Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Laboratorio de Comunicación y Redes

Aprenda Token Ring: Sistema Didáctico para la Enseñanza de la Arquitectura de Red Token Ring

Trabajo Especial de Grado
presentado ante la Ilustre
Universidad Central de Venezuela
por las Bachilleres:

Sheyla M. Pirela L.
C.I.: 15.540.798
E-mail: sheypi@gmail.com

Chen Yu Lin
C.I.: 25.263.822
E-mail: skyofstars@gmail.com

para optar al título de Licenciado en Computación

Tutores: Prof. Eric Gamess
Prof. Karima Velásquez

Caracas, Febrero 2009

Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Laboratorio de Comunicación y Redes

ACTA DEL VEREDICTO

Quienes suscriben, Miembros del Jurado designados por el Consejo de Escuela de Computación, para examinar el Trabajo Especial de Grado, presentado por las Bachilleres Sheyla M. Pirela L. C.I.: 15.540.798 y Chen Yu Lin C.I.: 25.263.822, con el título "**Aprenda Token Ring: Sistema Didáctico para la Enseñanza de la Arquitectura de Red Token Ring**", a los fines de cumplir con el requisito legal para optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído como fue dicho trabajo por cada uno de los Miembros del Jurado, se fijó el día 17 de Febrero de 2009, a las 2:00 PM, para que sus autores lo defiendan en forma pública, en Planta Alta III de la Escuela de Computación, mediante la exposición oral de su contenido, y luego de la cual respondieron satisfactoriamente a las preguntas que le fueron formuladas por el Jurado, todo ello conforme a lo dispuesto en la Ley de Universidades y demás normativas vigentes de la Universidad Central de Venezuela. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo.

En fe de lo cual se levanta la presente Acta, en Caracas a los diecisiete días del mes de Febrero del año dos mil nueve, dejándose también constancia de que actuó como Coordinador del Jurado el Profesor Tutor Eric Gamess.

Prof. Eric Gamess (Tutor)	Prof. Karima Velásquez (Tutora)
David Pérez (Jurado Principal)	Ana Morales (Jurado Principal)

AGRADECIMIENTOS

Gracias a Dios que me dio la fuerza, salud y esperanza para lograr mis metas, con su honorable enseñanza por medio de todas las personas, circunstancias y el tiempo, he aprendido y crecido de las experiencias buenas y malas. Las buenas para dejar como un ejemplo a los demás y, las malas se convierten en la base del éxito.

Gracias a mis padres, hermana y toda la familia que me han apoyado durante toda la vida, aunque no pueden estar a mi lado siempre, siento el cariño por el corazón. Especialmente a mi querido padre, sé que me estás viendo, cuidándome desde otra dimensión del universo, igual que yo comparto todos mis sentimientos que sean alegres o tristes contigo.

Gracias a mis tutores el Profesor Eric Gamess y la Profesora Karima Velásquez. Su apoyo y orientación a lo largo de la tesis fueron muy importantes para terminarla exitosamente. También a todos los otros profesores que me han educado durante el estudio de la carrera, sin ustedes no soy quien soy ahora con los conocimientos profesionales de computación.

Gracias a mi querida compañera Sheyla quien me ha brindado su gran apoyo y paciencia durante este largo tiempo de aprendizaje. Juntos hemos aprendido y conocido a Flash que era nuevo para nosotras. También a su querida madre quien siempre nos apoya física y mentalmente cuando necesitamos su ayuda.

Gracias a todos mis amigos y compañeros, los encuentros no fueron casuales, sino fueron destinados por nuestra afinidad. A pesar de que el tiempo de compañía no fue siempre largo, pero los recuerdos no se borran de mi mente ni de mi corazón. ¡Gracias por existir en este mundo y ser mis amigos!

¡Gracias a la UCV, sin ti no hubiera tenido la oportunidad de aprender y conocer tantas personas y cosas, ni tampoco de obtener una vida universitaria tan maravillosa! ¡¡Eres bella y generosa para caber tantos profesores, estudiantes y los personales administrativos, en ti se encuentra todo y me diste un recuerdo muy hermoso e inolvidable!!

UN MILLÓN DE GRACIAS!!!
CHEN YU LIN

AGRADECIMIENTOS

Primero que todo quiero agradecerle a mi madre, Miriam López, quien es la responsable de lo que soy hoy, ella fue quien me inculco los valores que tengo, se sacrificó para que yo pudiera llegar a donde estoy hoy, me ha apoyado siempre y me ha guiado por el buen camino. Igualmente a mi hermana Sherlin, ella es mi modelo a seguir, gracias hermana por estar siempre allí.

Gracias a mi abuela Carmen, que aunque no esta aquí físicamente nunca se ha apartado de mi corazón, y se que desde donde se encuentra me cuida y me escucha siempre. A toda mi familia, los quiero mucho, gracias por estar siempre pendiente de mi.

A Dios, por darme siempre la fuerza en los momentos difíciles, y por permitirme vivir cada día nuevas experiencias, sin ti no hubiera sido capaz de sobrepasar los obstáculos que se me han presentado, sin ti nada sería posible.

A mis tutores el Profesor Eric Gamess y la Profesora Karima Velásquez, quienes tuvieron mucha paciencia conmigo y me ayudaron siempre, me enseñaron a exigirme más, para poder lograr todos los proyectos que me plantee, gracias por guiarme, gracias por comprenderme, y sobre todo gracias por ayudarme a culminar mi carrera. En general, gracias a todos los profesores que me enseñaron tantas cosas, incluyendo algunos que aunque no me dieron clase, tuve la oportunidad de compartir y que dejaran una parte de sus vivencias en mí, específicamente gracias a la Profesora Carmen Elena Vera.

A mi compañera Chen Yu, por acompañarme en todo este tiempo, por siempre tener tu actitud positiva y pasiva, y confiar en mí.

A todos mis amigos y compañeros ucevistas, quienes compartieron conmigo toda esta etapa universitaria, sin ustedes mi carrera no hubiera sido tan placentera, no hubiera conocido lo que es el amor, la amistad y la lealtad, gracias por ser tan auténticos.

A la persona que nos ayudo en todo este mundo extraño que es flash, Manuel Herrera comprendo porque eres tan bueno es esto, tu personalidad es como el flash, un tanto abstracta, pero me encanta, gracias por ayudarnos tanto.

Gracias a todos!!!
Sheyla Pirela

RESUMEN

TÍTULO:
Aprenda Token Ring: Sistema Didáctico para la Enseñanza de la Arquitectura de Red Token Ring.

AUTORES:
Sheyla M. Pirela L.
Chen Yu Lin

TUTORES:
Prof. Eric Gamess
Prof. Karima Velásquez

El presente Trabajo Especial de Grado consiste en el desarrollo de una aplicación denominada *Aprenda Token Ring*, la cual será utilizada como apoyo en los cursos básicos de redes. *Aprenda Token Ring* está basada en el estándar IEEE 802.5 y provee un conjunto de herramientas que permiten a los estudiantes del área de Redes y Comunicaciones conocer el funcionamiento de la arquitectura de red Token Ring mediante simulaciones que muestran los procesos que se llevan a cabo en este tipo de redes dependiendo de un evento particular como lo es la inserción de una nueva estación, la eliminación de una estación, la eliminación de monitor activo, el envío de tramas, etc.

El trabajo se inicia con un estudio teórico de los aspectos más relevantes del estándar IEEE 802.5. Durante la implementación de la aplicación, se utilizó una metodología iterativa basada en las nuevas tendencias del modelo de desarrollo ágil que plantea la creación de cada una de las herramientas de la aplicación en una iteración del proceso que abarca las fases de Análisis, Diseño, Implementación y Pruebas.

Aprenda Token Ring incorpora también un conjunto de utilidades que permite que el estudiante se familiarice con el tema, como lo son un glosario de términos donde se da una breve explicación de la terminología más usada, una ayuda donde se explica cómo usar la aplicación, un módulo de evaluación donde se mide el aprendizaje del usuario.

El producto final junto a la documentación de la investigación serán ofrecidos a la Escuela de Computación de la Facultad de Ciencias de la UCV, para que sea utilizado por los profesores del área de Redes y Comunicaciones, como apoyo en la enseñanza de las redes Token Ring.

Palabras Claves: Token Ring, Sistema Didáctico, Topología de Anillo, Redes, Simulador.

Tabla de Contenido

Índice de Figuras

Índice de Tablas

Introducción

Actualmente la arquitectura de red LAN más utilizada es Ethernet, es por ello que Token Ring ha tendido a desaparecer; sin embargo aún se considera importante la enseñanza de este tipo de redes. Con el aumento de la popularidad de sistemas didácticos, se plantea la necesidad de contar con una herramienta que apoye a los profesores de la licenciatura en computación de la Universidad Central de Venezuela en la enseñanza de redes Token Ring.

Existen diferentes herramientas que muestran el funcionamiento de redes, pero sólo unas pocas cubren satisfactoriamente Token Ring. Es por ello que este Trabajo Especial de Grado se centra en el desarrollo de una herramienta que muestra de una forma más completa el funcionamiento de la capa MAC de las redes Token Ring.

Con este fin, se ha estructurado el documento en 5 capítulos que explican los diferentes aspectos tomados en cuenta durante la creación de la aplicación. A continuación se ofrece un breve resumen del contenido de cada uno de estos capítulos:

1. **Capítulo 1 (El Problema):** Plantea las razones que llevaron a realizar una aplicación que presenta de una manera profunda el funcionamiento de la capa MAC de las redes Token Ring, mostrando en detalle los objetivos y el alcance del presente Trabajo Especial de Grado.

2. **Capítulo 2 (Marco Teórico):** Presenta las bases teóricas que fueron estudiadas y en las cuales se basó el desarrollo de *Aprenda Token Ring*.

3. **Capítulo 3 (Marco Metodológico):** Explica la metodología utilizada durante el desarrollo de la aplicación.

4. **Capítulo 4 (Marco Aplicativo):** Detalla las diferentes etapas del proceso de implementación de cada uno de los módulos que conforman la aplicación mediante la metodología utilizada.

5. **Capítulo 5 (Conclusiones):** Presenta las conclusiones y recomendaciones obtenidas a partir del Trabajo Especial de Grado realizado.

1. El Problema

1.1 Planteamiento del problema

Las herramientas que actualmente simulan arquitecturas de redes en su mayoría se centran en redes Ethernet, por lo que muy pocas muestran el funcionamiento de redes Token Ring. Las pocas herramientas de simulación de Token Ring que existen no profundizan en el funcionamiento de la capa MAC.

Debido al gran auge que ha tenido Ethernet en el mercado, el uso de Token Ring ha tendido a desaparecer, por lo que las empresas que distribuyen el hardware necesario para implementar este tipo de red ya no los están fabricando, trayendo como consecuencia su extinción y por lo tanto la ausencia de un material palpable y visible para la enseñanza de esta tecnología. Sin embargo, Token Ring es enseñada en los cursos de introducción de redes ya que tiene conceptos importantes como la eliminación de colisiones, y una mejor escalabilidad que Ethernet.

En el pensum actual de la materia Redes de Computadores, la cual forma parte del componente electivo de la Licenciatura en Computación en la Universidad Central de Venezuela, está incluido un tema que muestra el funcionamiento de redes Token Ring. Actualmente, los profesores de esta cátedra no cuentan con ninguna aplicación didáctica en la que se puedan apoyar para enseñar esta tecnología de red. Es por ello que surge la necesidad de desarrollar una aplicación que muestre de forma sencilla y práctica el funcionamiento de Token Ring, específicamente de su capa MAC.

1.2 Objetivos

1.2.1 Objetivo General

Desarrollar una aplicación didáctica que sirva como apoyo en la enseñanza de Token Ring, la cual permita mediante el uso de simulaciones mostrar el funcionamiento de capa MAC de las redes en forma de anillo.

1.2.2 Objetivos Específicos

- Utilizar el lenguaje de programación Action Script 2.0, el cual viene inmerso en Flash CS3 para crear un sistema didáctico que permita simular redes Token Ring con una interfaz gráfica usable.
- Conocer y estudiar el estándar IEEE 802.5 para simular correctamente el funcionamiento de Token Ring.
- Implementar una herramienta que permita facilitar la enseñanza de las redes Token Ring en las materias del área de Redes y Comunicaciones.

- Desarrollar como parte de las herramientas de la aplicación el módulo de simulaciones, el módulo de evaluación, y el módulo de ayuda.
- Garantizar que todas las funcionalidades de simulación provistas por la aplicación ofrezcan el mayor acercamiento posible a la realidad del funcionamiento de Token Ring.
- Suministrar el producto final a los profesores del área de Redes y Comunicaciones para que sea utilizado.

1.3 Justificación

La finalidad de esta aplicación es principalmente dar apoyo en la enseñanza de Token Ring, el cual se enseña en materias del área de Tecnología en Comunicación y Redes de Computadores. Es de aclarar que en la actualidad no se cuenta con ninguna aplicación que simule el funcionamiento de la capa MAC de Token Ring.

Es importante la existencia de aplicaciones que simulen el funcionamiento de estas redes, ya que permite configurar y visualizar las tramas para lograr un mejor entendimiento de la capa MAC de Token Ring.

1.4 Alcance

La aplicación creada para el Trabajo Especial de Grado tiene el siguiente alcance:

- Mostrar el funcionamiento de los procesos de la capa MAC de las redes Token Ring. Las simulaciones de los procesos a mostrar son: Insertar estación, Eliminar Estación, Proceso de Contención, Proceso de Ring Poll o Notificación al Vecino y Envío de Tramas.

- Medir por medio de un Módulo de Evaluación, los conocimientos adquiridos por los estudiantes. Este módulo se divide en tres niveles, el básico donde se avalúan los conceptos básicos del tema, el intermedio donde se evalúa todo lo relacionado con los formatos de tramas y el avanzado donde se evalúan el funcionamiento de cada unos de los procesos simulados en la aplicación.

- Guiar al usuario mediante herramientas como la Ayuda, donde se explica como usar de la mejor manera la aplicación y un Glosario de Términos, para poder consultar el significado de los términos relacionados con el tema.

2. Marco Teórico

Este capítulo da una descripción de las redes Token Ring desde sus orígenes, pasando por su funcionamiento, especificando los procesos que permiten la comunicación entre los distintos dispositivos, así como también los mecanismos disponibles para la recuperación de la red luego de una falla. Se explicarán cada uno de los campos que conforman las tramas MAC, y cuales son los valores que pueden tomar cada uno de ellos dependiendo del estado en que se encuentre la red.

2.1 Descripción General

2.1.1 Historia

El primer diseño de una red de Token Ring es atribuido a E. E. Newhall en 1969. IBM publicó por primera vez su topología de Token Ring en marzo de 1982, cuando esta compañía presentó los papeles para el proyecto 802 del IEEE. IBM anunció un producto Token Ring en 1984, y en 1985 este llegó a ser un estándar de ANSI/IEEE, debido al apoyo de la primera empresa informática mundial [01].

La red Token Ring es una implementación del estándar IEEE 802.5, el cual se distingue más por su método de transmitir la información que por la forma en que se conectan las computadoras. El IEEE (Institute of Electrical and Electronics Engineers) ha desarrollado una serie de estándares (IEEE 802.X) en los que se definen los aspectos físicos (cableado, topología física y eléctrica) y de control de acceso al medio de redes de área local. Estos estándares son reconocidos internacionalmente (ANSI, ISO, etc.), y adoptados por ISO en una serie equivalente ISO 8802.X [01].

La norma 802.5 que ha realizado el IEEE define redes con anillo lógico en un anillo físico (también se puede configurar el anillo lógico sobre una topología física de estrella) y con protocolo MAC de *token passing*. La norma establece distintos niveles de prioridad (codificados mediante unos bits incluidos en el *token*). Las velocidades de transmisión normalizadas son de 1, 4, 16, 20 y 40 Mbps, siendo la más común la de 16 Mbps, y existen diferentes tipos de cableado como UTP, STP y cable coaxial [01].

Hasta finales de 1988, la máxima velocidad permitida en este tipo de redes era de 4 Mbps, con soporte físico de par trenzado. En esa fecha se presentó la segunda generación, Token Ring-II, con soporte físico de cable coaxial y de fibra óptica, y velocidades de hasta 16 Mbps. Sin embargo, las redes antiguas con cable de par trenzado debían recablearse si se querían utilizar las prestaciones de las de segunda generación, lo cual representa un buen ejemplo de la importancia que las decisiones sobre cableado tienen en la implantación de una red de área local [01].

En 1994, los principales proveedores de Token Ring formaron la Alianza para el Adelanto Estratégico y la Dirección de Token Ring, ASTRAL (siglas en inglés de Alliance for Strategic Token Ring Advancement and Leadership). La misión ASTRAL fue promotora de la tecnología Token Ring frente al aumento de la popularidad de Ethernet. Los miembros iniciales de ASTRAL eran 3Com, ACE/North Hills, Bay Networks (SynOptics and Wellfleet), Bytex, Cabletron, Centillion, Chipcom, Hewlett-Packard, IBM, Intel, Madge, Olicom, Proteon, Racore, SMC, Texas Instruments, Xircom, XPoint y UB Networks [02].

En 1997 el borrador del estándar 802.5r es presentado; en el se definió la operación del Token Ring dedicado (DTR) también conocido como Token Ring full duplex, que transforma el protocolo Token Passing para permitir a dos estaciones comunicarse en un enlace punto a punto. Este dobla eficazmente la tasa de transferencia permitiendo que cada estación transmita y reciba concurrentemente secuencias de datos separadas. Por ejemplo, una estación de Token Ring dedicado puede transmitir un flujo de datos a 16 Mbps al mismo tiempo que recibe un flujo de datos separado a 16 Mbps. Esto proporciona una tasa de transferencia de datos total de 32 Mbps [02].

En 1997, la Alianza de Token Ring de Alta Velocidad, HSTRA (con sus siglas en ingles, High Speed Token Ring Aliance) fue formada para perseguir un estándar de IEEE 802.5 para Token Ring dedicado de alta velocidad que fuera desde 100 Mbps hasta 1 Gbps. Los miembros primarios de HSTRA eran 3Com, Bay Networks, IBM, Madge Networks, Olicom, UNH Interoperability Lab, y Xylan [02].

En 1998, el borrador del estándar 802.5t es presentado. Allí se define la operación de Token Ring a 100 Mbps. El estándar de 100 Mbps tiene una operación dedicada restringida, es decir, este estándar no se planificó con la finalidad de compartir el medio. Los primeros productos de Token Ring de 100 Mbps fueron introducidos por Olicom e IBM en 1998 [02].

2.1.2 Comparación entre el estándar IEEE 802.5 y Token Ring

Token Ring es muy parecido al estándar IEEE 802.5, como se dijo anteriormente el primero en darse a conocer fue el Token Ring implementado por IBM, luego, el estándar se basó en el diseño de Newhall para desarrollar el estándar. Sin embargo en el estándar 802.5 no se especifica un tipo de topología en particular, mientras que Token Ring sí especifica la topología (en estrella) [03]. Hoy en día ambos son equivalentes, por lo que se usa el término Token Ring para referirse a ambos indistintamente [04].

Otra diferencia se basa en el tipo de medio a utilizar. Token Ring establece que el medio a usar es par trenzado mientras que 802.5 no especifica ninguno. Las diferencias existentes se resumen en la Tabla 2.1 (tomada de [03]).

	Token Ring IBM	IEEE 802.5
Velocidad de Transmisión	4/16 Mbps	4/16 Mbps
Estaciones por Segmento	270 STP y 72 UTP	250
Topología	Estrella	No Especificado
Medio	Par Trenzado	No Especificado
Señalización	Banda Base	Banda Base
Método de Acceso	Token Passing	Token Passing
Codificación	Manchester Diferencial	Manchester Diferencial

Tabla 2.1: Especificaciones de Token Ring / IEEE 802.5

2.1.3 Topología

Los nodos de red necesitan estar conectados para comunicarse. A la forma en que están conectados los nodos se le llama *topología*. Una red tiene dos diferentes topologías: una física y una lógica. La topología física es la disposición física de la red, la manera en que los nodos están conectados unos con otros. La topología lógica es el método que se usa para comunicarse con los demás nodos, la ruta que toman los datos de la red entre los diferentes nodos de la misma [05].

La topología física de Token Ring es estrella, donde todos los mensajes deben pasar a través de un dispositivo central de conexiones conocido como concentrador de cableado, el cual controla el flujo de datos [05]. En este caso particular, a este dispositivo se le conoce con el nombre de MAU (Multistation Access Unit).

La topología lógica de Token Ring tiene forma de anillo. Una topología anillo es una red punto a punto, en la cual los dispositivos están conectados máquina a máquina, en un circulo unidireccional cerrado.

La topología Token Ring usa un método de acceso llamado *token passing*. Ninguna estación puede transmitir a menos que esta posea el *token*. Debido a esta restricción para transmitir, Token Ring es un protocolo determinístico, por lo que se puede calcular exactamente el tiempo de retardo de la transmisión.

La topología de anillo puede ser naturalmente compleja. Aunque esta es relativamente fácil de expandir, usualmente hay que tomar con cuidado los cálculos de factores de diseño físico, el máximo de estaciones soportadas y la estructura del cableado por ejemplo, para mantener la red apegada a la especificación. La mayoría de las topologías de anillo se representan físicamente como una estrella.

Agregar o remover estaciones de la red es relativamente simple y puede ser hecho mientras la red está activa. El software incluido en las tarjetas de red Token Ring en cada estación del anillo automáticamente reconfigura el anillo lógico cuando se añade o remueve una estación. Conectores especiales son usados para mantener la integridad del anillo [06].

La configuración de la topología de Token Ring se puede describir de la manera siguiente [07]:

- Una serie de estaciones conectadas en forma serial al medio de transmisión.

- Cada estación en el anillo regenera y repite cada bit recibido.

- Cada estación tiene el puerto para transmitir y el puerto para recibir de forma separada, es decir la tarjeta de red (NIC – Network Interfaz Card) Token Ring posee dos puertos, el transmisor (TX) y el receptor (RX).

- Cada puerto es cableado (por medio de par trenzado) hacia un concentrador central conocido como MAU (Multistation Access Unit).

- El MAU proporciona la capacidad de conectar y desconectar hasta 8 estaciones eléctricamente. Además posee dos puertos extras para ser conectado a otros MAUs y extender el anillo. A estos puertos se le conoce como *Ring In* (RI) y *Ring Out* (RO).

- En caso que varios MAUs estén conectados, la última estación en un MAU debe enviar la señal de salida por el puerto RO y la información a este MAU entrará por el puerto RI hacia la primera estación del mismo.

La Figura 2.1 muestra tanto la topología lógica como la topología física en una red Token Ring.

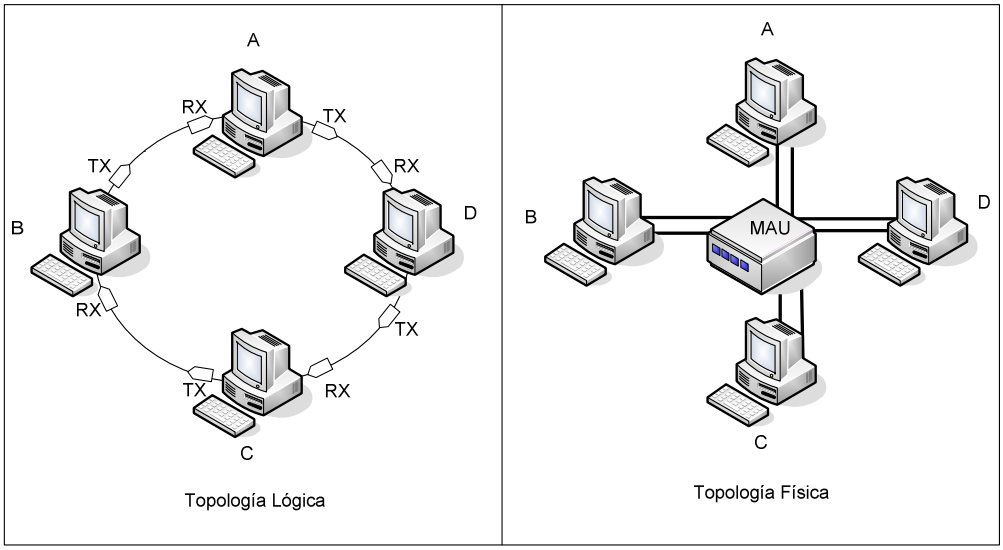

Figura 2.1: Topología Token Ring

2.1.4 Funcionamiento

Token Ring utiliza el método de acceso al medio conocido como *token passing*, el cual consiste en que una sola estación puede transmitir en determinado instante, esta estación es precisamente la que posee en ese momento el *token*, este es el encargado de asignar los permisos para transmitir los datos [01].

La información que viaja en el anillo va en una sola dirección a lo largo de la red. No requiere de enrutamiento, ya que cada paquete es pasado a su vecino y así consecutivamente. Por ejemplo, se tienen tres estaciones de trabajo A, B y C, si una estación A transmite un mensaje, este es pasado a B, luego es pasado a C y finalmente regresa a la estación de origen A [01].

El *token* se mantiene circulando constantemente a través de todo el anillo mientras ninguna estación necesite transmitir. Cuando alguna máquina desea enviar datos debe esperar a que le llegue el *token*. Cuando lo recibe, adjunta el mensaje al *token* formando una trama y activa una señal indicando que el bus está ocupado. La trama continúa su recorrido en orden hasta llegar a la estación destino. La estación que envió puede verificar si la trama encontró a la estación destino y si entregó la información correspondiente (copia de la información en un buffer). Cuando una estación recibe una trama, esta modifica algunos bits de la trama para informar a la estación receptora sobre la recepción y la copia en el buffer. Un dispositivo tiene que esperar hasta que el *token* llegue a el para poder enviar otra trama [01].

Si una estación desea enviar una trama, pero en ese momento ya existe una trama en la red, envía un comando de espera para indicar su deseo de transmitir (por lo general, transcurren sólo unas fracciones de segundo). Debido a que una computadora requiere el *token* para enviar información, no hay colisiones.

El *token* es una trama especial, que no debe confundirse con una trama de datos. Ninguna estación puede retener el *token* por más de un tiempo dado, este tiempo es controlado por el THT (Token Holding Timer), y el rango de este tiempo va desde los 8.9 hasta los 9.1 ms. El problema reside en el tiempo que debe esperar una estación para obtener el *token*. El *token* circula muy rápidamente, pero obviamente esto significa que la mayor parte de las veces, los dispositivos tendrán que esperar algo antes de poder enviar un mensaje [01].

La eficiencia en este sistema se debe a que las comunicaciones siempre viajan en una misma dirección y el sistema únicamente permite que una información este viajando por el cable en un momento dado [01].

El funcionamiento de Token Ring se puede resumir en los siguientes pasos [04]:

1. Si una estación que tiene información por transmitir recibe un token, lo retira de la red, y envía la trama de datos.

2. Mientras que la trama circula alrededor del anillo no existe otro *token* en la red, por lo tanto otras estaciones que deseen transmitir deberán esperar. Por esta razón, es difícil que se presenten colisiones.

3. La información de la trama circula en el anillo hasta que localiza la estación destino, la cuál copia la información para poder procesarla.

4. La información de la trama continúa circulando en el anillo y finalmente es eliminada por la estación origen.

5. La estación que envió la trama puede revisar si esta encontró a la estación destino y si la información correspondiente fue copiada (acuse de recibo).

A diferencia de las redes que utilizan CSMA/CD (como Ethernet), las redes token passing están caracterizadas por la posibilidad de calcular el máximo tiempo que puede permanecer una terminal esperando para transmitir datos [04].

2.1.5 Sistemas de prioridades

Token Ring soporta un mecanismo de prioridades que permite a las estaciones transmitir paquetes con ocho niveles diferentes de prioridad. Cada estación tiene asignada una prioridad de acceso que indica la prioridad máxima que puede tener un *token* enviado por dicha estación para transmitir datos. Una estación puede utilizar un *token* sólo si este tiene una prioridad menor o igual a la requerida por la estación.

Las tramas en las redes Token Ring tienen dos campos que controlan la prioridad: *Priority* y *Priority Reservation*. La prioridad es indicada en los tres primeros bits del byte de *Access Control*, mientras que si una estación desea reservar una prioridad, debe ser asignada en los últimos tres bits de dicho byte. Una estación

usa estos bits de *Priority Reservation* para requerir que un *token* sea originado en el anillo con la prioridad deseada.

Si otra estación tiene reservada una prioridad mayor o igual a la que se quiere reservar, la estación no puede cambiar la reservación en el *token*. Por el contrario si los tres bits de *Priority Reservation* no han sido modificados o si se le ha asignado una prioridad menor a la que se está requiriendo, la estación tiene la potestad de cambiar estos bits agregando la prioridad deseada.

Cuando una estación elimina una trama con un valor diferente a cero en el campo de *Priority Reservation*, esta debe liberar un nuevo *token* con la prioridad indicada en dicho campo. Para prevenir que una estación esté transmitiendo continuamente, se aplica un criterio de imparcialidad entre los diferentes niveles de prioridad. Una estación que origina un *token* con un incremento en la prioridad, eventualmente debe reemplazarla con la prioridad original del *token* [02].

2.1.6 Mantenimiento del anillo

Cuando una red Token Ring se inicia, todas las estaciones participan en una negociación para decidir quién controlará el anillo; a este proceso de negociación se le conoce con el nombre de *Monitor Contention*. La estación que gane esta contienda se denomina *monitor activo* o como sus siglas en inglés AM (Active Monitor). Esta elección es ganada por la estación con la dirección MAC más alta de todas las estaciones que estén participando en el procedimiento, el resto de las estaciones se convertirán en *monitores en espera* también conocidos por sus siglas en inglés SM (Standby Monitor).

El trabajo del *monitor activo* es cerciorarse de que ninguna de las estaciones esté causando problemas en la red, y reestablecer el anillo después de una ruptura o de un error. El AM realiza un proceso de *Neighbor Notification*, conocido como *Ring Poll* cada siete segundos y envía mensajes *Ring Purge* si ocurre un problema. El *Ring Poll* permite que todas las estaciones en la red descubran quién está participando en el anillo, además de aprender la dirección de su *NAUN* (Nearest Active Upstream Neighbor). El *Ring Purge* reajusta el anillo después de que una interrupción o una pérdida de datos, se divulgue.

Cuando una estación se incorpora al anillo realiza una prueba para verificar que su propia conexión está trabajando correctamente, si todo está correcto, envía un voltaje al MAU que funciona como una notificación para insertarla en el anillo.

2.1.7 Ventajas

- No requiere de enrutamiento.

- Es fácil de extender, ya que el nodo está diseñado como repetidor, por lo que permite amplificar la señal y enviarla a mayores distancias.

- Provee una especificación de funcionamiento determinística, por lo cual el tiempo de acceso (por ejemplo el máximo tiempo entre *i* y *i+1* usado por el *token*) puede ser determinado.

- Bajo cargas pesadas, Token Ring escala en forma eficiente.

- La capa física provee un número de características para detectar errores de hardware y mejorar su funcionamiento.

- Soporta tramas de gran tamaño, mayores a 18 Kb a 16 Mb.

2.1.8 Desventajas

- Altamente susceptible a fallas.

- Una falla en un nodo deshabilita toda la red (esto refiriéndose estrictamente al concepto puro de topología de anillo).

- El software de cada nodo es mucho más complejo.

- La complejidad de Token Ring requiere un mayor entendimiento de su operación para permitir la administración de la red.

- El número de nodos es relativamente bajo (260 por segmento de anillo) comparado con Ethernet (1024).

- Los equipos Token Ring son más costosos que los equipos Ethernet [08].

2.2 Formatos de Tramas

Token Ring soporta dos tipos de tramas: la de *token* y la de datos/comando. La de *token* está compuesta por tres bytes, los cuales son SD (Start Delimiter), AC (Access Control Byte), y ED (End Delimiter). La trama de datos/comando tiene un tamaño variable, el cual depende de la longitud que tenga el campo información. La trama de datos contiene información para protocolos de capa superior, se le conoce también como trama LLC (Logical Link Control), mientras que la trama de comando contiene información de control y no transporta datos para protocolos de capa superior. A la trama de control, también se le conoce como trama MAC (Media Access Control) [03]. Ambos formatos se muestran en la Figura 2.2.

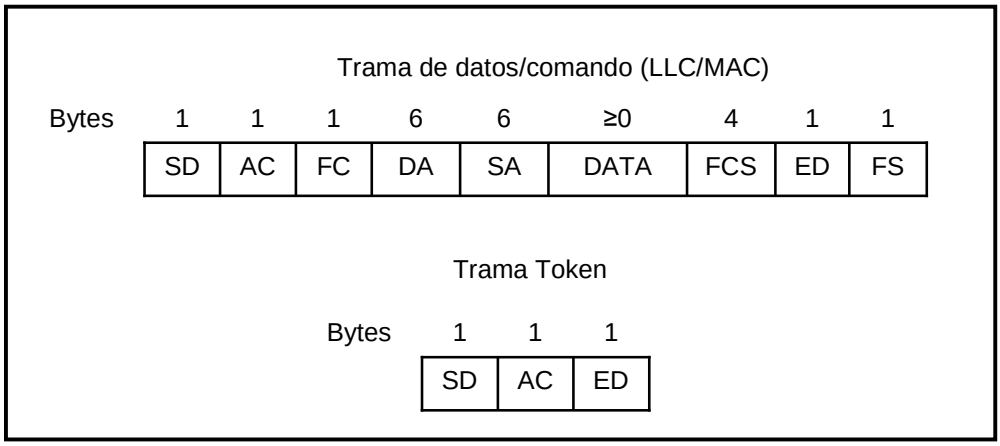

Figura 2.2: Formatos de tramas

2.2.1 Campos de un token

- **SD (Start Delimiter)**: Este byte indica el inicio de una trama, bien sea de *token* o datos/comando. Incluye una señal que lo distingue del resto de los bytes de la trama [03], violando el esquema de codificación Manchester Diferencial, donde cada señal tiene una transición en el medio, estas violaciones no presentan dicha transición [07]. La Figura 2.3 muestra la estructura de este byte, (con J como una señal que parece un 1 binario pero sin la transición y K como una señal que parece un 0 binario pero sin la transición).

J	K	0	J	K	0	0	0

Figura 2.3: Start Delimiter Byte

- **AC (Access Control Byte)**: Contiene *Priority* en los tres primeros bits, y *Priority Reservation* en los últimos tres bits. Los dos bits restantes corresponden al *Token Indicator*, el cual va a diferenciar un *token* de una trama datos/comando (si está en 1 significa que la trama es un *token*, si está en 0 entonces es una trama datos/comando), y al MC (Monitor Count), el cual es usado por el *monitor activo* para saber si la trama ha circulado más de una vez por el anillo [03]. El formato de este byte se muestra en la Figura 2.4.

Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
Priority			Token Indicator	Monitor Count	Priority Reservation		

Figura 2.4: Access Control Byte

- **ED (End Delimiter)**: Este byte indica el final de una trama, bien sea de *token* o datos/comando. Este campo además contiene un bit que indica si hay un error en la trama (bit E) y otro que identifica si es la última trama de una secuencia (bit I) [03]. El formato de este byte se muestra en la Figura 2.5 (con J y K teniendo el mismo significado que en el caso del SD).

J	K	1	J	K	1	I	E

Figura 2.5: End Delimiter Byte

2.2.2 Campos de la trama de datos/comando (LLC/MAC)

Los campos SD y ED no cambian en relación a la trama *token*, y el campo AC sólo cambia el bit de *Token Indicator* a uno cuando se trata de una trama datos/comando. A continuación se explican el resto de los campos.

- **FC (Frame Control Byte)**: Indica si la trama contiene datos o información de control. En caso de que los dos primeros bits sean 00, se trata de una trama MAC y los últimos cuatro bits se denominan PCF (Physical Control Field), los cuales muestran el código del tipo de información de control que se transmite. Si todos los bits están en cero indica que la trama será procesada usando un *buffer* de memoria normal; estos códigos se explican en la Tabla 2.2 [07].

Código	Mensaje
0001	Express Buffer
0010	Beacon
0011	Claim Token
0100	Ring Purge
0101	Active Monitor Present
0110	Standby Monitor Present

Tabla 2.2: Códigos de mensajes PCF

- **DA (Destination Address)**: Corresponde a un campo formado por 6 bytes, el cual contiene la dirección MAC de la estación destinataria. Incluye en la dirección dos bits que indican:
  - Si la DA pertenece a un grupo o es una dirección individual y
  - Si la DA es una dirección administrada local o universalmente [09].

  La Figura 2.6 muestra la estructura de esta dirección.

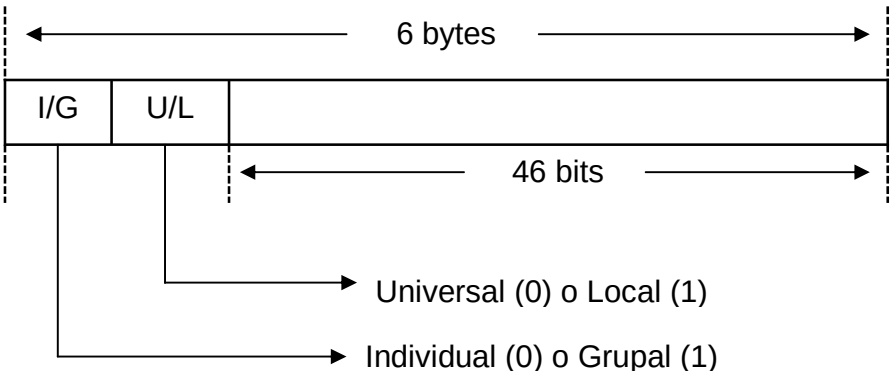

Figura 2.6: Formato de dirección MAC destino

- **SA (Source Address):** Corresponde a un campo formado por 6 bytes, el cual contiene la dirección MAC de la estación que envía la trama. En contraste al campo DA, no codifica el bit I/G ya que esta dirección siempre va a ser individual. En su lugar se coloca el bit RII (Routing Information Indicator), el cual si se encuentra en 1 indica que el campo de información de ruta está presente, por el contrario si está en 0 entonces el campo no está presente [09]. La Figura 2.7 muestra el formato de la dirección MAC fuente.

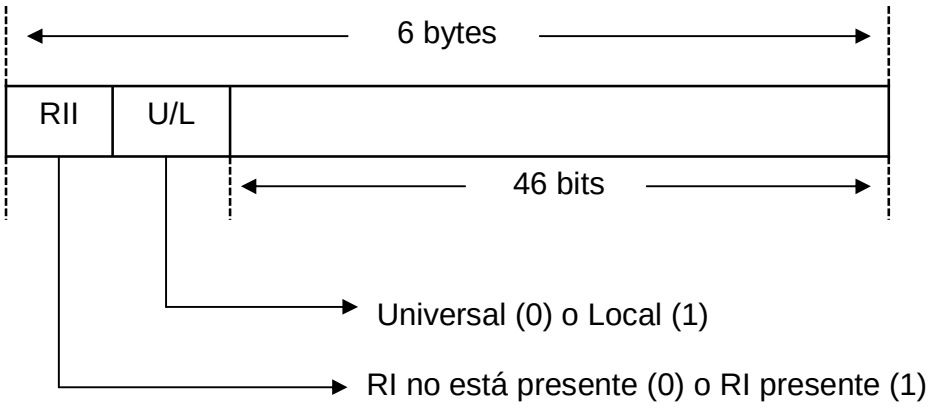

Figura 2.7: Formato de dirección MAC fuente

- **Data**: Este campo tiene longitud variable (≥0), depende de la información que se transmita, puede ser de control (MAC) o de protocolos de capas superiores (LLC). El tamaño máximo de este campo es limitado por el tamaño máximo de la trama. El campo FC (Frame Control Byte) es el que va a indicar cual es el formato de la información que se va a transmitir [02]. El formato de las tramas MAC será explicado en la Sección 2.4. El formato de la tramas LLC no está especificado en el estándar IEEE 802.5.

- **FCS (Frame Check Sequence)**: Este campo es agregado por la estación fuente con un valor calculado dependiente del contenido de la trama. La estación destino recalcula este valor para determinar si la trama ha sufrido

errores durante la transmisión. En caso de ser así, esta es descartada [03]. El cálculo de este valor se basa en el siguiente polinomio estándar de grado 32 [09], el cual se usa para aplicar la técnica de redundancia cíclica (CRC) [02]:

$$G(X)=X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^{8}+X^{7}+X^{5}+X^{4}+X^{2}+X+1$$

Un FCS válido es el complemento a uno de la suma (módulo 2) de lo siguiente [09]:

- El resto de $X^{k}(X^{31}+X^{30}+X^{29}+...+X^{2}+X+1)$ divido (módulo 2) por G(*X*), donde k es el número de bits en los campos FC, DA, SA, y de información, y
- El resto después de multiplicar por X^{32} y luego dividir (módulo 2) por G(*X*) el contenido (tratado como polinomio) de los campos FC, DA, SA y de información.

- **FS (Frame Status)**: Es un campo de un byte que está al final de la trama de datos/comando. El formato de este campo se muestra en la Figura 2.8.

A	C	r	r	A	C	r	r

Figura 2.8: Campo de estado de trama (FS)

Los bits A (address-recognized bits) y C (Frame-copied bits) deben ser transmitidos como 0 por la estación donde se origina la trama. Si otra estación reconoce la dirección DA como suya, o pertenece al grupo de direcciones, o es indicado por la interfaz del puente, estos bits deben ser cambiados a 1 [02].

Los bits r están reservados para estandarizaciones futuras, deben ser transmitidos como 0 en todo momento.

2.3 Operación de Token Ring

El estándar IEEE 802.5 proporciona a cada estación los siguientes 4 agentes de administración de la red:

1. Active or Standby Monitors (AM ó SM)
2. Ring Error Monitor (REM)
3. Configuration Report Server (CRS)
4. Ring Parameter Server (RPS)

Estos agentes son responsables de generar las 25 variaciones de las tramas MAC usadas para las operaciones del anillo. Inherentemente estos proveen una gran cantidad de estadísticas de la red que son usadas por los productos de administración de la red.

2.3.1 Active Monitor

La primera estación a inicializar en el anillo es el *monitor activo* (AM). Como parte de sus deberes, el AM inicializa el anillo (Ring Purge) y manda un token. El AM provee el reloj principal para la red y las demás estaciones reciben sus tiempos desde este reloj. Un estado latente del anillo de 24 ó 32 bits (para anillos de 4 Mbps ó 16 Mbps, respectivamente) es ofrecido por el AM. El bit *Monitor Count* es fijado y verificado por el AM para asegurar que las tramas no recirculen en el anillo.

2.3.2 Standby Monitor

Todas las demás estaciones son monitores en espera (SMs). Los SMs verifican que el AM está en el anillo cumpliendo sus deberes. Cada SM verifica los tokens válidos usando “el contador del tiempo del token válido” (2.6 s); los SMs también verifican que el AM está transmitiendo una trama *Active Monitor Present* utilizando el contador del tiempo de recepción del Poll (15 s). Si un contador de tiempo del token válido ó un contador del tiempo de recepción del Poll de los SMs expira, la estación envía una trama *Claim Token*. En el evento de una emisión de trama *Claim Token*, todas las estaciones entran a un diálogo de *Monitor Contention*, resultando una estación (la dirección activa más alta actualmente) como el AM.

2.4 Gerencia del anillo

Una estación en cada anillo es designada como el monitor activo (AM). Esta estación es responsable de monitorear el *token* y proveer otras funciones de gerencia del anillo. Todas las demás estaciones del anillo son monitores en espera, que son capaces de cumplir las funciones del AM de ser necesario.

Una de las tareas de gerencia más importante es *Ring Purge*, donde el AM puede generar una trama *Ring Purge* (PRG) para limpiar el anillo de cualquier error.

El proceso *Ring Purge* ocurre en respuesta a las siguientes condiciones:

- Una estación se ha convertido en AM recientemente.
- Una trama ha circulado más de una vez por el anillo, indicada por el bit *Monitor Count* a 1.
- El anillo está silencioso o con tráfico ilegal, entonces el tiempo válido de transmisión (TVX) ha expirado.

Cada estación en el anillo que recibe la trama *Ring Purge* cancela todos los contadores de tiempo y reinicia al modo de repetición normal si la trama no puede ser copiada. Si la trama puede ser copiada, una acción es tomada de acuerdo al campo información de la trama MAC. El campo información contiene la dirección de su *NAUN*.

Si la trama *Ring Purge* hace la circulación en el anillo exitosamente, el monitor ha sido reiniciado correctamente en el anillo; entonces un nuevo *token* será enviado por el monitor.

Es posible que la trama *Ring Purge* no haga la circulación en el anillo, en tal caso el contador del tiempo de purgación en el anillo del monitor expirará y el monitor entrará a *Monitor Contention Mode* (MCM) [06].

2.5 Ring Poll Process

2.5.1 Retiro de tramas circulantes

Para detectar tramas circulantes (tramas que circulan en forma indefinida) se usa el bit *Monitor Count* (MC), el cual está ubicado en el byte de *Access Control* (AC). Todas las tramas son inicialmente transmitidas con este bit en 0, y cuando la trama pasa por el monitor activo, este bit es cambiado a 1.

Con el anillo funcionando normalmente, una trama enviada debe ser retirada del anillo por la estación que la originó, por lo que no puede pasar por el monitor activo más de una vez. Si el monitor activo detecta una trama con el bit MC en uno, esto indica que la trama no ha sido retirada por la estación que la originó, por lo que el monitor activo tiene que retirar esta trama, enviar una trama de Ring Purge y generar un nuevo token.

Cada estación tiene la capacidad de cambiar y detectar el bit MC. En una operación normal del anillo, el retiro de tramas circulantes es estrictamente responsabilidad del monitor activo; es decir, sólo el puede remover tramas circulantes. Sin embargo, hay un modo de fallo en el anillo, en el cual se envían tramas *Beacon* durante el proceso de envío de estas tramas, las estaciones están capacitadas para detectar la circulación de las mismas. La primera estación que detecte una trama *Beacon* deberá asignarle al bit MC uno, y cambiarle este bit a todas las tramas *Beacon* que reciba. Cuando el anillo se recupere, puede haber una trama *Beacon* circulando, esta es detectada por la estación que activó el MC en la primera trama, y la estación comienza el proceso de recuperación del anillo. A este proceso se le conoce como *Monitor Contention*, descrito en la Sección 2.8.

2.5.2 El proceso Neighbor Notification

El proceso Neighbor Notification ocurre cada siete segundos durante el proceso Ring Poll. Cada estación aprende y recuerda quién es su *NAUN* por medio de los procesos de AMP (Active Monitor Present) y SMP (Standby Monitor Present). Cuando una estación reporta un problema, esta además reporta quién es su NAUN; esto ayuda al administrador de la red a encontrar el dominio de falla.

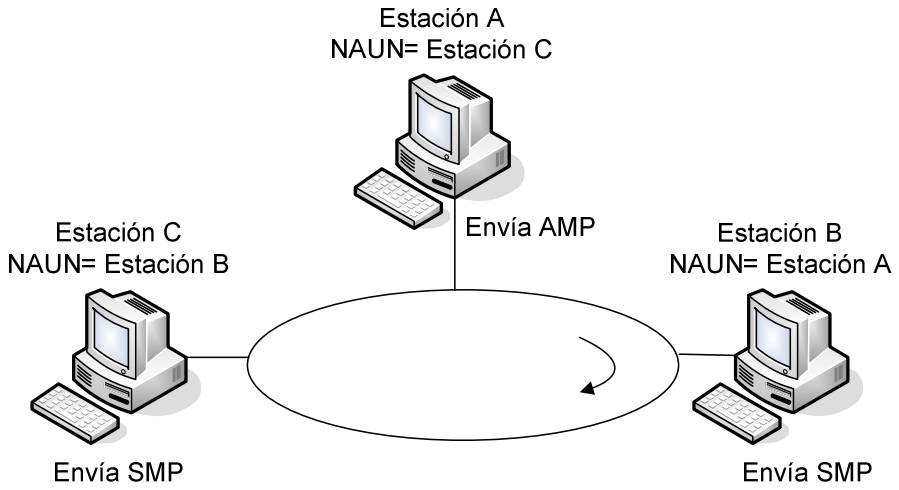

Figura 2.9: Proceso Neighbor Notification

En la Figura 2.9, la estación A es el monitor activo, esta envía una trama AMP, así es como la estación B aprende que su NAUN es la estación A. Cuando la trama AMP se retira por su remitente, la estación B envía una trama SMP, con ella es que la estación C aprende que su NAUN es la estación B. Después de que la estación B retire su trama SMP, la estación C repite lo mismo y el proceso Ring Poll continúa hasta que el monitor activo determine finalmente quién es su NAUN.

2.5.3 Funcionamiento del proceso Ring Poll

El funcionamiento del proceso Ring Poll se muestra a continuación:

1. Cada 7 segundos el tiempo del Ring Poll expira y el monitor activo se prepara para enviar una trama AMP cuando recibe un token libre (ver *Figura 2.10*).

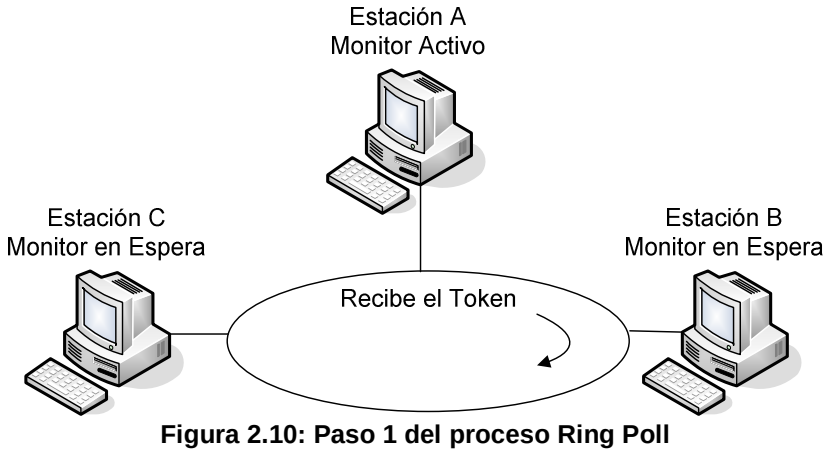

Figura 2.10: Paso 1 del proceso Ring Poll

2. El monitor activo (Estación A) envía una trama AMP con los bits ARI y FCI del byte Frame Status (FS) en cero. Esta trama es enviada a la dirección de broadcast 0xC000FFFFFFFF (ver *Figura 2.11*).

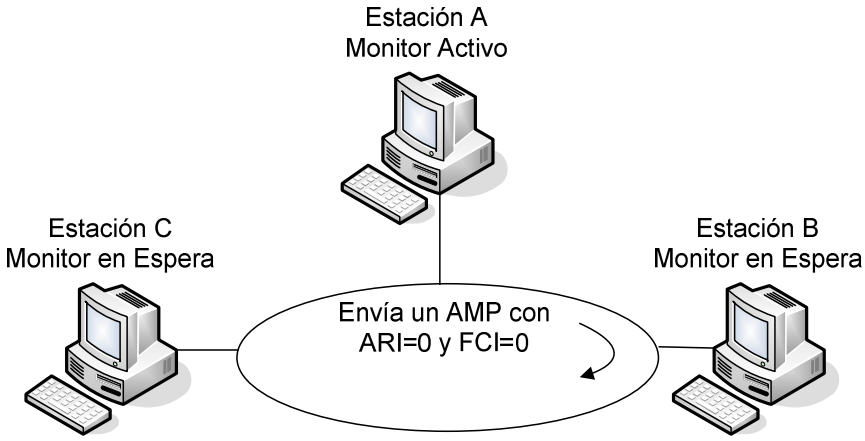

Figura 2.11: Paso 2 del proceso Ring Poll

3. El monitor en espera (Estación B) recibe la trama AMP; observa que los bits ARI y FCI están en cero, lo cual le indica que debe enviar una trama SMP una vez que ha expirado el tiempo TQP (Timer, Queue PDU). También revisa el registro SUA, el cual guarda la dirección de su *NAUN*, con la finalidad de comprobar si la estación que le está enviando la trama AMP es la misma que tenía registrada. En el caso de no ser la misma, actualiza el registro y envía una trama de reporte de cambio SUA dirigida a la dirección funcional del servidor de configuración una vez recibido un token libre (ver *Figura 2.12*).

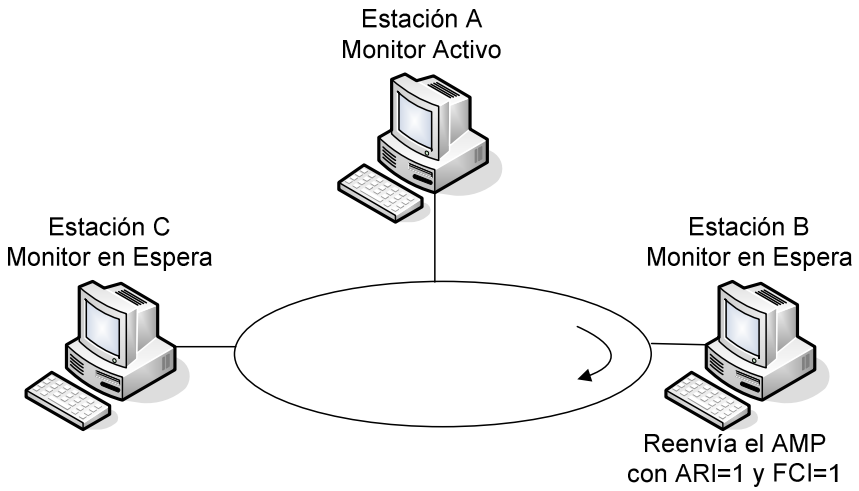

Figura 2.12: Paso 3 del proceso Ring Poll

4. La trama AMP continúa circulando por el anillo, las estaciones en espera que reciben esta trama notan que tiene los bits ARI y FCI en uno, por lo tanto saben que la estación remitente de esta trama no es su *NAUN*, por lo que la acción que toman es únicamente la de reenviarla. Una vez que la

trama llegue de vuelta al monitor activo, este la retira del anillo y libera un token (ver *Figura 2.13*).

Figura 2.13: Paso 4 del proceso Ring Poll

5. Una vez expirado el tiempo TQP, la primera estación que recibe AMP (la estación B) procede a enviar su SMP con los bits ARI y FCI en cero (ver *Figura 2.14*).

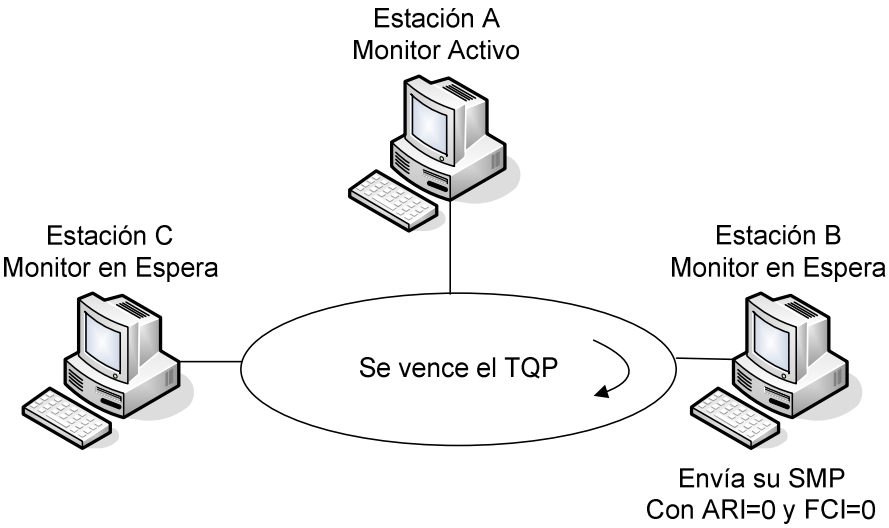

Figura 2.14: Paso 5 del proceso Ring Poll

6. La estación en espera que recibe la trama SMP con los bits ARI y FCI en cero sabe que la estación origen es su *NAUN*, por lo que actualiza su SUA, cambia los bits ARI y FCI a uno y regresa la trama al anillo, luego espera que el tiempo TQP expire para enviar su SMP (ver *Figura 2.15*).

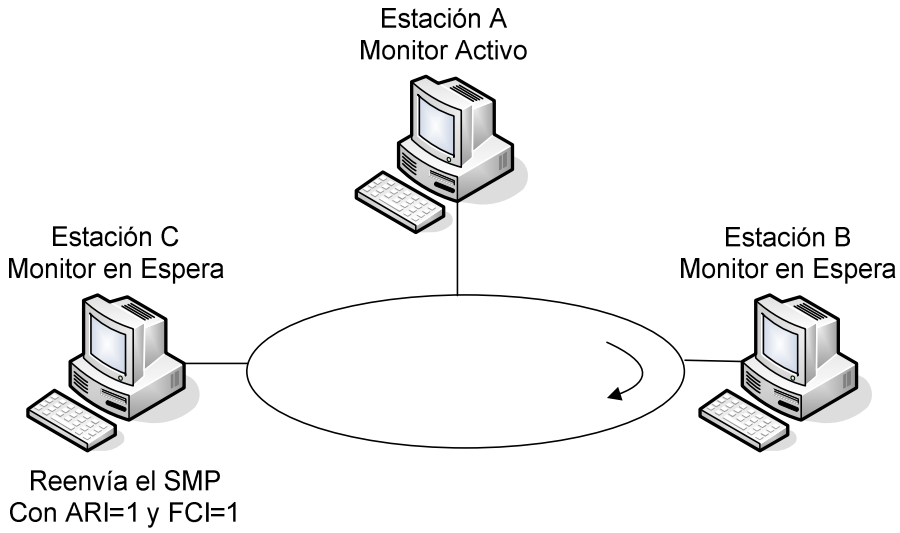

Figura 2.15: Paso 6 del proceso Ring Poll

7. El proceso se repite en cada una de las estaciones, hasta el momento en que el monitor activo recibe una trama SMP de su *NAUN*. Cuando esto sucede se da por sentado que el proceso Ring Poll ha finalizado satisfactoriamente (ver *Figura 2.16*).

Figura 2.16: Paso 7 del proceso Ring Poll

8. En el caso de que el monitor activo no reciba una trama SMP de su *NAUN* en un lapso mayor de siete segundos, este reporta un mensaje de falla del Ring Poll, y envía nuevamente una trama AMP para comenzar nuevamente el proceso.

2.6 Inserción y Eliminación de Estaciones en el anillo

2.6.1 Proceso de inserción de una estación en el anillo

Al momento de insertar una estación en el anillo, esta debe pasar primero por una serie de fases para poder comenzar a participar en el envío y recepción de tramas de las demás estaciones pertenecientes al anillo. A continuación se enumeran estas fases:

- **Fase 0, Lobe Test**: Una estación realiza una prueba del lobe antes de incorporarse al anillo. Una serie de tramas de prueba del lobe son enviadas sobre el cable del lobe de la estación para cerciorarse de que no hay averías en el cable. El cable del lobe es el cable que va desde la unidad del acceso múltiple (MAU) a la estación. Si la prueba del lobe termina con éxito, la estación se incorpora en el anillo enviando una señal de impulsión fantasma al concentrador, y continúa con la fase 1. Si la prueba del lobe falla, se termina el proceso de la inserción.

- **Fase 1, Monitor Verification:** Una estación inicia un contador de tiempo de asociación y espera una trama *Active Monitor Present* (AMP), *Standby Monitor Present* (SMP), o una trama Ring Purge del anillo. Si una de estas tramas se recibe antes de que expire el contador de tiempo, la estación asume que un monitor activo está presente y pasa a la fase 2. Si el contador de tiempo expira sin que se haya recibido alguna de estas tramas, la estación emite *Claim Token*, para luego entrar en el proceso *Monitor Contention*, de modo de elegir un nuevo monitor activo.

- **Fase 2, Duplicate Address Verification:** Una vez que se ha detectado el monitor activo, la estación recién insertada enviará una trama *Duplicate Address Test* (DAT) colocando su propia dirección en el campo *DA*. Esta trama debe regresar con los bits ARI y FCI en cero, ya que si ocurre lo contrario implica que en el anillo hay otra estación con la misma dirección; en caso de que suceda esto, la estación que deseaba insertarse deberá retirarse del anillo. Si no, procede con la fase 3.

- **Fase 3, Ring Poll:** Durante esta fase la estación participa en el proceso de Ring Poll. Esto permite que la estación aprenda cuál es la dirección su *NAUN* y se identifique con su vecino inferior más cercano. Por lo tanto durante el proceso de inserción de una estación al anillo, ocurren dos cambios en el SUA, esto se hace por medio del envío de tramas de cambio de SUA.

- **Fase 4, Request Initialization:** En esta fase la estación deberá transmitir una trama *Request Initialization* a la dirección funcional de Ring Parameter Server (RPS). Si el RPS está presente, es decir si los

bits ARI y FCI son iguales a uno, la estación espera recibir una trama de inicialización de estación válida o una trama de cambio de parámetros. Estas tramas MAC pueden fijar parámetros en la estación tales como la localización física, valor del contador de tiempo de reporte de errores, o el número del anillo. En el caso de que el RPS no esté presente, la estación hará un total de cuatro intentos y simplemente continuará con su operación normal.

2.6.2 Proceso de eliminación de una estación en el anillo

Cuando una estación del anillo es apagada, físicamente desconectada o el adaptador recibe un comando de cierre vía software, el puerto del MAU al cual estaba conectado corta automáticamente la señal, tanto de entrada como de salida. El vecino inferior de esta estación detecta una interferencia eléctrica y reporta el error aproximadamente dos minutos después.

Luego el vecino inferior inicia un proceso de cambio del SUA enviando una trama SUA_CHG, para luego comenzar el proceso Ring Poll y determinar quién es su nuevo *NAUN*. A diferencia del proceso de inserción, en este proceso sólo una estación cambia su SUA.

2.7 Transmisión de datos de usuario

Los datos de usuario son llevados en las tramas LLC y los bits ARI y FCI son fijados cuando los usuarios envían tramas.

Después de que una estación tiene acceso a la red, siempre y cuando la prioridad la permita, puede proceder con la transmisión de tramas [07].

2.7.1 Transmisión de tramas en 4 Mbps

Antes que una estación pueda transmitir los datos, tiene que transmitir un token. Mientras que una estación ocupa un token en el anillo, ninguna otra estación puede transmitir cualquier dato al mismo tiempo. Eso garantiza que cada estación activa es capaz de transmitir los datos en los intervalos regulares.

Una estación que tiene datos a transmitir empieza por capturar el token actual para obtener el acceso a la red, la cual puede ocupar el token por el tiempo definido para ello (normalmente 10 ms). La estación entonces convierte el token en una trama de datos cambiando el campo AC y su propia dirección, la dirección destinataria y los datos mismos. La trama de datos es recibida y repetida por cada estación inferior activa. La estación receptora reconoce su dirección, copia los datos, configura el campo del estado de la trama (para indicar la recepción exitosa de la trama) y transmite la trama. La estación emisora tiene la responsabilidad de remover la trama de datos, y libera un token listo para que la próxima estación lo tome.

Cada bit que llega a una estación es copiado al buffer de 1 bit, y luego es copiado en el anillo otra vez. Mientras que está en el buffer, el bit es analizado y puede ser modificado antes de ser escrito otra vez. Este paso regenera el bit y produce un retardo de 1 bit [07].

2.7.2 Transmisión de tramas en 16 Mbps

Early Token Release (ETR): Aunque sólo un token es permitido, con ETR la estación emisora libera el token inmediatamente después de que ha transmitido su trama de datos. Esto permite a otra estación tomar el token antes de que la primera estación después de la estación emisora reciba su trama de datos de regreso con confirmación de que la trama ha sido copiada por la estación receptora. Mientras más grande sea el anillo, más tiempo se necesita para que la trama de datos lo atraviese y entonces más de una trama de datos puede existir en el mismo momento [10].

En Token Ring de 4 Mbps, el token se libera cuando una trama circula completamente el anillo, el Token Ring de 16 Mbps permite un sólo token siguiendo detrás de una trama, y partes de otras tramas pueden estar después del token en el mismo momento en el anillo. En el último caso, el uso de prioridad puede no estar disponible y se puede ver el retraso por el tamaño físico del anillo, número de estaciones activas y la tasa de los datos. Las estaciones ETR y no-ETR pueden coexistir en el mismo anillo sin ningún problema. El ETR esencialmente ofrece ahorros de una rotación del anillo transmitiendo una trama, y mejora la transmisión sobre el anillo.

2.8 Monitor Contention Process

Todas las estaciones son capaces de ser AM, pero solamente hay uno en un anillo. Todas las demás estaciones están en el modo SM. Un caso especial es que una estación puede configurarse a repetir tramas de solicitud de token, entonces nunca se convertirá en AM. Es decir que no todas las estaciones son requeridas para participar en el proceso de contención.

El establecimiento del AM usualmente se hace en la inicialización. Generalmente es la primera estación que acceda al anillo. Sin embargo, hay situaciones cuando un nuevo monitor activo es requerido. Este proceso de seleccionar un nuevo AM es conocido como *Monitor Contention* y puede ser inicializado si cualquiera de las siguientes condiciones ocurre:

1. Un monitor en espera no detecta el token o una trama de datos cada 2.6 s.

2. Una trama AMP (Active Monitor Present) no es detectada por un monitor en espera cada 15 s.

3. Ningún AMP, SMP (Standby Monitor Present) ó Ring Purge detectado dentro de 18 s. en la inserción.

4. El AM no pudo enviar su trama Ring Purge y por lo tanto no puede generar un nuevo token exitosamente dentro de 2.6 s.

5. Un monitor en espera detecta frecuencia del reloj fuera de la especificación ó degradación de señal.

6. Transmisor de Beacon recibe Beacon con MC encendido.

7. Pérdida de señal.

Monitor Contention usa un proceso conocido como *Claim Token* para determinar que estación va a ser AM [06].

2.8.1 Claim Token

El proceso de *Claim Token* es igual sin importar la condición que lo causó. Como los contadores de tiempo expiran, cada estación empieza a transmitir una trama *Claim Token* en el anillo.

Por ejemplo, la Figura 2.17 muestra las estaciones de una red local con el protocolo Token Ring. La estación 33, el Monitor Activo, salió de la red y su vecino inferior, la estación 4, emite una trama *Claim Token* con su propia dirección, en este caso es 4.

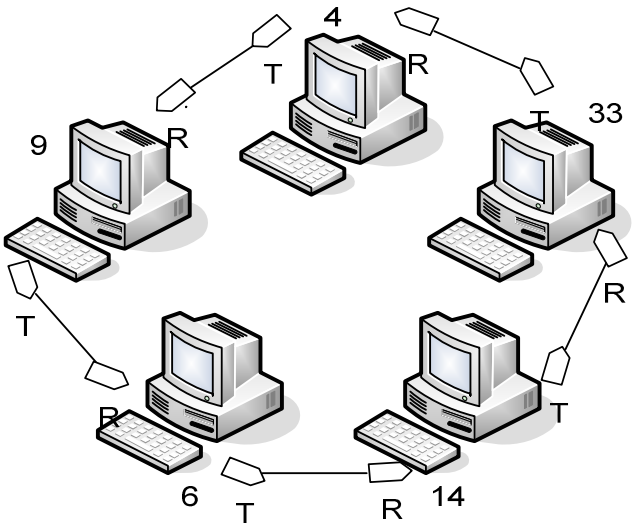

Figura 2.17: Diseño de Token Ring

Así, cada estación también recibe tramas *Claim Token* y compara la dirección fuente en la trama y la suya. Una estación que tiene su dirección menor a la de la fuente de la trama *Claim Token* numéricamente repite la trama y termina de mandar a sí mismo ya que nadie sustituye al suyo. Si su dirección es numéricamente mayor, genera su propia trama *Claim Token* y no repite la trama recibida.

Por eso, la estación con la dirección mayor eventualmente recibe el retorno de su propia *Claim Token*, en este caso es la estación 14, permitiendo enviar 3 veces tramas *Claim Token* sucesivas para asegurar la integridad del anillo. Entonces, la estación 14 se convierte en el nuevo AM.

Para asegurar que todas las estaciones reciben información sobre el estado del AM, una trama AMP es transmitida por el AM periódicamente. Cuando este contador expira en 7 s, una trama AMP es emitida por el AM. Si las estaciones no reciben una trama AMP por 18 s, entonces el proceso *Monitor Contention* es introducido. La trama AMP también permite a las estaciones recibir la dirección de su *NAUN*.

Si el anillo falla mientras un monitor activo está presente, este no participará en el proceso oferta y no asumirá el rol del monitor activo después de *Monitor Contention*. Si *Monitor Contention* no resuelve el problema dentro de 1 s (1000 ms), la primera estación al expirar el tiempo del proceso empieza a enviar tramas Beacon y el anillo entra al modo Beacon. Cuando el anillo se recupera de la condición de Beacon, regresa al modo *Monitor Contention* en el que un nuevo monitor activo es seleccionado.

3. Marco Metodológico

Con la finalidad de cumplir a cabalidad los objetivos planteados en el Capítulo 1, se debe seguir una metodología que permita un desarrollo eficiente y exitoso de la aplicación. En este capítulo se dan a conocer los detalles de la metodología en la cual se basó el desarrollo de la aplicación *Aprenda Token Ring*.

3.1 Metodología de Desarrollo

La mayoría de las metodologías tradicionales se basan en normas provenientes de estándares seguidos por el entorno de desarrollo, es por ello que muestran ciertas resistencias a los cambios, haciendo que el proceso de desarrollo sea más controlado y por ende más laborioso. En la actualidad se plantean proyectos donde se requieren que el resultado final sea un producto de gran calidad en el menor tiempo posible, por esta razón surgen las metodologías ágiles, las cuales se basan en el desarrollo incremental del software con iteraciones muy cortas, adaptándose de una mejor manera a los constantes cambios que se van presentando durante el desarrollo [11].

Un proceso de desarrollo de software iterativo es aquel que se representa por una serie de tareas agrupadas en pequeñas etapas repetitivas, conocidas como iteraciones. Para cada una de las etapas que conformen el desarrollo final de la aplicación, se define una serie de pasos para completarla, estos pasos son usados en los modelos de desarrollos comunes y se conocen como Análisis, Diseño, Codificación y Pruebas.

El desarrollo de la aplicación *Aprenda Token Ring* se basa en la metodología de desarrollo ágil, donde se definen una serie de iteraciones, en las cuales se cumplen cada uno de los pasos nombrados anteriormente y que se detallarán a continuación.

3.1.1 Análisis

Durante la fase de análisis se define la funcionalidad de la iteración o etapa, para ello hay que especificar quiénes son los actores o usuarios que interactúan con la aplicación, sumado a la actividad que ellos realizan.

La manera de mostrar la relación entre los usuarios y la aplicación es realizando los casos de uso, los cuales muestran cada uno de los pasos o actividades que el usuario puede realizar en cada módulo de la aplicación.

Durante la primera iteración se hace el análisis basado en una primera impresión de lo que debe realizar el módulo, mientras que las iteraciones siguientes están basadas en el resultado de las pruebas realizadas en las iteraciones pasadas.

3.1.2 Diseño

En el diseño preliminar se trata de establecer la arquitectura del programa. La arquitectura es un esquema donde se muestra en qué módulos se divide la aplicación, si la aplicación es cliente/servidor o por el contrario es *stand alone*.

Una vez definida la estructura se define cuáles clases y métodos se usan. Se observa la relación entre cada uno, de manera de poder definir que se puede reutilizar, es decir si hay un proceso que sea utilizado por más de un método.

Inicialmente cada módulo tiene un diseño de clases prototipo, sin embargo, este diseño puede variar dependiendo de los resultados de la iteración anterior, al igual que los resultados de la fase de pruebas.

3.1.3 Codificación

Luego de haber realizado el análisis de los casos de uso, las clases y métodos a utilizar, se procede a la implementación de los mismos. Para eso, se utiliza un lenguaje de programación que se adapte mejor a la solución que se quiere presentar.

3.1.4 Pruebas

Una vez realizada la codificación, se procede a realizar las pruebas correspondientes, las cuales determinan si la iteración ha finalizado, o por el contrario es necesario realizar nuevas implementaciones y por ende una nueva iteración.

Esta fase debe arrojar los resultados esperados a cada uno de los requerimientos planteados en la fase de análisis, para cada módulo y también la integración de cada uno de ellos, es decir se debe garantizar mediante estas pruebas que cada módulo funciona bien independientemente, al igual que en conjunto, formando una aplicación completa.

4. Marco Aplicativo

En el Capítulo 3 se describió la metodología a utilizar para la implementación de la aplicación, la cual consiste en una metodología iterativa, donde cada iteración muestra un módulo de la aplicación.

Con el fin de facilitar la comprensión de los resultados obtenidos durante cada iteración, estos son documentados por fases (Análisis, Diseño, Codificación y Pruebas), especificando en cada una de ellas el detalle de todas las iteraciones. A continuación, se describe el proceso práctico que se siguió a lo largo del desarrollo de la aplicación.

4.1 Fase de análisis

En esta fase se define la funcionalidad de la iteración o etapa, especificando la relación entre los usuarios y la aplicación mediante los casos de uso, de acuerdo a los pasos o actividades que el usuario puede realizar en cada módulo de la aplicación.

En la Figura 4.1 se muestra el nivel 0 de los casos de uso, el cual explica que un usuario quiere interactuar con el sistema llamado "*Aprenda Token Ring*". El usuario puede ser un estudiante, un profesor o alguien que quiere aprender la arquitectura de red Token Ring.

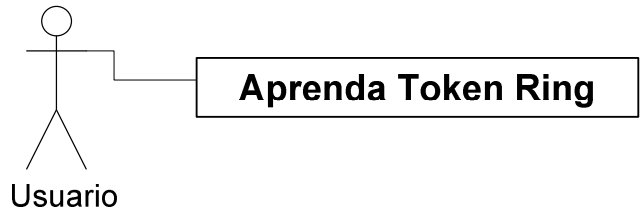

Figura 4.1: Nivel 0 de los Casos de uso

4.1.1 Iteración 1: General

En la primera iteración se hace el análisis global de lo que debe cumplir la aplicación, mientras que las iteraciones siguientes están basadas en el resultado de las pruebas realizadas en las iteraciones pasadas.

En el nivel 1 de los casos de uso mostrado en la Figura 4.2, se definen las principales funcionalidades o actividades del sistema.

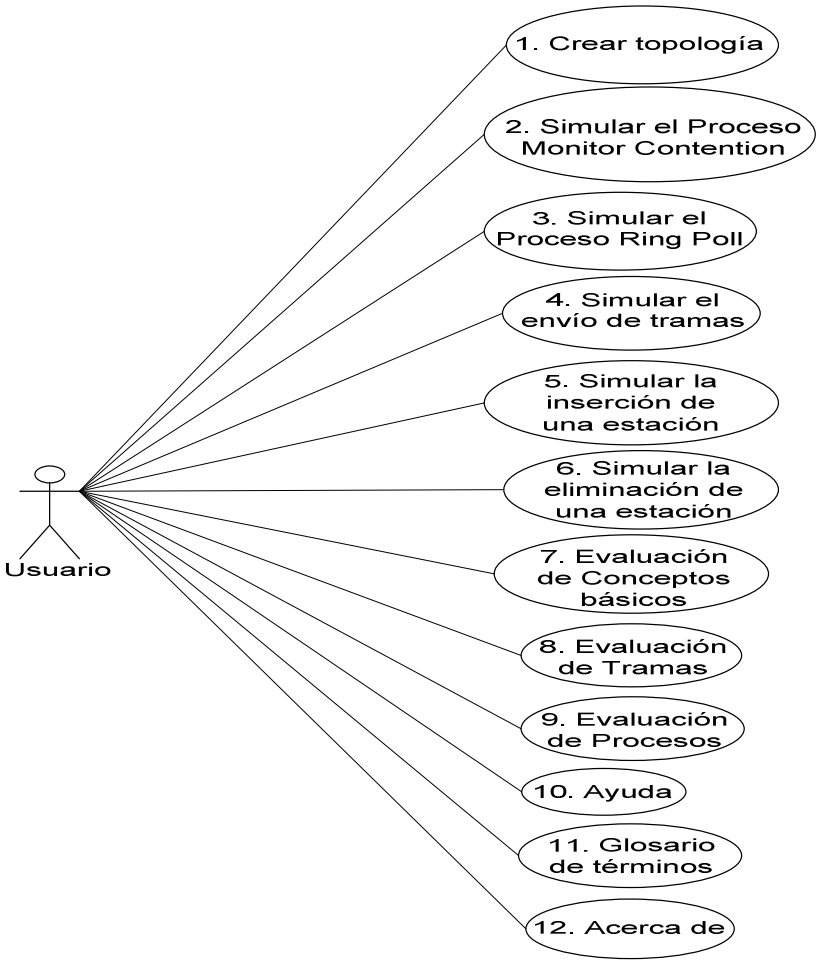

Figura 4.2: Nivel 1 de los Casos de uso

A continuación se describe las especificaciones de los casos de uso en este nivel:

Caso de uso	1. Crear Topología
Actores	Usuario
Descripción	Crear una red Token Ring especificando el número de estaciones para iniciar la simulación.
Flujo básico	• Seleccionar en la barra de menú o directamente en la barra de herramientas la opción Crear Topología • Escoger el número de estaciones para crear la red.

Tabla 4.1: Especificación de Caso de uso 1, Crear Topología

Caso de uso	2. Simular el Proceso Monitor Contention
Actores	Usuario
Descripción	Permite iniciar el Proceso Monitor Contention para seleccionar el nuevo monitor activo.
Flujo básico	• Seleccionar la Simulación del Proceso Monitor Contention. • Avanzar con la simulación hasta terminar el proceso.
Puntos de exclusión	• Retroceder el avance.

Tabla 4.2: Especificación de Caso de uso 2, Simular el Proceso Monitor Contention

Caso de uso	3. Simular el Proceso Ring Poll
Actores	Usuario
Descripción	Permite mantener actualizado el registro NAUN.
Flujo básico	• Seleccionar en la barra de menú la opción Simulación/Proceso Ring Poll. • Avanzar con la simulación hasta terminar el proceso.
Puntos de exclusión	• Retroceder el avance.

Tabla 4.3: Especificación de Caso de uso 3, Simular el Proceso Ring Poll

Caso de uso	4. Simular el envío de tramas
Actores	Usuario
Descripción	Permite seleccionar hasta 3 orígenes, destinos y prioridades para cada envío.
Flujo básico	• Seleccionar la Simulación de Enviar Trama. • Seleccionar uno, dos o tres orígenes, destinos y prioridades. • Avanzar con la simulación hasta terminar el proceso.
Puntos de exclusión	• Retroceder el avance. • Enviar otra trama durante la simulación

Tabla 4.4: Especificación de Caso de uso 4, Simular el envío de tramas

Caso de uso	5. Simular la inserción de una estación
Actores	Usuario
Descripción	Permite insertar una nueva estación al anillo.
Flujo básico	• Seleccionar la opción Agregar estación. • Elegir entre cuales estaciones desea insertar la nueva estación. • Avanzar con la simulación hasta terminar el proceso.
Puntos de exclusión	• Retroceder el avance.

Tabla 4.5: Especificación de Caso de uso 5, Simular la inserción de una estación

Caso de uso	6. Simular la eliminación de una estación
Actores	Usuario
Descripción	Permite eliminar una estación en espera.
Flujo básico	• Seleccionar la opción Eliminar estación • Escoger la estación en espera a ser eliminada • Avanzar con la simulación hasta terminar el proceso.
Puntos de exclusión	• Retroceder el avance.

Tabla 4.6: Especificación de Caso de uso 6, Simular la eliminación de una estación

Caso de uso	7. Evaluación de Conceptos básicos
Actores	Usuario
Descripción	Permite al usuario evaluar los conocimientos adquiridos sobre conceptos básicos de Token Ring.
Flujo básico	• Seleccionar la opción Evaluación/Conceptos Básicos. • Seleccionar una respuesta y revisarla. • Seguir las preguntas hasta que se da el resultado.
Puntos de inclusión	• Revisar la respuesta y avanzar las preguntas.

Tabla 4.7: Especificación de Caso de uso 7, Evaluación de Conceptos básicos

Caso de uso	8. Evaluación de Tramas
Actores	Usuario
Descripción	Permite al usuario evaluar los conocimientos adquiridos sobre tramas.
Flujo básico	• Seleccionar la opción Evaluación/Tramas. • Seleccionar una respuesta y revisarla. • Seguir las preguntas hasta que se da el resultado.
Puntos de inclusión	• Revisar la respuesta y avanzar las preguntas.

Tabla 4.8: Especificación de Caso de uso 8, Evaluación de Tramas

Caso de uso	9. Evaluación de Procesos
Actores	Usuario
Descripción	Permite al usuario evaluar los conocimientos adquiridos sobre los procesos.
Flujo básico	• Seleccionar la opción Evaluación/Procesos. • Seleccionar una respuesta y revisarla. • Seguir las preguntas hasta que se da el resultado.
Puntos de inclusión	• Revisar la respuesta y avanzar las preguntas.

Tabla 4.9: Especificación de Caso de uso 9, Evaluación de Procesos

Caso de uso	10. Ayuda
Actores	Usuario
Descripción	Orientar al usuario a usar la aplicación.
Flujo básico	• Seleccionar en la barra de menú la opción Ayuda/Ayuda • Seleccionar un tema y seguir avanzando con la explicación.
Puntos de exclusión	• Retroceder la explicación.

Tabla 4.10: Especificación de Caso de uso 10, Ayuda

Caso de uso	11. Glosario de términos
Actores	Usuario
Descripción	Permite conocer el significado de los términos sobre Token Ring.
Flujo básico	• Seleccionar en la barra de menú la opción Ayuda/Glosario • Seleccionar un término a conocer.
Puntos de inclusión	• Seleccionar un término.

Tabla 4.11: Especificación de Caso de uso 11, Glosario de términos

Caso de uso	12. Acerca de
Actores	Usuario
Descripción	Permite conocer los autores y versión de la aplicación.
Flujo básico	• Seleccionar en la barra de menú la opción Ayuda/Acerca de.

Tabla 4.12: Especificación de Caso de uso 12, Acerca de

En las siguientes iteraciones se describe el análisis de cada caso de uso en el nivel 2.

4.1.2 Iteración 2: Crear Topología

En la Figura 4.3 se muestra el detalle del caso de uso *Crear Topología* en el nivel 2 de abstracción, y en la Tabla 4.13 y Tabla 4.14 se muestran las especificaciones correspondientes de cada uno de ellos.

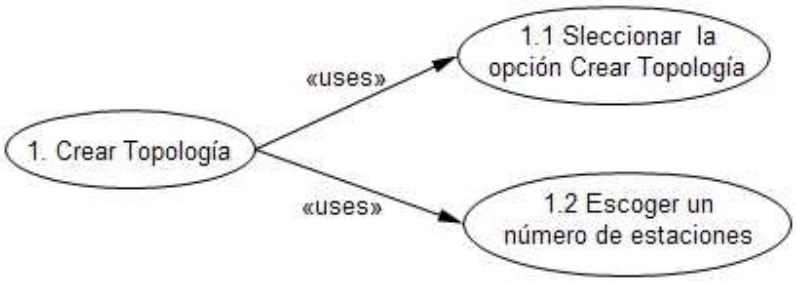

Figura 4.3: Nivel 2 del Caso de uso 1, Crear Topología

Caso de uso	1.1 Seleccionar la opción Crear Topología
Actores	Usuario
Descripción	Permite crear una red Token Ring.
Flujo básico	• Seleccionar en la barra de menú la opción Archivo/Nuevo o desde la barra de herramientas el icono Crear Topología.

Tabla 4.13: Especificación de Caso de uso 1.1

Caso de uso	1.2 Escoger un número de estaciones
Actores	Usuario
Descripción	Permite seleccionar la cantidad de estaciones a crear en la red.
Flujo básico	• Seleccionar un número entre 2 y 12 y presionar el botón Aceptar.

Tabla 4.14: Especificación de Caso de uso 1.2

4.1.3 Iteración 3: Simular el Proceso Monitor Contention

En la Figura 4.4 se muestra el nivel 2 del caso de uso *el Proceso Monitor Contention* y sus especificaciones en la Tabla 4.15, Tabla 4.16 y Tabla 4.17.

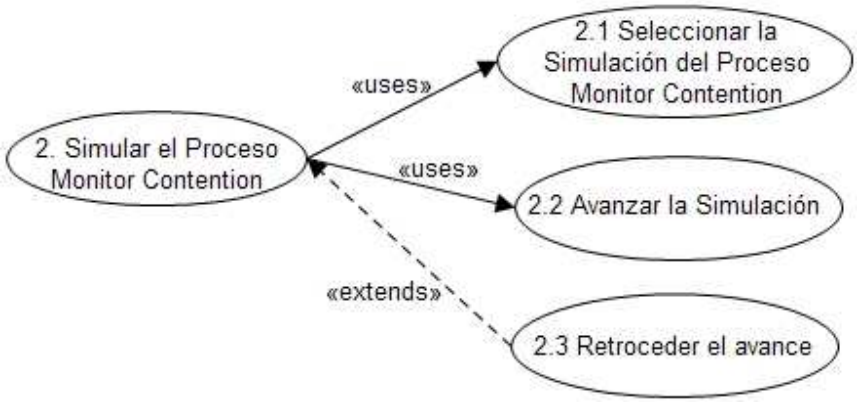

Figura 4.4: Nivel 2 del Caso de uso 2, Simular el Proceso Monitor Contention

Caso de uso	2.1 Seleccionar la Simulación del Proceso Monitor Contention
Actores	Usuario
Descripción	Permite iniciar el Proceso Monitor Contention.
Flujo básico	• Seleccionar en la barra de menú la opción Simulaciones / Proceso Monitor Contention.

Tabla 4.15: Especificación del Caso de uso 2.1

Caso de uso	2.2 Avanzar la Simulación
Actores	Usuario
Descripción	Permite avanzar la simulación.
Flujo básico	• Presionar en el botón Siguiente para avanzar la simulación hasta que se termine con un mensaje del fin de la simulación.

Tabla 4.16: Especificación del Caso de uso 2.2

Caso de uso	2.3 Retroceder el avance
Actores	Usuario
Descripción	Permite retroceder un paso hacia atrás.
Flujo básico	• Presionar en el botón Atrás para visualizar el paso anterior.

Tabla 4.17: Especificación del Caso de uso 2.3

4.1.4 Iteración 4: Simular el Proceso Ring Poll

En la Figura 4.5 se muestra el nivel 2 del caso de uso *Simular el Proceso Ring Poll* y en la Tabla 4.18, Tabla 4.19 y Tabla 4.20 se especifican los detalles de cada uno.

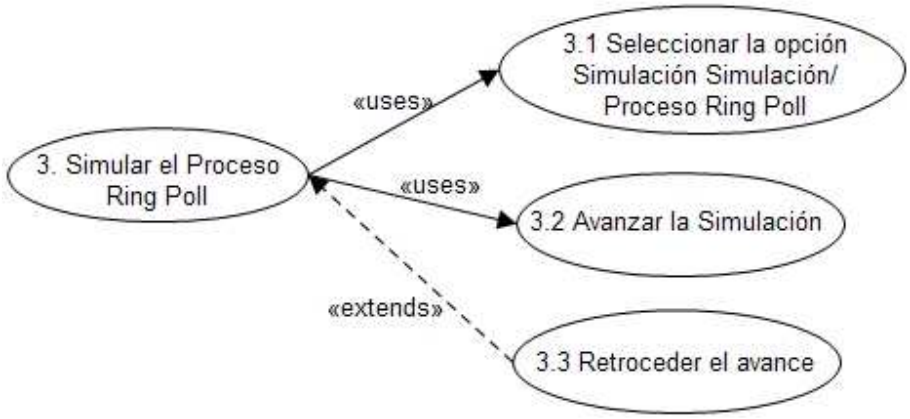

Figura 4.5: Nivel 2 de Caso de uso 3, Simular el Proceso Ring Poll

Caso de uso	3.1 Seleccionar la opción Simulaciones/Proceso Ring Poll
Actores	Usuario
Descripción	Permite iniciar la Simulación del Proceso Ring Poll.
Flujo básico	• Seleccionar en la barra de menú la opción Simulaciones / Proceso Ring Poll.

Tabla 4.18: Especificación del Caso de uso 3.1

Caso de uso	3.2 Avanzar la simulación
Actores	Usuario
Descripción	Permite avanzar la simulación hasta terminar el proceso.
Flujo básico	• Presionar el botón Siguiente para avanzar la simulación hasta que termine mostrando un mensaje del fin de la simulación.

Tabla 4.19: Especificación del Caso de uso 3.2

Caso de uso	3.3 Retroceder el avance
Actores	Usuario
Descripción	Permite regresar al paso anterior.
Flujo básico	• Presionar el botón Atrás para visualizar el paso anterior.

Tabla 4.20: Especificación del Caso de uso 3.3

4.1.5 Iteración 5: Simular el envío de tramas

En la Figura 4.6 se muestra el nivel 2 del caso de uso *Simular el envío de tramas* y sus especificaciones en la Tabla 4.21, Tabla 4.22, Tabla 4.23 y Tabla 4.24 respectivamente.

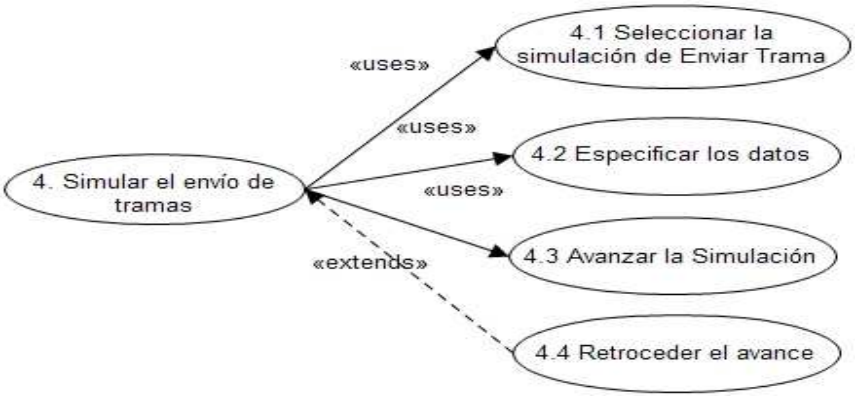

Figura 4.6: Nivel 2 de Caso de uso 4, Simular el envío de tramas

Caso de uso	4.1 Seleccionar la Simulación de Enviar Trama
Actores	Usuario
Descripción	Permite iniciar la simulación de Enviar Trama.
Flujo básico	• Seleccionar en la barra de menú la opción Simulaciones / Enviar Trama o desde la barra de herramientas el icono Enviar Trama.

Tabla 4.21: Especificación del Caso de uso 4.1

Caso de uso	4.2 Especificar los datos
Actores	Usuario
Descripción	Permite especificar hasta 3 envíos de trama.
Flujo básico	• Seleccionar el origen, destino y la prioridad de cada envío y luego presionar el botón Aceptar para iniciar la simulación.

Tabla 4.22: Especificación del Caso de uso 4.2

Caso de uso	4.3 Avanzar la simulación
Actores	Usuario
Descripción	Permite avanzar la simulación.
Flujo básico	• Presionar el botón Siguiente para avanzar la simulación hasta que se muestre el mensaje del fin de simulación.

Tabla 4.23: Especificación del Caso de uso 4.3

Caso de uso	4.4 Retroceder el avance
Actores	Usuario
Descripción	Permite regresar al paso anterior de la simulación.
Flujo básico	• Presionar el botón Atrás.

Tabla 4.24: Especificación del Caso de uso 4.4

4.1.6 Iteración 6: Simular la inserción de una estación

En la Figura 2.1 se muestra el nivel 2 del caso de uso *Simular la inserción de una estación* y sus especificaciones en la Tabla 4.25, Tabla 4.26, Tabla 4.27 y Tabla 4.28 respectivamente.

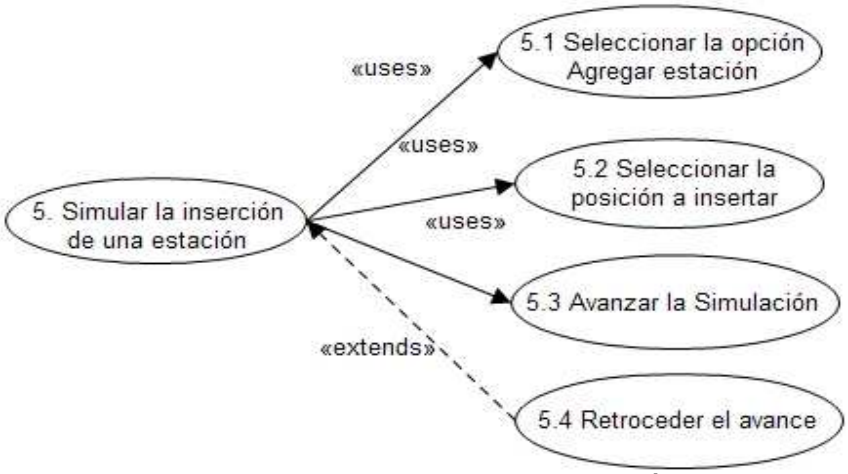

Figura 4.7: Nivel 2 del Caso de uso 5, Simular la inserción de una estación

Caso de uso	5.1 Seleccionar la opción Agregar estación
Actores	Usuario
Descripción	Permite agregar una estación en la red.
Flujo básico	• Seleccionar en la barra de menú la opción Simulaciones / Agregar estación o desde la barra de herramientas el icono Agregar Estación.

Tabla 4.25: Especificación del Caso de uso 5.1

Caso de uso	5.2 Seleccionar la posición a insertar
Actores	Usuario
Descripción	Permite seleccionar la posición donde insertar la nueva estación.
Flujo básico	• Seleccionar las 2 estaciones vecinas para insertar la nueva estación y presionar el botón Aceptar para iniciar el proceso.

Tabla 4.26: Especificación del Caso de uso 5.2

Caso de uso	5.3 Avanzar la simulación
Actores	Usuario
Descripción	Permite avanzar la simulación.
Flujo básico	• Presionar el botón Siguiente para avanzar la simulación hasta que termine el proceso con un mensaje del fin de simulación.

Tabla 4.27: Especificación del Caso de uso 5.3

Caso de uso	5.4 Retroceder el avance
Actores	Usuario
Descripción	Permite regresar al paso anterior.
Flujo básico	• Presionar en el botón Atrás para regresar al paso anterior.

Tabla 4.28: Especificación del Caso de uso 5.4

4.1.7 Iteración 7: Simular la eliminación de una estación

En la Figura 4.8 se muestra el caso de uso *Simular la eliminación de una estación* y sus especificaciones respectivamente en la Tabla 4.29, Tabla 4.30, Tabla 4.31 y Tabla 4.32.

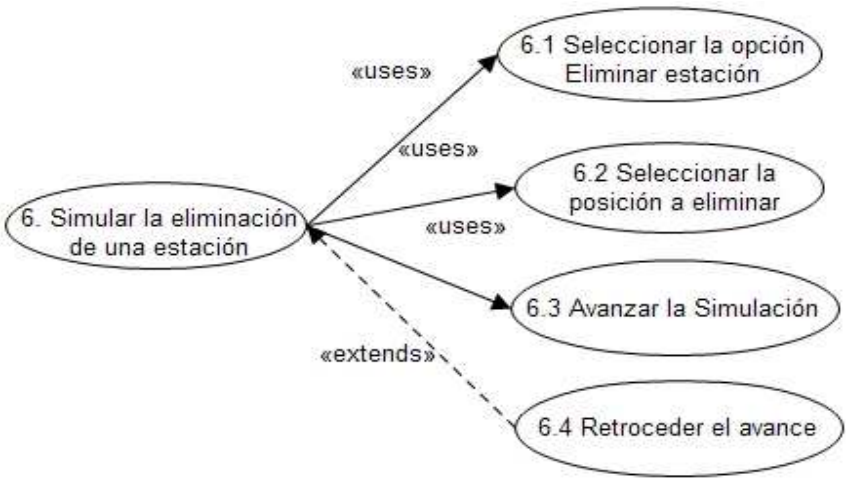

Figura 4.8: Nivel 2 del Caso de uso 6, Simular la eliminación de una estación

Caso de uso	6.1 Seleccionar la opción Eliminar estación
Actores	Usuario
Descripción	Permite eliminar una estación de la red.
Flujo básico	• Seleccionar en la barra de menú la opción Simulaciones / Eliminar estación o desde la barra de herramientas el icono Eliminar Estación.

Tabla 4.29: Especificación del Caso de uso 6.1

Caso de uso	6.2 Seleccionar la estación a eliminar
Actores	Usuario
Descripción	Permite seleccionar la estación a eliminar.
Flujo básico	• Seleccionar una estación en espera de la lista y presionar el botón Aceptar para iniciar la simulación.

Tabla 4.30: Especificación del Caso de uso 6.2

Caso de uso	6.3 Avanzar la simulación
Actores	Usuario
Descripción	Permite avanzar la simulación hasta que termine la simulación.
Flujo básico	• Presionar el botón Siguiente para avanzar la simulación hasta que muestre el mensaje del fin de simulación.

Tabla 4.31: Especificación del Caso de uso 6.3

Caso de uso	6.4 Retroceder el avance
Actores	Usuario
Descripción	Permite regresar al paso anterior de la simulación.
Flujo básico	• Presionar el botón Atrás.

Tabla 4.32: Especificación del Caso de uso 6.4

4.1.8 Iteración 8: Evaluación de Conceptos Básicos

En la Figura 4.9 se muestra el nivel 2 del caso de uso *Evaluación de Conceptos Básicos* y sus especificaciones en la Tabla 4.33, Tabla 4.34 y Tabla 4.35, respectivamente.

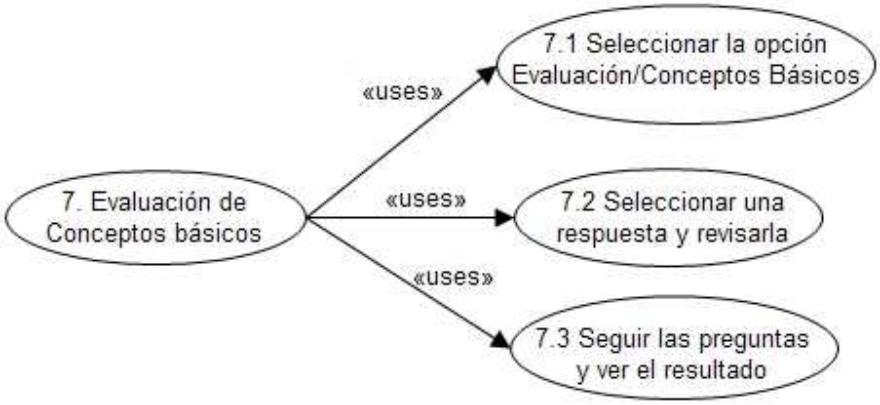

Figura 4.9: Nivel 2 del Caso de uso 7, Evaluación de Conceptos básicos

Caso de uso	7.1 Seleccionar la opción Evaluación/Conceptos Básicos
Actores	Usuario
Descripción	Permite iniciar la evaluación sobre conceptos básicos de Token Ring.
Flujo básico	• Seleccionar en la barra de menú la opción Evaluación / Conceptos Básicos.

Tabla 4.33: Especificación del Caso de uso 7.1

Caso de uso	7.2 Seleccionar una respuesta y revisarla
Actores	Usuario
Descripción	Permite revisar la respuesta en cada pregunta y obtener 2 puntos por cada respuesta correcta.
Flujo básico	• Seleccionar una respuesta y presionar el botón Revisar.

Tabla 4.34: Especificación del Caso de uso 7.2

Caso de uso	7.3 Seguir las preguntas y ver el resultado
Actores	Usuario
Descripción	Permite seguir la evaluación y ver el resultado al final.
Flujo básico	• Presionar el botón Siguiente para avanzar la evaluación hasta que termine la evaluación y muestre el resultado.

Tabla 4.35: Especificación del Caso de uso 7.3

4.1.9 Iteración 9: Evaluación de Tramas

En la Figura 4.10 se muestra el caso de uso *Evaluación de Tramas* y sus especificaciones en la Tabla 4.36, Tabla 4.37 y Tabla 4.38, respectivamente.

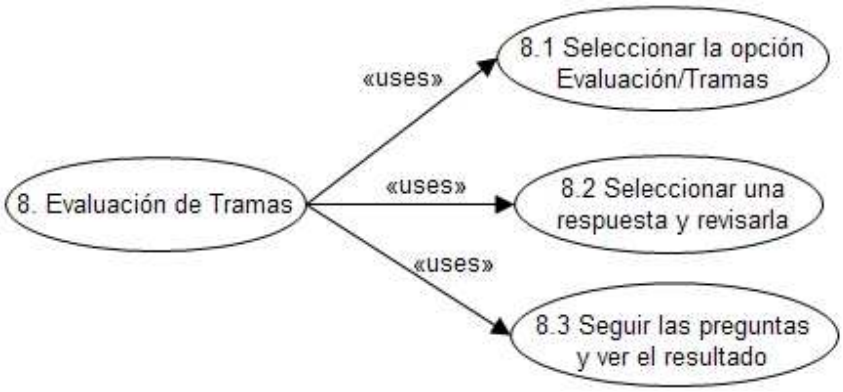

Figura 4.10: Nivel 2 del Caso de uso 8, Evaluación de Tramas

Caso de uso	8.1 Seleccionar la opción Evaluación/Tramas
Actores	Usuario
Descripción	Permite iniciar la evaluación sobre tramas de Token Ring.
Flujo básico	• Seleccionar en la barra de menú la opción Evaluación / Tramas.

Tabla 4.36: Especificación del Caso de uso 8.1

Caso de uso	8.2 Seleccionar una respuesta y revisarla
Actores	Usuario
Descripción	Permite revisar la respuesta en cada pregunta y obtener 2 puntos por cada respuesta correcta.
Flujo básico	• Seleccionar una respuesta y presionar en el botón Revisar.

Tabla 4.37: Especificación del Caso de uso 8.2

Caso de uso	8.3 Seguir las preguntas y ver el resultado
Actores	Usuario
Descripción	Permite avanzar la evaluación hasta que se muestre el resultado de la evaluación.
Flujo básico	• Presionar el botón Siguiente hasta mostrar el resultado de la evaluación.

Tabla 4.38: Especificación del Caso de uso 8.3

4.1.10 Iteración 10: Evaluación de Procesos

En la Figura 4.11 se muestra el nivel 2 del caso de uso *Evaluación de Procesos* y sus especificaciones en la Tabla 4.39, Tabla 4.40 y Tabla 4.41, respectivamente.

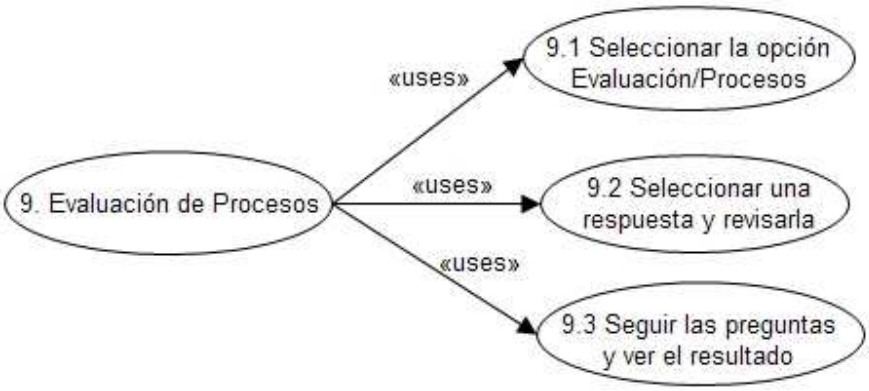

Figura 4.11: Nivel del Caso de uso 9, Evaluación de Procesos

Caso de uso	9.1 Seleccionar la opción Evaluación/Procesos
Actores	Usuario
Descripción	Permite iniciar la evaluación sobre Procesos de Token Ring.
Flujo básico	• Seleccionar en la barra de menú la opción Evaluación / Procesos.

Tabla 4.39: Especificación del Caso de uso 9.1

Caso de uso	9.2 Seleccionar una respuesta y revisarla
Actores	Usuario
Descripción	Permite revisar la respuesta en cada pregunta y obtener 2 puntos por cada respuesta correcta.
Flujo básico	• Seleccionar una respuesta y presionar el botón Revisar.

Tabla 4.40: Especificación del Caso de uso 9.2

Caso de uso	9.3 Seguir las preguntas y ver el resultado
Actores	Usuario
Descripción	Permite seguir la evaluación hasta que termine mostrando el resultado.
Flujo básico	• Presionar el botón Siguiente hasta que se muestre el resultado de la evaluación.

Tabla 4.41: Especificación del Caso de uso 9.3

4.1.11 Iteración 11: Ayuda

En la Figura 4.12 se muestra el caso de uso *Ayuda* y sus especificaciones en la Tabla 4.42, Tabla 4.43 y Tabla 4.44 respectivamente.

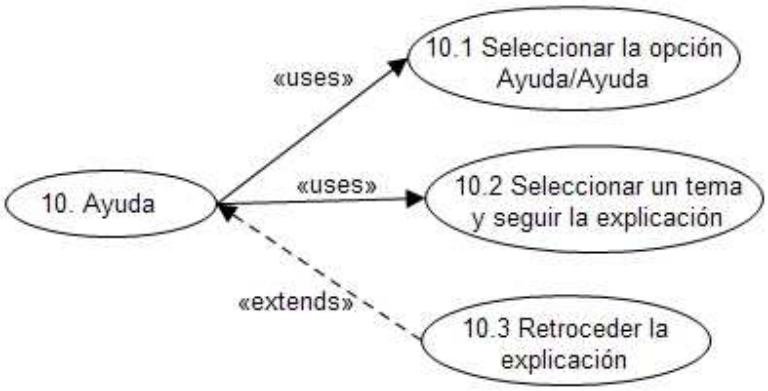

Figura 4.12: Nivel 2 del Caso de uso 10, Ayuda

Caso de uso	10.1 Seleccionar la opción Ayuda/Ayuda
Actores	Usuario
Descripción	Permite abrir la ayuda.
Flujo básico	• Seleccionar en la barra de menú la opción Ayuda/Ayuda.

Tabla 4.42: Especificación del Caso de uso 10.1

Caso de uso	10.2 Seleccionar un tema y seguir la explicación
Actores	Usuario
Descripción	Permite ver la explicación de cada tema.
Flujo básico	• Seleccionar un tema a conocer y presionar el botón Siguiente para seguir la explicación.

Tabla 4.43: Especificación del Caso de uso 10.2

Caso de uso	10.3 Retroceder la explicación
Actores	Usuario
Descripción	Permite regresar a la página anterior de la explicación del mismo tema.
Flujo básico	• Presionar el botón Atrás.

Tabla 4.44: Especificación del Caso de uso 10.3

4.1.12 Iteración 12: Glosario de términos

En la Figura 4.13 se muestra el nivel 2 del caso de uso *Glosario de términos* y sus especificaciones en la Tabla 4.45 y Tabla 4.46 respectivamente.

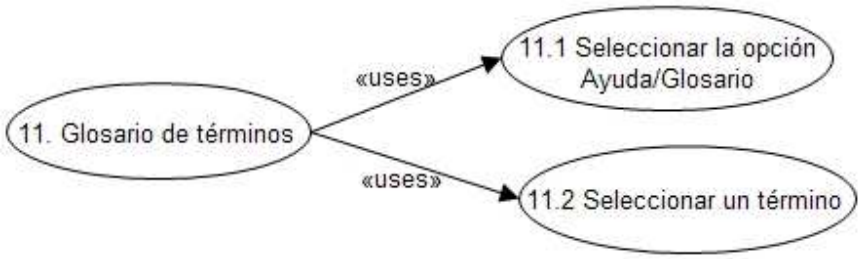

Figura 4.13: Nivel 2 del Caso de uso 11, Glosario de términos

Caso de uso	11.1 Seleccionar la opción Ayuda/Glosario
Actores	Usuario
Descripción	Permite abrir el glosario.
Flujo básico	• Seleccionar en la barra de menú la opción Ayuda/Glosario.

Tabla 4.45: Especificación del Caso de uso 11.1

Caso de uso	11.2 Seleccionar un término
Actores	Usuario
Descripción	Permite conocer el significado de los términos sobre Token Ring.
Flujo básico	• Seleccionar un término y leer su explicación.

Tabla 4.46: Especificación del Caso de uso 11.2

4.1.13 Iteración 13: Acerca de

En la Figura 4.14 se muestra el nivel 2 del caso de uso *Acerca de* y su especificación en la Tabla 4.47

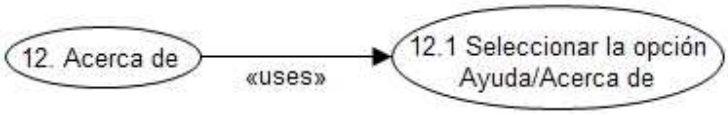

Figura 4.14: Nivel 2 del Caso de uso 12, Acerca de

Caso de uso	12.1 Seleccionar la opción Ayuda/Acerca de
Actores	Usuario
Descripción	Permite conocer la información sobre la aplicación.
Flujo básico	• Seleccionar en la barra de menú la opción Ayuda/Acerca de.

Tabla 4.47: Especificación del Caso de uso 12.1

4.2 Fase de Diseño

Luego de finalizar la fase de análisis, la cual mediante la utilización de diagramas de casos de usos representa cada una de las funcionalidades del sistema, se procede a diseñar la estructura que mejor se adapte a las necesidades planteadas.

Debido a que la aplicación fue implementada en Flash CS3, utilizando el lenguaje Action Script 2.0, la estructura general del programa se basa en un escenario que contiene una serie de clips de películas, estos clips son considerados como clases en nuestro sistema, ya que los mismos están compuestos por atributos, métodos predefinidos y métodos definidos durante el desarrollo para la obtención de los resultados esperados. Según lo expuesto, la estructura de la aplicación se muestra en la Figura 4.15:

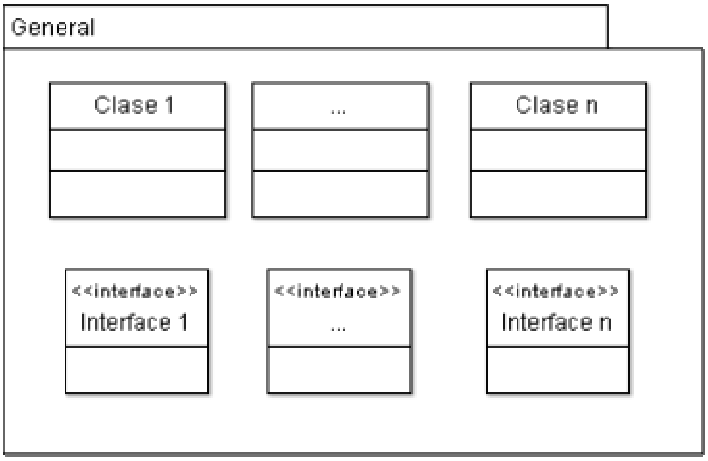

Figura 4.15: Diagrama de Clase General

Como se puede ver en la estructura, la iteración General corresponde al escenario, para el resto de las iteraciones se usaran interfaces y clases. A continuación se detallarán cada una de las iteraciones mostrando el diagrama de clases asociado a cada una, de manera de simplificar la documentación.

4.2.1 Iteración 1: General

En esta primera iteración se desarrolla la estructura general de la aplicación y se definen todas las interfaces necesarias para acceder a cada uno de los módulos que conforman la aplicación.

Dentro del escenario se crea la clase principal, la cual contiene los atributos y métodos más comunes de cada módulo. Allí se definen los botones y menús que

le permiten al usuario acceder a cada una de las funcionales. La Figura 4.16 muestra el diagrama representativo de esta clase.

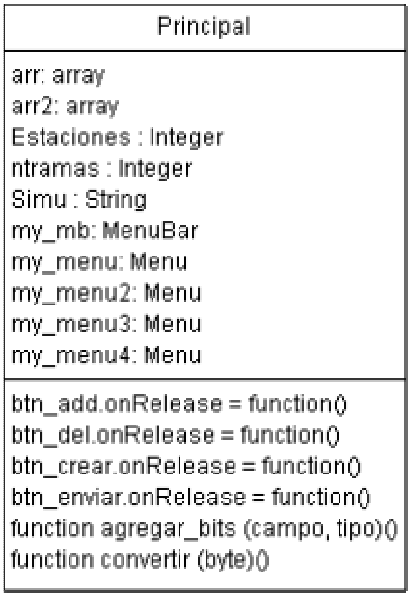

Figura 4.16: Diagrama de Clase Principal

Descripción de atributos:

- Mediante los arreglos **arr** y **arr2** se maneja la información de cada una de las estaciones, **arr** contiene el detalle (nombre, MAC, NAUN y tipo) mientras que **arr2** es un arreglo de enteros que contiene el orden en que se encuentran las estaciones en un momento determinado.
- **Estaciones:** es un entero que guarda la cantidad de estaciones presente en la topología utilizada.
- **ntramas:** corresponde a la cantidad de tramas asociada a una simulación.
- **Simu:** indica la simulación que esta siendo utilizada, en caso de que no se haya iniciado ninguna simulación, esta variable se encuentra en blanco.
- **my_mb:** corresponde a la barra de menú que contiene *my_menu* (correspondiente al menú archivo), *my_menu2* (correspondiente al menú simulaciones), *my_menu3* (correspondiente al menú evaluación) y *my_menu4* (correspondiente al menú ayuda).

Descripción de los métodos:

- **btn_add.onRelease = function ()**: Invoca a la interfaz *Vent_agregar_estación*
- **btn_del.onRelease = function ()**: Invoca a la interfaz *Vent_eliminar_estación*
- **btn_crear.onRelease = function ()**: Invoca a la interfaz *Vent_N_estaciones*

- **btn_enviar.onRelease = function ()**: Invoca a la interfaz *Vent_enviar_tramas*
- **function agregar_bits (campo, tipo)**: Agrega a la ventana de formato trama, los campos correspondientes al tipo de trama
- **function Convertir (byte)**: Convierte un valor de byte a bits

4.2.2 Iteración 2: Crear Topología

Esta iteración corresponde a la segunda más importante de la aplicación, ya que para iniciar cualquiera de las simulaciones es necesario crear una topología. En ella se le solicita al usuario la cantidad de estaciones que desea agregar, que van desde 2 hasta 12 estaciones.

El diagrama de clases asociado a esta iteración se muestra en la Figura 4.17:

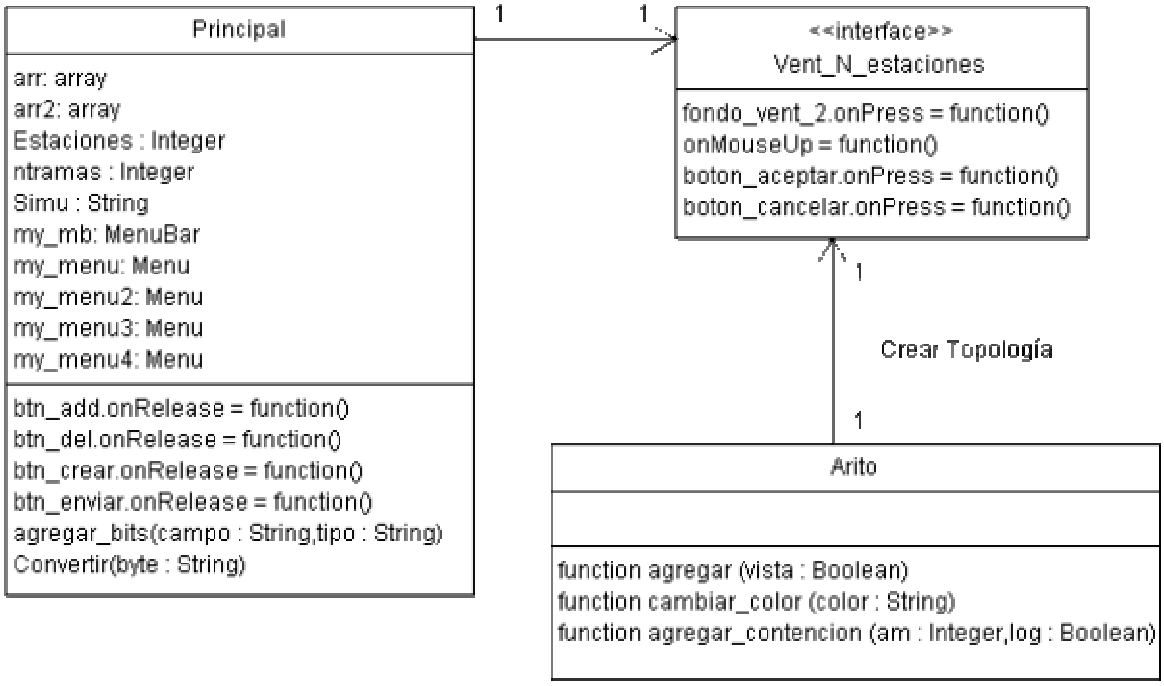

Figura 4.17: Diagrama de Clase Crear Topología

Descripción de los métodos de la interfaz *Vent_N_estaciones*:

- **fondo_vent_2.onPress = function ()**: Permite mover la ventana a través del escenario (Drag).
- **OnMouseUp = function ()**: Detiene la ventana una vez que el usuario suelta el botón derecho del Mouse (Drop).
- **boton_aceptar.onPress = function ()**: Invoca a la función agregar (vista) de la clase *Arito*.
- **boton_cancelar.onPress = function ()**: Cierra la ventana.

Descripción de los métodos de la clase *Arito*:

- **function agregar (vista: Boolean)**: Crea el anillo con la cantidad de estaciones seleccionada por el usuario, si *vista* es verdadero mostrará la flecha que guía las tramas, si es falso no las mostrará.
- **function cambiar_color (color: String)**: Cambia el color de la flecha guía, si es un token la coloca en rojo, si es una trama la coloca en azul.
- **function agregar_contencion (am: Integer, log: Boolean)**: Crea una topología sin monitor activo inicialmente, y luego de haber elegido el monitor activo, es invocada nuevamente con el *log* igual a verdadero y el número *am* identifica a la estación que le corresponde ser el nuevo monitor activo.

4.2.3 Iteración 3: Simular el proceso Monitor Contention

Esta iteración corresponde al grupo de simulaciones, todas las iteraciones que pertenecen a este grupo tendrá como diagrama base el que se muestra en la Figura 4.18. En este interactúan las clases *Principal* y *Arito*, ya explicadas anteriormente, la clase *Formato_trama*, la cual se encarga de mostrar durante toda la simulación el formato asociado a cada trama o token y la interfaz *CampoAC_bits*, donde se muestra el detalle de los bytes SD, AC, FC, FS y ED, dependiendo del caso seleccionado por el usuario.

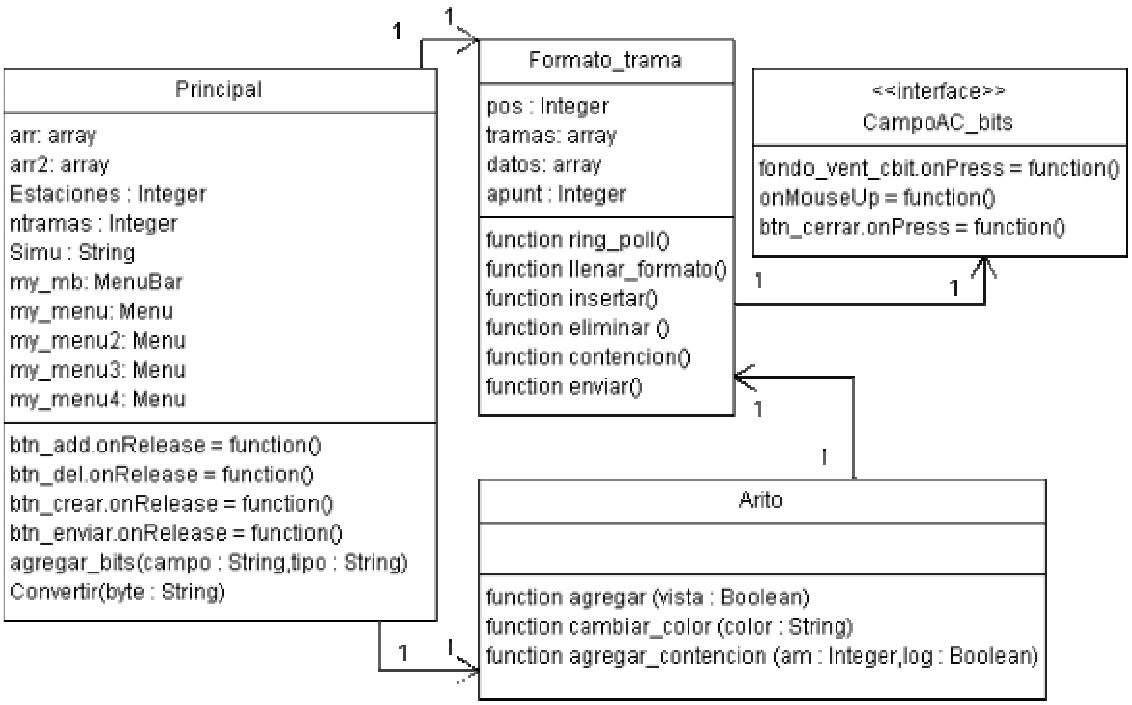

Figura 4.18: Diagrama de Clase Base para Simulaciones

Descripción de los métodos de la interfaz *CampoAC_bits*:

- **fondo_vent_cbit.onPress = function ()**: Permite mover la ventana a través del escenario (Drag).
- **OnMouseUp = function ()**: Detiene la ventana una vez que el usuario suelta el botón derecho del mouse (Drop).
- **btn_cerrar.onPress = function ()**: Cierra la ventana.

Descripción de atributos de la clase *Formato_trama*:

- **pos:** es un entero que cumple la función de un apuntador, ya que lleva la posición que se está utilizando del arreglo *arr2* perteneciente a la clase principal.
- **tramas:** es un arreglo de String, que contiene el nombre de todas las tramas utilizadas dentro de cada simulación.
- **datos:** es un arreglo que contiene la información de todas las tramas enviadas en una simulación, está conformado por los siguientes campos:
    - Origen: String que contiene la dirección MAC de la estación que origina la trama.
    - Destino: String que contiene la dirección MAC de la estación a quien va destinada la trama.
    - Tipo de trama: String que identifica el tipo de trama a enviar, obtenido del arreglo Tramas.
    - Prioridad: Entero que guarda la prioridad de la trama.
    - Reservación: Entero que guarda la reservación de prioridad de la trama.
    - MC: Boolean que indica si el bit MC debe estar 1 o en 0.
    - ColorAC: String que contiene un valor hexadecimal que corresponde al color a asignar en el campo AC durante la transmisión de la trama.
    - ColorFS: String que contiene un valor hexadecimal que corresponde al color a asignar en el campo FS durante la transmisión de la trama.
    - Color MC: String que contiene un valor hexadecimal que corresponde al color a asignar en el byte MC durante la transmisión de la trama.
    - ColorPrior: String que contiene un valor hexadecimal que corresponde al color a asignar en el campo Prioridad durante la transmisión de la trama.
    - ColorReser: String que contiene un valor hexadecimal que corresponde al color a asignar en el campo Reservación de Prioridad durante la transmisión de la trama.
    - Desc: Texto que describe la transmisión de la trama, que se muestra en la parte superior derecha de la ventana durante la simulación

- **apunt:** entero que cumple la función de apuntador del arreglo *datos*.

Descripción de los métodos de la clase *Formato_trama*:

- **function ring_poll ()**: Se encarga de llenar el arreglo *datos*, con todas las tramas correspondientes a la simulación Ring Poll.
- **function llenar_formato ()**: Toma los datos contenidos en el registro ubicado en la posición **apunt** del arreglo *datos*, y los muestra en la ventana de formato trama.
- **function insertar ()**: Se encarga de llenar el arreglo *datos*, con todas las tramas correspondientes a la simulación *Insertar una estación*, dependiendo de la posición que fue suministrada por el usuario.
- **function eliminar ()**: Se encarga de llenar el arreglo *datos*, con todas las tramas correspondientes a la simulación *Eliminar una estación*, dependiendo de la estación seleccionada por el usuario.
- **function contencion ()**: Se encarga de llenar el arreglo *datos*, con todas las tramas correspondientes a la simulación *Proceso de Monitor Contention*.
- **function enviar ()**: Se encarga de llenar el arreglo *datos*, con todas las tramas correspondientes a la simulación *Enviar Trama*, dependiendo de los orígenes, destinos y prioridades suministrados por el usuario.

4.2.4 Iteración 4: Simular el proceso Ring Poll

Al igual que la iteración 3, esta iteración corresponde al grupo de simulaciones, y no necesita de una interacción mayor del usuario, lo único que hace falta es seleccionar la simulación en el menú y luego ir avanzando por cada una de las tramas enviadas, es por ello que el diagrama de clases asociado a esta simulación es el que se muestra en la Figura 4.18, el cual corresponde al diagrama básico de simulación.

4.2.5 Iteración 5: Simular el envío de tramas

Al igual que las dos iteraciones anteriores, esta iteración corresponde al grupo de simulaciones, pero a diferencia del *Proceso de Monitor Contention* y *Ring Poll*, esta simulación requiere de una interacción extra del usuario, ya que como se explicó en los casos de uso, es necesario elegir los datos necesarios para enviar de una a tres tramas, es por ello que el diagrama de clases asociado a esta simulación varía, y se puede ver en la Figura 4.19:

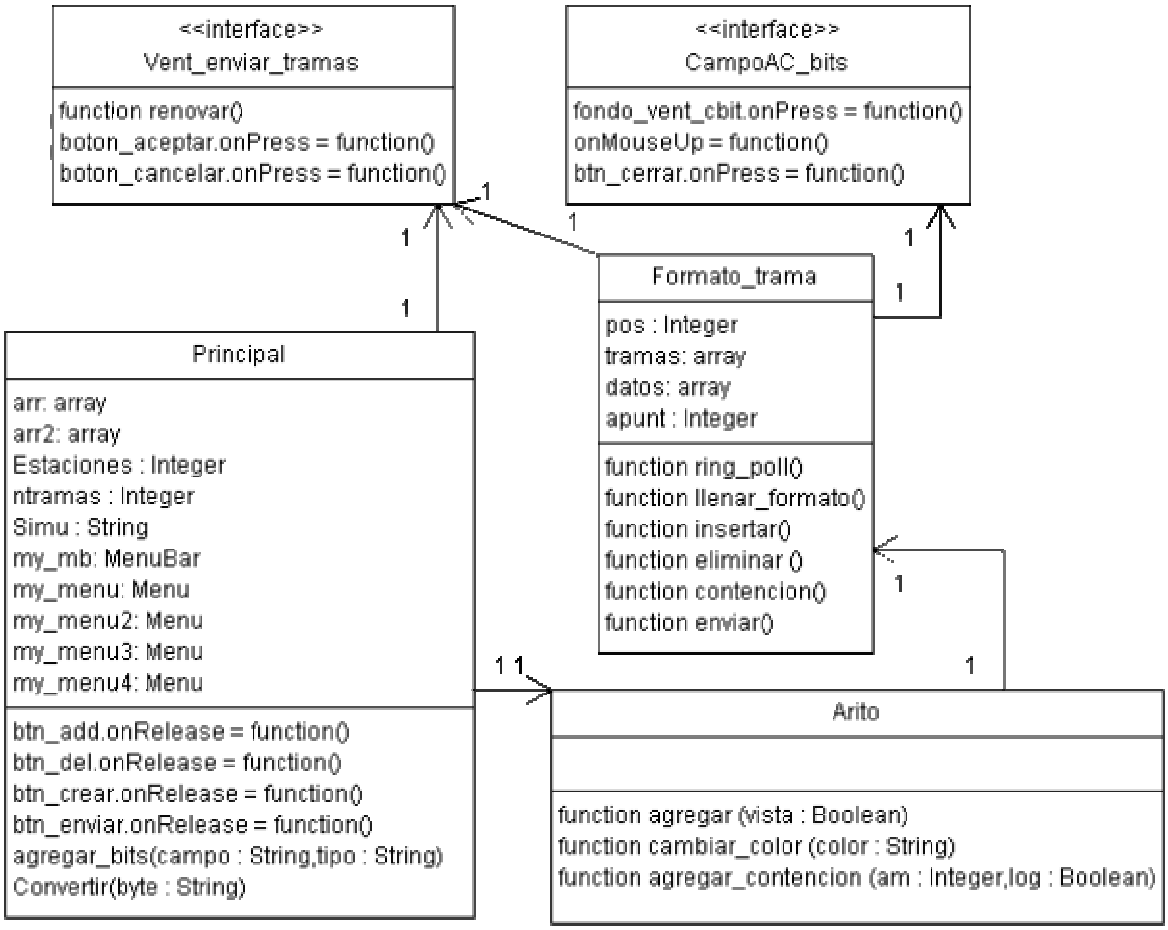

Figura 4.19: Diagrama de Clase Simular Enviar trama

Descripción de los métodos de la interfaz *Vent_enviar_tramas*:

- **function renovar ()**: Actualiza los valores correspondientes a origen y destino mostrados en los ComboBox que se encuentran en la ventana.
- **boton_aceptar.onPress = function ():** Llama a la función enviar perteneciente a la clase *Formato_Trama*, con los datos de las tramas a enviar seleccionados por el usuario.
- **boton_cerrar.onPress = function ()**: Cierra la ventana, y termina la simulación.

4.2.6 Iteración 6: Simular la inserción de una estación

Al igual que la iteración *Simular el envío de tramas*, esta iteración corresponde al grupo de simulaciones y requiere de una interacción extra del usuario, ya que como se explicó en los casos de uso, es necesario que el usuario elija la posición

en la cual será insertada la nueva estación, es por ello que el diagrama de clases asociado a esta simulación varía, y se puede ver en la Figura 4.20:

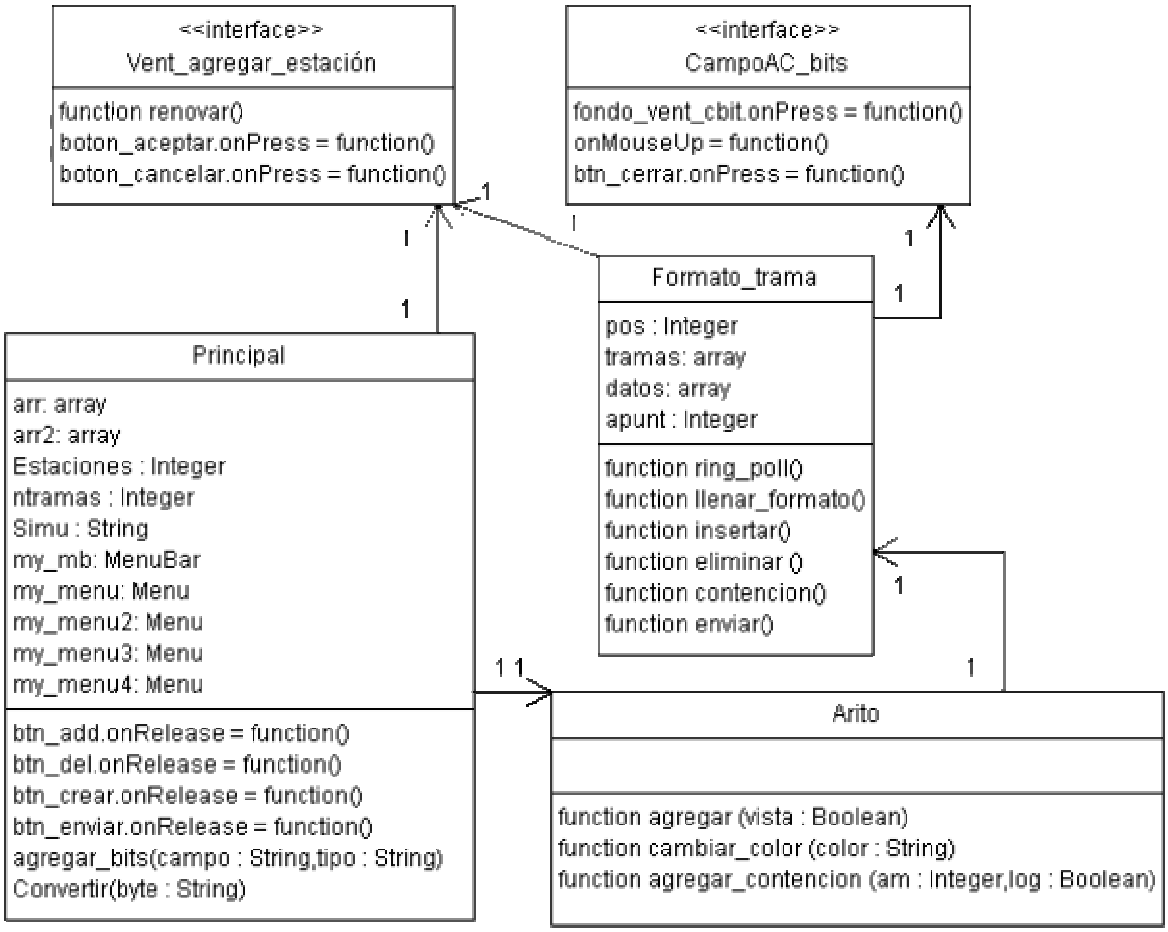

Figura 4.20: Diagrama de Clase Simular la inserción de una estación

Descripción de los métodos de la interfaz *Vent_agregar_estación*:

- **function renovar ()**: Actualiza las estaciones que serán mostradas en los ComboBox para que el usuario elija entre cuales estaciones desea insertar la nueva estación.
- **boton_aceptar.onPress = function ():** Llama a la función insertar perteneciente a la clase *Formato_Trama* para el inicio de la simulación.
- **boton_cerrar.onPress = function ()**: Cierra la ventana.

4.2.7 Iteración 7: Simular la eliminación de una estación

Esta iteración también corresponde al grupo de simulaciones y requiere de una interacción extra del usuario, ya que como se explicó en los casos de uso, es

necesario que el usuario elija cual será la estación que será eliminada, es por ello que el diagrama de clases asociado a esta simulación varía, y se puede ver en la Figura 4.21:

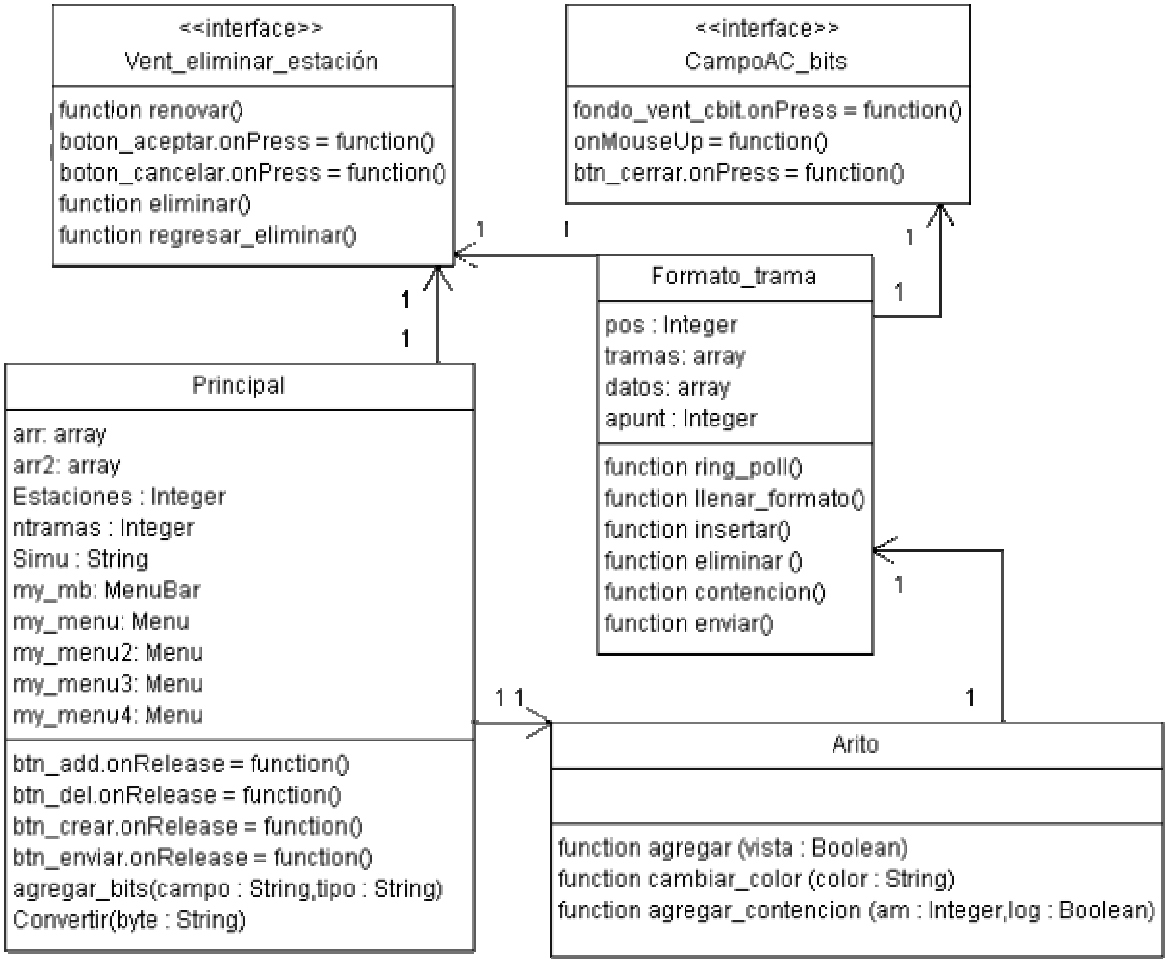

Figura 4.21: Diagrama de Clase Simular la eliminación de una estación

Descripción de los métodos de la interfaz *Vent_eliminar_estación*:

- **function renovar ()**: Actualiza las estaciones que serán mostradas en el ComboBox para que el usuario elija cual es la estación que desea eliminar. Cabe destacar que dentro de esta lista no se encuentra el monitor activo.
- **boton_aceptar.onPress = function ():** Llama a la función eliminar perteneciente a la clase *Formato_Trama* para el inicio de la simulación.
- **boton_cancelar.onPress = function ()**: Cierra la ventana.
- **function eliminar ()**: Elimina la estación seleccionada por el usuario de la topología.
- **function regresar_eliminar ()**: Regresa la topología al estado original, es decir, muestra la topología sin haber eliminado la estación.

4.2.8 Iteración 8: Evaluación de conceptos básicos

Esta iteración corresponde al grupo de evaluación, todas las iteraciones pertenecientes a este grupo tendrá como diagrama base el que se muestra en la Figura 4.22. En este interactúan las clases *Principa*l (ya explicada anteriormente) y *Vent_eval*, además de la interfaz *Vent_result*, las cuales serán detalladas a continuación:

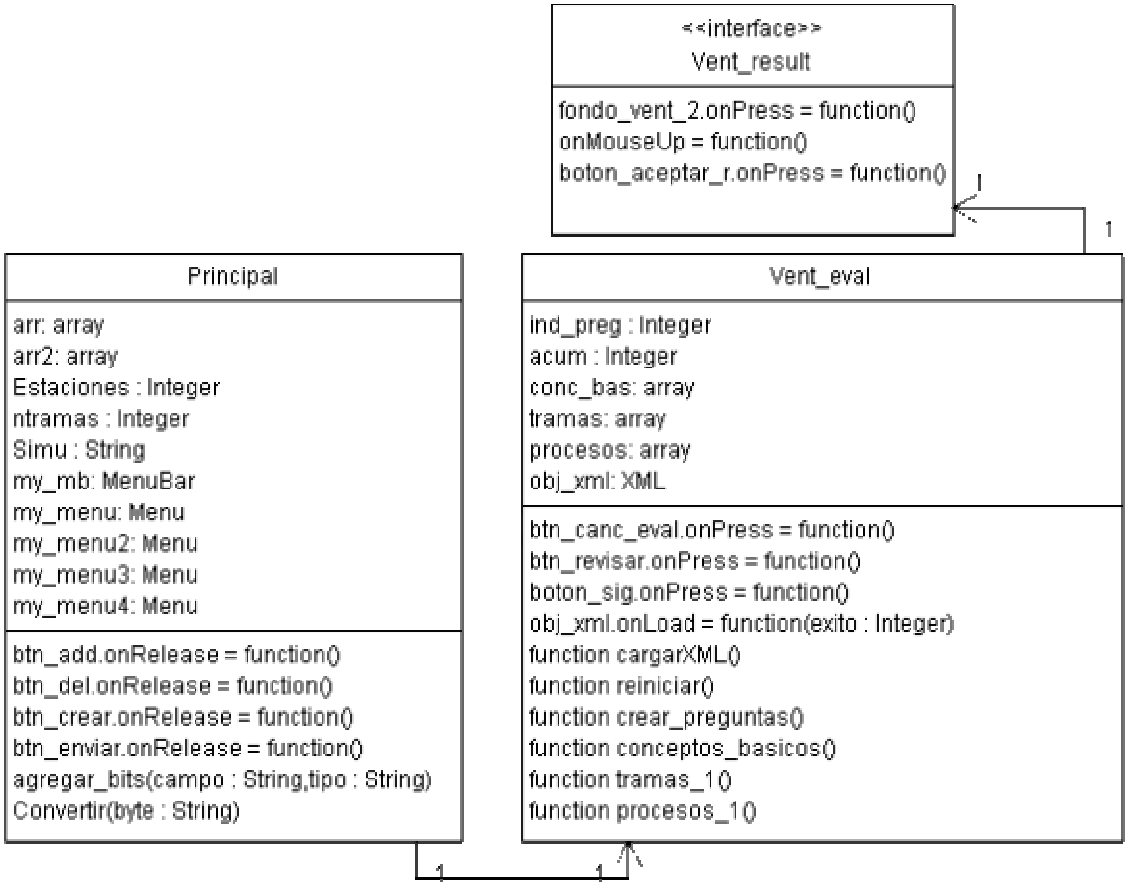

Figura 4.22: Diagrama de Clase Evaluación

Descripción de los métodos de la interfaz *Vent_result*:

- **fondo_vent_2.onPress = function ()**: Permite mover la ventana a través del escenario (Drag).
- **OnMouseUp = function ()**: Detiene la ventana una vez que el usuario suelta el botón derecho del mouse (Drop).
- **boton_aceptar_r.onPress = function ()**: Cierra la ventana.

Descripción de atributos de la clase *Vent_eval*:

- **ind_preg:** es un entero que cumple la función de un apuntador, ya que lleva la posición que se está utilizando del arreglo que contiene las preguntas, el cual puede ser *conc_bas*, *tramas* ó *procesos* dependiendo de la selección hecha por el usuario.
- **acum:** corresponde a un entero que guarda la puntuación que lleva acumulada el usuario.
- **conc_bas:** es un arreglo que contiene las preguntas y respuestas correspondientes al módulo de evaluación de conceptos básicos.
- **tramas:** es un arreglo que contiene las preguntas y respuestas correspondientes al módulo de evaluación de tramas.
- **procesos:** es un arreglo que contiene las preguntas y respuestas correspondientes al módulo de evaluación de procesos.

Descripción de los métodos de la clase *Vent_eval*:

- **btn_canc_eval.onPress = function ()**: Cierra la ventana y culmina la evaluación sin dar resultado.
- **btn_revisar.onPress = function ():** Es usada para verificar si la respuesta del usuario es correcta o no, de ser correcta suma dos puntos a la variable **Acum**.
- **boton_sig.onPress = function ()**: Sólo se habilita una vez que se haya revisado la respuesta, muestra la siguiente pregunta, si no existe más preguntas, llama a la interfaz *Vent_result* para que muestre los resultados de la evaluación.
- **obj_xml.onLoad = function (exito):** Carga en los arreglos *conc_bas*, *tramas* ó *procesos* las preguntas y respuestas contenidas en los archivos XML, dependiendo de la opción escogida por el usuario.
- **function cargarXML ()**: Invoca a la función *load* del objeto XML, pasándole por parámetro la ruta del archivo XML.
- **function reiniciar ()**: Regresa todos los valores de las variables y contadores a 0.
- **Function crear_preguntas ()**: Toma de forma aleatoria una pregunta de los arreglos *conc_bas*, *tramas* o *procesos* y la muestra.
- **function conceptos_basicos ()**: Toma la pregunta ubicada en la posición *ind_preg* del arreglo *conc_bas*.
- **function tramas_1 ()**: Toma la pregunta ubicada en la posición *ind_preg* del arreglo *tramas*.
- **function procesos_1 ()**: Toma la pregunta ubicada en la posición *ind_preg* del arreglo *procesos*.

4.2.9 Iteración 9: Evaluación de Tramas

Esta iteración corresponde al grupo de evaluación, por lo que el diagrama de clase asociado es el que se muestra en la Figura 4.22.

4.2.10 Iteración 10: Evaluación de Procesos

Esta iteración corresponde al grupo de evaluación, por lo que el diagrama de clase asociado es el que se muestra en la Figura 4.22.

4.2.11 Iteración 11: Ayuda

En esta interacción participan las clases *Principal*, y *Vent_ayuda*. El diagrama de clase asociado a esta iteración se muestra en la Figura 4.23.

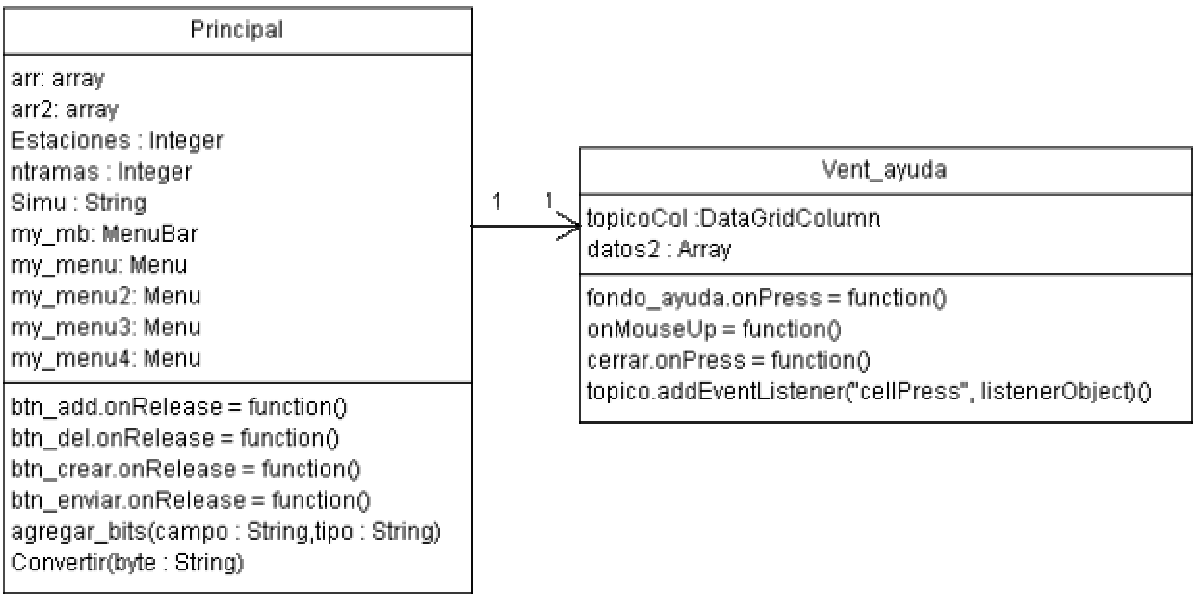

Figura 4.23: Diagrama de Clase Ayuda

Descripción de Atributos de la clase *Vent_ayuda*:

- **topicoCol:** es un componente de flash que permite la agregación de una columna en un DataGrid, corresponde al listado donde se especifican los diferentes tópicos de ayuda que se ofrecen.
- **datos2:** corresponde a un arreglo que contiene la lista de términos a asociar a la columna *topicoCol*.

Descripción de los métodos de la clase *Vent_ayuda*:

- **fondo_ayuda.onPress = function ()**: Permite mover la ventana a través del escenario (Drag).
- **onMouseUp = function ():** Detiene la ventana una vez que el usuario suelta el botón derecho del mouse (Drop).
- **cerrar.onPress = function ()**: Cierra la ventana.

- **topico.addEventListener ("cellPress", listenerObject)**: Se encarga de mostrar la explicación asociada al tópico seleccionado por el usuario.

4.2.12 Iteración 12: Glosario de términos

En esta interacción participan las clases *Principal*, y *Vent_glosario*. El diagrama de clase asociado a esta iteración se muestra en la Figura 4.24.

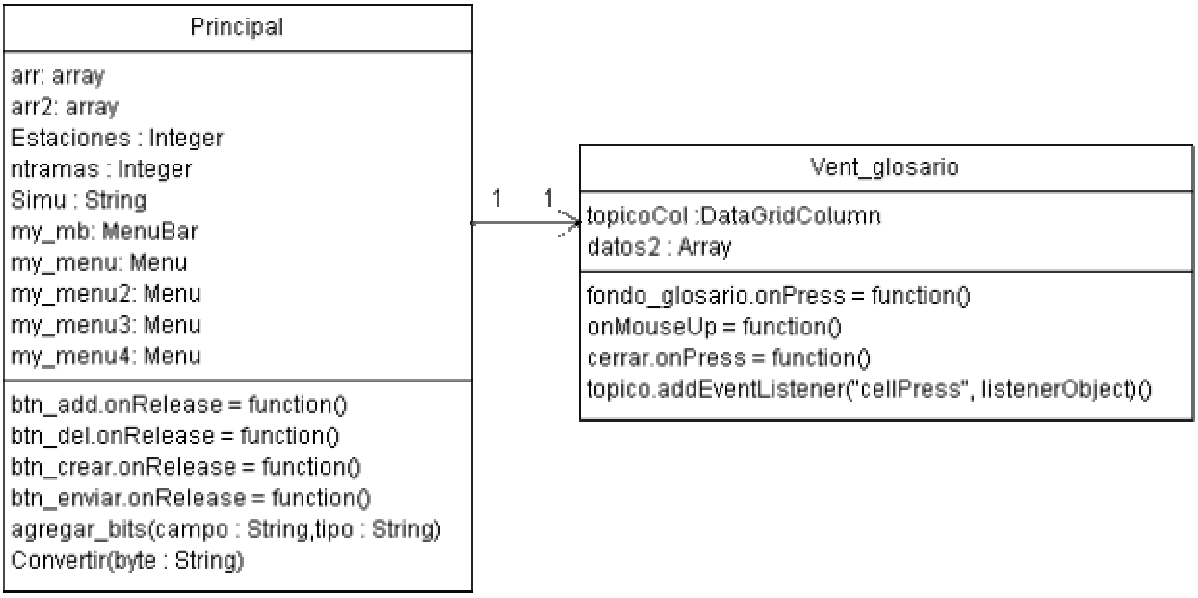

Figura 4.24: Diagrama de Clase Glosario de términos

La descripción de los atributos y métodos de la clase *Vent_glosario* es igual al de la clase *Vent_ayuda*, por lo cual no se detallarán en esta sección.

4.2.13 Iteración 13: Acerca de

En esta interacción participan la clase *Principal*, y la interfaz *Vent_acerca_de*. El diagrama de clase asociado a esta iteración se muestra en la Figura 4.25.

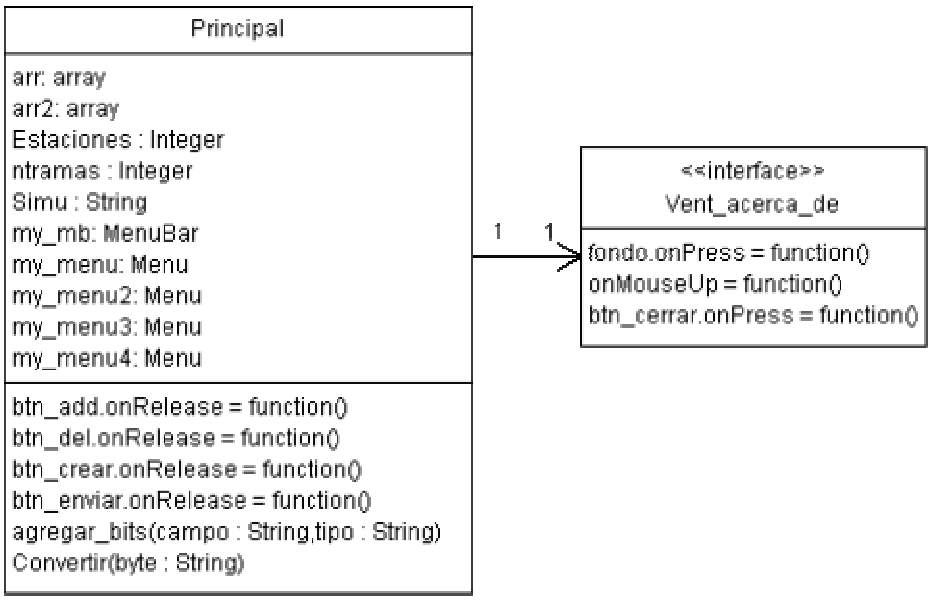

Figura 4.25: Diagrama de Clase Acerca de

Descripción de los métodos de la interfaz *Vent_acerca_de*:

- **fondo.onPress = function ()**: Permite mover la ventana a través del escenario (Drag).
- **onMouseUp = function ():** Detiene la ventana una vez que el usuario suelta el botón derecho del mouse (Drop).
- **btn_cerrar.onPress = function ()**: Cierra la ventana.

Finalmente, todo el diseño de la aplicación, se puede observar en el diagrama de clase completo, el cual se muestra en la Figura 4.26, allí se ve la relación de todas las clases pertenecientes a cada iteración.

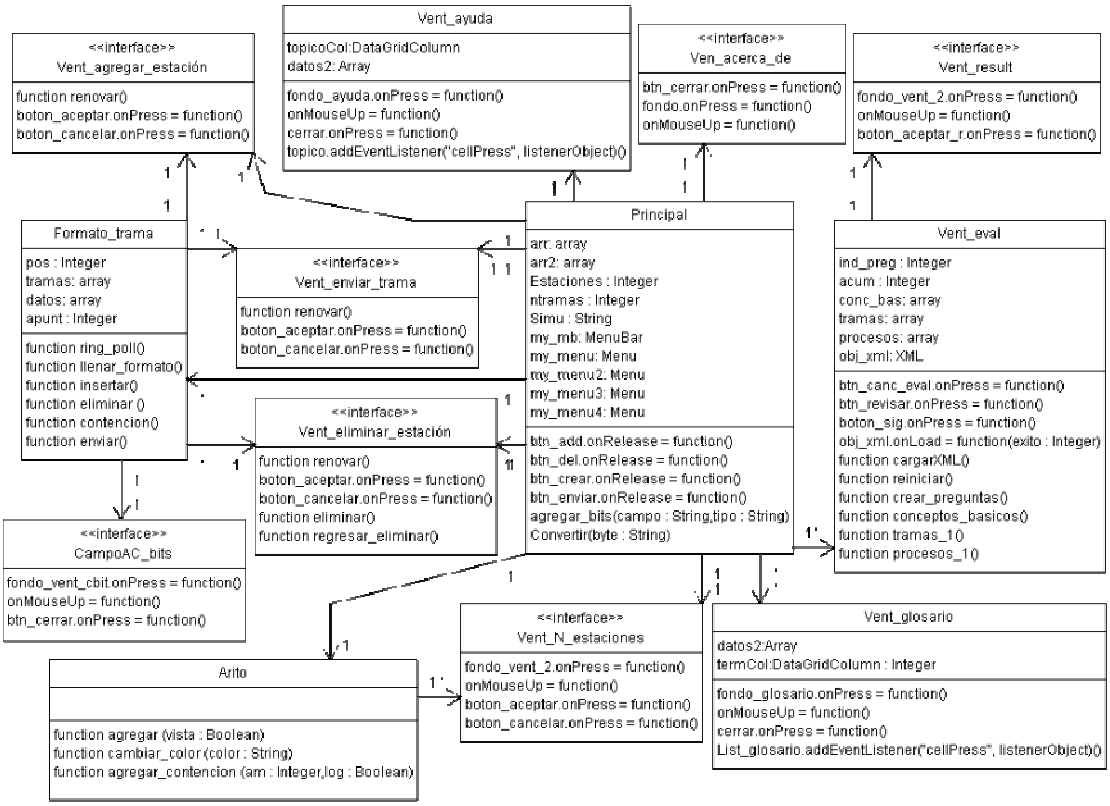

Figura 4.26: Diagrama de Clase completo

4.3 Fase de codificación

Durante esta fase, se llevó a cabo la elaboración del código en flash CS3, específicamente en el lenguaje Action Script 2.0. Como se mencionó anteriormente, en este lenguaje se manejan las clases por medio de Movie Clips, donde cada uno tiene una porción de código que le corresponde. En esta sección se mostrarán los fragmentos de códigos más importantes pertenecientes a los Movie Clips asociados a cada iteración.

4.3.1 Iteración 1: General

Durante esta iteración se definió todo lo que representaba la interfaz principal de la aplicación, para ello se crearon lo que fue el menú principal, ubicado en la parte superior de la ventana, y la barra de menú rápido ubicado en la parte derecha de la ventana.

```
16  */
17  var my_mb:mx.controls.MenuBar;
18
19  var my_menu:mx.controls.Menu = my_mb.addMenu ("Archivo     ");
20
21  my_menu.addMenuItem ({label:"Nuevo   ", instanceName:"newInstance"});
22
23  my_menu.setSize (500,100,true);
24  var fileListener1:Object = new Object ();
25  fileListener1.change = function (eventObj:Object):Void
26  {
27      if (eventObj.menuItem == eventObj.menu.newInstance)
28      {
29          vent_2._visible = true;
30      }
31  };
32  my_menu.addEventListener ("change",fileListener1);
33
34
35  my_menu.addMenuItem ({label:"Salir", instanceName:"exitInstance"});
36  var fileListener2:Object = new Object ();
37  fileListener2.change = function (eventObj:Object):Void
38  {
39      if (eventObj.menuItem == eventObj.menu.exitInstance)
40      {
41          fscommand ("quit", true);
42      }
43  };
44  my_menu.addEventListener ("change",fileListener2);
45
46  var my_menu2:mx.controls.Menu = my_mb.addMenu ("Simulaciones     ");
47  my_menu2.setSize (500,500,true);
48  my_menu2.addMenuItem ({label:"Proceso Monitor Contention    ", instanceName:"contencionInstance"});
49  var fileListener6:Object = new Object ();
50  fileListener6.change = function (eventObj:Object):Void
```

Figura 4.27: Segmento de código de creación del menú principal

La Figura 4.27 muestra el segmento de código correspondiente a la construcción del menú principal, específicamente se puede observar la creación del menú *Archivo*, que contiene los ítems *Nuevo* y *Salir*, además de la creación del menú *Simulaciones* que contiene el ítem de *Proceso de Monitor Contention*.

Por otro lado, la Figura 4.28 muestra el segmento de código correspondiente a la asociación de actividad de los botones pertenecientes a la barra de menú rápido, específicamente en este caso se muestra las acciones asociadas al botón agregar una estación, se puede ver que se realiza la validación que permite no agregar más de 12 estaciones, ya que de ser así envía un mensaje de error, notificando la violación, en caso de que aún no se haya alcanzado el límite de estaciones, se procede a mostrar la interfaz que permite escoger la posición en la cual será insertada la nueva estación.

```
346 btn_add.onRelease = function ()
347 {
348
349     simu = "insertar";
350
351     if (estaciones<12)
352     {
353
354         btn_add.enabled = false;
355         btn_add._alpha = 50;
356         btn_del.enabled = false;
357         btn_del._alpha = 50;
358         btn_crear.enabled = false;
359         btn_crear._alpha = 50;
360         btn_enviar.enabled = false;
361         btn_enviar._alpha = 50;
362         my_mb.setMenuEnabledAt (1,false);
363         my_mb.setMenuEnabledAt (0,false);
364         formato_trama.apunt = 0;
365         formato_trama.datos.splice (0,datos.length);
366         Vent_agregar_estacion._visible = true;
367
368     }
369     else
370     {
371         estaciones = estaciones-1;
372         vent_error.mensaje_error.text = "No es posible agregar más \n estaciones.";
373         vent_error._visible = true;
374     }
375 };
376
377 btn_del.onRelease = function ()
378 {
379
```

Figura 4.28: Segmento de código asociado a botones de barra de menú

4.3.2 Iteración 2: Crear Topología

En esta iteración se crea el código que mostrará la topología de anillo con la cantidad de estaciones que haya seleccionado el usuario. En la Figura 4.29 se muestra un segmento de código perteneciente al método *Agregar* del movie clip *Arito*, donde se crea mediante un ciclo; el cual tiene tantas iteraciones como estaciones a agregar en el escenario y se le asigna sus propiedades, como lo son: nombre, tipo, dirección MAC y NAUN.

Cabe destacar que el método *Agregar* es llamado al iniciar cualquier simulación, ya que dependiendo de la misma, las estaciones serán mostradas de manera diferente, por ejemplo en la simulación contención, al inicio se muestran sólo los monitores en espera, en cambio en el resto se muestra siempre el monitor activo y los monitores en espera.

```
424     for (n=0; n<_parent.estaciones; n++)
425     {
426         if (_parent.simu == "insertar")
427         {
428             if (n == _parent.Vent_agregar_estacion.ind_pc+1)
429             {
430                 mipc.innerpc.texto.textColor = "0xFFCC99";
431             }
432         }
433         if (n == 0)
434         {
435             if (_parent.simu == "contencion")
436             {
437                 mipc = cont.attachMovie ("PC", "pc"+cnt, cnt++);
438                 mipc.innerpc.texto.text = _parent.arr[_parent.arr2[n]].nombre;
439                 mipc.innerpc._xscale = 50;
440                 mipc.innerpc._yscale = 50;
441                 mipc.innerpc._rotation = -(gr*n);
442             }
443             else
444             {
445
446                 mipc = cont.attachMovie ("PC2", "pc"+cnt, cnt++);
447                 mipc.innerpc2.texto.text = _parent.arr[_parent.arr2[n]].nombre;
448                 mipc.innerpc2._xscale = 50;
449                 mipc.innerpc2._yscale = 50;
450                 mipc.innerpc2._rotation = -(gr*n);
451             }
452             mipc._x = arito._x;
453             mipc._y = arito._y;
454             mipc._rotation = gr*n;
455
456             mipc._xscale = 100;
457             mipc._yscale = 100;
458
```

Figura 4.29: Segmento de código del método Agregar perteneciente al movie clip Arito

4.3.3 Iteración 3: Simular el Proceso Monitor Contention

En el desarrollo de esta simulación se involucran varios métodos y clases, sin embargo el método que lleva el mayor control de esta iteración es *Contencion*, de la cual se muestra un extracto de código en la Figura 4.30. Allí se va construyendo el arreglo *datos*, el cual contiene todos los campos pertenecientes a las tramas que son enviadas durante esta simulación. Esta iteración se diferencia del resto de simulaciones, ya que inicialmente no existe un monitor activo en la topología, es por ello que hace uso del método *Agregar* y *Agregar_contencion* del movie clip *Arito*.

En la Figura 4.30 se puede observar la utilización de la variable *am*, en la cual se va guardando el identificador del monitor activo, inicialmente es 0, sin embargo luego en cada iteración del ciclo, lo cual es equivalente a cada trama enviada, se comprara la direcciones MAC (línea 1430), y si esta es mayor el identificador del monitor activo cambia. Finalmente una vez que se han enviado todas las tramas Claim token (*tipo_trama* [8]) y se define quien es el nuevo monitor activo, se procede a mostrar la topología con el monitor activo identificado, mediante el método *Agregar_contencion*.

```
1411
1412  function contencion ()
1413  {
1414
1415      ori = _parent.arr[_parent.arr2[0]].MAC;
1416      des = "Todos";
1417      am = 0;
1418
1419      for (n=0; n<_parent.estaciones; n++)
1420      {
1421          if (n == 0)
1422          {
1423              datos.addItem ({origen:ori, destino:des, tipo_tram:tramas[8], prio:"0", reserv:"0", MC:fal
1424              _parent.ntramas = _parent.ntramas+1;
1425
1426          }
1427          else
1428          {
1429
1430              if (_parent.arr[_parent.arr2[am]].MAC<_parent.arr[_parent.arr2[n]].MAC)
1431              {
1432
1433                  mens = "Como la MAC de la estación "+_parent.arr[_parent.arr2[n]].nombre+" ("+_parent.
1434                  ori = _parent.arr[_parent.arr2[n]].MAC;
1435                  am = n;
1436              }
1437              else
1438              {
1439                  mens = "Como la MAC de la estación "+_parent.arr[_parent.arr2[n]].nombre+" ("+_parent.
1440              }
1441              color = "0x3366FF";
1442              if (n == 1)
1443              {
1444                  color = "0xFF0033";
1445              }
```

Figura 4.30: Segmento de código de método contencion

4.3.4 Iteración 4: Simular el Proceso Ring Poll

Al igual que la iteración anterior, durante el desarrollo de la simulación Ring Poll, se hace uso de varios métodos y clases, sin embargo el método que cumple el rol más importante dentro de esta iteración es *Ring_poll*. Tal y como se muestra en la Figura 4.31, por medio de este método que es invocado desde el menú principal, se va creando el arreglo *datos* con todos los datos correspondientes a las tramas enviadas durante esta simulación.

En esta iteración no existe variación alguna, exceptuando la cantidad de estaciones que participan en el proceso, es decir lo único que varía es la cantidad de veces que el ciclo se repetirá, todo esto debido a que el usuario no tiene mayor participación durante esta simulación.

```
551 function ring_poll ()
552 {
553     for (n=0; n<_parent.estaciones; n++)
554     {
555 
556         ori = _parent.arr[_parent.arr2[n]].MAC;
557         des = "Todos";
558 
559         if (n == 0)
560         {
561             datos.addItem ({origen:ori, destino:des, tipo_tram:tramas[0], prio:"0", reserv:"0", MC:fal
562             for (k=n+1; k<_parent.estaciones; k++)
563             {
564                 color = "0x3366FF";
565                 mens = "Continúa circulando la trama AMP por cada una de las estaciones con los bits A
566                 if (k == n+1)
567                 {
568                     mens = "La estación "+_parent.arr[_parent.arr2[1]].nombre+" recibe la trama AMP, c
569                     color = "0xFF0033";
570                 }
571                 if (k == _parent.estaciones-1)
572                 {
573                     datos.addItem ({origen:ori, destino:des, tipo_tram:tramas[0], prio:"0", reserv:"0"
574                 }
575                 else
576                 {
577                     datos.addItem ({origen:ori, destino:des, tipo_tram:tramas[0], prio:"0", reserv:"0"
578                 }
579             }
580             for (k=0; k<_parent.estaciones+1; k++)
581             {
582                 if (k == 0)
583                 {
584                     datos.addItem ({origen:ori, destino:des, tipo_tram:tramas[7], prio:"0", reserv:"0"
```

Figura 4.31: Segmento de código de método ring_poll

4.3.5 Iteración 5: Simular el Envío de Tramas

En esta simulación se interactúa con la interfaz *Vent_enviar_trama*, donde se toman y validan los datos correspondientes a las tramas a ser envidas. En la Figura 4.32 se puede observar un segmento de código perteneciente a esta interfaz, específicamente, donde son extraídos los valores de los combo box y son validados.

```
295 boton_aceptar.onPress = function ()
296 {
297     contador = contador+1;
298     sigo = true;
299     if (contador>1)
300     {
301         if (l4 & l5 & l6)
302         {
303             dos = true;
304             if (ori2.selectedItem.label == "-" || des2.selectedItem.label == "-" || prio_ori2.selected
305             {
306                 if (ori2.selectedItem.label == "-")
307                 {
308                     _parent.vent_error.mensaje_error.text = "El valor del campo origen debe ser especí
309                 }
310                 if (des2.selectedItem.label == "-")
311                 {
312                     _parent.vent_error.mensaje_error.text = "El valor del campo destino debe ser espec
313                 }
314                 if (prio_ori2.selectedItem.label == "-")
315                 {
316                     _parent.vent_error.mensaje_error.text = "El valor del campo prioridad debe ser esp
317                 }
318                 _parent.vent_error._visible = true;
319                 sigo = false;
320 
321             }
```

Figura 4.32: Segmento de código de Interfaz Vent_enviar_trama

En la Figura 4.33 se muestra el segmento de código del método *Enviar*, el cual, al igual que el resto de las iteraciones que pertenecen al grupo de simulaciones, se encarga de construir el arreglo de tramas de acuerdo a los datos proporcionado por el usuario.

```
1619 function enviar ()
1620 {
1621 
1622     i = 0;
1623     ptoken_ant = 0;
1624     //Ciclo para buscar la primera estación que desea envíar, únicamente se envía token con prioridad
1625     while (_parent.arr[_parent.arr2[i]].nombre != _parent.Vent_enviar_tramas.tramas[0].ori)
1626     {
1627         if (i == 0)
1628         {
1629             datos.addItem ({origen:ori, destino:des, tipo_tram:tramas[7], prio:"0", reserv:"0", MC:fal
1630         }
1631         else
1632         {
1633             datos.addItem ({origen:ori, destino:des, tipo_tram:tramas[7], prio:"0", reserv:"0", MC:fal
1634 
1635         }
1636         i = i+1;//trace(i);
1637         _parent.ntramas = _parent.ntramas+1;
1638 
1639     }
1640     reserv = 0;
1641     prioridad = 0;
1642     //Ciclo de envio de las tres tramas
1643     for (j=0; j<3; j++)
1644     {
1645         //si no estan las tres tramas se sale del ciclo
1646         if (_parent.Vent_enviar_tramas.tramas[j].ori == "-")
1647         {
1648             break;
1649         }
1650         else
1651         {
```

Figura 4.33: Segmento de código de método enviar

4.3.6 Iteración 6: Simular la Inserción de una Estación

En esta iteración se hace uso del método *Insertar*, perteneciente al movie clip *Formato_Trama*, el cual es invocado desde la interfaz *Vent_agregar_estación*, luego de haber definido el lugar donde será insertada la nueva estación. Tal y como se puede ver en la Figura 4.34, luego de que el usuario ha seleccionado las dos estaciones entre las cuales estará ubicada, se procede a invocar el método *Agregar* del movie clip *Arito*, con los datos de la nueva estación.

```
55  boton_aceptar.onPress = function() {
56
57      if (my_cb1.selectedIndex == _parent.estaciones-1) {
58          switch (my_cb2.selectedIndex) {
59              case 0 :
60              if(_parent.estaciones!=2)
61              {
62                  aux = _parent.arr2[_parent.estaciones];
63                  _parent.arr2.splice(_parent.estaciones,1);
64                  _parent.arr2.splice(my_cb1.selectedIndex ,0,aux);
65                  ind_pc=my_cb1.selectedIndex;
66                  _parent.estaciones = _parent.estaciones+1;
67              }
68              else
69              {
70                  _parent.estaciones = _parent.estaciones+1;
71                  ind_pc=_parent.estaciones-1;
72
73              }
74
75                  break;
76
77              case 1 :
78                  _parent.estaciones = _parent.estaciones+1;
79                  ind_pc=_parent.estaciones-1;
80                  break;
81          }
82      }
83      else
84      {
85          switch (my_cb2.selectedIndex) {
```

Figura 4.34: Segmento de código de interfaz Vent_agregar_estación

La Figura 4.35 muestra el segmento de código correspondiente al inicio del método *Insertar*, donde al igual que el resto de los métodos generadores de simulaciones, se encarga de construir una serie de tramas que luego serán mostradas en pantalla, a medida que el usuario va avanzando, o por el contrario retrocediendo.

```
893  function insertar ()
894  {
895
896      ori = _parent.arr[0].MAC;
897      des = "Todos";
898      //El monitor activo envía una trama de AMP
899      for (n=0; n<_parent.estaciones; n++)
900      {
901          if (n == 0)
902          {
903              datos.addItem ({origen:ori, destino:des, tipo_tram:tramas[0], prio:"0", reserv:"0", MC:fal
904              _parent.ntramas = _parent.ntramas+1;
905          }
906          else
907          {
908              color = "0x3366FF";
909              if (n == 1)
910              {
911                  color = "0xFF0033";
912              }
913              if (n == _parent.Vent_agregar_estacion.ind_pc)
914              {
915                  datos.addItem ({origen:ori, destino:des, tipo_tram:tramas[0], prio:"0", reserv:"0", MC
916              }
917              else
918              {
919                  datos.addItem ({origen:ori, destino:des, tipo_tram:tramas[0], prio:"0", reserv:"0", MC
920
921              }
922              _parent.ntramas = _parent.ntramas+1;
923          }
924      }
925      datos.addItem ({origen:ori, destino:des, tipo_tram:tramas[7], prio:"0", reserv:"0", MC:false, bit:
926      _parent.ntramas = _parent.ntramas+1;
```

Figura 4.35: Segmento de código de método insertar

4.3.7 Iteración 7: Simular la Eliminación de una Estación

En esta iteración participa la interfaz *Vent_eliminar_estación*, que es la que permite al usuario escoger la estación que desea eliminar, en la Figura 4.36 se muestra el extracto de código que se ejecuta luego de que el usuario a presionado el botón aceptar, es decir, luego de haber elegido la estación a eliminar.

```
22  boton_aceptar.onPress = function () {
23
24      _parent.ntramas=(5*_parent.estaciones)-(my_cb.selectedItem.data)-2;
25      _parent.formato_trama._visible = true;
26      _parent.arito.agregar(true);
27      _parent.formato_trama.eliminar();
28      _visible = false;
29
30  }
31
32  boton_cancelar.onPress = function () {
33      _parent.btn_add.enabled =true;
34      _parent.btn_add._alpha = 100;
35      _parent.btn_del.enabled =true;
36      _parent.btn_del._alpha = 100;
37      _parent.btn_crear.enabled =true;
38      _parent.btn_crear._alpha = 100;
39      _parent.btn_enviar.enabled = true;
40      _parent.btn_enviar._alpha = 100;
41      _parent.my_mb.setMenuEnabledAt(1,true);
42      _parent.my_mb.setMenuEnabledAt(0,true);
43      _visible = false;
44      _parent.arito.agregar(false);
45  }
46
47  function eliminar()
48  {
49      aux=_parent.arr2[my_cb.selectedItem.data];
50      _parent.arr2.splice(my_cb.selectedItem.data,1);
51      _parent.arr2.push(aux);
52      _parent.estaciones=_parent.estaciones-1;
```

Figura 4.36: Segmento de código de Vent_eliminar_estación

Finalmente es llamado el método *Eliminar* perteneciente al movie clip *Formato_trama*, del cual se muestra un extracto de código en la Figura 4.37, y tiene como objetivo construir el arreglo con las tramas que son enviadas durante esta simulación.

```
1199  function eliminar ()
1200  {
1201
1202      ori = _parent.arr[_parent.arr2[_parent.Vent_eliminar_N_estaciones.my_cb.selectedItem.data]].MAC;
1203      des = "Todos";
1204
1205      a = _parent.arr[_parent.arr2[_parent.Vent_eliminar_N_estaciones.my_cb.selectedItem.data]].NAUN;
1206      if(_parent.Vent_eliminar_N_estaciones.my_cb.selectedItem.data!=_parent.estaciones-1)
1207      {
1208          b = _parent.arr[_parent.arr2[_parent.Vent_eliminar_N_estaciones.my_cb.selectedItem.data+1]].nc
1209      }
1210      else
1211      {
1212          b = _parent.arr[_parent.arr2[0]].nombre;
1213      }
1214      retiro.text = "La estación que será eliminada es "+_parent.arr[_parent.arr2[_parent.Vent_eliminar_
1215
1216      datos.addItem ({origen:ori, destino:des, tipo_tram:tramas[1], prio:"0", reserv:"0", MC:false, bit:
1217
1218      //ciclo donde la trama SMP enviada por la estación saliente recorre el anillo
1219      for (n=_parent.Vent_eliminar_N_estaciones.my_cb.selectedItem.data+1; n<_parent.estaciones; n++)
1220      {
1221          if (n == _parent.Vent_eliminar_N_estaciones.my_cb.selectedItem.data+1)
1222          {
1223              datos.addItem ({origen:ori, destino:des, tipo_tram:tramas[1], prio:"0", reserv:"0", MC:fal
1224          }
1225          else
1226          {
1227              datos.addItem ({origen:ori, destino:des, tipo_tram:tramas[1], prio:"0", reserv:"0", MC:fal
1228          }
```

Figura 4.37: Segmento de código de método eliminar

4.3.8 Iteración 8: Evaluación de Conceptos Básicos

Con esta iteración se inicia el grupo de evaluación, aquí principalmente interactúan métodos del movie clip Vent_eval. Inicialmente se debe cargar las preguntas correspondientes a cada nivel de la evaluación de los archivos XML, esto se hace mediante el método *obj_xml.onLoad = function (exito)*, del cual se muestra un segmento de código en la Figura 4.38.

```
116 obj_xml.onLoad = function (exito)
117 {
118     if (exito)
119     {
120         i = 0;
121         while (obj_xml.firstChild.childNodes[i].attributes["enun"] != null)
122         {
123             switch(_parent.tipo_eva){
124                 case "basico":
125                     conc_bas.addItem ({preg:obj_xml.firstChild.childNodes[i].attributes["enun"], opc1:
126                 break;
127                 case "tramas":
128                     tramas.addItem ({preg:obj_xml.firstChild.childNodes[i].attributes["enun"], opc1:ob
129                 break;
130                 case "procesos":
131                     procesos.addItem ({preg:obj_xml.firstChild.childNodes[i].attributes["enun"], opc1:
132                 break;
133             }
134             i = i+1;
135         }
136         _visible = true;
137         crear_preguntas ();
138         switch (_parent.tipo_eva){
139             case "basico":
140                 conceptos_basicos ();
141             break;
142             case "tramas":
143                 tramas_1 ();
144             break;
145             case "procesos":
146                 procesos_1 ();
```

Figura 4.38: Segmento de código de método obj_xml.onLoad = function (exito)

Luego de que cada arreglo contenga las respectivas preguntas, dependiendo de la selección del usuario se invoca el método que le corresponda, en el caso de esta iteración es el método *Conceptos básicos*, el cual se muestra en su totalidad en la Figura 4.39.

```
254
255 function conceptos_basicos ()
256 {
257     pregunta.text = conc_bas[indices[ind_preg]].preg;
258     opcion1.label = conc_bas[indices[ind_preg]].opc1;
259     opcion2.label = conc_bas[indices[ind_preg]].opc2;
260     opcion3.label = conc_bas[indices[ind_preg]].opc3;
261     opcion4.label = conc_bas[indices[ind_preg]].opc4;
262 }
263
```

Figura 4.39: Segmento de código de método conceptos_basicos

4.3.9 Iteración 9: Evaluación de Tramas

En esta iteración, por pertenecer al grupo de evaluación, se hace uso del método *obj_xml.onLoad = function (exito)*, mostrado en la Figura 4.38 y del método particular *Tramas_1*, el cual se puede observar el código correspondiente a la asignación de las preguntas sobre tramas en la Figura 4.40.

```
265
266 function tramas_1 ()
267 {
268     pregunta.text = tramas[indices[ind_preg]].preg;
269     opcion1.label = tramas[indices[ind_preg]].opc1;
270     opcion2.label = tramas[indices[ind_preg]].opc2;
271     opcion3.label = tramas[indices[ind_preg]].opc3;
272     opcion4.label = tramas[indices[ind_preg]].opc4;
273 }
```

Figura 4.40: Segmento de código de método tramas_1

4.3.10 Iteración 10: Evaluación de Procesos

En esta iteración al igual que la anterior, por pertenecer al grupo de evaluación, se hace uso del método *obj_xml.onLoad = function (exito)*, mostrado en la Figura 4.38 y del método particular *Procesos_1*, el cual se puede observar el código correspondiente a la asignación de las preguntas sobre procesos en la Figura 4.41.

```
275 function procesos_1 ()
276 {
277     pregunta.text = procesos[indices[ind_preg]].preg;
278     opcion1.label = procesos[indices[ind_preg]].opc1;
279     opcion2.label = procesos[indices[ind_preg]].opc2;
280     opcion3.label = procesos[indices[ind_preg]].opc3;
281     opcion4.label = procesos[indices[ind_preg]].opc4;
282 }
283
```

Figura 4.41: Segmento de código de método procesos_1

4.3.11 Iteración 11: Ayuda

En esta iteración, por ser en su mayoría sólo mostrar texto e imágenes estáticas, lo que se realizó fue la asignación de cada tópico en forma estática, el cual se va mostrando u ocultando dependiendo de la selección del usuario. En la Figura 4.42 se muestra el código asociado al movie clip *Vent_ayuda*, allí se puede ver la asignación de cada tema.

```
23  var datos2:Array = new Array();
24  datos2.addItem({tema:"Crear Topología"});
25  datos2.addItem({tema:"Inserción de una estación en el anillo"});
26  datos2.addItem({tema:"Eliminación de una estación en el anillo"});
27  datos2.addItem({tema:"Simulaciones"});
28  datos2.addItem({tema:"Evaluación"});
29  topico.dataProvider = datos2;
30
31  actual._visible = true;
32  Pant_crear._visible = false;
33  agregar._visible = false;
34  eliminar._visible = false;
35  prueba._visible = false;
36  simu1._visible = false;
37
38  var listenerObject = new Object();
39  listenerObject.cellPress = function(eventObject : Object) : Void{
40
41      switch(topico.selectedItem.tema)
42      {
43          case "Crear Topología":
44              actual.text = "Crear Topología";
45              prueba._visible = false;
46              agregar._visible = false;
47              simu1._visible = false;
48              eliminar._visible = false;
49              Pant_crear._visible = true;
50              break;
51          case "Inserción de una estación en el anillo":
52              actual.text = "Inserción de una estación en el anillo";
53              prueba._visible = false;
```

Figura 4.42: Segmento de código del movie clip Vent_ayuda

4.3.12 Iteración 12: Glosario de Términos

En esta iteración, por ser en su mayoría sólo mostrar texto, lo que se realizó fue la asignación de cada tópico en forma estática, el cual se va mostrando u ocultando dependiendo de la selección del usuario. En la Figura 4.43 se muestra el código asociado al movie clip *Vent_glosario*, allí se puede ver la asignación de cada término perteneciente al glosario.

```
var datos2:Array = new Array();

var termCol:DataGridColumn = new DataGridColumn("termino");
termCol.headerText = "Lista de términos";
termCol.width = 250;
List_glosario.rowHeight=40;
List_glosario.headerHeight=40;
List_glosario.addColumn(termCol);

//datos2=["AMP","SMP","DAT","RNC","RI","RP","RE"];
datos2.addItem({termino:"AC"});
datos2.addItem({termino:"AMP"});
datos2.addItem({termino:"Anillo"});
datos2.addItem({termino:"Beaconing"});
datos2.addItem({termino:"DAT"});
datos2.addItem({termino:"ED"});
datos2.addItem({termino:"FC"});
datos2.addItem({termino:"FCS"});
datos2.addItem({termino:"FS"});
datos2.addItem({termino:"MAC"});
datos2.addItem({termino:"MAU"});
datos2.addItem({termino:"MC"});
datos2.addItem({termino:"Monitor Contention Process"});
datos2.addItem({termino:"NAUN"});
datos2.addItem({termino:"PCF"});
datos2.addItem({termino:"Priority"});
datos2.addItem({termino:"Priority Reservation"});
datos2.addItem({termino:"RI"});
datos2.addItem({termino:"Ring Poll"});
datos2.addItem({termino:"RNC"});
datos2.addItem({termino:"RP"});
datos2.addItem({termino:"SD"});
```

Figura 4.43: Segmento de código del movie clip Vent_glosario

4.3.13 Iteración 13: Acerca de

Finalmente la iteración Acerca de, no necesitó un código mayor, por lo que únicamente se le asoció el texto estático y las imágenes a la interfaz. El código correspondiente a esta iteración se muestra en la Figura 4.44.

```
btn_cerrar.onPress = function() {
        _visible = false;
};

fondo.onPress = function(){
    startDrag(this._parent);
}

onMouseUp = function() {
    stopDrag();
}
```

Figura 4.44: Código perteneciente al movie clip Vent_acerca_de

4.4 Fase de Pruebas

Al terminar cada iteración, se realizan pruebas funcionales para probar si se obtiene el resultado requerido. En las siguientes subsecciones se presentan las pruebas correspondientes a cada iteración.

4.4.1 Iteración 1: General

Las pruebas de esta iteración consisten en verificar si la barra de menú y la barra de herramientas muestran las funcionalidades oportunas y los atajos llevan al módulo apropiado. En la Figura 4.45 se muestra la ventana inicial de la aplicación donde la opción “Simulaciones” en la barra de menú y las opciones “Agregar Estación”, “Eliminar Estación” y “Enviar Trama” en la barra de herramientas están inhabilitadas hasta que se cree la topología, se habilitan automáticamente luego de haber creado una topología.

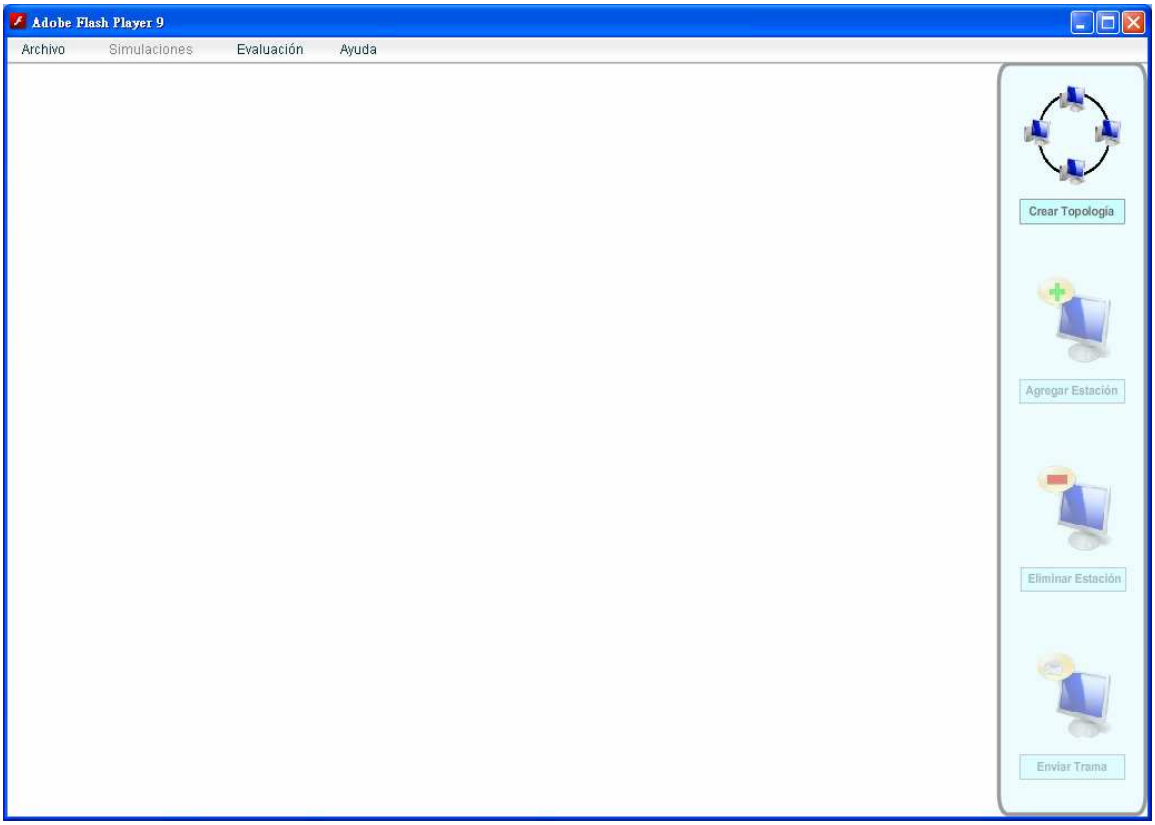

Figura 4.45 : Ventana inicial de la aplicación.

4.4.2 Iteración 2: Crear Topología

La prueba de “Crear Topología” inicia pulsando el botón de “Crear Topología” en la barra de herramientas o desde la barra de menú “Archivo/Nuevo”. Una vez seleccionado el número de estaciones y presionado el botón “Aceptar”, se debe mostrar la topología con el número de estaciones indicado, en este caso es 5 como se muestra en la Figura 4.46.

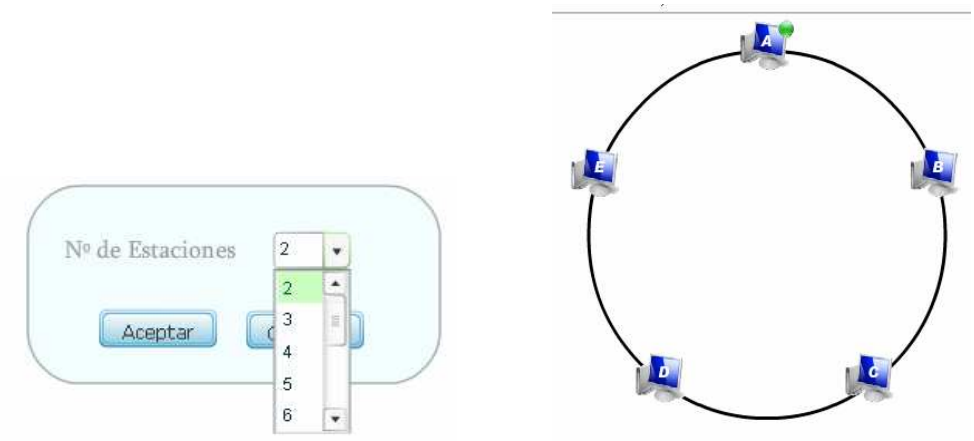

Figura 4.46: Crear Topología

4.4.3 Iteración 3: Simular el Proceso Monitor Contention

La prueba del *Proceso Monitor Contention* consiste en eliminar el monitor activo actual y seleccionar otro nuevo. Esta se inicia pulsando la opción "Simulaciones/Proceso Monitor Contention" en la barra de menú. Durante el proceso se reflejan los cambios de valor de los campos con el color rojo y se muestra una leyenda de cada avance del proceso. Al presionar el nombre de un campo, inmediatamente se muestra la información asociada. Los monitores en espera se comparan quién tiene la dirección mayor para convertirse en el monitor activo. Cuando todos los nodos reconocen el nuevo monitor activo como se muestra en la Figura 4.47, el proceso se termina.

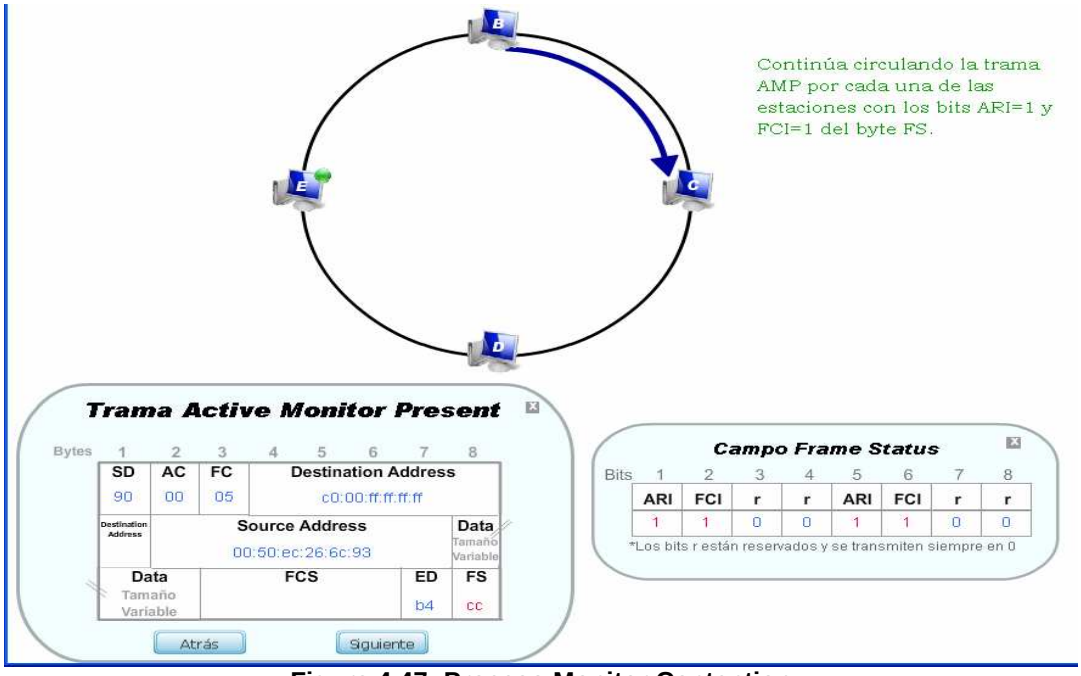

Figura 4.47: Proceso Monitor Contention

4.4.4 Iteración 4: Simular el Proceso Ring Poll

El Proceso Ring Poll consiste en actualizar el registro NAUN de todas las estaciones. La prueba inicia por el monitor activo enviando una trama AMP circulando por el anillo. Después de regresar la trama a el, la cual se retira del anillo. Luego cada uno de los monitores en espera envía su trama SMP hasta que el monitor activo actualice su registro NAUN, se termina el proceso. Se verifica el registro NAUN pulsando cada imagen de estaciones como se muestra en la Figura 4.48.

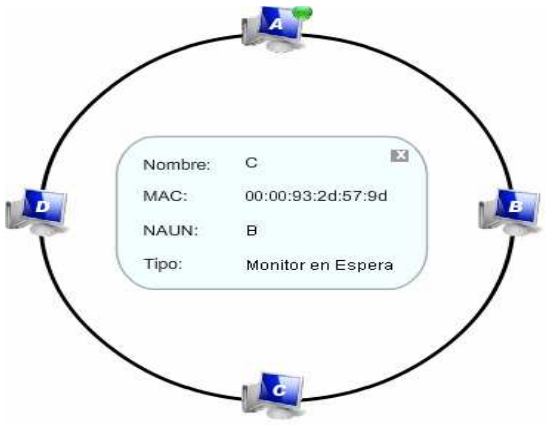

Figura 4.48: Información de la estación C

4.4.5 Iteración 5: Simular el Envío de Tramas

La prueba empieza con seleccionar la opción “Simulaciones/Enviar trama” en la barra de menú o desde la barra de herramientas con el botón “Enviar Trama”. Después de programar el origen, el destino y la prioridad de cada envío y presionar el botón “Aceptar”, comienza la simulación por el monitor activo. Durante el proceso se verifica *Priority Reservation* por las estaciones que desean enviar tramas, comparando su prioridad si es mayor o igual que la reservada para poder reservar el siguiente candidato de enviar tramas. Al terminar de enviar todas las tramas programadas y volver a recuperar la prioridad original, se culmina la simulación. En este caso (ver *Figura 4.49*), se programan los 3 envíos siguientes: de B a A con la prioridad 1, de C a B con la prioridad 6 y, de A a D con la prioridad 0. Como la simulación empieza con el monitor activo A, tiene la prioridad de ocupar el token y envía su trama, aunque la trama pasa primero a B y B cambia Priority Reservation a 1, cuando la trama pasa por C, cambia otra vez *Priority Reservation* a 6 ya que su prioridad es mayor que la de B. Cuando A termine de enviar su trama, le toca a C, luego a B. Por último, A regresa *Priority Reservation* a 0, su estado original y se termina la simulación.

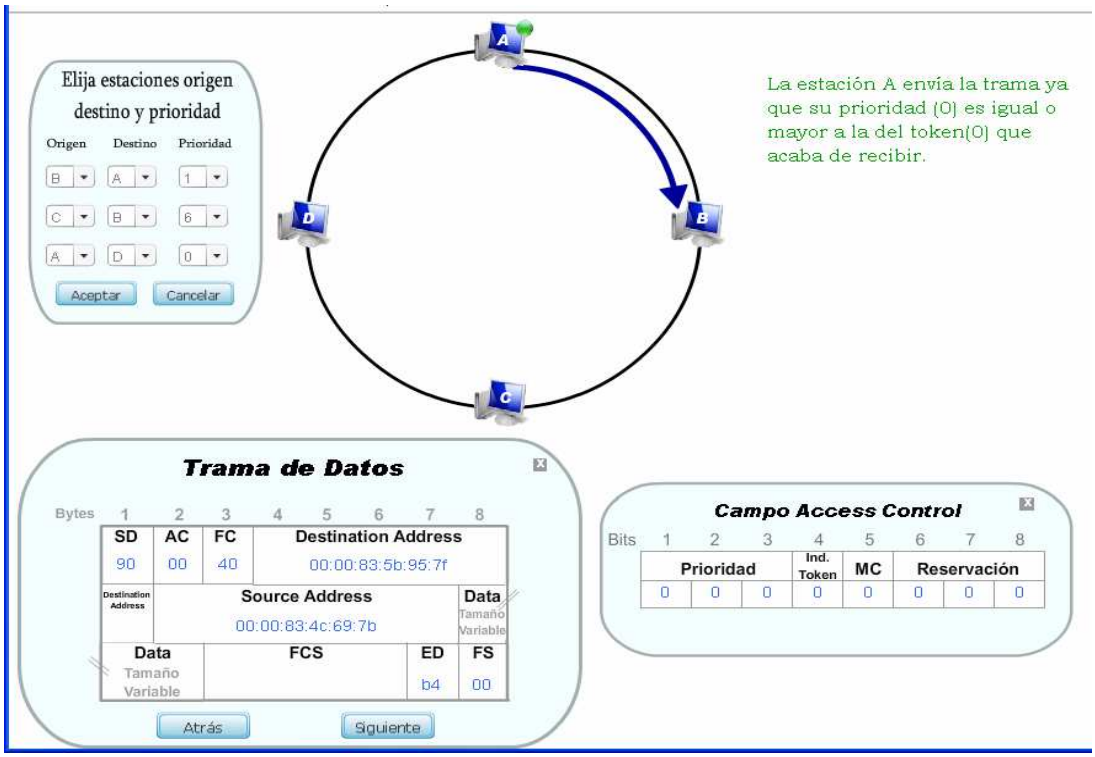

Figura 4.49: Enviar Trama

4.4.6 Iteración 6: Simular la Inserción de una Estación

La prueba de la inserción de una estación se inicia pulsando el botón de “Agregar Estación” en la barra de herramientas o desde la barra de menú “Simulaciones/Agregar estación”. Una vez seleccionada la posición de la nueva estación como se muestra en la Figura 4.50 y presionada el botón “Aceptar”, se verifica si se inserta en la posición correcta y se reconoce por las demás estaciones actualizando sus registros NAUN.

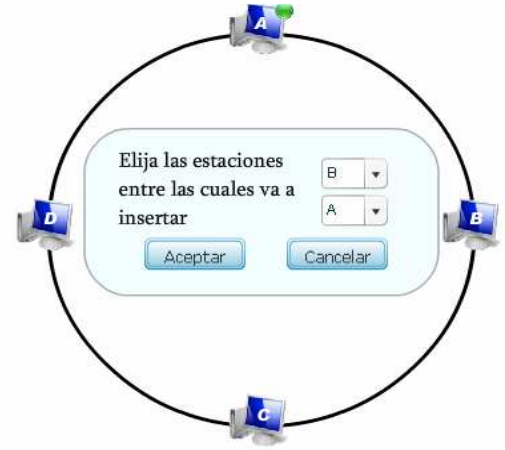

Figura 4.50: Ventana de la posición a insertar la nueva estación

4.4.7 Iteración 7: Simular la Eliminación de una Estación

La prueba de la eliminación de una estación, monitor en espera, se inicia pulsando el botón “Eliminar Estación” en la barra de herramientas o desde la barra de menú la opción “Simulaciones/Eliminar estación”. Una vez seleccionada la estación a eliminar, en este caso es la estación B como se muestra en la Figura 4.51, y presionada el botón “Aceptar”, comienza la simulación empezando con la estación que será eliminada hasta que todas las estaciones actualicen sus registros NAUN reconociendo sus nuevos vecinos como se muestra en la Figura 4.52.

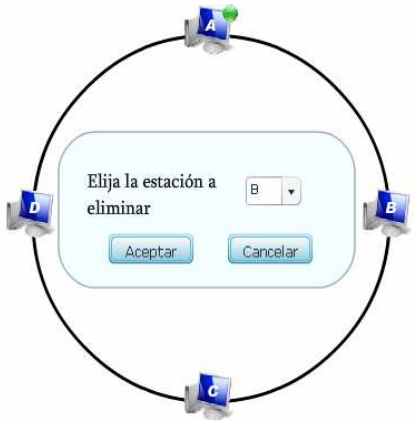

Figura 4.51: Ventana de eliminar una estación

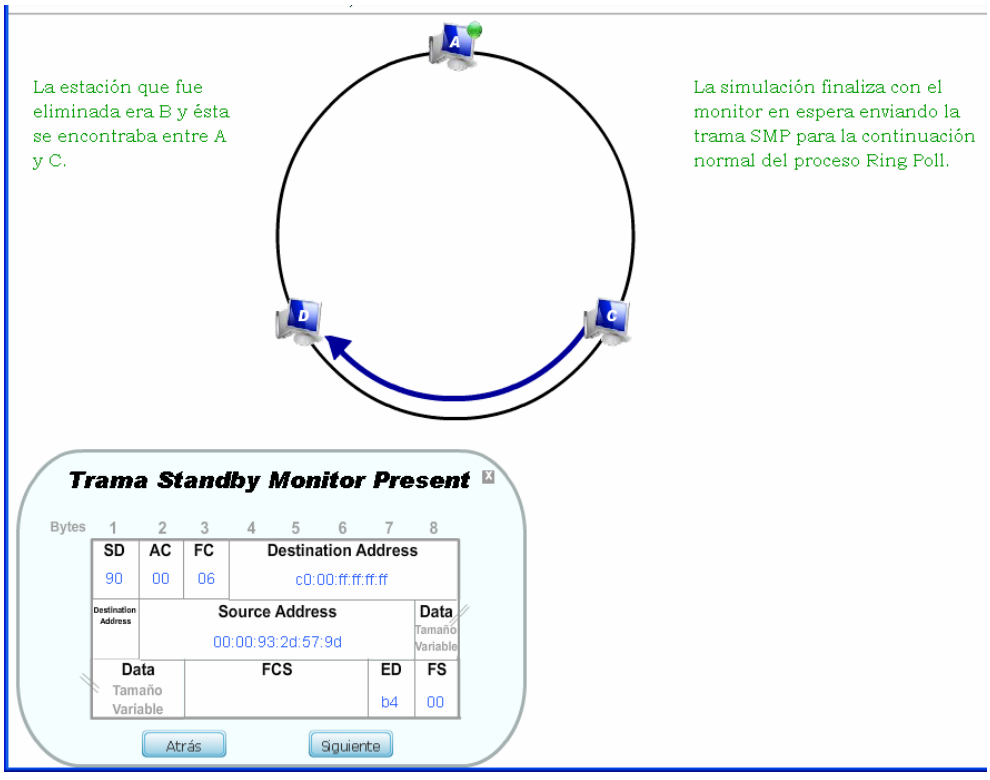

Figura 4.52: El resultado de la eliminación de una estación.

4.4.8 Iteración 8: Evaluación de Conceptos Básicos

La prueba de la evaluación de conceptos básicos se inicia pulsando la opción "Evaluación/Conceptos Básicos", allí se muestra la pantalla con la pregunta correspondiente, tal y como se puede ver en la Figura 4.53. Luego se verifica si suma 2 puntos por cada respuesta correcta y sigue avanzando la evaluación hasta que se muestren el resultado y la recomendación sobre las preguntas como en la Figura 4.54.

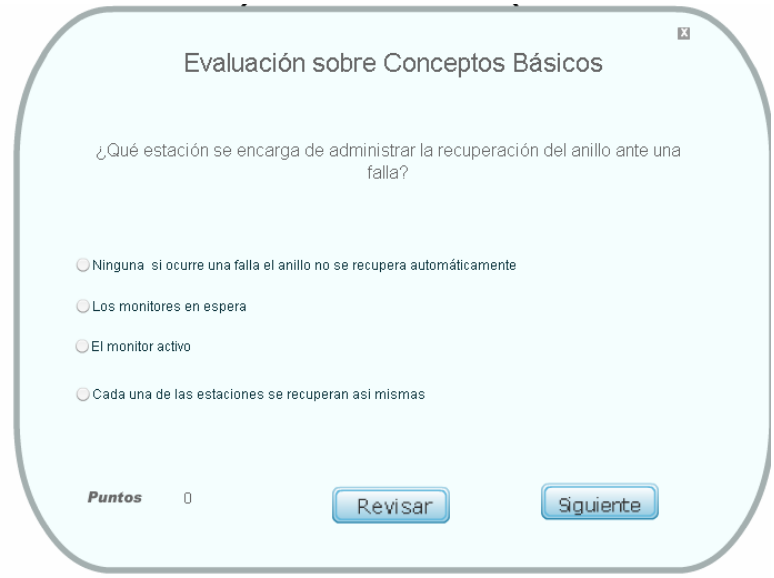

Figura 4.53: Ventana de la evaluación de Conceptos Básicos

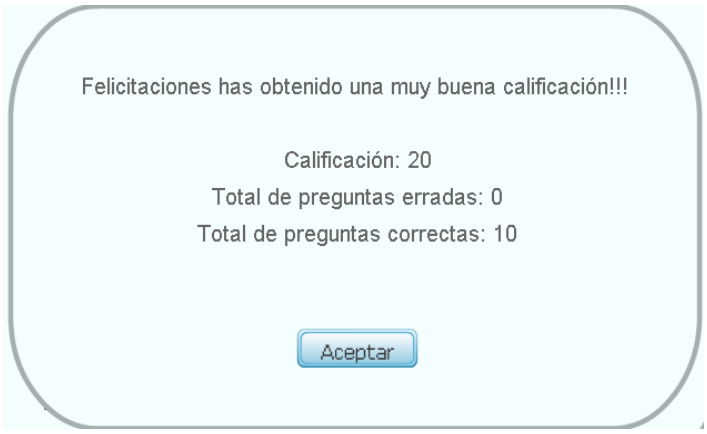

Figura 4.54: El resultado de la evaluación de Conceptos Básicos

4.4.9 Iteración 9: Evaluación de Tramas

La prueba se inicia pulsando la opción *Evaluación/Tramas* en la barra de menú. Igual que la evaluación de *Conceptos Básicos*, se verifican los puntos y el resultado de la evaluación y, también las preguntas correspondientes a las tramas de Token Ring.

4.4.10 Iteración 10: Evaluación de Procesos

La prueba de la evaluación de procesos se inicia por la opción *Evaluación/Procesos* en la barra de menú. Lo mismo de las 2 iteraciones anteriores, se verifican la suma de los puntos, las preguntas adecuadas y el resultado de la evaluación.

4.4.11 Iteración 11: Ayuda

La prueba se inicia pulsando la opción *Ayuda/Ayuda* en la barra de menú. Se verifica si al seleccionar un tema se muestra la explicación respectiva y el enlace de las páginas coherentes, tal y como se muestra en la Figura 4.55.

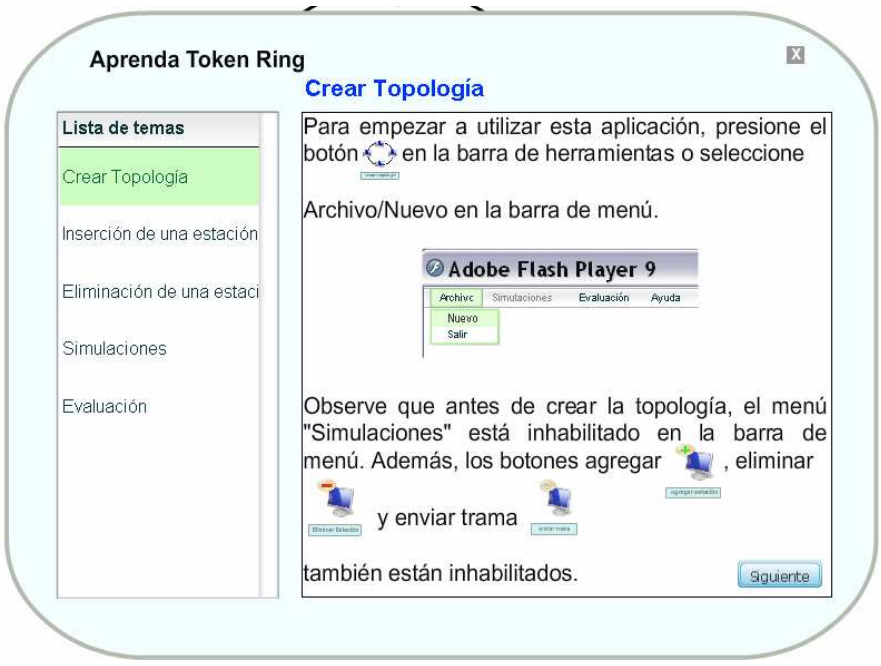

Figura 4.55: Ventana de Ayuda

4.4.12 Iteración 12: Glosario de Términos

La prueba se inicia pulsando la opción *Ayuda/Glosario de términos* en la barra de menú. Se verifica si al seleccionar un término se muestra el título y la explicación relativos como se muestra en la Figura 4.56 con AC.

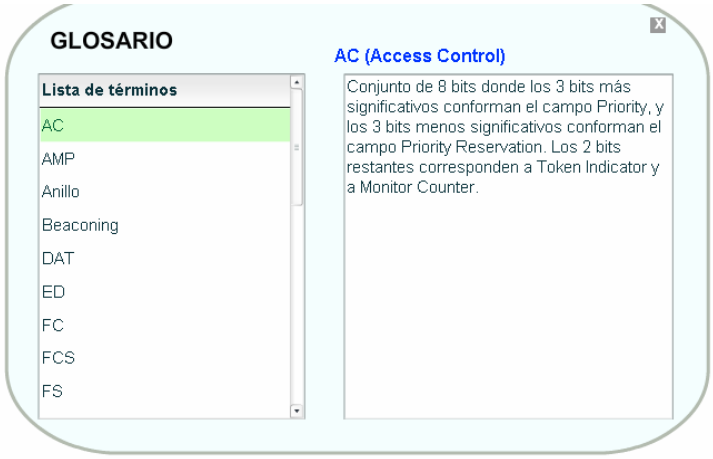

Figura 4.56: Ventana del glosario de términos

4.4.13 Iteración 13: Acerca de

La prueba se inicia pulsando la opción *Ayuda/Acerca de* en la barra de menú. Se verifica si se muestra la información correcta sobre la aplicación como se muestra en la Figura 4.57.

Figura 4.57: Ventana Acerca de

Todas las pruebas mencionadas de las iteraciones tuvieron resultados satisfactorios.

5. Conclusiones

Aprenda Token Ring es una aplicación didáctica que tiene como principal propósito mostrar el funcionamiento del estándar IEEE 802.5, mejor conocido como Token Ring, mediante simulaciones que van detallando el cambio en los campos pertenecientes a cada una de las tramas transmitidas. Adicionalmente, se cuenta con un módulo de evaluación que permite medir si se han obtenido los conocimientos relacionados con conceptos básicos, tramas y procesos. Y finalmente tiene un módulo de ayuda que permite al usuario contar con una guía para la utilización de esta herramienta y un glosario de términos que refresca el significado de la terminología utilizada en la aplicación.

Para su implementación fue utilizado Flash CS3 como lenguaje de programación, el cual permitió cubrir en su totalidad los objetivos planteados como Trabajo Especial de Grado. Dicho lenguaje, por ser relativamente nuevo, cuenta con un API que ha pesar de no ser muy extenso tiene lo necesario para el desarrollo de una aplicación visual, que provee facilidades y herramientas para desarrollar una interfaz usable.

Durante la etapa de análisis, se estudió de una manera profunda el funcionamiento del estándar IEEE 802.5, que es bastante completo y complejo, y que ha sido descontinuado por parte de los constructores. La documentación existente no está actualizada, sin embargo los estándares detallan todo el funcionamiento de las redes Token Ring.

Para el diseño de la aplicación, el enfoque fue tener una interfaz sencilla, usable, que permitiera entender de una manera más visual los procesos que ocurren en una red en forma de anillo.

El proceso de implementación de la herramienta se basó en iteraciones, de manera de seguir la metodología de desarrollo ágil. En cada una de estas iteraciones se realizó un segmento de código, el cual se fue integrando a la aplicación de manera de realizar las pruebas correspondientes. En caso de que existiera alguna discrepancia o algún problema en los datos, o bien sea en la interfaz de la aplicación, se realizaban los cambios necesarios hasta obtener los resultados esperados en cada iteración.

Si se comprara *Aprenda Token Ring* con el resto de los simuladores existentes en el mercado, se podrá notar que hasta ahora no existía una herramienta que mostrara el cambio en el formato de cada una de las tramas. Las pocas aplicaciones existentes no daban mayor detalle, es por ello que *Aprenda Token Ring*, viene a llenar un vacío existente en el ámbito didáctico.

Referencias Bibliográficas

[01] **Token Ring – FDDI**. Documento en línea:
www.monografias.com/trabajos27/redes-token-ring/redes-token-ring.shtml.

[02] **Token-Ring Technical Summary**. Documento en línea:
http://www.techfest.com/networking/lan/token.htm.

[03] **Token Ring / IEEE 802.5**. Cisco System. Documento en línea:
http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/tokenrng.pdf.

[04] **Redes y Teleprocesos, tecnología Token Ring**. Documento en línea:
www.monografias.com/trabajos/tokenring/tokenring.shtml.

[05] **Fundamentos de una red de área local**. Documento en línea:
http://www.geocities.com/SiliconValley/8195/redes.html.

[06] **Token Ring Technology Overview**. Cabletron Systems. Documento en línea:
http://secure.enterasys.com/support/manuals/hardware/1941.pdf.

[07] **Token Ring Network Analysis & Troubleshooting**. Network Associates. 1998

[08] **Introduction to Data Communications. Token Ring**. Documento en línea:
http://www.cadvision.com/blanchas/Intro2dcRev2/page176.html#176.

[09] **ISO 8802-5 Information Technology - Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks - Specific Requirements - Part 5: Token ring access method and Physical Layer specifications.** Robert D. Love, David W. Wilson y otros**.** Estándar ANSI/ IEEE 802.5 1998.

[10] **Token Ring**. Data Network Resource. Documento en línea:
http://www.rhyshaden.com/tokenr.htm.

[11] **Metodologías Ágiles en el desarrollo del Software.** José H. Canós, Patricio Letelier y M. Carmen Penadés. 2006. Documento en Línea:
http://www.willydev.net/descargas/prev/TodoAgil.pdf.

Glosario de Términos

AC (Access Control): Conjunto de 8 bits donde los 3 bits más significativos conforman el campo *Priority*, y los 3 bits menos significativos conforman el campo *Priority Reservation*. Los 2 bits restantes corresponden a *Token Indicator* y a *Monitor Counter*.

AMP (Active Monitor Present): Es una trama transmitida periódicamente (cada 7 segundos) por el monitor activo para indicar su presencia en el anillo y para inicializar el proceso de *Ring Poll*.

Anillo: Es una red punto a punto, en la cual los dispositivos están conectados máquina a máquina, en un circulo unidireccional cerrado.

Beaconing: El proceso de Beaconing es inicializado cuando una estación en el anillo ha detectado una falla grave (generalmente necesita la intervención humana) en el anillo. Una trama beacon es transmitida por una estación para notificar a otras estaciones en el anillo de la falla y la trama lleva la dirección de la estación que ejecuta el proceso y la dirección del vecino superior (NAUN). El objetivo del proceso es encontrar la estación que está ocupando el beacon en el anillo por el período de tiempo más largo.

DAT (Duplicate Address Test): Es una trama enviada por una estación durante su inserción al anillo para verificar que su dirección MAC es única.

ED (End Delimiter): Conjunto de 8 bits que indica el final de una trama, bien sea de un token o una trama de datos/comandos. Tiene un campo de 1 bit (E) que indica si hay un error en la trama y otro de 1 bit (I) que identifica si es la última trama de una secuencia.

FC (Frame Control): Indica si la trama contiene datos o información de control. En caso de que los dos primeros bits sean 00, se trata de una trama MAC. Si todos los bits están en 0 indica que la trama MAC será procesada usando un buffer de memoria normal. De lo contrario, si los dos primeros bits son 01, se trata de una trama LLC y los últimos tres bits corresponden al campo Priority de la trama.

FCS (Frame Check Sequence): Este campo es agregado por la estación fuente con un valor calculado dependiente del contenido de la trama. La estación destino recalcula este valor para determinar si la trama ha sufrido errores durante la transmisión. En caso de ser así, esta es descartada. El cálculo de este valor se basa en un polinomio estándar de grado 32, el cual se usa para aplicar la técnica de redundancia cíclica (CRC).

FS (Frame Status): Campo de 8 bits que está al final de la trama de datos/comandos. Se utiliza para identificar si la trama fue reconocida y copiada por el destinatario.

MAC (Medium Access Control): Una dirección MAC (Control de Acceso al Medio) es un identificador hexadecimal de 48 bits que está asignada de forma única a una interfaz de red.

MAU (Multistation Access Unit): Una unidad de interfaz que contiene un pequeño transformador de aislamiento para cada dispositivo conectado, el cual controla el flujo de datos.

MC (Monitor Count): Campo de un bit usado por el monitor activo para saber si la trama ha circulado más de una vez por el anillo.

Monitor Contention Process: El proceso se inicia para definir el monitor activo de una red Token Ring y todas las estaciones participan en la negociación para decidir quién controlará el anillo.

NAUN (Nearest Active Upstream Neighbor): Es el vecino superior más cercano de una estación en un anillo. Cada estación aprende quién es su NAUN por medio del proceso de *Ring Poll* donde tramas *AMP* y *SMP* son enviadas.

PCF (Physical Control Field): Los últimos cuatro bits de *FC*, los cuales muestran el código del tipo de información de control que se transmite.

Priority: Los primeros tres bits de *AC* corresponden al campo *Priority*. La prioridad permite a las estaciones transmitir paquetes con ocho niveles diferentes, donde '000' es la prioridad más baja y '111' es la más alta.

Priority Reservation: Priority Reservation se ubica en los últimos tres bits de *AC*. Una estación usa estos bits de reservación para requerir que un token sea originado en el anillo con la prioridad deseada. Una estación puede cambiar la reservación siempre y cuando necesita enviar una trama cuya prioridad es mayor a la reservación. Una estación que origina un token con un incremento en la prioridad, más adelante, deberá bajar la prioridad a su valor original.

RI (Request Inicialization): La trama *MAC RQ_INIT* es enviada por una estación al *RPS* (Ring Parameter Server) como parte del proceso de agregación al anillo. Esta informa que una estación ha sido insertada y aceptará los parámetros notificados desde *RPS* o *CRS* (Configuration Report Server).

Ring Poll: Es un proceso de notificación al vecino que es iniciado por el monitor activo cada siete segundos. Permite que todas las estaciones en la red descubran quién está participando en el anillo además de aprender la dirección de su NAUN.

RNC (Report NAUN Change): Mensaje enviado por una estación a todas las demás estaciones notificándoles el cambio del NAUN de la estación emisora.

RP (Ring Purge): Consiste en que el monitor activo envía un mensaje de este tipo para reajustar el anillo después de que una interrupción o una pérdida de datos se divulgue. Cuando reciben un mensaje de este tipo, las estaciones se ponen en modo de repetición normal, cancelando y reiniciando todos los temporizadores correspondientes.

SD (Start Delimiter): Un campo de 8 bits que indica el inicio de un token o de una trama, e incluye una señal que lo distingue del resto de los bytes de la trama.

SMP (Standby Monitor Present): La trama *MAC SMP* es transmitida por los monitores en espera como parte del proceso *Ring Poll*.

Token: Es una trama corta (3 bytes) que le permite transmitir a la estación que la tiene.

Token Indicador: Es un campo de un bit que permite diferenciar un token de una trama de datos/comandos. Si está en '1' significa que es un token, si está en '0' entonces es una trama datos/comandos.

Trama: Es el PDU (Protocol Data Unit) de la capa de enlace de datos.