN
        CerrarAsociacion (MapaAsociaciones)
  END ;



{ No hay asociaciones establecidas con el MTA remoto ni asociaciones
  temporales en estado Free ni asociaciones en estado Repose ni MPDUs esperando

  esperando transferencia. Se encola la MPDU . }
  PROVIDED (NOT HayAsociacion (Nombre_MTA, MapaAsociaciones)
            AND
            NOT HayAsociacionFREE (MapaAsociaciones)
            AND
            NOT HayEstadoREPOSE (MapaAsociaciones)
            AND
            NOT HayMPDUsEsperando (Nombre_MTA))
  BEGIN
     CrearMPDU (MessageReference.TipoMpdu, Umpdu, DRmpdu, Pmpdu, MPDU) ;
     EncolarParaRTS (PrioridadMPDU (MPDU), FechaHoraSist, Nombre_MTA, MPDU,
                     MessageReference) ;
  END ;
```

```
{ No hay asociaciones establecidas con el MTA remoto pero hay MPDUs esperando
  transferencia a ese MTA. Se encola la MPDU. }
  PROVIDED (NOT HayAsociacion (Nombre_MTA, MapaAsociaciones)
            AND
            HayMPDUsEsperando (Nombre_MTA))
  BEGIN
     CrearMPDU (MessageReference.TipoMpdu, Umpdu, DRmpdu, Pmpdu, MPDU) ;
     EncolarParaRTS (PrioridadMPDU (MPDU), FechaHoraSist, Nombre_MTA, MPDU,
                     MessageReference) ;
  END ;
```

```
WHEN D.OPEN.Indication
FROM EstadoUnico TO SAME
   { No se tiene el turno de transferir. Si la asociacion es valida y hay
     recursos, se da credito (derecho de transferir). Se solicita el turno
     si hay MPDUs esperando transferencia al MTA remoto y se envia la
     primitiva WAITING_Call si hay alguna asociacion temporal en estado Free.
     Si la asociacion no es valida se envia la primitiva WAITING_Call. }
   PROVIDED ((MapaAsociaciones[K].AssociationState = waitingcall)
             AND
             (TurnoInicial = Iniciador))
   BEGIN
      IF AsociacionValida (DatosUsuario, ModoDialogo, MapaAsociaciones) THEN
         BEGIN
            EnviarOpenResp (K, MapaAsociaciones, MapaAsociaciones[K].Name,
                            DirIniciador, ModoDialogo, TurnoInicial,
                            Aceptado, { undefined } NIL, NomEsteMTA,
                            PasswEsteMTA, DirecSesionEsteMTA) ;
            MapaAsociaciones[K].HaveIturn := FALSE ;
            IF ExisteRecursos THEN
               EnvieUSER_Credit (K, MapaAsociaciones)
            ELSE
               MapaAsociaciones[K].AssociationState := prereceiver ;
            IF (HayMPDUsEsperando (MapaAsociaciones[K].Name) AND
                (MapaAsociaciones[K].Dialogue = BidirecAlter)) THEN
               EnviarTurnPleaseReq (K, MapaAsociaciones,
                                    PrdadMaxEsperando (MapaAsociaciones[K].Name),
                                    DirecSesionEsteMTA) ;
            IF HayAsociacionFREE (MapaAsociaciones) THEN
               PrepareWaitingCall (MapaAsociaciones)
         END
      ELSE
        BEGIN
           EnviarOpenResp (K, MapaAsociaciones, MapaAsociaciones[K].Name,
                           DirIniciador, ModoDialogo, TurnoInicial,
                           Rechazado, AverigueMotivoRechazo (DatosUsuario,
                           ModoDialogo, MapaAsociaciones),
                           NomEsteMTA, PasswEsteMTA, DirecSesionEsteMTA) ;
           EnviarWAITING_Call (K, MapaAsociaciones)
        END
   END ;
```

```
{ Se tiene le turno de transferir y hay MPDUs esperando transferencia
  al MTA remoto. Se desencola y se transfiere una MPDU si la asociacion
  es valida y se envia la primitiva WAITING_Call si hay alguna asociacion
  temporal libre.
  Pero si la asociacion no es valida se envia la primitiva WAITING_Call. }
PROVIDED ((MapaAsociaciones[K].AssociationState = waitingcall)
          AND
          (TurnoInicial = Respondedor)
          AND
          HayMPDUsEsperando (MapaAsociaciones[K].Name))
BEGIN
   IF AsociacionValida (DatosUsuario, ModoDialogo, MapaAsociaciones) THEN
      BEGIN
         EnviarOpenResp (K, MapaAsociaciones, MapaAsociaciones[K].Name,
                         DirIniciador, ModoDialogo, TurnoInicial,
                         Aceptado, { undefined } NIL, NomEsteMTA, PasswEsteMTA,
                         DirecSesionEsteMTA) ;
         MapaAsociaciones[K].HaveIturn := TRUE ;
         DepositeMPDUenRTS (K, MapaAsociaciones) ;
         IF HayAsociacionFREE (MapaAsociaciones) THEN
            PrepareWaitingCall (MapaAsociaciones)
      END
   ELSE
      BEGIN
         EnviarOpenResp (K, MapaAsociaciones, MapaAsociaciones[K].Name,
                         DirIniciador, ModoDialogo, TurnoInicial,
                         Rechazado, AverigueMotivoRechazo (DatosUsuario,
                         ModoDialogo, MapaAsociaciones),
                         NomEsteMTA, PasswEsteMTA, DirecSesionEsteMTA) ;
         EnviarWAITING_Call (K, MapaAsociaciones)
      END
END ;
```

```
{ Se tiene el turno de transferir y no hay MPDUs esperando transferencia
  al MTA remoto. Se da el turno y credito si hay recursos. Si hay
  alguna asociacion temporal libre se envia la primitiva WAITING_Call.
  Pero si la asociacion no es valida, envia la primitiva WAITING_Call. }
   PROVIDED ((MapaAsociaciones[K].AssociationState = waitingcall)
             AND
             (TurnoInicial = Respondedor)
             AND
             NOT HayMPDUsEsperando (MapaAsociaciones[K].Name))
   BEGIN
      IF AsociacionValida (DatosUsuario, ModoDialogo, MapaAsociaciones) THEN
         BEGIN
            EnviarOpenResp (K, MapaAsociaciones, MapaAsociaciones[K].Name,
                            DirIniciador, ModoDialogo, TurnoInicial,
                            Aceptado, { undefined } NIL, NomEsteMTA, PasswEsteMTA,
                            DirecSesionEsteMTA) ;
            MapaAsociaciones[K].HaveIturn := FALSE ;
            EnviaTURN_GIVEreq (K, MapaAsociaciones, DirecSesionEsteMTA) ;
            IF ExisteRecursos THEN
               EnvieUSER_Credit (K, MapaAsociaciones) ;
            IF HayAsociacionFREE (MapaAsociaciones) THEN
               PrepareWaitingCall (MapaAsociaciones)
         END
      ELSE
         BEGIN
            EnviarOpenResp (K, MapaAsociaciones, MapaAsociaciones[K].Name,
                            DirIniciador, ModoDialogo, TurnoInicial,
                            Rechazado, AverigueMotivoRechazo (DatosUsuario,
                            ModoDialogo, MapaAsociaciones),
                            NomEsteMTA, PasswEsteMTA, DirecSesionEsteMTA) ;
            EnviarWAITING_Call (K, MapaAsociaciones)
         END
   END ;
```

```
WHEN D.OPEN.Confirmation
FROM EstadoUnico TO SAME
   { Acepta el establecimiento de la asociacion y tiene el turno y hay
     MPDUs esperando transferencia al MTA.
     Desencola y transfiere la MPDU con la prioridad mayor. }
   PROVIDED ((Disposicion = Aceptado)
             AND
             HayAsociacionOpening (DatosUsuario, K, MapaAsociaciones)
             AND
             MapaAsociaciones[K].HaveIturn
             AND
             HayMPDUsEsperando (MapaAsociaciones[K].Name))
   BEGIN
      RegistreOpen (DirecSesionEsteMTA, MapaAsociaciones[K].Address, Disposicion,
                    MotivoRechazo, MapaAsociaciones[K].Dialogue, Iniciador, P1) ;
      DepositeMPDUenRTS (K, MapaAsociaciones)
   END ;



   { Acepta el establecimiento de la asociacion, tiene el turno y el tipo
     de la asociacion es permanente pero no hay MPDUs esperando transferencia
     al MTA. La asociacion se coloca en estado 'repose'. }
   PROVIDED ((Disposicion = Aceptado)
             AND
             HayAsociacionOpening (DatosUsuario, K, MapaAsociaciones)
             AND
             MapaAsociaciones[K].HaveIturn
             AND
             NOT HayMPDUsEsperando (MapaAsociaciones[K].Name)
             AND
             (MapaAsociaciones[K].Association = permanent))
   BEGIN
      RegistreOpen (DirecSesionEsteMTA, MapaAsociaciones[K].Address, Disposicion,
                    MotivoRechazo, MapaAsociaciones[K].Dialogue, Iniciador, P1) ;
      MapaAsociaciones[K].AssociationState := repose
   END ;
```

```
{ Acepta el establecimiento de la asociacion, tiene el turno y el tipo
  de la asociacion es temporal pero no hay MPDUs esperando transferencia
  al MTA. La asociacion se coloca en estado 'repose' y se envia la
  primitiva TIMER.Request. }
PROVIDED ((Disposicion = Aceptado)
          AND
          HayAsociacionOpening (DatosUsuario, K, MapaAsociaciones)
          AND
          MapaAsociaciones[K].HaveIturn
          AND
          NOT HayMPDUsEsperando (MapaAsociaciones[K].Name)
          AND
          (MapaAsociaciones[K].Association = temporal))
BEGIN
   RegistreOpen (DirecSesionEsteMTA, MapaAsociaciones[K].Address, Disposicion,
                 MotivoRechazo, MapaAsociaciones[K].Dialogue, Iniciador, P1) ;
   OUTPUT E.TIMER.Request (K, TimerReposeAssociation (K, MapaAsociaciones)) ;
   MapaAsociaciones[K].AssociationState := repose
END ;



{ Acepta el establecimiento de la asociacion pero no tiene el turno.
  Si hay recursos, concede credito al RTS. }
PROVIDED ((Disposicion = Aceptado)
          AND
          HayAsociacionOpening (DatosUsuario, K, MapaAsociaciones)
          AND
          NOT MapaAsociaciones[K].HaveIturn)
BEGIN
   RegistreOpen (DirecSesionEsteMTA, MapaAsociaciones[K].Address, Disposicion,
                 MotivoRechazo, MapaAsociaciones[K].Dialogue, Respondedor, P1) ;
   IF ExisteRecursos THEN
      EnvieUSER_Credit (K, MapaAsociaciones)
   ELSE
      MapaAsociaciones[K].AssociationState := prereceiver
END ;
```

```
{ El establecimiento de la asociacion no se acepto y el tipo de la
  asociacion es temporal. Libera la asociacion y si va a intentar abrir
  nuevamente la asociacion, encola el nombre de MTA remoto y el
  tiempo de rechazo de apertura .En caso contrario desencola todas las
  MPDUs que esperan transferencia al MTA remoto y envia primitivas
  de excepcion al Despachador de Mensajes. }
PROVIDED ((Disposicion = Rechazado)
          AND
          HayAsociacionOpening (DatosUsuario, K, MapaAsociaciones)
          AND
          (MapaAsociaciones[K].Association = temporal))
BEGIN
   IF MapaAsociaciones[K].HaveIturn THEN
      RegistreOpen (DirecSesionEsteMTA, MapaAsociaciones[K].Address, Disposicion,
                    MotivoRechazo, MapaAsociaciones[K].Dialogue, Iniciador, P1)
   ELSE
      RegistreOpen (DirecSesionEsteMTA, MapaAsociaciones[K].Address, Disposicion,
                    MotivoRechazo, MapaAsociaciones[K].Dialogue, Respondedor, P1)
   IF IntentarAbrirAsNuevamente (K, MapaAsociaciones, MotivoRechazo)
   THEN
      EncolarParaAbrir (MapaAsociaciones[K].Name, FechaHoraSist)
   ELSE
      WHILE HayMPDUsEsperando (MapaAsociaciones[K].Name) DO
          EnviaMPDUaMD (K, MapaAsociaciones,
                        IndiqueRazonMD (MotivoRechazo)) ;
   DespuesDeLiberarAsoc (K, MapaAsociaciones, NomEsteMTA, PasswEsteMTA)
END ;
```

```
{ No se acepta el establecimiento de la asociacion y el tipo de la
  asociacion es permanente. Si no va a intentar abrir nuevamente,
  desencola todas las MPDUs que esperan transferencia al MTA remoto
  y envia primitivas EXCEPTION al Despachador de Mensajes si es posible.
  En caso contrario , encola para el Despachador de Mensajes. }
PROVIDED ((Disposicion = Rechazado)
          AND
          HayAsociacionOpening (DatosUsuario, K, MapaAsociaciones)
          AND
          (MapaAsociaciones[K].Association = permanent))
BEGIN
   IF MapaAsociaciones[K].HaveIturn THEN
      RegistreOpen (DirecSesionEsteMTA, MapaAsociaciones[K].Address, Disposicion,
                    MotivoRechazo, MapaAsociaciones[K].Dialogue, Iniciador, P1)
   ELSE
      RegistreOpen (DirecSesionEsteMTA, MapaAsociaciones[K].Address, Disposicion,
                    MotivoRechazo, MapaAsociaciones[K].Dialogue, Respondedor, P1)
   IF IntentarAbrirAsNuevamente (K, MapaAsociaciones, MotivoRechazo) THEN
      BEGIN
         OUTPUT E.TIMER.Request (K,
                                 PeriodoTiempoAsSinAbrir (K, MapaAsociaciones))
      END
   ELSE
      WHILE HayMPDUsEsperando (MapaAsociaciones[K].Name) DO
         BEGIN
            EnviaMPDUaMD (K, MapaAsociaciones,
                          IndiqueRazonMD (MotivoRechazo)) ;
            MapaAsociaciones[K].AssociationState := free
         END
END ;
```