PROTOCOLO PDM-RING PARA REDES EN ANILLO – SIN COLISIONES Y BASADO EN MULTIPLEXACIÓN DE POTENCIA

ALEXANDER GARCÍA DÁVALOS
CODIGO 974622
MARY ELIZABETH RAMÍREZ CANO
CODIGO 974563

INSTITUTO TECNOLÓGICO DE MONTERREY
UNIVERSIDAD AUTONOMA DE BUCARAMANGA
MAESTRIA EN CIENCIAS COMPUTACIONALES
SANTIAGO DE CALI – COLOMBIA
2005

PROTOCOLO PDM-RING PARA REDES EN ANILLO – SIN COLISIONES Y BASADO EN MULTIPLEXACIÓN DE POTENCIA

ALEXANDER GARCÍA DÁVALOS
CODIGO 974622
MARY ELIZABETH RAMÍREZ CANO
CODIGO 974563

Tesis de Maestría para optar al título de Magíster en Ciencias Computacionales

Director de Tesis:
Dr. Arturo Galván Rdz. Ph.D
Profesor Investigador del Centro de Sistemas Inteligentes
Instituto Tecnológico de Monterrey

INSTITUTO TECNOLÓGICO DE MONTERREY
UNIVERSIDAD AUTONOMA DE BUCARAMANGA
MAESTRIA EN CIENCIAS COMPUTACIONALES
SANTIAGO DE CALI – COLOMBIA
2006

Nota de aceptación:

_____
_____
_____
_____
_____
_____

_____
Firma del presidente del Jurado

_____
Firma del Jurado

_____
Firma del Jurado

Santiago de Cali, 24 de Marzo de 2006

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

RESUMEN

En la actualidad el reto de las comunicaciones es satisfacer la demanda de transmisión de gran cantidad de datos en el menor tiempo posible. Esto implica, que los protocolos que se diseñan deben usar de forma eficiente el medio de transmisión, maximizando el ancho de banda disponible. Ejemplos clásicos de esta situación se presentan en la transmisión de vídeo de alta calidad ó el control en redes industriales.

En particular en el sector de las telecomunicaciones, se ha invertido gran cantidad de tiempo y esfuerzo en maximizar el aprovechamiento de los medios físicos de comunicación, sin importar su naturaleza, este hecho conlleva a que la investigación en este campo nunca termine y se hagan esfuerzos y propuestas cada vez mejores que buscan transmitir mayor cantidad de datos en el menor tiempo posible.

El presente trabajo es una propuesta en este sentido, ya que trata acerca del diseño de un nuevo protocolo de comunicaciones denominado *PDM-Ring* que hace uso eficiente del medio físico y minimiza la pérdida de datos basándose en la tecnología multiplexación por división de potencia, concepto que tienen amplia aplicación en el campo de la comunicación por radio, pero que gracias a los avances tecnológicos que ofrecen los nuevos dispositivos electrónicos tales como, alta capacidad de almacenamiento y gran velocidad, es posible ahora aplicar en las redes cableadas.

Es importante aclarar que, el protocolo propuesto será implementado en un nuevo dispositivo electrónico de comunicaciones que está en proceso de construcción, pero que no hizo parte del trabajo, por tanto, el interés fundamental en este proyecto fue demostrar que el diseño del protocolo PDM-Ring para redes en anillo sin colisiones y basado en Multiplexación de Potencia es correcto, es decir, el protocolo no posee inconsistencias y su implementación es completamente viable desde el punto de vista lógico.

Para la validación del protocolo propuesto se utilizo una herramienta de validación automatizada qe se denomina SPIN y los resultados obtenidos en este proceso permiten afirmar que el diseño del protocolo propuesto es correcto.

1. INTRODUCCIÓN

1.1. ANTECEDENTES

La motivación del presente proyecto surgió de la necesidad del ingeniero Iván Herrera[1] de construir un dispositivo electrónico, denominado *DCIH*[2] , que permite la comunicación en Tiempo Real debido a que se aprovecha al máximo el ancho de banda del medio físico que comunica varios de estos dispositivos. Este aprovechamiento se logra porque el dispositivo permite que varios usuarios envíen, de forma simultánea, los datos que requieren transmitir en un mismo instante de tiempo, evitando así, que se vean afectados por los tiempos de espera a los que se verían enfrentados si tuviesen que esperar a que el medio esté libre porque otro usuario se ha apoderado de él.

El proceso relacionado con la propiedad intelectual del dispositivo DCIH se encuentra en proceso de patentización según documento radicado ante la Superintendencia de Industria y Comercio bajo el número 46525 del 2002.

De otra parte, para el correcto funcionamiento del dispositivo DCIH se requiere del diseño y la validación del protocolo de comunicación que permite transmitir, en Tiempo Real, datos entre dos o más dispositivos DCIH dispuestos en una red tipo anillo. Esta fase es la que es de interés en este proyecto, ya que el diseño del dispositivo está completo y a hoy se han realizado pruebas de laboratorio y se encuentra en construcción.

El diseño y la validación de protocolos de comunicación es una tarea que requiere del seguimiento meticuloso de varias fases, entre las que se incluye el conocimiento del servicio que presta, las reglas que rigen su comportamiento y la verificación y validación de las mismas; generando por tanto el diseño detallado de un producto de software que cumple con lo especificado anteriormente. Debido a las características del presente trabajo, el grupo que lo desarrolló considera que posee las características para ser considerado como proyecto de tesis de Maestría dado el aporte al conocimiento en el área de la comunicación en redes.

[1] Iván Herrera Murgueitio, Ingeniero en Electrónica graduado en la Universidad del Cauca – Popayán (Colombia) y Especialista en Redes de la Universidad del Valle – Cali (Colombia).
[2] *Dispositivo de Comunicaciones Iván Herrera*

1.2. PROBLEMA

Actualmente, en las redes que se utilizan para el control de los procesos industriales se cuenta con varios niveles, a saber:

- Nivel 0, los procesos industriales.
- Nivel 1, red de sensores y actuadores (nivel de campo).
- Nivel 2, equipos para control y monitoreo.
- Nivel 3, sistema informático para administración (almacenamiento de datos e históricos).
- Nivel 4, red empresarial (Intranet) en la cual se tienen las aplicaciones del negocio (p.ej. ERP – **Enterprise Resource Planing**).

Ethernet se utiliza en los niveles altos (nivel 3 y 4) y en el nivel 2 para la conexión de los equipos de control y monitoreo. Adicionalmente, algunas organizaciones han especulado acerca de la posible convergencia de **Ethernet** + **Fieldbus**[3] en el nivel de campo, lo cual es atractivo para las redes industriales debido a la sencillez de la estructura de **Ethernet** y la posibilidad de conexión de un gran número de dispositivos, pero desafortunadamente, éste no garantiza algunas de las caracteristicas requeridas en el nivel de campo[4] (v.gr. tiempo de respuesta) y además **Ethernet** presenta un pobre desempeño durante los instantes de sobrecarga de la red[5]. Desde la perspectiva del grupo de trabajo, las causas principales de este hecho son:

- El mecanismo de acceso al medio, que conlleva a que los usuarios tienen que someterse a tiempos de espera para acceder a él.
- El diseño del protocolo, que es el encargado de definir la estructura del mensaje y las reglas que rigen el tratamiento del mismo.

Ante el problema planteado por Ethernet y su mecanismo de acceso al medio, el Ing. Ivan Herrera Murgueitio propuso una solución consistente en el diseño de un nuevo dispositivo de comunicaciones orientado a las redes industriales denominado DCIH, el cual tiene como particularidad que se configura en una red en anillo y que para el acceso al medio utiliza un tipo de multiplexación diferente a

---

[3] Fieldbus - Bus de campo, es una red del nivel de campo (sensores/actuadores) que permite la conexión de gran cantidad de dispositivos en una topología tipo bus. Existen diversas implementaciones de este tipo de red como por ejemplo Modbus, Profibus, y Foundation FieldBus.

[4] Berge J., «FieldBus, Ethernet and the Reality of Convergence», technical article, The Industrial Ethernet Book, Issue 23, November 2004. Disponible en la URL: http://ethernet.industrial-networking.com/ieb/articles.asp (última consulta marzo 3 de 2006).

[5] LONWORKS, White Paper. «Determinism in Industrial Computer Control Network Applications». Echelon, Enero1995. http://www.echelon.com/support/documentation/bulletin/005-0060-01A.pdf (consultada el 11 de noviembre de 2003)

los tradicionales, el cual permite aprovechar mejor el canal de comunicaciones. A partir de esta solución, el problema se reduce a *¿ cuál es el diseño del protocolo que permite establecer comunicación en Tiempo Real entre dispositivos DCIH de una red tipo anillo, que hace uso eficiente del medio y que minimiza la pérdida de datos teniendo canales únicos e independientes ?*.

1.3. JUSTIFICACIÓN

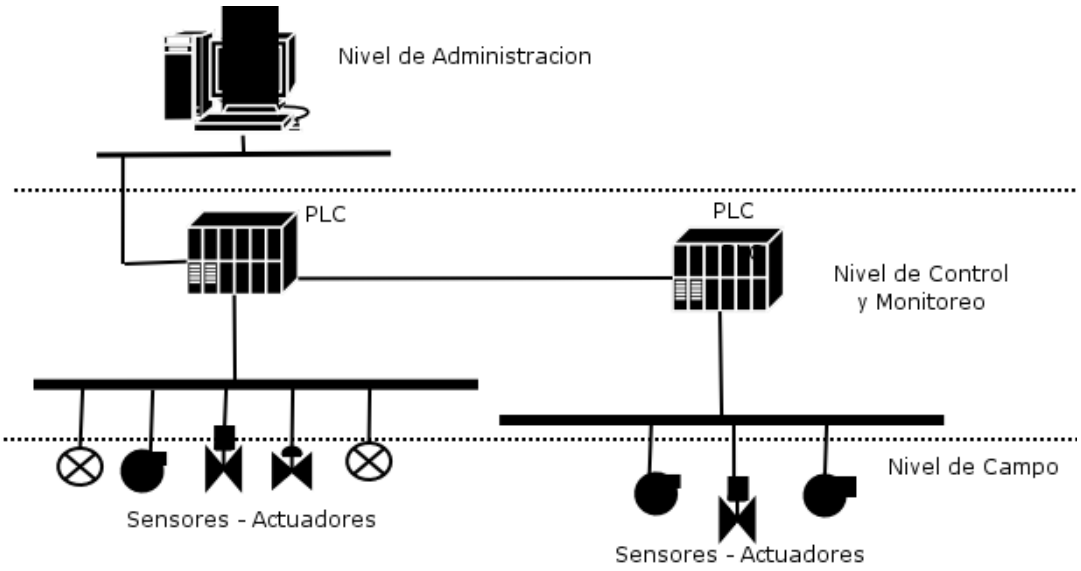
La justificación del proyecto se basa en el beneficio económico desde varias perspectivas:

- Disminución en costos del hardware y material requerido para la instalación de la red.
- Disminución en costos de la mano de obra en la instalación de la red.
- Mejora en los tiempos de transmisión, respuesta y espera.
- Mejora la calidad en el servicio de transmisión.
- Apertura de una alternativa de mercado de telecomunicaciones en Tiempo Real.

1.3.1 Costos en hardware y materiales.
Para ejemplificar el ahorro en este rubro, se ilustra un caso en el que se tipifica una red industrial, para procesos que requieren de Tiempo Real, basada en componentes disponibles a hoy. Posteriormente se compara dicha red con una que cumple los mismos requerimientos diseñada con base en dispositivos DCIH.

Una red industrial que soporta procesos que requieren Tiempo Real se muestra en la Figura 1.

Figura 1. Red Industrial Típica



El nivel de administración no tiene incidencia en el proyecto y tampoco será de interés, pero el nivel de control y monitoreo si, por esta razón, son los componentes que lo conforman los que se consideran en este análisis económico.

Típicamente se establecen dos componentes en el nivel de control y monitoreo y son el sistema central, compuesto por los **PLC**[6] o Controladores Centrales y los periféricos (tarjetas que se insertan en el PLC), encargados de recibir ó enviar datos analógicos desde los sensores o hacia los actuadores.

Para el caso específico de los periféricos, el del tipo **PXI-6115** de **National Instruments** con 4 entradas analógicas tiene un costo de US$4395, lo que implica un costo promedio de US$1099 por entrada analógica. Dado que este esquema funciona centralizado, la unidad de Control Central también debe adquirirse, esta unidad varía de acuerdo a las utilidades que ofrece, para el caso se toma como ejemplo la **PXI-8156B/333 RT** de **National Instruments** a un costo de US$ 3845, lo que daría un total de US$ 8240. Los costos[7] aquí planteados no incluyen ningún rubro por el software requerido para administrar los equipos identificados.

En este modelo una vez la información de cada uno de los dispositivos se ha digitalizado y sale del PLC hacia el Controlador Central, se comparte el mismo medio de comunicación, lo que implica que no hay disponibilidad completa del medio para cada dispositivo que transmitió.

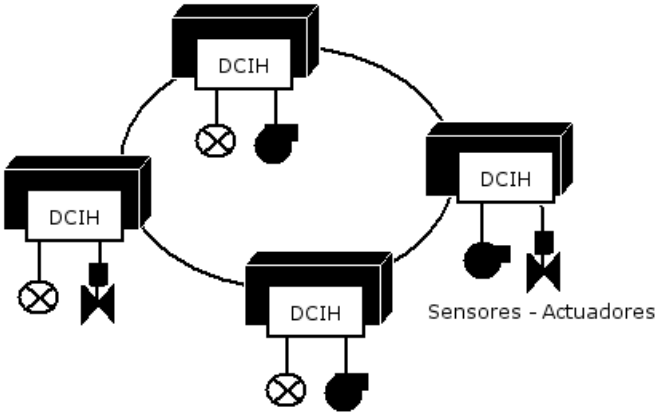
Si se piensa en una red similar a la planteada anteriormente, habrá equivalencia entre los PLC y los dispositivos DCIH, pero ahora estos se conectarán en anillo. Para el caso específico se hizo mención de un PLC de 4 entradas analógicas éste se reemplazaría por 4 dispositivos DCIH dispuestos en una red en anillo. En la Figura 2 se ilustra este planteamiento.

Un dispositivo DCIH cuesta aproximadamente US$ 200[8], mientras que el costo de las entradas analógicas del PLC es de US$1200.

Aunque es evidente el ahorro en este rubro, la mayor justificación económica del proyecto está directamente relacionada con el protocolo de comunicacion, por sus características de tiempo y calidad.

[6] Programmable Logic Controller – Autómata Programable o Controlador Lógico Programable.
[7] Los valores fueron consultados en el sitio web del fabricante – http://sine.ni.com/nips/cds/view/p/lang/en/nid/11885 (última consulta diciembre 2 de 2005)
[8] Valor calculado por el Ing. Iván Herrera Murgueitio.

Figura 2. Red de dispositivos DCIH

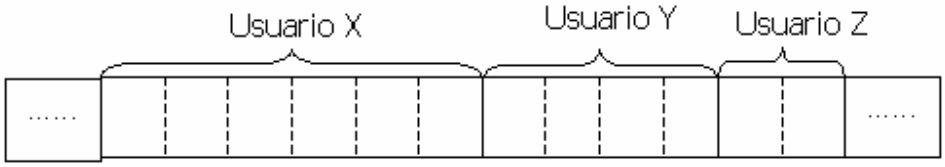


1.3.2 Mejora en el tiempo de transmisión.

Como se mencionó anteriormente, en muchas de las redes industriales en el nivel de administración se utiliza **Ethernet**, la cual se complementa con la pila de protocolos TCP/IP brindando la posibilidad de implementación de diferentes soluciones en el nivel de aplicacion[9]. A este tipo de redes se les conoce como Ethernet Industrial.

En las redes **Ethernet** se utiliza una técnica de acceso al medio conocida como CSMA/CD[10] (**Carrier Sense Múltiple Access with Collision Detection** – Acceso Múltiple con Detección de Portadora y Detección de Colisiones), en la cual cada vez que un usuario ocupa el medio lo hace completamente, lo que se traduce en que la simultaneidad de transmisión en realidad no se da, lo que sucede entonces es que cada usuario ocupa el medio completamente por un instante de tiempo y otro usuario que desea transmitir lo hace detrás del último que tuvo el derecho de hacerlo, dando la apariencia de simultaneidad porque ambos usuarios transmitieron. La Figura 3 ilustra lo anteriormente expuesto.

Figura 3. Transmisión de datos en Ethernet



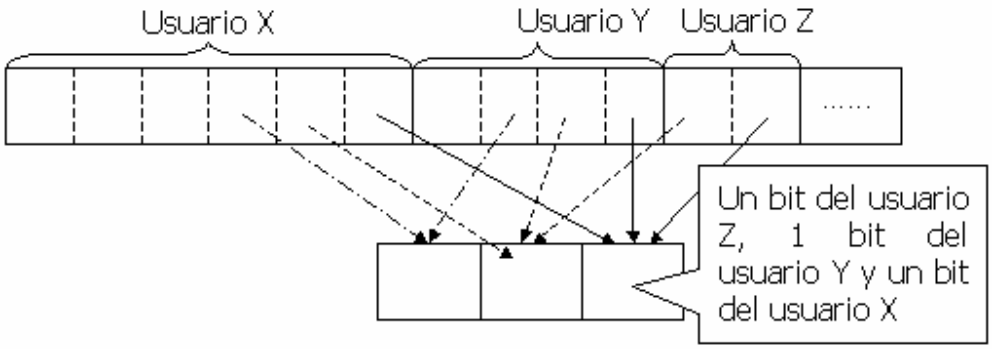
[9] Berge J., «Ethernet in Process Control», Technical article, The Industrial Ethernet Book, Issue 3, June 2000. Disponible en la URL: http://ethernet.industrial-networking.com/ieb/articles.asp (última consulta marzo 3 de 2006).

[10] STALLINGS, William. «Comunicaciones y Redes de Computadores», Sexta Edición, Editorial Prentice Hall, Madrid (España), 2000. Pag. 438-443.

Para simplificar el ejemplo, se supone que cada casilla de la Figura 3 representa la transmisión de un bit de cada usuario (en realidad es mayor el número de bits que transmite cada usuario y esto depende del protocolo bajo el que se han transmitido los datos, en **Ethernet** la trama mínima es de 512 bits), como se evidencia en la Figura 3, el usuario *Z* transmite 2 bits, luego el usuario *Y* transmite 4 bits, posteriormente el usuario *X* transmite 6 bits. Lo que refleja la Figura 3 es que durante el periodo que dura la transmisión de cada usuario, el medio está ocupado solo por el bit que ese usuario transmite y el usuario que desee transmitir deberá esperar a que el medio se libere para poder iniciar su transmisión.

Desde el punto de vista del protocolo que se propone, una de las mejoras fundamentales se refiere al hecho de transmitir un bit de cada uno de los usuarios que desean transmitir en un momento determinado y de forma simultánea. Ahora la misma situación puede apreciarse en la Figura 4.

Figura 4. Transmisión de datos en el Protocolo Propuesto

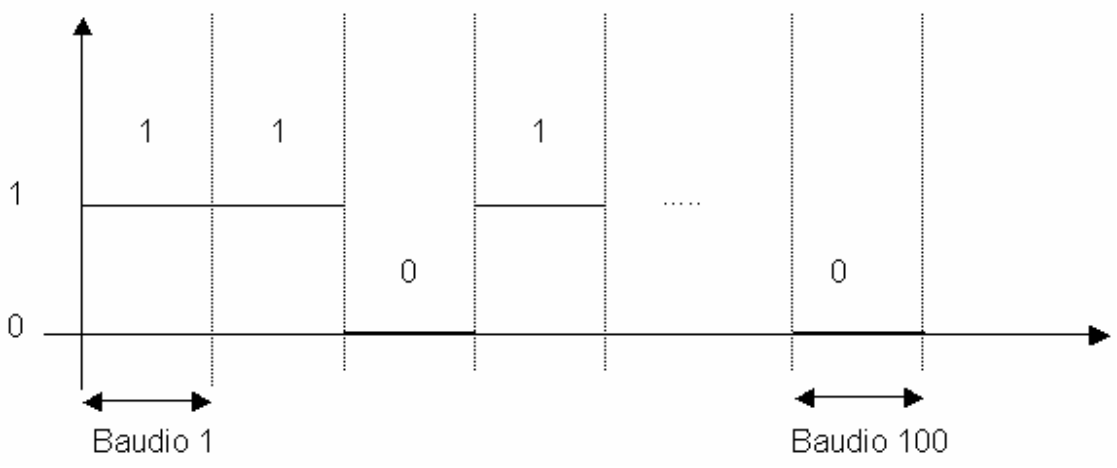


Para este caso se supone que los usuarios *X*, *Y* y *Z* desean transmitir en el mismo instante de tiempo, entonces el protocolo se encarga de empaquetar los tres bits de tres usuarios distintos en un mismo paquete. Si se sigue con este proceso es evidente que el tiempo que demora la transmisión de dicha información es mucho menor que para el caso anterior. De otra parte, se mejora el aprovechamiento del medio, ya que varios usuarios transmiten realmente de forma simultánea sus bits y ninguno de ellos se ve abocado a la espera de la disponibilidad del medio para su uso.

Para ejemplificar se plantea la siguiente situación:

Suponga que se cuenta con un enlace serial dedicado punto a punto basado en multiplexación de Tiempo (TDM), en el que no se presenta colisión y suponga también que la velocidad de señalización es de 1 Mega Baudio por segundo implicando esto que, por cada baudio de señalización se transmite un solo bit. Si se desea transmitir una trama de 100 bits (para este caso equivalentes a 100 baudios) se requiere de 0.0001 segundos, es decir 100 μs. La Figura 5 ilustra este hecho.

Figura 5. Transmisión de datos tradicional



Si se asumen las mismas condiciones del caso anterior, pero ahora el sistema de multiplexación se basa en PDM[11], y si se supone que se tienen 5 canales sobre un mismo medio (haciendo uso de PDM), para transmitir los mismos 100 bits no se requerirían los 100 baudios del caso anterior, se podrían multiplexar haciendo uso de los 5 canales, dado que para una señal analógica es posible que en un baudio viajen varios bits, entonces se requerirían de 0.00002 segundos, equivalentes a 20 μs. En la Figura 6 se ilustra este hecho.

1.3.3 Mejora en gestión de calidad de los datos

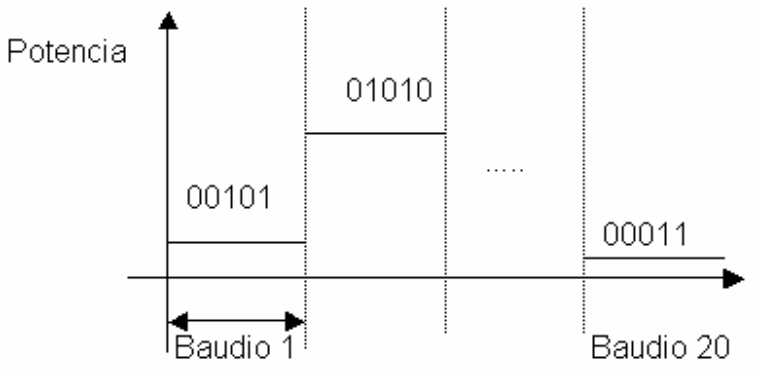
Es evidente que si se mejora el tiempo de transmisión se obtiene una mejora en la calidad del servicio, ya que el tiempo que toma en transmitirse un paquete de bits disminuye y en consecuencia el tiempo de transmisión de los datos también lo hará.

Por otra parte, no se presenta tiempo de espera en la transmisión de un usuario en particular. Para explicar este punto, se toma como base el mismo ejemplo del numeral anterior. Si se observa la Figura 3, el usuario *X* estuvo en capacidad de

[11] Power Division Multiplex – Múltiplexación por División de Potencia. En el capítulo 2 se explica el concepto en detalle.

transmitir porque el usuario Y culminó su transmisión, esto ocasiona un tiempo de espera en la transmisión para el usuario *X*. Ahora si se analiza la misma situación en la Figura 4, cada usuario transmite cada uno de sus bits sin tener que esperar a que el medio se encuentre disponible, en este sentido la calidad del servicio se incrementa ya que no hay retardos de transmisión adicionales por esperar la disponibilidad del medio.

Figura 6. Transmisión Propuesta



1.4. OBJETIVOS

1.4.1 General

Diseñar, validar y simular, a nivel lógico, un protocolo para la comunicación entre dispositivos DCIH en una red tipo anillo, teniendo canales únicos e independientes para cada usuario, por un mismo medio de transmisión y utilizando Multiplexación de Potencia.

1.4.2 Específicos

- Identificar las características de operación del dispositivo DCIH a través de sus especificaciones de diseño.
- Diseñar el protocolo de comunicación para una red tipo anillo de dispositivos DCIH.
- Validar el protocolo a nivel lógico.
- Someter el protocolo a pruebas de simulación.

1.5. HIPOTESIS

Consiste en la validación del protocolo diseñado para la comunicación entre dispositivos DCIH en una red tipo anillo, teniendo canales únicos e independientes para cada usuario, por un mismo medio de transmisión y utilizando Multiplexación de Potencia.

1.6. ORGANIZACIÓN DEL DOCUMENTO

El desarrollo del presente trabajo se ha oganizado de la siguiente forma:

- Marco Teórico, resume el estado del arte de las investigaciones en el área de protocolos de comunicación y se describen claramente los conceptos fundamentales del proyecto, tales como PDM y DCIH entre otros.

- Metodología, describe y explica en detalle el diseño del protocolo siguiendo la metodología que se adoptó.

- Conclusiones, se identifican los aspectos relevantes basados en los resultados del trabajo que se desarrolló.

- Trabajos futuros, se describen los proyectos futuros que pueden basar su desarrollo en el diseño del protocolo propuesto.

- Anexos, en ellos se muestra el código fuente de los modelos implementados, el resultado de la simulación y la validación correspondientes.

2. MARCO TEÓRICO

Se entiende por una red de comunicaciones, a un conjunto de nodos (dispositivos) que permiten la transmisión de datos entre ellos, a través de canales establecidos. Existen diversas maneras de estructurar la conexión entre los nodos, lo cual se conoce comúnmente con el término de topología de red. Una topología común a nivel industrial es la denominada topología en anillo, en la cual los dispositivos se encuentran conectados a través de un medio de transmisión en el que el flujo de datos sigue una ruta circular y en la mayoría de los casos fluyen en una sola dirección. Cada vez que un nodo recibe un mensaje, ejecuta una subrutina que le permiten decidir si este mensaje es para él (y lo toma) o no, y en cualquier caso después de este proceso lo retransmite al siguiente nodo en el anillo.

Dentro de la topología en anillo, existen dos tipos de configuraciones comunes: **Token Ring** y FDDI (**Fiber Distributed Data Interface**), siendo la más popular la primera de ellas.

Una red **Token Ring** consiste en un anillo de nodos conectados punto a punto que basa su funcionamiento en que un pequeño paquete (sin datos) denominado "**token**" (*testigo*), circula a través de la red mientras no haya ningún nodo que requiera el medio para transmitir. Cuando algún nodo requiera efectuar la transmisión de un paquete, debe obtener la posesión del token, modificar el bit que indica que está ocupado (se convierte en un "**frame**"), insertar los datos e iniciar la transmisión. Si un nodo recibe el **token** y no tiene información para transmitir, éste lo pasa al siguiente nodo. Dado que solo existe un **token**, solo un nodo podrá transmitir en un momento determinado.

Este tipo de red representa algunas ventajas:

- Son raros los embotellamientos, bastante frecuentes en otro tipo de topología, ya que en el medio de transmisión en un instante de tiempo, solo hay un nodo transmitiendo.
- Lo anterior implica que no es necesario implementar un estricto control de colisiones.
- Tienen un desempeño relativamente estable.

En contraprestación presenta algunas desventajas:

- Todos los nodos del anillo están conectados entre sí por un mismo medio de transmisión, lo que implica que si se daña alguna sección del anillo, se interrumpe todo el flujo de comunicación.
- El medio de transmisión se encuentra ocupado en un instante de tiempo por un solo nodo, implicando desperdicio en el uso del ancho de banda del mismo.

- Puede presentarse con frecuencia un retraso entre el instante de tiempo en el que el nodo desea transmitir y la obtención del token para poder hacerlo. Si por alguna razón el **token** se ha perdido, habrá un tiempo de retraso mayor.

Además, de la topología usada para conectar los nodos de una red, es importante definir las reglas que rigen el acceso al medio que comunica los nodos, en este sentido es común utilizar la técnica conocida como TDM[12] (**Time Division Multiplexing** - *Multiplex por División de Tiempo*), y en especial una técnica asociada a la anterior denominada CSMA/CD (**Carrier Sense Múltiple Access with Collision Detection** – *Acceso Múltiple con Detección de Portadora y Detección de Colisiones*), que consiste en que cada nodo tiene asignado un periodo de tiempo en el medio de transmisión que se reparte de manera equitativa entre los distintos nodos; u otra técnica similar a la anterior, llamada CSMA/CA (**Carrier Sense Múltiple Access with Collision Avoidance** - *Acceso Múltiple con Detección de Portadora Evitando Colisión*), consistente en que un dispositivo que desee transmitir evalúa la disponibilidad del medio hasta que logre acceder a él, pero a diferencia de CSMA/CD, ésta provee un mecanismo de prioridades con el ánimo de evitar las colisiones[13]. Ambas técnicas tienen la desventaja de no ser Tiempo Real debido a que, al utilizar señalización binaria por el canal para la transmisión de mensajes entre los nodos, el sistema debe someter a turnos a los nodos que desean transmitir al mismo tiempo; esto se explica porque el canal solo admite la transmisión de un solo nodo al mismo tiempo. Esta situación llevó a que en las décadas de los años ochenta y noventa, se adelantara, a nivel mundial, una ola de investigaciones en el campo de los protocolos de comunicaciones para sistemas en Tiempo Real, teniendo como principal requerimiento el tiempo de respuesta, que debe ser mínimo dado que las aplicaciones en las que se hace uso del mismo se catalogan como de misión crítica.

En la mayoría de los casos, los investigadores tomaron como base protocolos existentes tales como TDMA (**Time Division Multiple Access** - *Múltiple Acceso por División de Tiempo*), Token Passing (paso de testigo) y CSMA para el diseño de los nuevos protocolos.

Una de las versiones más conocidas de CSMA es el protocolo Ethernet (IEEE 802.3), utilizado ampliamente en las redes de computadores, pero que presenta un bajo desempeño en las redes industriales con alta carga de trabajo. CSMA se puede considerar como un protocolo determinístico si se usa en entornos maestro-esclavo, en cambio cuando se usa en comunicaciones par-a-par (**peer-to-peer**)

[12] BLACK, Uyless. Redes de Computadoras Protocolos, normas e interfaces. Segunda edición. Rama, 1997.
[13] TANENBAUM, A. S. «Redes de Computadores» Tercera Edición. Prentice Hall, 1997. pag. 252-254.

se comporta como un protocolo no determinístico debido a que los nodos no tienen un acceso equitativo a la red[14].

A nivel académico se han realizado estudios interesantes como el de M. R. Lima en la Universidad de Campinas (Sao Paulo - Brasil), quien propone un protocolo en Tiempo Real que utiliza CSMA/CD para el modo de operación normal, es decir, cuando no hay colisiones en el canal, y otra técnica basada en TDMA, denominada TDMA/O (**TDMA Optimised**) cuando se presenten colisiones. Con esta técnica, a cada nodo se le asigna una "ventana de tiempo" (**time window**) para transmitir. Al terminar la transmisión de los mensajes que originaron la colisión, la red regresa a su estado de operación normal con CSMA/CD[15].

En los años 80 a nivel industrial surge el protocolo **HART** que se basa en el sistema de comunicación telefónica estándar **BELL 202** y opera usando modulación de frecuencia (FSK). Este protocolo funciona en un ambiente de conexión con topología en bus, y permite dos regímenes de trabajo punto a punto y multipunto con configuración maestro-esclavo. Entre sus ventajas se puede mencionar que brinda una alternativa económica de comunicación digital e implica un ahorro considerable en materiales eléctricos en las instalaciones multipunto[16].

Uno de los nuevos protocolos ha sido CAN (**Controller Area Network**), creado por la firma alemana Bosch y la norteamericana Intel, que soporta Tiempo Real. Este protocolo opera a una velocidad de 1 Mbit/seg, usa CSMA/CA y una topología tipo bus de difusión (broadcast) para la interconexión de los nodos. Los mensajes que son transmitidos a través del bus, poseen una identificación única y una prioridad[17]. Básicamente, se utiliza en sistemas empotrados (**embeded systems**), y particularmente en la industria automotriz.

En cuanto a las investigaciones realizadas tomando como base los protocolos de paso de testigo (**token**), se puede mencionar el trabajo de los investigadores italianos M. Conti y L. Donatiello, cuya propuesta está basada en Token Ring y consiste en un protocolo denominado RT-Ring, diseñado para operar en una topología de red tipo anillo, y que puede manejar tráfico tanto en entornos normales (genéricos), como de Tiempo Real. El protocolo RT-Ring, provee acceso concurrente a la red y una política de reutilización del espacio en el canal

[14] LONWORKS, White Paper. «Determinism in Industrial Computer Control Network Applications». Echelon, Enero1995. http://www.echelon.com/support/documentation/bulletin/005-0060-01A.pdf (consultada el 11 de noviembre de 2003)
[15] LIMA, M.R., LEITE J.C.B., LOQUES O. G. «A Real-Time Communication Protocol», Real Time, 1990. Proceedings. Euromicro '90 Workshop on . 6-8 June 1990. pp 129 - 134.
[16] HART Communication Foundation. HART Protocol Overview. http://www.hartcomm.org/ . (última consulta el 3 de diciembre de 2005).
[17] ISO/DIS 11898. «Road Vehicles – Interchange of Digital Information – Controller Area Network (CAN) for High Speed Communications». Febrero 1992.

"*ranurado*"[18]. Una de las ventajas de este protocolo, según sus creadores es su compatibilidad con algunas arquitecturas de servicios diferenciados, lo cual le permite ser usado en conexiones de una red de área local (LAN – **Local Area Network**) con una red de área extensa (WAN – **Wide Area Network**).

Luego, con el auge de Internet y la demanda de servicios tales como la transmisión de video de alta calidad, los investigadores comenzaron a explorar acerca de los protocolos de Internet y su utilización en sistemas denominados de Tiempo Real Blando (**Soft Real-Time Systems**). Su diferencia con el Tiempo Real propiamente dicho, o el denominado Tiempo Real Duro (**Hard Real-Time Systems**) tiene que ver con el tiempo de respuesta (se permiten ciertos retrasos) y el tipo de aplicaciones (no son aplicaciones de misión crítica). El protocolo que se usa en este caso es TCP/IP, y se introduce una especie de subconjunto de servicios denominados *servicios diferenciados*, para lograr soportar demandas como la transmisión de vídeo[19].

Han aparecido algunos estudios acerca de protocolos de comunicaciones híbridos, es decir, que soportan tanto sistemas de Tiempo Real Duro (TRD) como Blando (TRB). Una de las investigaciones en este sentido, utiliza los protocolos TDMA y CSMA/CA, y entre sus características avanzadas están la flexibilidad y una mejor utilización del medio, debido a que utiliza las ranuras no usadas por los "*marcos*" de TRD para los "*marcos*" de TRB[20].

2.1. PDM (Power Division Multiplex - Multiplex por División de Potencia)

Otro de los campos de investigación ha sido el de **PDM** (**Power Division Multiplex** - Multiplex por División de Potencia), donde se han realizado estudios en el campo de las radiocomunicaciones. En 1958 la Dra. Elie J. Baghdady del MIT utilizo por primera vez el concepto de PDM[21]. En dicha ocasión se planteó el uso de PDM, para transmitir por radio FM varios mensajes simultáneos, utilizando la misma banda de frecuencias. Aunque inicialmente fue formulado para ser

[18] CONTI, M. y DONATIELLO L. « Design an Analysis of RT – Ring: a protocol for supporting real-time communications ». Industrial Electronics, IEEE Transactions on . Volume: 49 , Issue: 6 . Dec. 2002. pp 1214 - 1226.

[19] BANERJEA, Anindo. FERRARI, Domenico. MAAH, Bruce A., MORAN, Mark. VERMA, Dinesh C. y ZHANG, Hui. « The Tenet Real - Time Protocol Suite: Design, Implementation, and Experiences». Networking, IEEE/ACM Transactions on. Volume: 4 , Issue: 1 . Feb. 1996. pag. 1 – 10.

[20] ERIKSSON, Christer. THANE, Henrik. GUSTAFSSON, Mikael. «A Communication Protocol for Hard and Soft Real-Systems», IEEE Real-Time Systems, 1996. Proceedings of the Eighth Euromicro Workshop on . 12-14 June 1996. pag. 187 - 192.

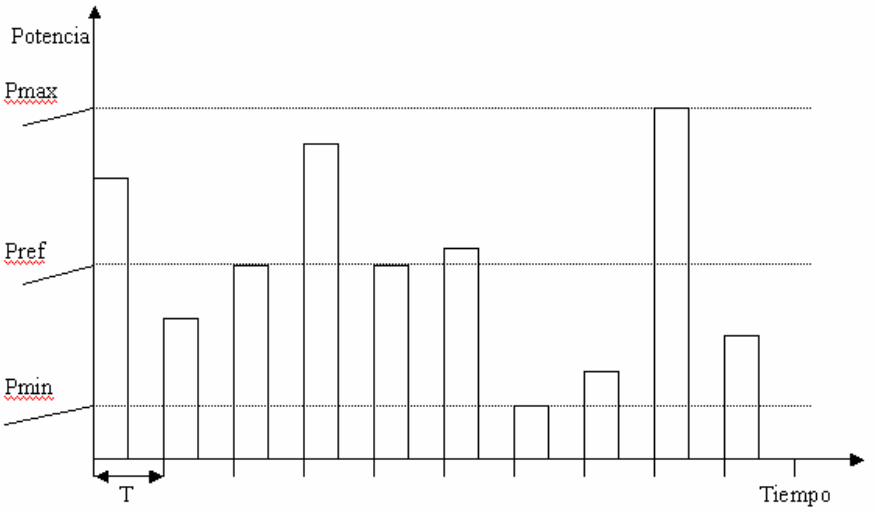
[21] BAGHDADY, Elie J. «New development in FM Reception and their application to the realization of a system of "Power division multiplex" ». Communications, IEEE Transactions on [legacy, pre - 1988]. Volume: 7 , Issue: 3 . Sep 1959. pag. 147 - 161.

utilizado en radiocomunicaciones, es cierto que el concepto es aplicable en cualquier medio de transmisión y puede ser usado en redes con líneas de transmisión de par de cobre, cable coaxial o fibra óptica.

PDM se basa en la transmisión análoga multinivel de potencias y su funcionamiento se resume en que la forma de onda de la señal de potencia como función del tiempo que viaja por el canal de comunicaciones, se transmite como pulsos a intervalos de tiempo regulares e iguales (*T*) y la energía se transmite en la primera mitad de ese intervalo *T*. La energía contenida en cada pulso de potencia, es variable y representa los datos binarios transmitidos por uno o más dispositivos durante cualquier intervalo *T*. La potencia de cada pulso, no puede ser menor a un nivel mínimo de potencia, *Pmin*, determinado por el nivel de ruido en el canal de comunicaciones, ni ser mayor a un nivel máximo de potencia, *Pmáx*, determinado por las características físicas de los componentes electrónicos del dispositivo.

La mitad de la diferencia entre *Pmáx* y *Pmin*, se denomina la potencia de referencia *Pref*, y la utilizan los dispositivos para calibrar sus parámetros internos de amplificación de la señal. Al nivel de potencia de referencia *Pref*, ningún dispositivo puede transmitir datos binarios válidos. En la Figura 7 se ilustra lo expuesto anteriormente.

Figura 7. Múltiplexación por División de Potencia



Recientemente, algunos investigadores japoneses han hecho estudios relacionados con un método de acceso múltiple al medio basado en PDM, denominado PDMA (**Power División Múltiple Access**), donde analizan el

rendimiento de este tipo de acceso que, se diferencia de los métodos anteriores en que se comparte un canal entre los nodos de la red, pero dividiéndolo en diferentes niveles de potencia[22]. La propuesta, básicamente consiste en dos tipos de esquemas para el acceso al canal: en el primer esquema, cada nodo selecciona de manera aleatoria los niveles de potencia para transmitir los paquetes (**Randomly Selected Power Level**). El segundo esquema, asigna los niveles de poder de manera secuencial (**Sequentially Assigned Power Level**), es decir, cada nodo transmite un mensaje en un determinado nivel de potencia, que es superior al usado en el anterior envío, hasta alcanzar un nivel máximo, cuando deberá volver a enviar mensajes en el nivel de potencia más bajo que se haya establecido.

Como resultado de los estudios realizados, los investigadores concluyeron que el esquema con niveles de potencia seleccionados aleatoriamente, es sencillo en su implementación y presenta resolución de colisiones entre niveles, pero solo es recomendable en entornos de bajo tráfico. Mientras que, el esquema con asignación secuencial de niveles de potencia, presenta un buen rendimiento incluso en un entorno con alto tráfico (todos los paquetes serán enviados exitosamente), y en momentos de bajo tráfico, el retraso será pequeño.

Al igual que en PDMA, la presente investigación se fundamentó en la técnica PDM y fue un gran apoyo el trabajo realizado por los investigadores japoneses con su propuesta y la evaluación de rendimiento.

Sin embargo, a diferencia de los trabajos de PDMA mencionados anteriormente que, están orientados a la comunicación a través de ondas de radio, la propuesta de la tesis actual fue diseñada para utilizarse en redes LAN cableadas y teniendo como uno de sus objetivos la implementación en ambientes industriales, donde el ruido producido por los equipos afecta notablemente el rendimiento de la red.

Al sintetizar los resultados de las investigaciones consultadas sobre los diferentes protocolos de comunicaciones y técnicas de acceso múltiple al medio, se tiene que:

- CSMA/CD a diferencia del paso de testigo (**token passing**), se adapta mejor a la tolerancia a fallas.
- Los protocolos de paso de testigo garantizan acceso determinístico durante el modo normal de operación, pero son vulnerables a fallas ante situaciones como la adición y/o eliminación de nodos y la reconfiguración del anillo lógico después de los errores.

[22] SHIMAMOTO, Shigeru. ONOZATO, Yoshikuni. TESHIGAWARA, Yoshimi. « Perfomance Evaluation of Power Level Division Multiple Access (PDMA) Scheme », IEEE , International Conference of Communications. 1992. pp. 1333 - 1337.

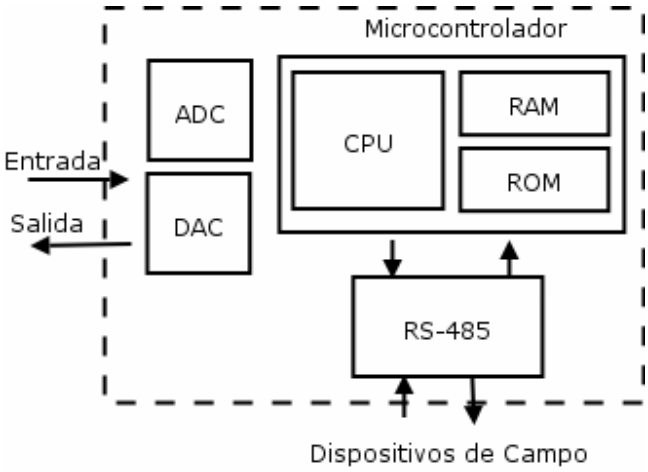
- El protocolo CSMA/CD que se usa ampliamente en las LAN, ofrece ciertas dificultades (v.g. permite colisiones) para ser utilizado como protocolo de comunicaciones en Tiempo Real.
- Los estudios realizados en torno a PDMA, muestran que ésta es una alternativa interesante para implementar en sistemas de comunicación, incluso de alto tráfico.

2.2. Dispositivo DCIH

DCIH – *Dispositivo de Comunicaciones Iván Herrera*, es el dispositivo de comunicaciones diseñado por el Ing. Iván Herrera Murgueitio, cuya finalidad es transmitir datos entre varios de estos dispositivos en Tiempo Real (TRD), por un canal de comunicaciones, utilizando PDM. Los dispositivos se disponen en forma de una red de comunicaciones tipo anillo y cada uno puede transmitir datos por el canal de comunicaciones con destino a uno o varios de ellos, controlando la cantidad de energía contenida en un pulso de potencia transmitido a intervalos regulares. La cantidad de energía presente en este pulso de potencia, representa los datos transmitidos por el canal de comunicaciones, por uno o varios dispositivos, en un mismo instante de tiempo. Utilizando PDM se puede realizar la transmisión simultánea de datos de varios nodos, cada uno de ellos utilizando prácticamente su propio canal, sin experimentar colisión.

En la Figura 8 se puede observar un diagrama de bloques que de manera abstracta ilustra el diseño del dispositivo DCIH.

Figura 8. Diagrama de Bloques del DCIH

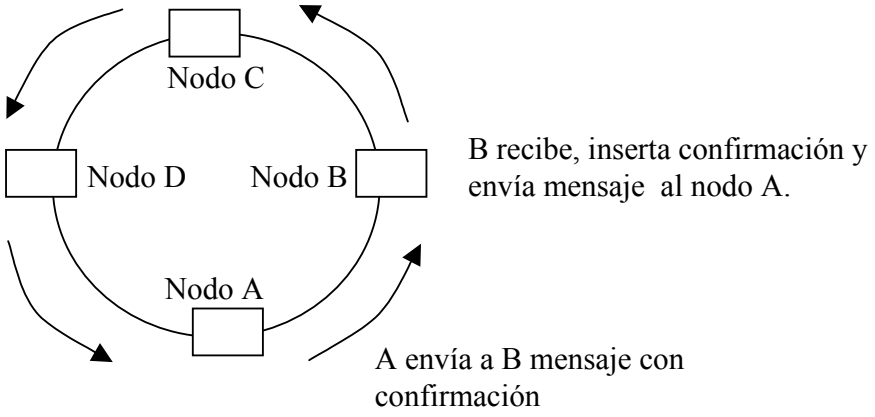


Según el diseño del Ing. Ivan Herrera Murgueitio, el dispositivo se compone de un Microcontrolador, una interfaz RS-485 para la comunicación con los dispositivos de campo (sensores/actuadores), conversores Analogo/Digital (ADC) y Digital/Analogo (DAC) y la entrada/salidad del dispositivo.

3. METODOLOGÍA

El protocolo diseñado en el presente trabajo permite la transmisión de datos entre varios dispositivos *DCIH* dispuestos en una red en anillo (ver Figura 9), cada uno de ellos utilizando el ancho de banda completo de su propio canal de comunicaciones, sin experimentar colisiones, facilitando así la transmisión en Tiempo Real. Para lograr ésto, el método que usa el dispositivo se basa en la Multiplexación de la Potencia, lo que implica que la señalización en el canal no es binaria, sino de tipo analógico. El principio de su funcionamiento se basa en el uso de MASK *(***M-ary Amplitude Shift Keying**), método de señalización que consiste en variar la amplitud de la señal de acuerdo al valor representado por un número (*M*) de bits; esto implica por ejemplo que si se manejan 3 bits, se tienen 8 niveles posibles de amplitud. La diferencia en este planteamiento, radica en que cada bit que se maneja en la muestra, pertenece a un nodo distinto.

Figura 9. Red en anillo de dispositivos DCIH



Utilizando el protocolo PDM-Ring se pueden transmitir a través del anillo de dispositivos *DCIH* dos tipos de mensajes – con confirmación y sin confirmación. En la Figura 9 se muestra como los mensajes circulan por el anillo en un solo sentido y cuando el nodo *A* envía un mensaje al nodo *B* que requiere confirmación, al llegar a *B* el mensaje es recibido, y se inserta la confirmación que circula por el anillo hasta llegar al nodo *A*.

En el caso de los mensajes sin confirmación, si el nodo *A* transmite al nodo *B* un mensaje, este circula por el anillo hasta llegar al nodo *B*, el cual lo recibe y lo transmite al siguiente nodo en el anillo hasta que el mensaje llega nuevamente al nodo A.

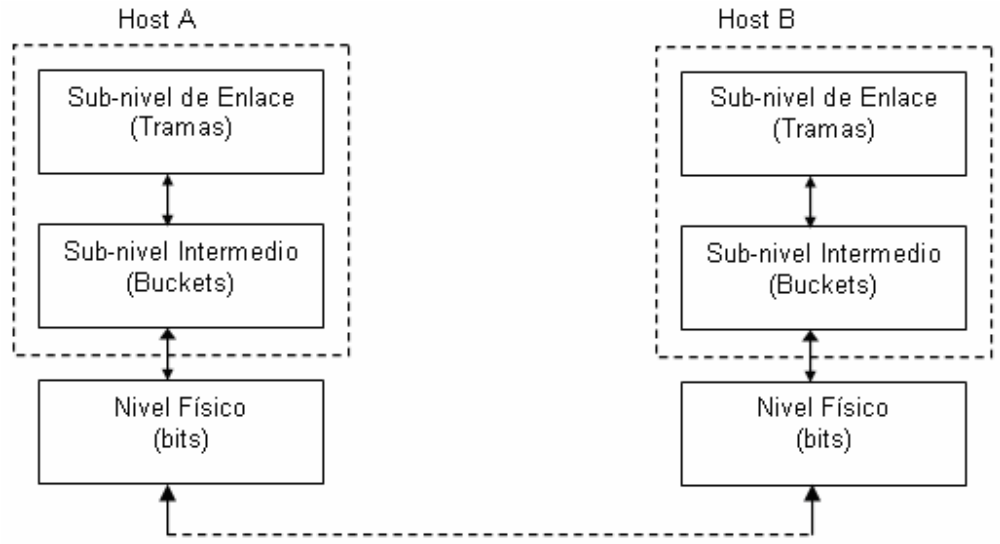
Aparte de los mensajes antes entre los nodos, estos pueden enviar un tipo especial de mensaje para indicar su estado actual a todos los nodos del anillo.

Debido a la complejidad del protocolo necesario para la comunicación de los dispositivos *DCIH*, se decidió diseñar una pila de protocolos de dos niveles, de tal manera que cada uno de ellos tuviera una funcionalidad bien definida y facilitara el diseño. Utilizar un solo protocolo que cumpliera con toda la funcionalidad requerida, habría complicado enormemente las fases de diseño y validación del mismo y la implementación de tal diseño sería muy robusta y demandaría gran poder de cómputo en cada uno de los dispositivos *DCIH*.

Para el diseño del protocolo se tomaron como base los dos primeros niveles del modelo OSI[23], y principalmente se consideró la funcionalidad del nivel 2 ó nivel de enlace de datos.

El protocolo PDM-Ring diseñado consta de dos sub-niveles: el *sub-nivel de enlace* y el *sub-nivel intermedio* (*sub-nivel de datos*), que está entre el nivel físico y el nivel de enlace. En la Figura 10 se puede apreciar el esquema general de la pila de protocolos PDM-Ring.

Figura 10. Pila de Protocolos PDM-Ring



[23] TANENBAUM, Andrew S. Redes de Computadores. Tercera Edición. Prentice Hall, 1997. pag. 28-38

Como se observa en la Figura 10, el diseño de la pila de protocolos PDM-Ring utiliza una estructura por capas o niveles, donde cada una de ellas presta un servicio a la capa superior. De esta manera cada sub-nivel tiene definida su funcionalidad y ésta puede ser modificada sin tener que modificar al otro sub-nivel, lo cual es una gran ventaja para el diseño y mantenimiento de la pila de protocolos, y fue evidenciado durante el presente trabajo.

El esquema propuesto para el sub-nivel de enlace de datos utiliza la "*trama*" como unidad de transmisión y ésta se le pasa al sub-nivel de datos o nivel intermedio para que sea codificada y transferida al nivel físico que, finalmente la coloca en el medio físico para su transmisión por el anillo. Este sub-nivel puede considerarse como un "nivel alto" donde se maneja el control de flujo y la verificación de errores, es decir es equivalente al nivel de enlace de datos del modelo OSI.

En el sub-nivel intermedio hay un concepto novedoso y es el de "***Bucket***"[24], el cual es la unidad de transmisión que agrupa dos bits de cada uno de los nodos que transmite en cada ranura de tiempo. Es decir, es una especie de "*nivel bajo*" donde se "*codifican*" las tramas entregadas por el sub-nivel de enlace de datos para su transmisión por el anillo de dispositivos *DCIH*. Esta codificación es necesaria debido a la utilización de la técnica PDM, lo cual demanda cierto procesamiento, que de haber sido incluido en el sub-nivel de datos lo habría hecho muy robusto y complejo para su validación.

3.1. DISEÑO DEL PROTOCOLO PDM-RING

En el diseño de la pila de protocolos PDM-Ring se siguió la metodología tradicional, la cual menciona cinco elementos[25] para la definición de un protocolo:

1. Los servicios que provee el protocolo.
2. Las asunciones o suposiciones, acerca del medio en el cual se implementa el protocolo.
3. El vocabulario de los mensajes utilizados para implementar el protocolo.
4. El formato de los mensajes.
5. Las reglas de procedimiento o de intercambio de mensajes.

Así mismo, G. Holzmann Op. Cit. 25 que, es uno de los autores más reconocidos en el campo del diseño de protocolos, propone diez reglas básicas a seguir para lograr un buen diseño:

[24] En el capítulo 3.4 se trata con más detalle este concepto.
[25] HOLZMANN, Gerard J.. «Design and Validation of Computer Protocols», Prentice Hall, 1991. pag. 21.

1. Definir bien el problema.
2. Definir el servicio que se provee en cada nivel de abstracción.
3. Diseñar la funcionalidad externa antes de la interna.
4. Mantener el diseño simple.
5. No conectar lo que es independiente. Separar conceptos ortogonales.
6. Diseño genérico.
7. Construir un prototipo y verificar el diseño.
8. Implementación del diseño, medir desempeño y si es necesario optimizar.
9. Verificar que la implementación es equivalente al diseño.
10. No saltarse de la regla 1 a la 7.

En el diseño del protocolo PDM-Ring se observaron las reglas mencionadas y esto facilitó la realización del proyecto. Sin embargo, una de las reglas más difíciles de cumplir fue la relacionada con "*mantener el diseño simple*", ya que al tratar de considerar todas las posibles situaciones, se volvía más complejo el diseño del protocolo y ésto hizo que en ocasiones hubiese que devolverse y retomar versiones anteriores del diseño.

Inicialmente, se definieron los servicios y las asunciones generales de la pila de protocolos PDM-Ring, y luego se continúo con la descripción de cada uno de los sub-niveles y la especificación de los otros tres elementos del diseño respectivamente.

En cuanto a las reglas de procedimiento, se especificaron usando un lenguaje natural y se complementaron con los diagramas que representan cada una de las posibles situaciones que debe afrontar el protocolo. Esto se realizó con diagramas de flujo haciendo uso de un sub-conjunto del lenguaje SDL[26] (**Specification and Description Lenguaje**), para cada sub-nivel de la pila de protocolos.

En las secciones siguientes se presentan los elementos del protocolo en orden y considerando cada uno de los sub-niveles.

3.2. SERVICIOS

El propósito del protocolo PDM-Ring es permitir la transmisión de datos en una red en anillo, con la particularidad de que se contará con varios canales y en condiciones ideales cada usuario contará con su propio canal dependiendo del medio de transmisión. La transmisión se hace en dos regímenes - seguro y rápido.

[26] Lenguaje de Especificación y Descripción orientado a objetos estandarizado por la ITU (*International Telecommunication Union*) mediante la Recomendación Z.100. En Internet se puede encontrar un manual en línea de SDL - http://www.iec.org/online/tutorials/sdl/index.html.

En el modo seguro se tendrá confirmación de las tramas, mientras que en el modo rápido no, lo cual es muy similar a lo que sucede con los protocolos de Internet TCP y UDP, ya que este último protocolo no garantiza la entrega de las tramas, en tanto que el primero si lo hace.

El protocolo verifica errores y determina en que momento una trama debe ser reenviada; evita las colisiones, y provee una estrategia de expropiación, basada en la prioridad asignada a cada nodo, para establecer cual de ellos transmite cuando no haya canales disponibles.

El protocolo es no orientado a conexión y la transmisión es **full-duplex**, en consecuencia un nodo podrá transmitir y recibir al mismo tiempo, pero no por el mismo canal. Además, soporta la opción de transmisión tipo multidifusión (**broadcast**).

3.3. ASUNCIONES

En el diseño de la pila del protocolo PDM-Ring se asumió lo siguiente:

- Memoria suficiente: la capacidad de memoria del dispositivo de comunicaciones DCIH es suficiente para las necesidades del protocolo que se diseña. Aunque este dispositivo no se ha construido, si se ha contemplado en su diseño una capacidad de memoria adecuada.
- Conexión estable en el anillo. Si se tiene una red cableada, la conectividad se garantiza la mayor parte del tiempo.
- No hay colisiones.Si dos nodos coinciden en la transmisión por el mismo canal, sólo uno de ellos transmitirá, el que tenga la mayor prioridad.
- Número de estados suficiente para diferenciar claramente lo que cada usuario transmite.
- Número finito de canales.
- Número finito de nodos (hasta 128, dado que se tienen 7 bits para la dirección).
- No se pierden tramas, dado que se asume un canal ideal, pero en caso de expropiación el nodo con menor prioridad pierde lo transmitido.

3.4. SUB-NIVEL INTERMEDIO

Este sub-nivel facilita la implementación de PDM, y maneja una especie de trama muy pequeña denominada "**Bucket**". El protocolo de este nivel es orientado a bits.

3.4.1 Requerimientos:

- La transmisión de la trama es asíncrona: cada nodo inicia la transmisión cuando lo requiere.
- Evaluar la disponibilidad de los canales.
- Empaquetar y desempaquetar la transmisión de un bit de cada usuario de la red.
- Cada trama es transmitida por un único canal.
- Mantener en un búfer de entrada y de salida los bits que se reciben y transmiten por cada nodo en cada uno de los canales.
- Si se tienen datos para transmitir y existen canales vacíos se debe informar a los demas nodos que el canal se ha ocupado.
- Si no existen canales disponibles, en este nivel se debe evaluar la estrategia de expropiación, basada en prioridades y el tipo de trama que está en proceso de transmisión.
- En cada ranura de tiempo un nodo puede expropiar solo a un nodo, sin importar que tenga varias tramas pendientes por transmitir.
- Informar a los otros nodos de la red cuando un canal sobre el que otro nodo estaba transmitiendo ha sido expropiado.
- Informar a los otros nodos de la red que ignoren lo que un nodo ha transmitido para los casos en que :
    - Dos nodos se apoderan del mismo canal en un mismo instante de tiempo.
    - Un nodo se apodera de un canal que otro nodo ya poseía.
- En ambos casos el nodo que se mantiene será aquel que tenga mayor prioridad y los datos que se ignoran son del nodo que ha sido desalojado.
- Identificar si una trama que se recibe es para el nodo actual.
- Entregar la trama completa al nivel de enlace.
- Informar a los otros nodos cuando la transmisión de una trama ha terminado.
- La transmisión del Bucket es síncrona y se ejecuta para todos los nodos en cada ranura de tiempo.
- Posee un búfer de Transmisión el cuál es llenado por el sub-nivel de Enlace y que le indica al sub-nivel Intermedio que tramas tiene por transmitir.

3.4.2 Vocabulario.

El vocabulario definido para este nivel es:

V = {00,01,10,11}

En la siguiente Tabla 1 se puede apreciar una descripción de los elementos del vocabulario.

Tabla 1. Vocabulario del Protocolo de Sub-nivel Intermedio

Mensaje	Descripción
00	Canal Libre
01	Expropiación
10	Transmite 0
11	Transmite 1

3.4.3 Formato de Mensajes.
Cada mensaje del nivel intermedio, consiste de un campo de control que identifica el tipo de mensaje y un campo de datos. El formato general es el siguiente:

F = {control, datos}

En donde,

control : 1 bit, datos : 1 bit

3.4.4 Reglas de Procedimiento.
Para el nivel intermedio las reglas de procedimiento definidas son:

Proceso de recepción:

- Los bits que se reciben se ubican en un búfer de entrada.
- Si el canal por el que se recibe datos está siendo usado por el nodo actual ya sea por transmisión o expropiación, se deben descartar los bits que se reciben, siempre y cuando sean del nodo actual.
- Si el bit que se recibe es el último y la trama no es para el nodo actual, se transmite el último bit a través del bucket y se limpia los Búfer de Entrada y Salida.
- Si el nodo recibe un mensaje que indica expropiación de otro nodo sobre un canal, identifica este hecho y no considera el canal como disponible.
- Si los bits que se reciben por el canal no son del nodo actual, significa que otro nodo se apoderó del mismo canal, en este caso debe almacenarlos y después de poseer toda la dirección de origen del otro nodo, permanecerá con el canal aquel que tenga mayor prioridad.

- Si un nodo posee un canal y detecta que otro nodo de mayor prioridad se apoderó del mismo canal, lo que hará es enviar un mensaje indicando a los de nodos que ignoren lo que él había transmitido y posteriormente dejará pasar los bits que el nodo de mayor prioridad había transmitido y tenía almacenados.
- Si un nodo posee un canal y detecta que otro nodo de menor prioridad se apoderó del mismo canal, lo que hará es ignorar la transmisión del nodo de menor prioridad hasta que este se detenga y seguirá transmitiendo sus propios bits.
- Si lo que se recibe son datos (propios de la trama) se ubica el dato correspondiente en el búfer de entrada (cada nodo tiene todas las tramas que se transmiten por el anillo).
- Si el bit que se recibe es el último y la trama es para el nodo actual, se debe entregar al nivel de enlace y se limpia el búfer de entrada y el búfer de salida; si la trama no es para el nodo actual solamente se limpia el búfer de entrada y de salida.

El diagrama de flujo que representa este proceso, se muestran en las Figura 11 y Figura 12.

Proceso de transmisión:

- Cada nodo posee un búfer de salida sobre el que se almacena lo que el nodo envía por el canal.
- Cuando el nodo posee tramas en proceso de transmisión: Se debe ubicar en el Bucket y en el búfer de salida el bit correspondiente. Si este bit es final, se deben limpiar los búferes y se debe preparar para enviar en la siguiente ranura de tiempo el mensaje que indica que el canal ha quedado disponible.
- Si no se encuentran canales disponibles, se procede a evaluar la estrategia de expropiación.
- Cuando el nodo posee nuevas tramas para transmitir: Si existe una o varias tramas en el búfer de Transmisión a las que no se le haya asignado ningún canal, se procede a organizarlas por tipo de mensaje (con confirmación) y por estampilla de tiempo. Una vez hecho esto se procede a ocupar solo un canal. Si no se encuentran canales disponibles, se procede a evaluar la estrategia de expropiación.
- Cuando el nodo tiene tramas por transmitir, pero no hay canales disponibles: En este caso se ejecuta la estrategia de expropiación que consiste en desalojar a un nodo con menor prioridad que el actual. La trama del nodo que se expropia, no debe requerir confirmación y no se debe haber transmitido del 50% de la misma.
- Aunque se ejecute la estrategia de expropiación es posible que queden tramas pendientes por transmitir, en este caso, en la siguiente ranura de

tiempo nuevamente se tendrán en cuenta dentro del proceso normal de transmisión.
- Una vez que la trama es transmitida completamente, se debe limpiar el búfer de Transmisión.

El diagrama de flujo que representa este proceso, se muestran en las Figura 13 y Figura 14.

3.4.5 Diagramas de Flujo

Figura 11. Proceso de Recepción

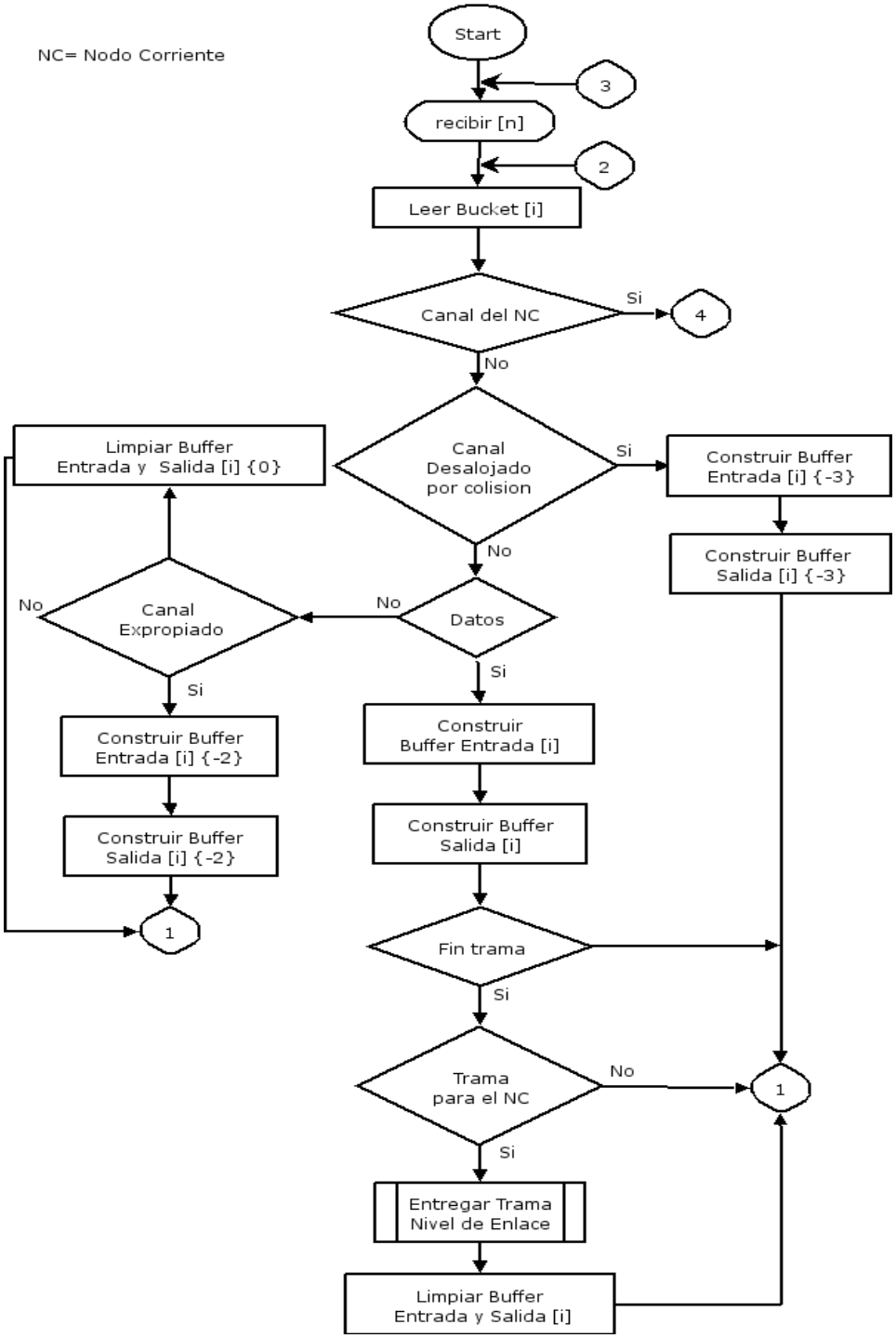
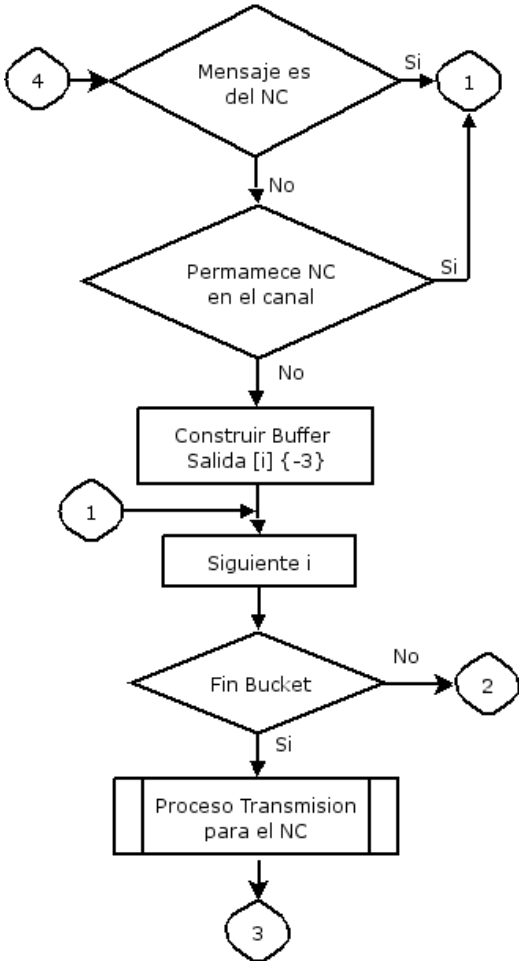


Figura 12. Proceso de Recepción continúa…



Esta figura ilustra el proceso de Recepción del sub-nivel Intermedio que en general debe evaluar que tipo de mensaje recibe po cada uno de los canales, a través del **Bucket**.

Para mayor claridad en el diagrama se ha incluido una etiqueta diferenciadora de tipo -3, que se obtiene cuando dos nodos se apoderan de un mismo canal, pero este hecho no se ha incluido en la verificación y la simulación, dado que el procedimiento que se sigue es el mismo que cuando se dá una expropiación y esta situación si ha sido verificada.

Figura 13 . Proceso de Transmisión

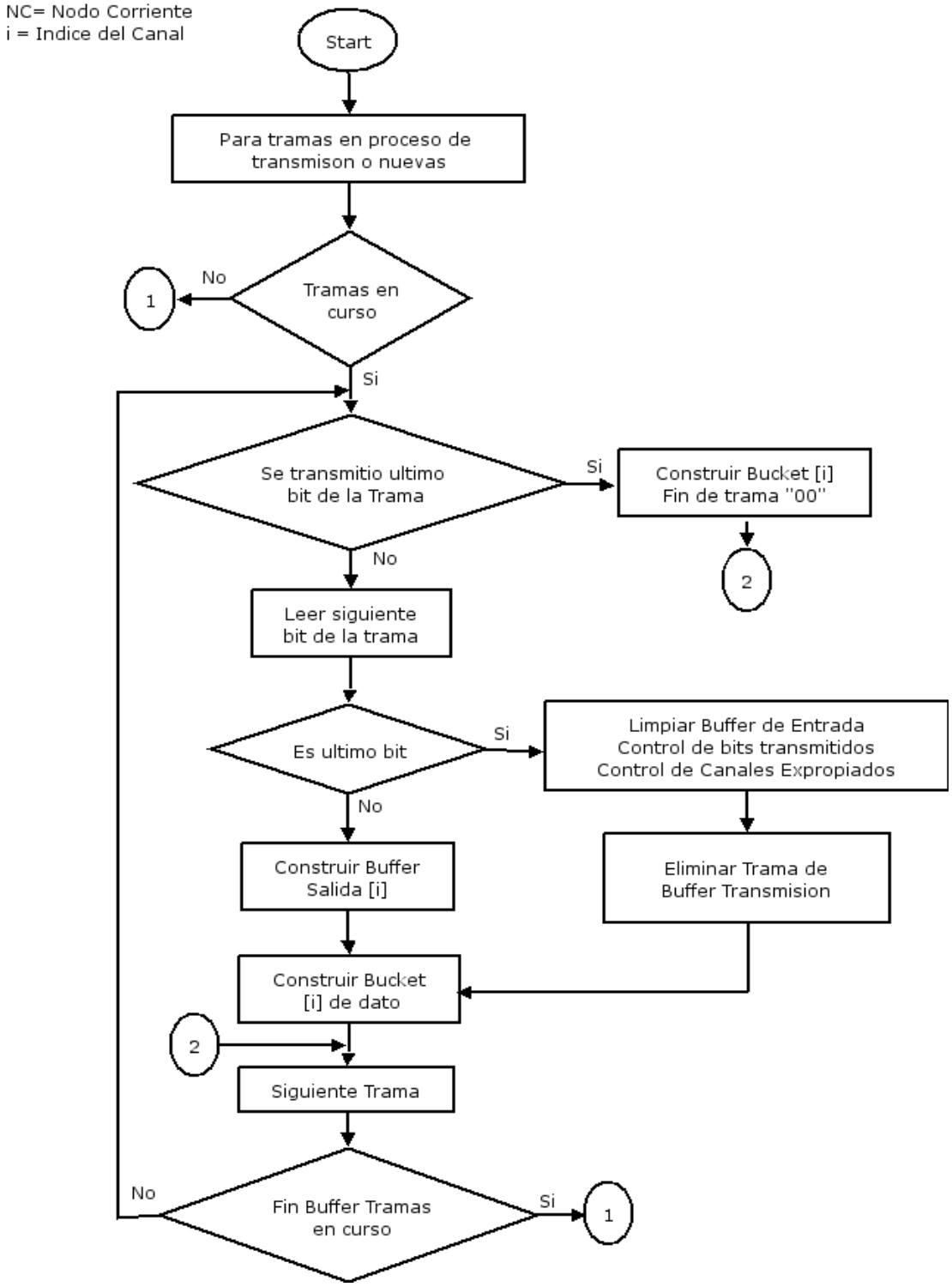
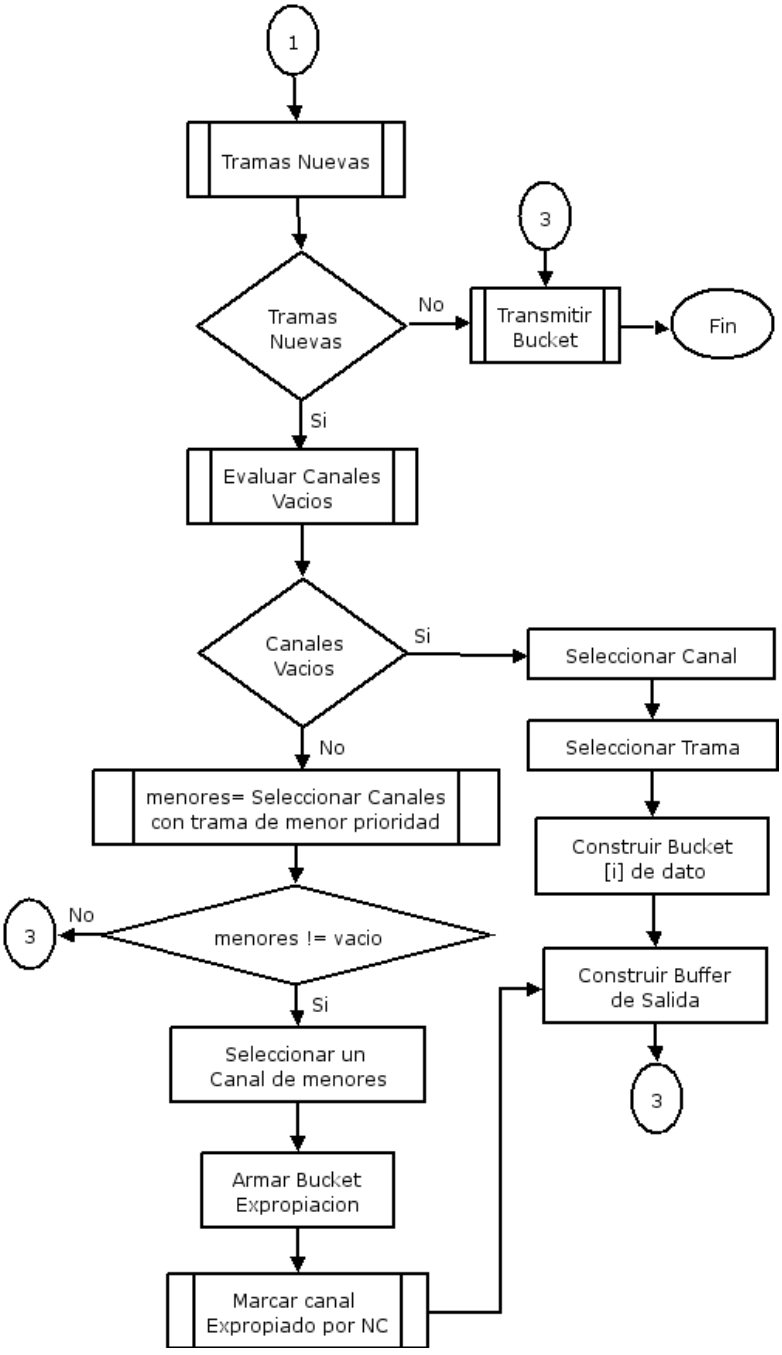


Figura 14. Proceso de Transmisión continua…



La figura ilusta el proceso de transmisión, en el que en general el nodo debe verificar si posee tramas en proceso de transmisión y nuevas por transmitir, teniendo ciudado de verificar que el canal que posee no haya sido expropiado.

Otra característica especial hacer referencia al proceso de expropiación, ya que si el nodo no encuentra canales disponibles, debe verificar la posibilidad de apoderarse de un canal que otro nodo posea.

3.4.6 Bucket

Es el componente central del sub-nivel Intermedio, ya que a través de él se transmiten los tipos de mensajes de este nivel. Como se mostró en el diseño del protocolo en la Tabla 1, cada combinación de bits tiene un significado diferente, a través del cual el nodo toma decisiones tanto en el proceso de recepción como en el proceso de transmisión.

En cada ranura de tiempo, el nodo debe evaluar el contenido del bucket, de esta forma logra identificar en que estado se encuentran todos los canales interpretando el tipo de mensaje que ha recibido.

Una vez esto se ha hecho, el nodo procede a verificar si tiene tramas en proceso de transmisión para así ubicar su mensaje en el **Bucket**, en el canal que le corresponda, así como también debe verificar si tiene tramas por transmitir para proceder a buscar un canal disponible y nuevamente ubicar su mensaje en el **Bucket**.

A continuación se detalla el proceso de recepción y transmisión, contemplando todos los casos posibles, para así aclarar su funcionamiento.

Proceso de Recepción

En el búfer de entrada, los bits sombreados indican que estos bits no hacen parte de la trama, sino que indican el estado anterior del canal, para este caso el cero (0) indica que el canal estaba vacío y no se confundirá con el dato cero dado que cada nodo ha recibido por el **Bucket** el mensaje que le permite diferenciar estas situación.

Paso 1, el medio físico transporta el código que se decodifica para conformar el **Bucket**. Cada uno de estos mensajes del **Bucket** tiene un significado diferente (00 – canal vacío; 10 - canal transmite un 0; 11 - canal transmite un 1; 01 - canal ha sido expropiado).

Paso 2, el nodo lee el **Bucket** de recepción y ubica el bit en búfer de entrada de acuerdo al canal que le corresponda y en la posición adecuada

Paso 3, se evalúa si algunas de las tramas en el búfer de entrada ya están completas, es decir están en el bit *m*, si es así se evalúa que la dirección destino sea la del nodo corriente, si esto se cumple, el nodo baja la trama al búfer de recepción (Búfer Rx).

Paso 4, si la trama es enviada al nodo corriente, se entrega al sub-nivel de enlace.

La Figura 15 ilustra varios casos, según el contenido del **Bucket**:

1. Por el canal 1 se detecta que no hay transmisión de datos, es decir el canal está libre (canal marcado con 0).

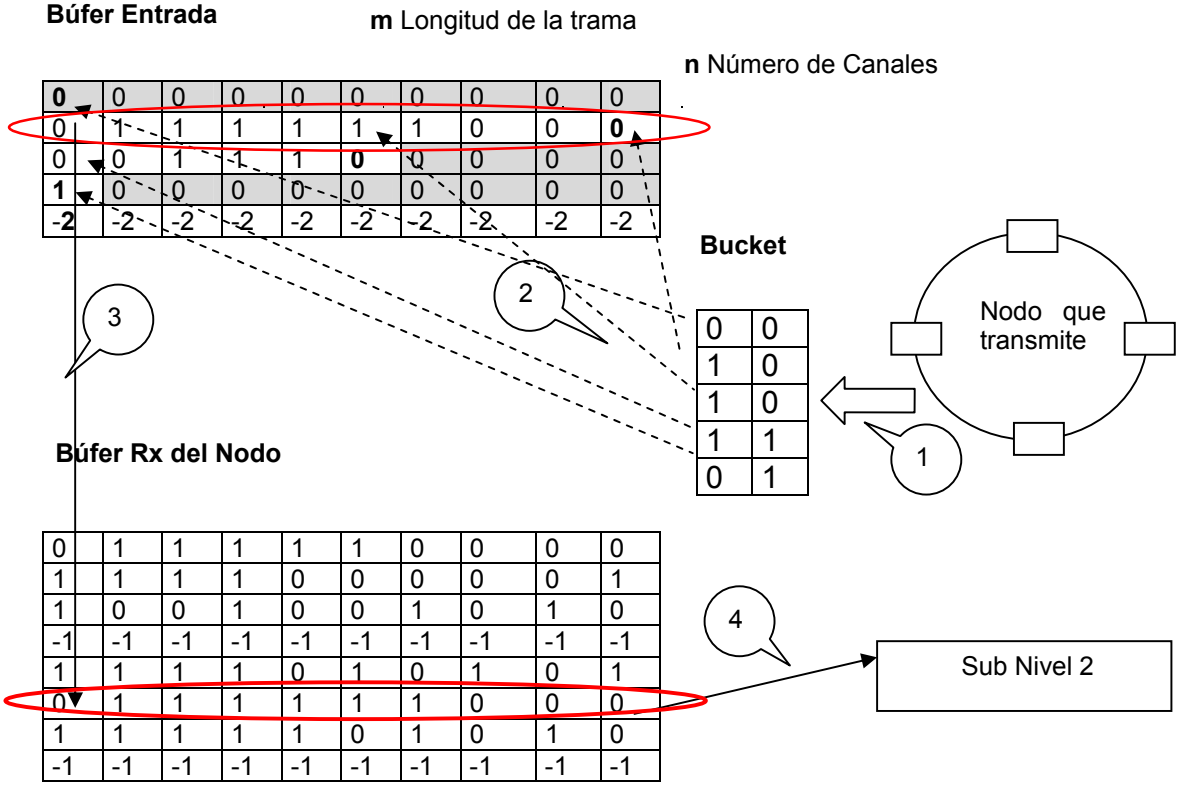
2. Por el canal 2 se detecta que llega el último bit de una trama y es para el nodo en corriente, por tanto debe ser ubica en el Búfer de Recepción.

3. Por el canal 3 se recibe un bit (no final) de una trama que no necesariamente es para el nodo corriente.

4. Por el canal 4 llega el primer bit de una trama que no necesariamente es para el nodo corriente.

Por el canal 5 se recibe la señal de expropiación, indicando que el canal ha sido tomado por otro nodo, por lo tanto se marca con -2.

Figura 15. Estado del Bucket en el Proceso de Recepción



Proceso de Transmisión

La Figura 15 ilustra la forma como el **Bucket** es modificado durante el proceso de transmisión. Los pasos que este proceso sigue son:

Paso 1, el sub-nivel de enlace ubica en el búfer de transmisión las tramas que desea transmitir.

Paso 2, el sub-nivel intermedio se encarga de ubicar en el búfer de entrada, en cada ranura de tiempo, un bit de las tramas que están en proceso de transmisión, así como también de ubicar las tramas nuevas que el sub-nivel de enlace haya ubicado en el búfer de transmisión. En este último caso, se ubica un canal disponible para ubicar esta trama, si lo hay ubica el primer bit de esta trama en al canal; sino entonces se evalúa la posibilidad de expropiar un canal en el que se transmitan tramas de menor prioridad que no requieran confirmación. Si no es posible ubicar un canal disponible, la trama se encola para ser transmitida en otra ranura de tiempo.

Paso 3, se arma el **Bucket**, conformado por todos los bits de búfer de Entrada que deben ser transmitidos.

Paso 4, se codifica el **Bucket** de acuerdo para ser enviado por el medio físico.

La Figura 16 ilustra varios casos:

1. Se transmite la señal de canal disponible (canal número 1, marcado con 0).

2. Se transmite el último bit de una trama que pertenece a otro nodo (canal número 2).

3. Se continúa con el proceso de transmisión de una trama del nodo actual: Se ubica el siguiente bit de la trama en el búfer de Entrada (trama 4, canal número 3).

4. Se detecta una trama nueva en el búfer de transmisión (trama número 7) del nodo actual, por tanto el primer bit de esta trama se ubica Búfer de Entrada en uno de los canales disponibles (canal número 4 marcado con 0).

5. Un canal ha sido expropiado por el nodo corriente u otro nodo (canal número 5 marcado con –2).

3.5. SUB-NIVEL DE ENLACE DATOS

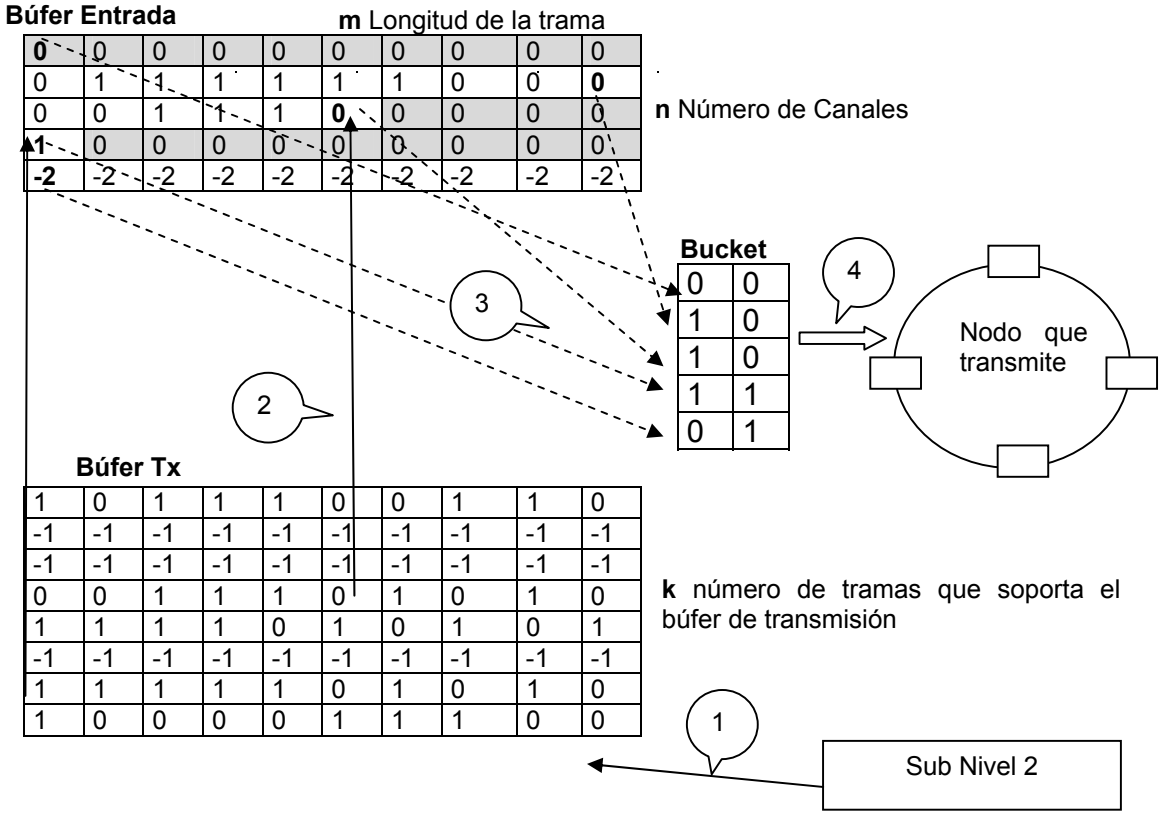
En este sub-nivel se realizan las funciones típicas del nivel de enlace del modelo OSI[27], es decir el control de flujo y la verificación de errores.

Los requerimientos que se definieron para este sub-nivel fueron los siguientes:

- Realizar verificación de errores usando una suma de verificación (**checksum**).
- Las tramas deben ser pequeñas (poco **overhead**) y la transmisión es asíncrona.
- Permitir la retransmisión de tramas.

[27] STALLINGS, William. «Comunicaciones y Redes de Computadores», Sexta Edición, Editorial Prentice Hall, Madrid (España), 2000. Pag. 182-210.

Figura 16. Estado del Bucket en el Proceso de Transmisión



- Proveer reconocimiento de la recepción de tramas (**ACK**) en el modo seguro (**Secure Mode**).
- Controlar el tiempo de recepción de las tramas de reconocimiento (**ACK**) utilizando un contador de tiempo (temporizador).
- Proveer una tabla con información acerca del estado de todos los nodos presentes en la red.
- Guardar copia de las tramas enviadas.
- La dirección 127 (1111111) está reservada para los mensajes de multidifusión (**broadcast**).
- Proveer números de secuencia para las tramas con confirmación.

3.5.1 Vocabulario.

Para el nivel de enlace el Vocabulario definido fue el siguiente:

V = { I, RR, RNR, ACK, NN, SOK, BY }

En la Tabla 2 se presenta la descripción de los elementos del Vocabulario definido.

Tabla 2. Vocabulario del Sub-nivel de Enlace

Mensaje	Descripción
I	Trama de información
RR	Nodo listo para recibir datos
RNR	El nodo No está listo para recibir datos.
ACK	Reconocimiento de trama
NN	Nuevo Nodo en la red
SOK	El nodo está en status O.K
BY	Salida controlada de un nodo de la red

3.5.2 Formato de Mensajes.
Las tramas son de longitud fija y constan de 79 bits, cuya estructura es la siguiente:

F = { encabezado, control, datos, ACK, CRC }

En donde,

encabezado, está conformado por la dirección destino, y la dirección origen, que es a su vez la prioridad del nodo.

control, define los tres tipos de mensajes que se transmiten - información, control de flujo, y control (estado).

ACK, reconocimiento de trama recibida. Si la trama se recibió con errores, se informará al emisor de esta situación.

CRC (**Ciclyc Redundancy Check** – Código de Redundancia Cíclica)[28] , campo de verificación de errores.

Es importante aclarar que, el protocolo está orientado al ámbito industrial, donde uno de los requerimientos es que las tramas deben ser pequeñas y con poco "**overhead**" (*encabezado*), ya que los datos a transmitir generalmente son valores

[28] TANENBAUM, A. S. , «Redes de Computadores». Tercera Edición. Prentice Hall, 1997. p.186-190.

numéricos y deben ser transferidos con gran velocidad debido a la frecuencia con que se producen (v. gr. lectura de sensores[29]). Por ejemplo, en el caso de las redes Profibus la velocidad de transmisión va desde 9.6 Kbps hasta 12 Mbps[30].

Además, debido a la utilización de una estrategia de prioridades en el sub-nivel intermedio, se planteó la utilización de la dirección del nodo como prioridad, porque definiendo prioridades únicas se facilita la solución de las posibles colisiones en los canales durante los momentos de alto tráfico de datos en el anillo.

En la Tabla 3 se puede apreciar un resumen de los campos que conforman la trama del sub-nivel de enlace.

Tabla 3. Estructura de trama del sub-nivel de Enlace.

Campo	Descripción	Longitud (bits)
Do	Dirección origen	7
Dd	Dirección destino	7
Tm	Tipo de mensaje	2
nseq	Número de secuencia	10
Tt	Tipo de trama	3
datos	Datos	32
ACK	Reconocimiento de mensaje	2
CRC	Suma de verificación	16

El campo tm – tipo de mensaje, está relacionado con aquellos mensajes que requieren reconocimiento (ACK) y su significado se describe en la Tabla 4

Tabla 4. Campo tipo de mensajes (tm)

Valor	Descripción
00	No requiere reconocimiento – ACK
01	----------
10	----------
11	Requiere reconocimiento – ACK

[29] En las redes industriales los datos (mediciones físicas) se leen desde las entradas de los PLC para ser procesadas con una frecuencia de muestreo que va desde los 10 mseg. hasta los 200 mseg. Dicha frecuencia de muestreo depende de la dinámica del proceso, considerandose un valor intermedio 100 mseg. para procesos que no poseen una gran dinámica.

[30] Catálogo de Comunicaciones Industriales IK PI – 2005, Referencia E86060-K6710-A101-B4-7800, SIEMENS AG, República Federal de Alemania, 2005. pag. 5/12.

El campo nseq – número de secuencia de trama, se utiliza en las tramas que requieren confirmación (ACK) para que el receptor indique el número de la trama recibida.

Los posibles valores del campo tt - tipo de trama se describe en la Tabla 5.

Tabla 5. Campo tipo de tramas (tt)

Valor	Significado	Descripción
000	NN	Nuevo Nodo en la red.
001	SOK	Status O.K
010	RR	El Nodo está listo para recibir datos
011	RNR	El Nodo NO está listo para recibir datos
100	I	Trama de datos.
101	BY	Salida controlada de un Nodo de la red.
110	ACK	Reconocimiento de trama.

El campo ACK – reconocimiento, está relacionado con el reconocimiento de mensajes y su significado se describe en la Tabla 6.

Tabla 6. Campo de reconocimiento de mensajes – ACK

Valor	Descripción
00	Trama recibida con errores
01	----------
10	----------
11	Trama recibida correctamente (ACK)

3.5.3 Reglas de Procedimiento.

Las reglas de procedimiento definidas para el sub-nivel de enlace fueron las siguientes:

- Al momento de enviar una trama se deberá especificar si es “sin” o “con” confirmación. Si se envía una trama con confirmación ésta deberá tener un número de secuencia y se activará el temporizador (con decremento) que sirve para controlar su reenvío.
- Se pueden transmitir/recibir 4 tipos de tramas: estado del nodo, confirmación de trama (ACK) y tramas con información (datos).
- En las tramas de estado la dirección destino serán todos los nodos de la red y estas serán la únicas de tipo “**broadcast**”.

- Las tramas que serán transmitidas son almacenados en un búfer - búfer de transmisión, y luego serán organizadas de acuerdo a un cierto criterio – en primer lugar las tramas que contienen mensajes de estado (*SOK, BY, RNR, NN*), en segundo lugar las tramas de confirmación (*ACK*), y en tercer lugar las tramas de información.
- Cada nodo almacenará en un búfer especial - búfer de tramas pendientes por *ACK*, copia de las tramas con confirmación enviadas. Las tramas almacenadas en este búfer se irán dando de baja en la medida en que se vayan recibiendo los mensajes de confirmación (*ACK*) de los nodos receptores.
- Si al momento de enviar una trama con confirmación el búfer está lleno, ésta se almacenará en un búfer de tramas pendientes por transmitir.
- Cuando sea necesario la retransmisión de una trama con confirmación, se verificará su existencia en el búfer de tramas pendientes por confirmación, y se retransmitirá su copia.
- Si la confirmación de una trama supera el tiempo de espera (temporizador) especificado en la configuración del nodo, entonces la trama deberá ser retransmitida. En el nodo emisor, el temporizador se activa al momento preciso del envío de las trama con confirmación y su duración será parte de la configuración del nodo.
- Cuando un nodo recibe una trama que requiere confirmación, devuelve un mensaje "**ACK**". Al enviar el mensaje de confirmación – **ACK**, la dirección origen de la trama es el nodo que está enviando y la dirección destino corresponde al nodo del cual se recibió la trama original.
- Con una frecuencia de tiempo definida en los parámetros de configuración de cada nodo del anillo, estos reportan su estado enviando un mensaje "**SOK**" a todos los nodos de la red. Los posibles estados de un nodo son: **OK** (normal - activo), **BUSY** (ocupado), y **OUT** (fuera de servicio).
- Cuando un nodo es agregado a la red, éste debe enviar un mensaje "**NN**" a todos los nodos para reportar su presencia y estado. Los nodos que reciben un mensaje "**NN**", crean una nueva entrada con la dirección del nodo emisor en su tabla de estados y se define el estado "**OK**" para el nuevo nodo.
- Si un nodo va a ser retirado de la red (mantenimiento, revisión, etc) debe enviar antes un mensaje "**BY**" a todos los nodos informando de este suceso. Los nodos que reciben un mensaje "*BY*", editan el estado del nodo emisor en la tabla de estados y el nuevo estado será "*OUT*".
- Para su reactivación en la red un nodo que salió de servicio, deberá enviar un mensaje "*SOK*" a todos los de nodos de la red. Los nodos que reciben un mensaje "SOK", editan el estado del nodo emisor en tabla de estados y el nuevo estado será "*OK*".
- Si un nodo no puede continuar recibiendo datos debido a la falta de memoria (búfer de recepción) para almacenar las tramas, éste enviará un mensaje "*RNR*" a todos los nodos de la red, lo cual indicará que no está listo para recibir datos. Los nodos que reciben un mensaje "*RNR*", editan el

estado del nodo emisor en la tabla de estados y el nuevo estado será "*BUSY*".

- El búfer de recepción de tramas de cada nodo tendrá un porcentaje de espacio mínimo de memoria libre (5%) definido en la configuración del nodo, y en el momento que se llegue a ese umbral, se deberá esperar a que se entreguen las tramas almacenadas en el búfer para enviar el mensaje "RR" a todos los nodos de la red. Mientras se libera el búfer de recepción del nodo, no se enviarán tramas con confirmación.
- Si al momento de recibir una trama con confirmación, el espacio de memoria libre en el búfer de recepción está en el umbral (5%), la trama se recibe y se almacena.
- Un nodo que se encuentra en estado "*RNR*" deberá procesar las tramas pendientes en el búfer de recepción hasta que vuelva a tener espacio disponible para recibir tramas de información (espacio mayor al umbral). Cuando tenga espacio disponible enviará el mensaje "SOK" a todos los nodos de la red.
- Cuando un nodo se encuentra ocupado – "BUSY", continuara recibiendo tramas, pero únicamente de tipo **broadcast**.
- Una trama con confirmación será retransmitida, pero el número de retransmisiones será limitado (máximo 3 veces), pasado este limite la trama será eliminada del búfer.
- Si un nodo después de tiempo *T* (*frecuencia de estado - fq*) no reporta su estado a los de nodos, éstos cambiaran el estado del nodo en cuestión a "*BY*" (ausente) en la tabla de estados de cada uno de ellos.
- Si una trama con confirmación (*ACK*) se va a enviar a un nodo que no está habilitado, será almacenada en un búfer - búfer de tramas con confirmación pendientes.
- El búfer de tramas con confirmación pendientes por transmitir será revisado con una cierta frecuencia ($\Delta Tpendientes$) y las tramas que hayan superado el tiempo de espera (*Twait*) en el búfer serán descartadas. Las tramas serán clasificadas de acuerdo al tiempo de espera en el búfer, y se transmitirán siempre y cuando el búfer de transmisión esté libre.
- En cada nodo existe un procedimiento para verificar los temporizadores (estado de los nodos, tramas pendientes por transmitir, tramas con confirmación pendientes por transmitir).

3.5.4 Diagramas de Flujo.

A continuación se presentan los diagramas de los procesos de transmisión y recepción de mensajes del sub-nivel de enlace.

a. Proceso de Transmisión

En la Figura 17 se presenta el diagrama del proceso de transmisión de manera completa. Como se aprecia en la figura en cuestión, éste incluye una serie de subrutinas (sub-procedimientos), las cuales son detalladas en lo sucesivo

El proceso de transmisión se inicia con la verificación de los temporizadores[31] (**time out**) de las tramas con confirmación y la capacidad del búfer de transmisión (ver Figura 18 y Figura 19). Luego, se selecciona el tipo de mensaje a enviar y se ejecuta la subrutina de transmisión del mensaje seleccionado (*TxNN, TxSOK, TxRNR, TxRR, TxBY, TxACK, TxI*).

Las subrutinas para la trasmisión de los diferentes tipos de mensajes se ilustran en la Figura 20, Figura 21, Figura 22, Figura 23, Figura 24, Figura 25 y la Figura 26. En algunas de ellas, es común la subrutina "*Generar estampilla de tiempo*", la cual se utiliza para la verificación del tiempo de generación de la trama, y la subrutina "*Entregar trama*" que define las operaciones a realizar para la entrega de las tramas al sub-nivel intermedio.

[31] Son temporizadores con decremento, es decir que se vencen cuando alcanzan el valor 0.

Figura 17. Transmisión de Mensajes del Sub-nivel de Enlace de Datos

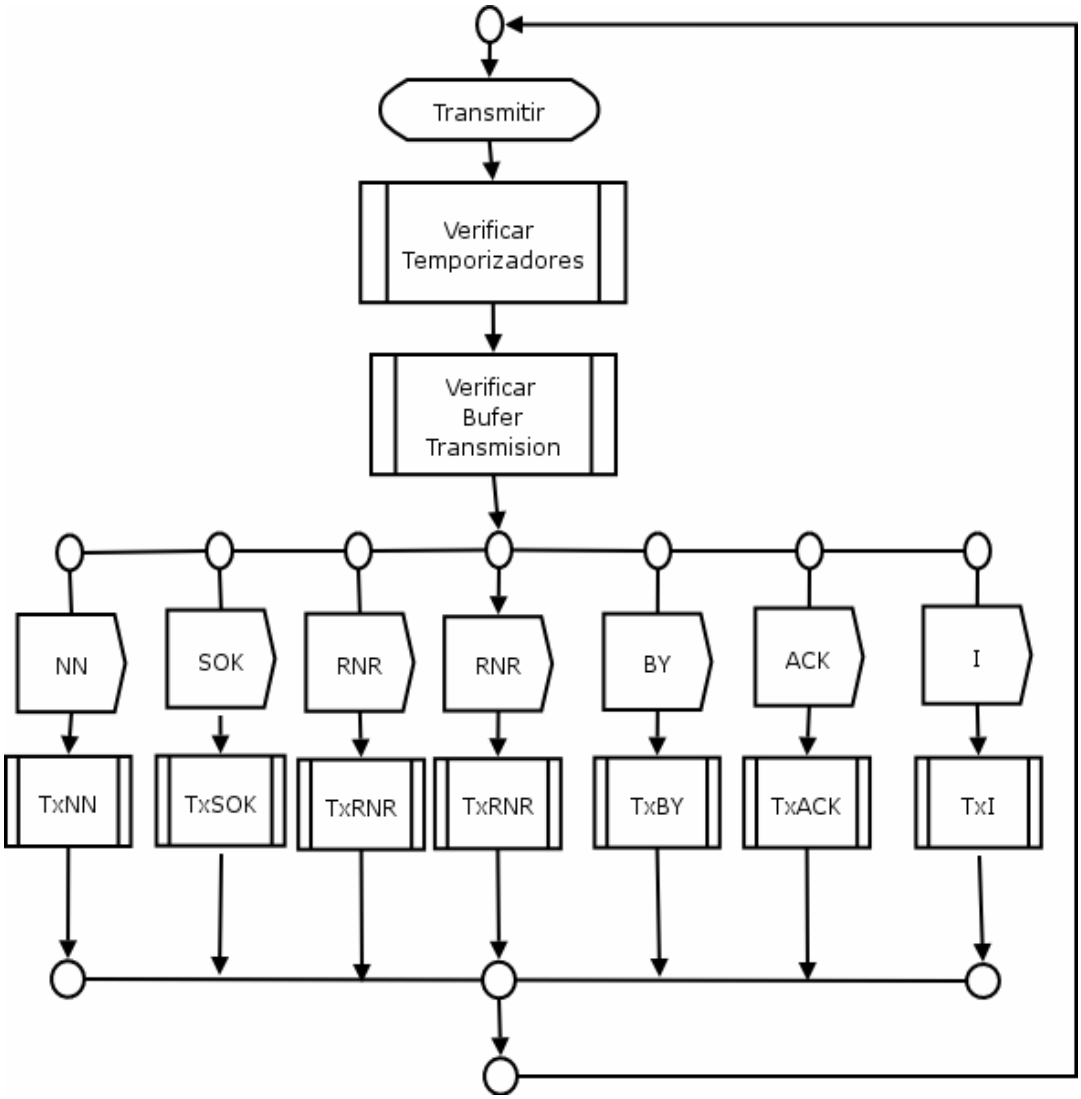
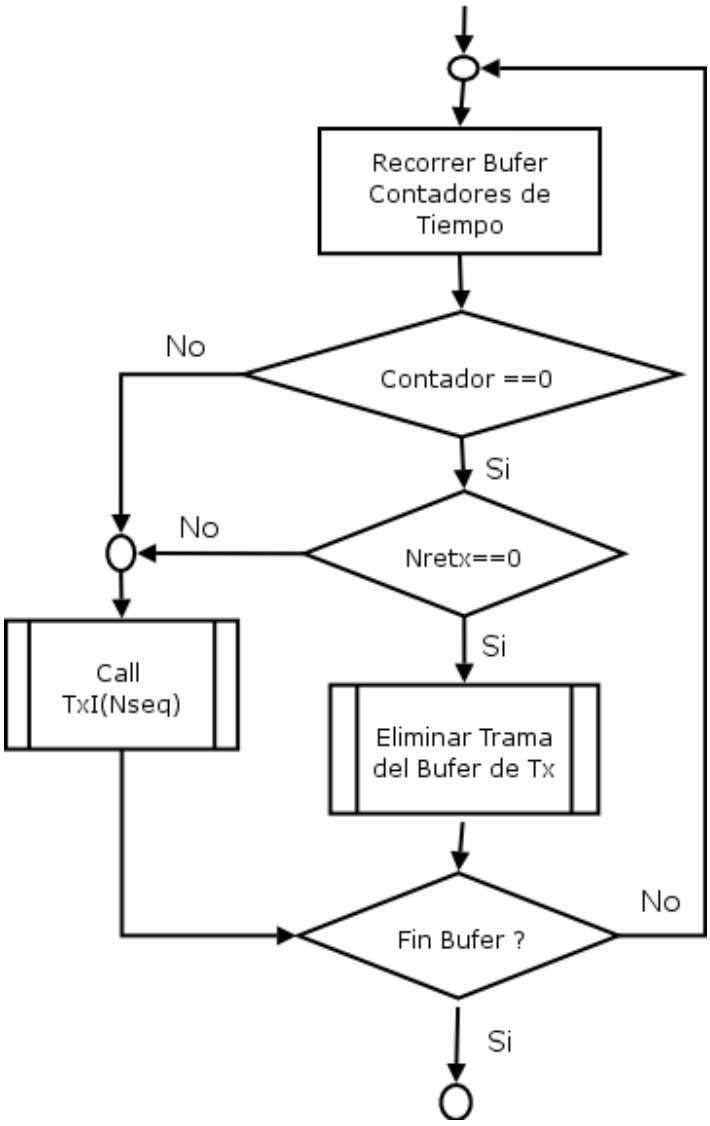


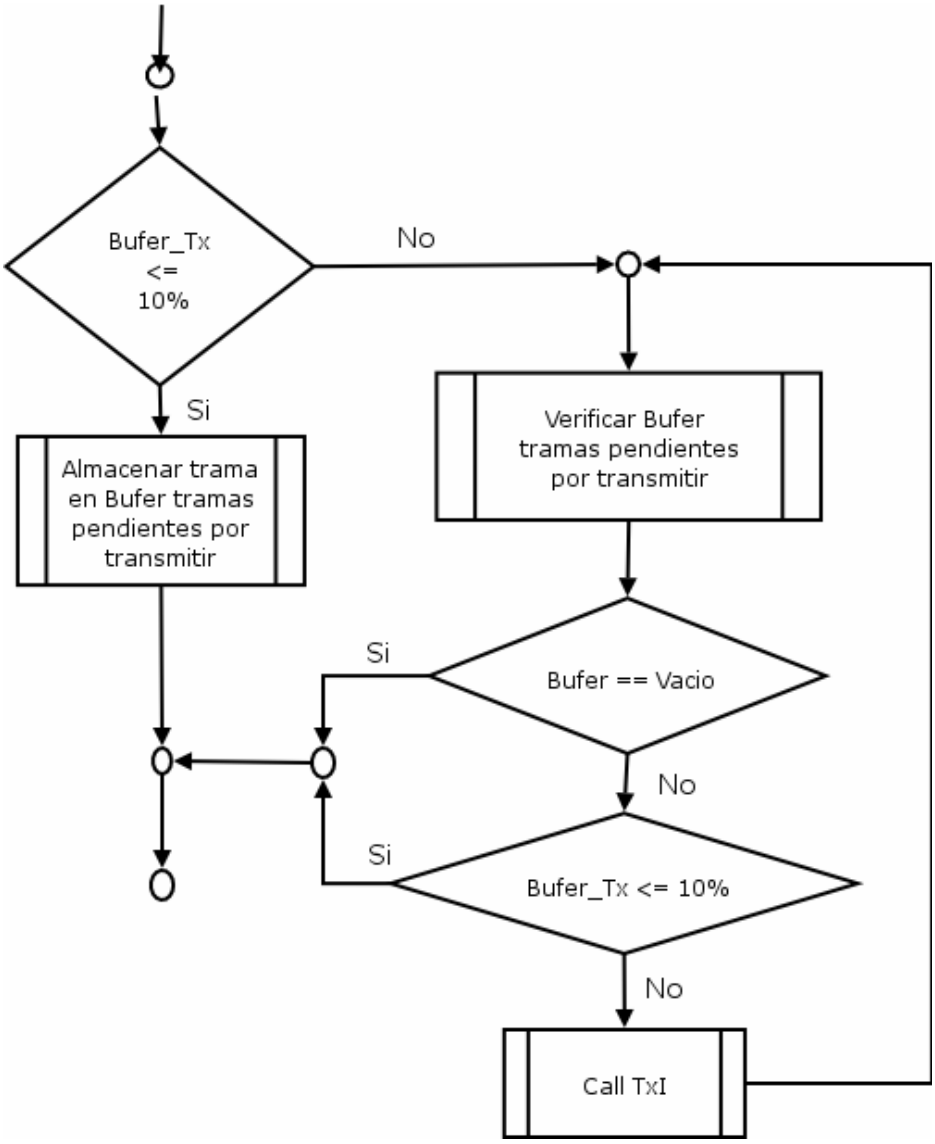
Figura 18. Subrutina Verificar Temporizadores



La Figura 18 ilustra el diagrama de la subrutina para la verificación de los temporizadores de las tramas con confirmación, en ella se aprecia que los valores de los contadores de tiempo están almacenados en un búfer, el cual es recorrido en su totalidad y si se hallan valores 0 (**time out**) y ade el contador de re-envíos de trama con confirmación ha alcanzado su limite[32], entonces la trama es eliminada del búfer de tramas pendientes por confirmación. En el caso contrario, se invoca a la subrutina *TxI* para el reenvío de la trama.

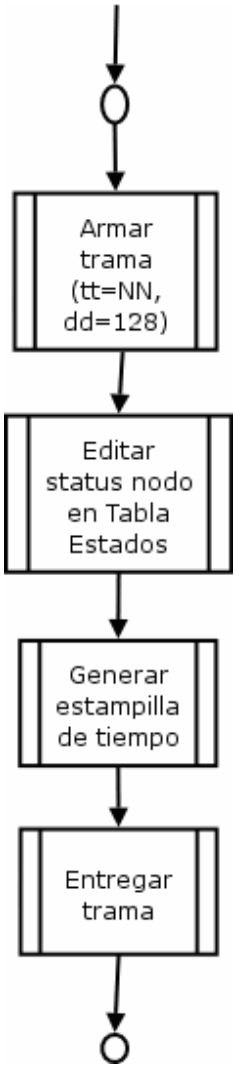
[32] Este contador inicia en 3, que es el máximo de reenvíos, y luego se va decrementado hasta alcanzar el valor 0.

Figura 19. Subrutina Verificar Capacidad de Búfer de Transmisión.



En la Figura 19 se presenta el diagrama de la verificación de la capacidad del búfer de transmisión, en la cual se muestra que si la capacidad del búfer es menor al 10% y hay una trama para enviar, ésta se almacena en un búfer especial que se denomina *búfer de tramas pendientes por transmitir* (tramas con o sin confirmación). En caso contrario, es decir si la capacidad del *búfer de transmisión* es superior al 10%, entonces se verifica el *búfer de tramas pendientes* por transmitir, y si hay tramas almacenadas en él, son transmitidas, pero considerando la capacidad del *búfer de transmisión*.

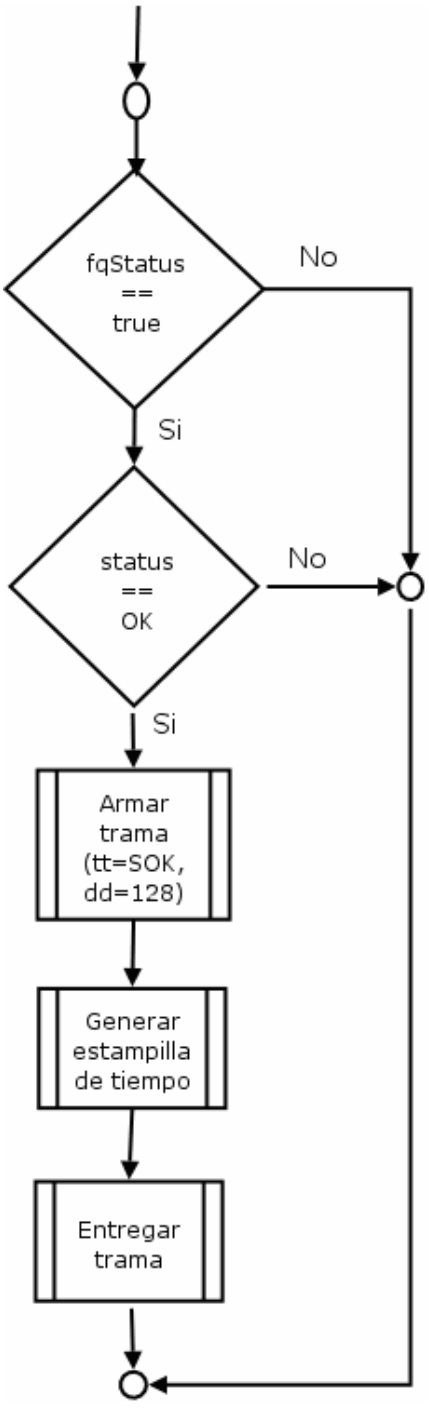
Figura 20. Subrutina Transmitir Mensaje NN (Nuevo Nodo)



La Figura 20 ilustra el diagrama de la subrutina "*Armar trama*", donde se especifica el tipo de trama (tt) que en este caso es NN (*Nuevo Nodo*), y la dirección destino (dd) que toma el valor 127, es decir está dirigido a todos los nodos del anillo. Y luego, se edita el estado del nodo en la "*tabla Estados*" que, a nivel local en cada uno de los nodos mantiene la información del estado actual de los nodos del anillo.

En la Figura 21 se puede apreciar que, al inicio de la subrutina de transmisión de estado **OK** (*TxSOK*) se verifica la frecuencia de estado (*fqStatus*), la cual indica cuando enviar este tipo de mensaje, y que está definida como un parámetro en la configuración de cada uno de los nodos del anillo.

Figura 21. Subrutina Transmitir mensaje SOK (Nodo en estado OK)



La Figura 22 y la Figura 23 presentan los diagramas de flujo de las subrutinas de transmisión de los mensajes *RNR* y *RR* respectivamente, los cuales son muy similares y solo se diferencian por el parámetro *tipo de trama* (tt) y la modificación del estado del nodo (**OK** o **BUSY**). En este tipo de mensajes la dirección destino (dd) toma el valor 127, es decir están dirigidos a todos los nodos del anillo.

Figura 22. Subrutina Transmitir Mensaje RNR

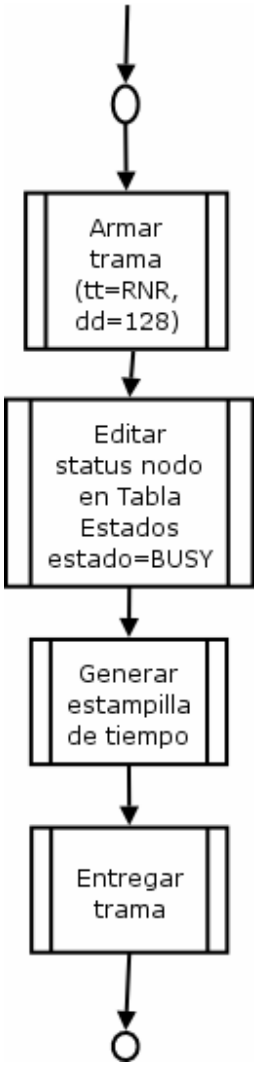


Figura 23. Subrutina Transmitir Mensaje RR

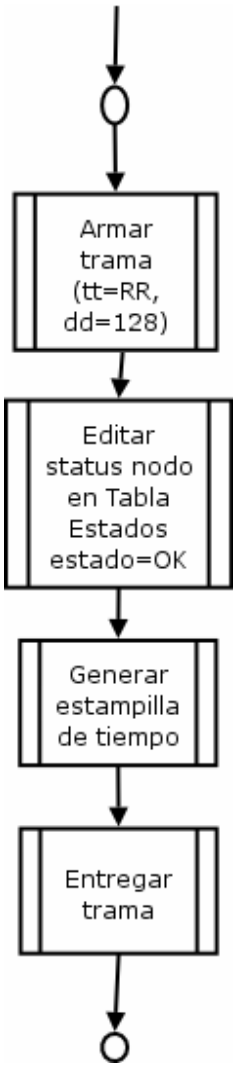
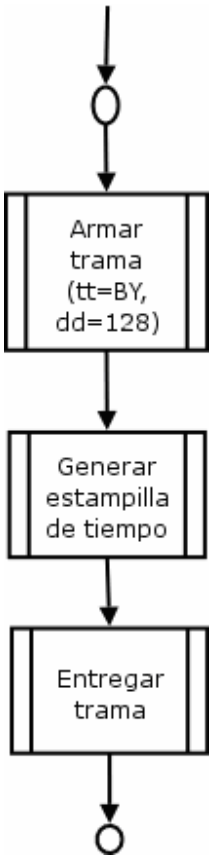
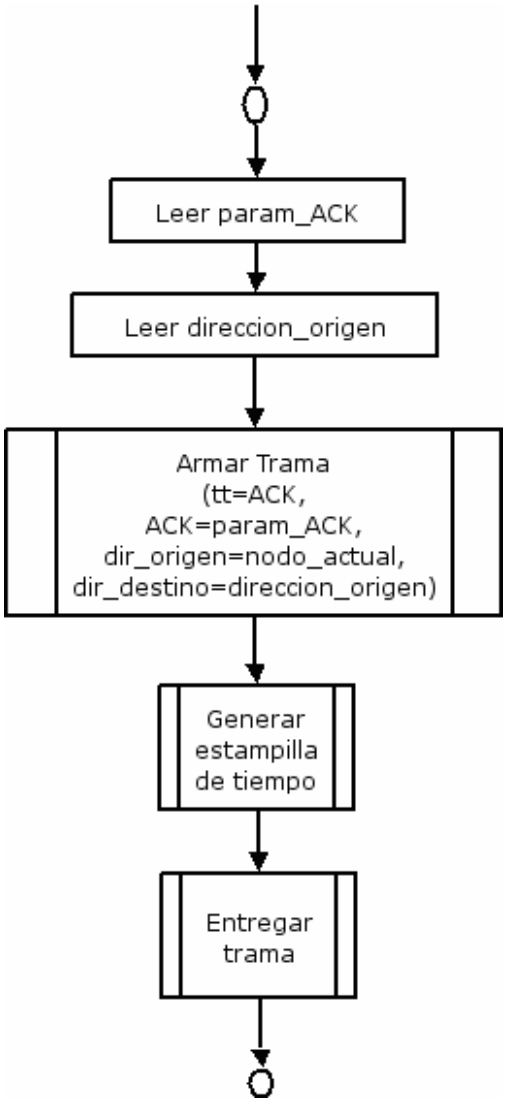


Figura 24. Subrutina Transmitir Mensaje BY



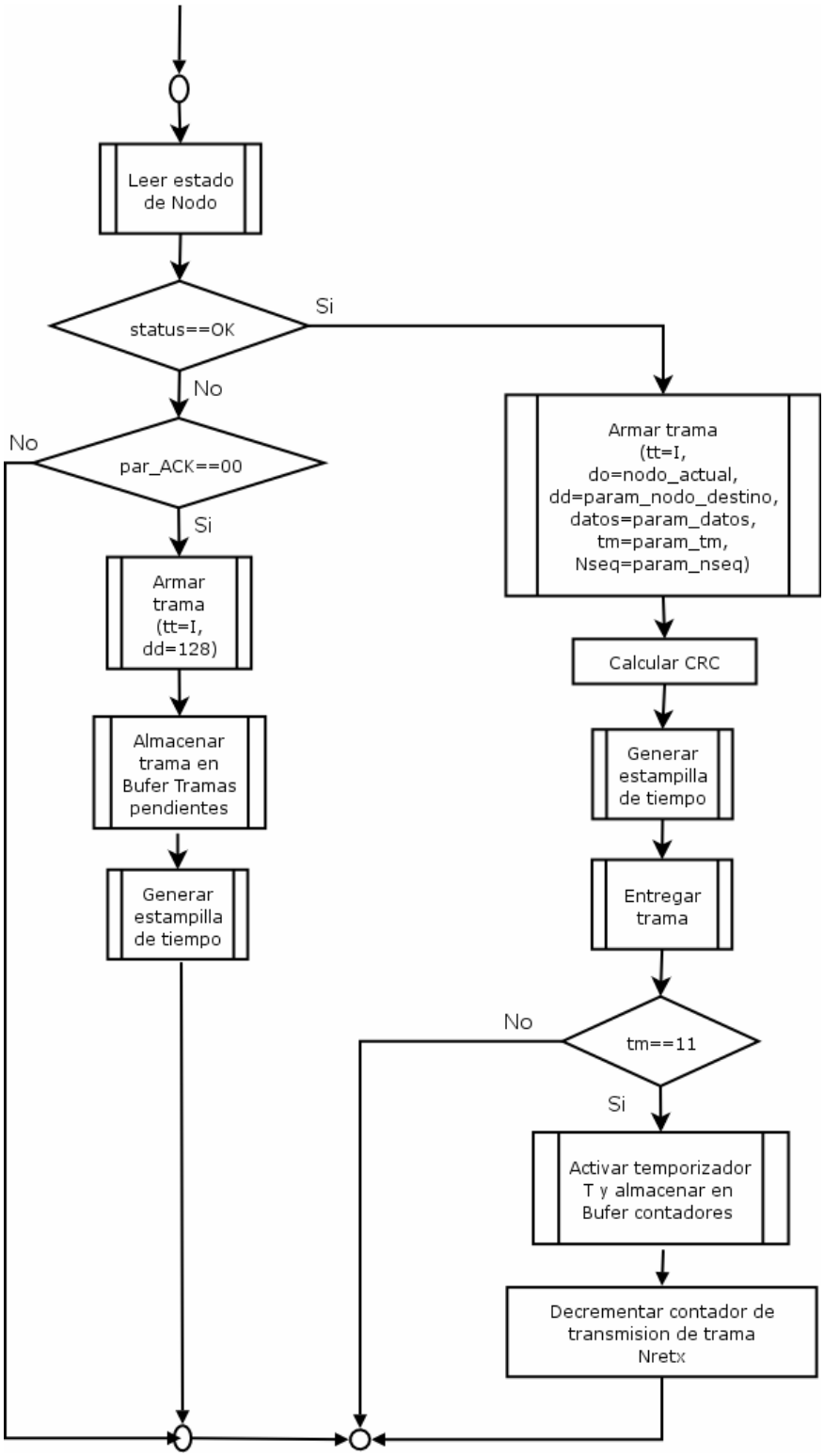
La Figura 24 ilustra el diagrama de la subrutina *TxBY*, que envía el mensaje tipo **BY** a todos los nodos del anillo. Este es un tipo de mensaje especial para la salida ordenada de un nodo de la red.

Figura 25. Subrutina Transmitir ACK



En la Figura 25 se presenta el diagrama de la subrutina de transmisión de mensajes de confirmación de recepción de trama (*TxACK*), en el cual inicialmente se leen los parámetros del encabezado (**header**) de la trama con confirmación recibida. Estos parámetros se colocan en el *encabezado* de la nueva trama a transmitir y se toma uno particular que, es la dirección origen, la cual será asignada como la dirección destino del nuevo mensaje.

Figura 26. Subrutina Transmitir Datos



La subrutina de transmisión de mensajes de datos (*TxI*) se ilustra en el diagrama de la Figura 26, en donde se puede apreciar que antes de transmitir el mensaje se verifica el estado del nodo y solo se transmite en caso de ser **OK**. De lo contrario, y si se trata de una *trama con confirmación*, será almacenada en el *búfer de tramas pendientes por transmitir*. Las tramas sin confirmación, se desechan debido a que los datos se están renovando con una frecuencia alta (lectura de sensores industriales) y si se retarda la transmisión, luego al ser enviados estarían muy desactualizados.

Si el estado del nodo es **OK**, se ejecuta la subrutina "*Armar trama*", en donde se lee la dirección del nodo actual y se configura el encabezado de la nueva trama con la dirección destino y en caso de ser necesario el número de secuencia. Luego, se agregan los datos a transmitir y se calcula el **CRC**. Y si se trata de una trama con confirmación, se debe activar un temporizador para llevar la contabilidad del tiempo de recepción del mensaje de parte del nodo destino. Dicho temporizador, es almacenado en un *búfer de temporizadores*, cuya subrutina de verificación se ilustra en el diagrama de la Figura 18. Finalmente, se decrementa el contador de reenvío de trama con confirmación.

b. Proceso de Recepción

La recepción de mensajes en el sub-nivel de enlace se ilustra en el diagrama de la Figura 27, donde se aprecia que inicialmente se verifica el estado del nodo y si está "*ocupado*" (**BUSY**), no se recibe la trama, sino que se hará el procesamiento de las tramas que están actualmente pendientes en el búfer de recepción. Cabe aclarar que la trama que en ese instante de tiempo acaba de llegar no será desechada, sino que se almacenará en un búfer temporal para luego ser pasada al búfer de recepción. A continuación, después de procesar las tramas pendientes se verifica la capacidad del búfer de recepción y si se tiene un porcentaje superior al 5%, se cambia el estado del nodo a **OK** y se envía un mensaje RR a todos los nodos del anillo, lo cual indica que el nodo está listo para recibir datos.

En caso de que el nodo esté en estado **OK**, se selecciona el tipo de mensaje recibido y se invoca a la subrutina de recepción respectiva. Si la trama contiene información acerca del estado de nodo (Rx*NN, RxSOK, RxRNR, RxRR* y *RxBY*) se actualiza la tabla de estados de acuerdo a la dirección origen. Cuando se trata de tramas tipo *I* (tramas con datos normales), luego de ser procesadas se verifica la capacidad del búfer de recepción para validar si se envía un mensaje de tipo RR o RNR indicando la disponibilidad de recepción del nodo.

Las subrutinas para la recepción de tramas con datos (*RxI* y *RxACK*) se ilustran en los diagramas de la Figura 28 y la Figura 29.

Figura 27. Procedimiento de Recepción Sub-nivel de Enlace

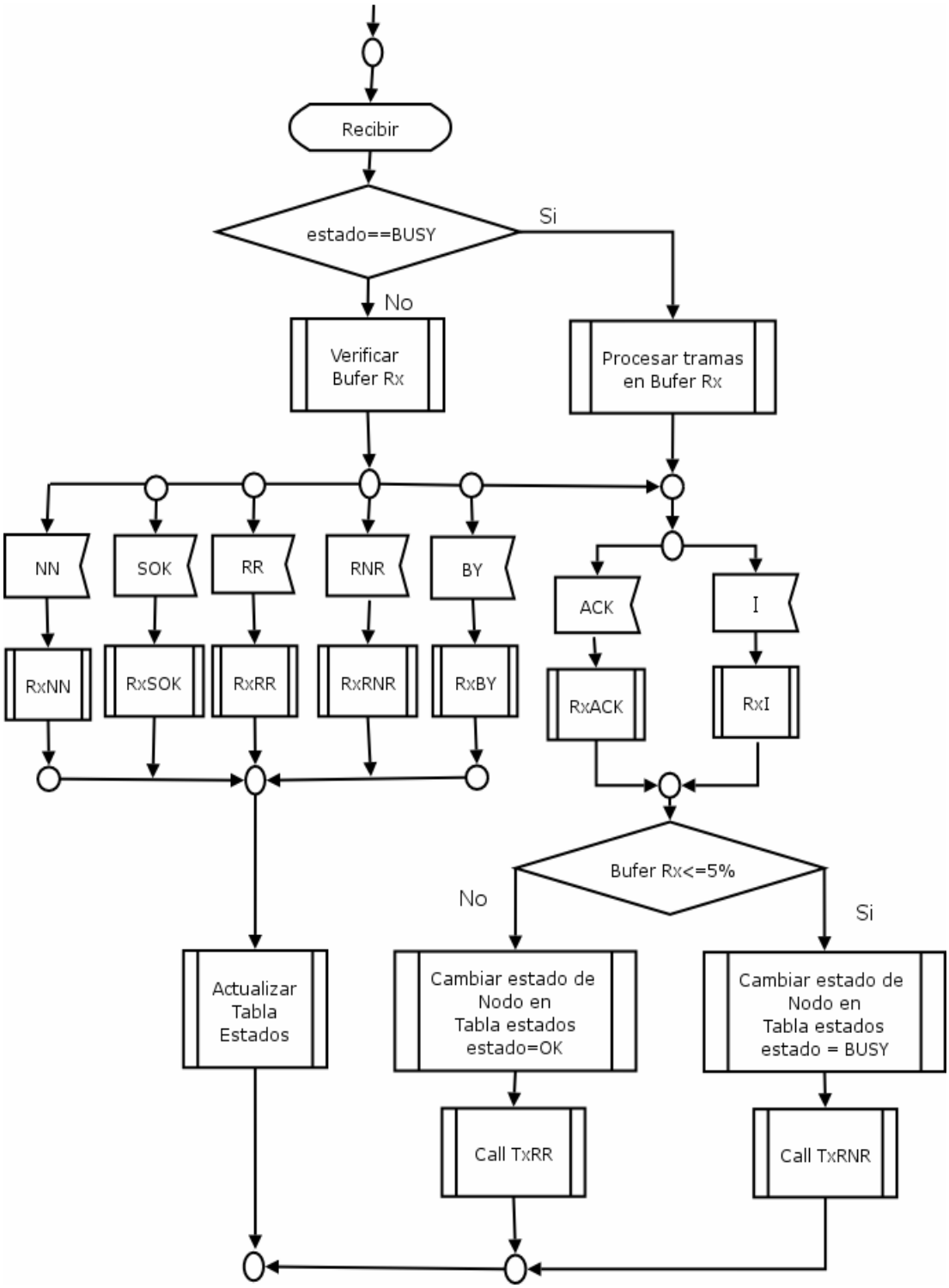
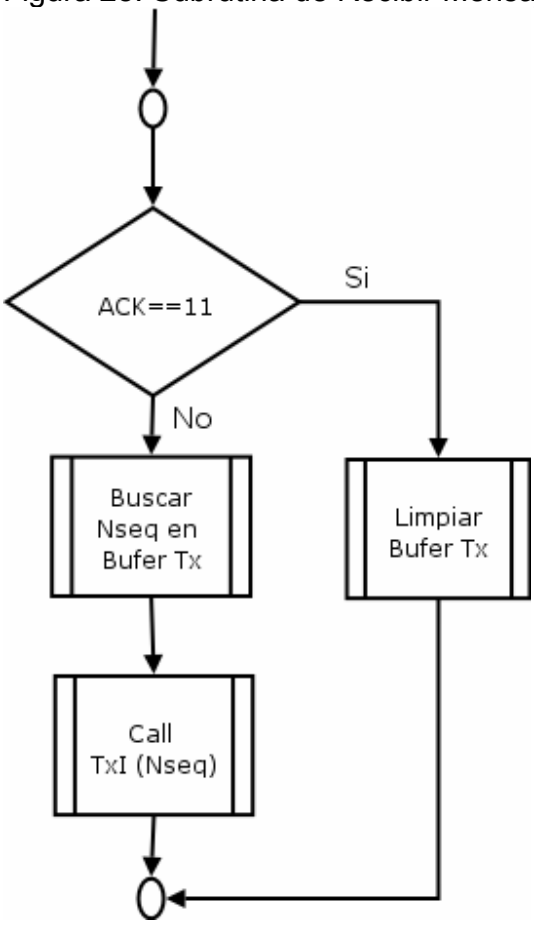
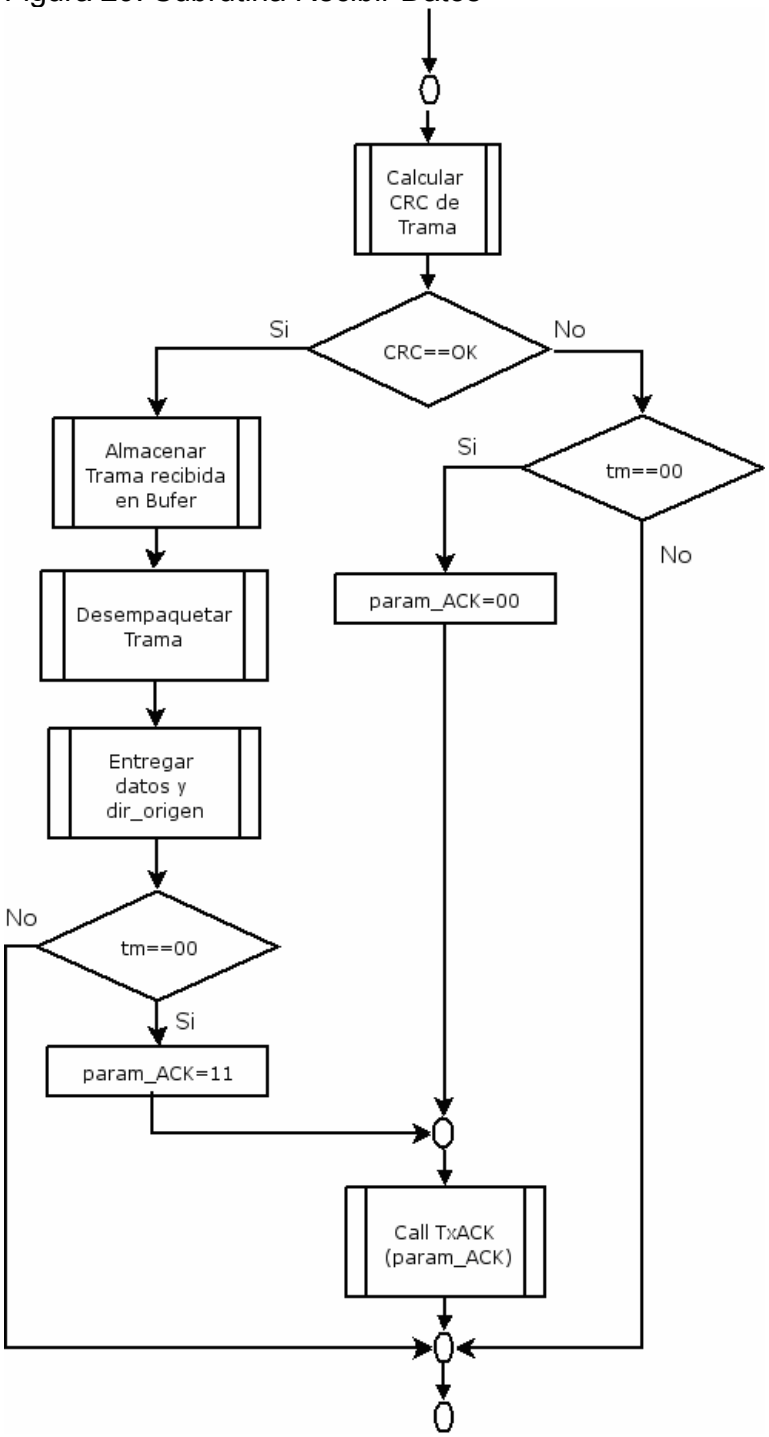


Figura 28. Subrutina de Recibir Mensajes de Confirmación (ACK)



En el diagrama de la subrutina de recepción de tramas de confirmación de mensaje que se aprecia en la Figura 28, se verifica si el mensaje con confirmación enviado se recibió sin errores por parte del destinatario (*ACK = 11*), en cuyo caso se ejecutan las subrutinas que permiten eliminar del búfer de tramas pendientes por confirmación, la trama de la cual se está recibiendo el ACK. Si hubo errores, entonces se ejecutan las subrutinas para realizar el reenvió de la trama en cuestión.

Figura 29. Subrutina Recibir Datos



La Figura 29 ilustra el diagrama de la subrutina de recepción de mensajes, y en ella se puede apreciar que inicialmente se calcula y verifica el **CRC** para validar que la trama llego sin errores. Si hay errores en la trama recibida, entonces se verifica el tipo de mensaje (con confirmación o sin confirmación), dependiendo de lo cual se enviará o no el mensaje tipo **ACK** (confirmación).

En caso de que no haya errores se almacena la trama en el búfer, y se verifica el tipo de mensaje, porque si se trata de una trama con confirmación se deberá enviar un mensaje tipo **ACK**.

Los diagramas de las subrutinas *RxNN, RxSOK, RxRNR, RxRR,* y *RxBY* que, se utilizan para conocer el estado de los nodos del anillo, se presentan en la Figura 30, Figura 31, Figura 32, Figura 33 y Figura 34 respectivamente. En ellos, se puede apreciar que hay dos subrutinas que son comunes "*Leer dirección origen*" que, lee del encabezado de la trama recibida la dirección del nodo origen y "*Buscar dirección origen en Tabla Estados*", la cual realiza la búsqueda de la dirección dada en la tabla de estados para luego realizar la operación de agregar, editar, o eliminar la entrada correspondiente en la tabla de estados del nodo receptor.

Figura 30. Subrutina Recepción Mensaje NN

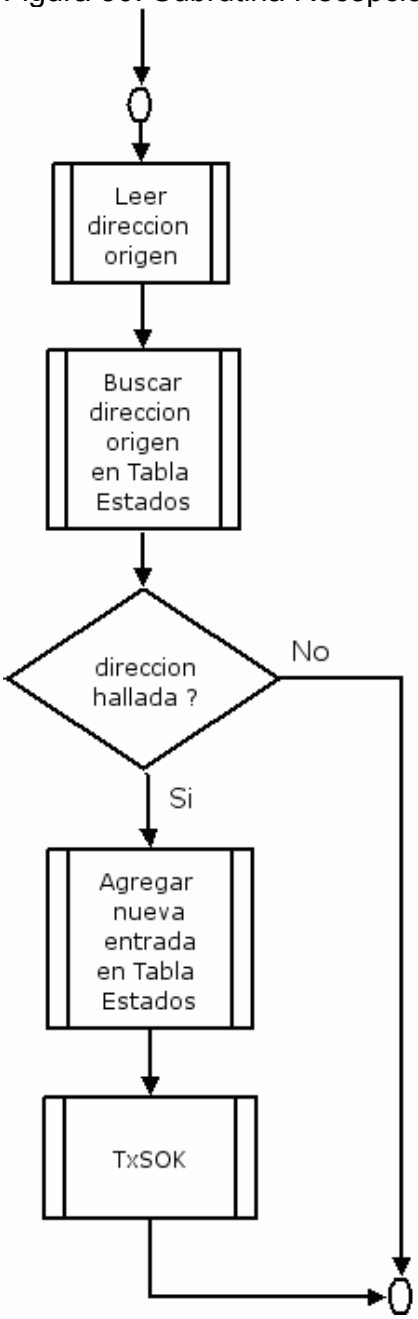


Figura 31. Subrutina Recepción de Mensaje SOK

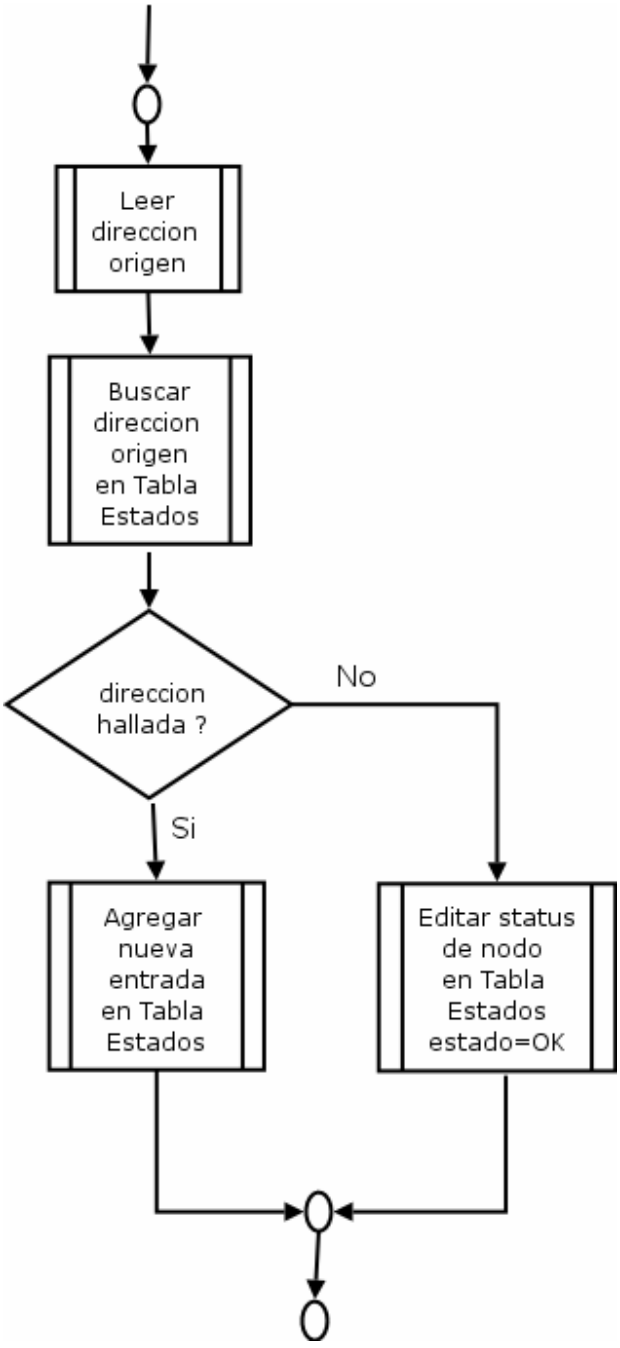


Figura 32. Subrutina Recepción Mensaje RNR

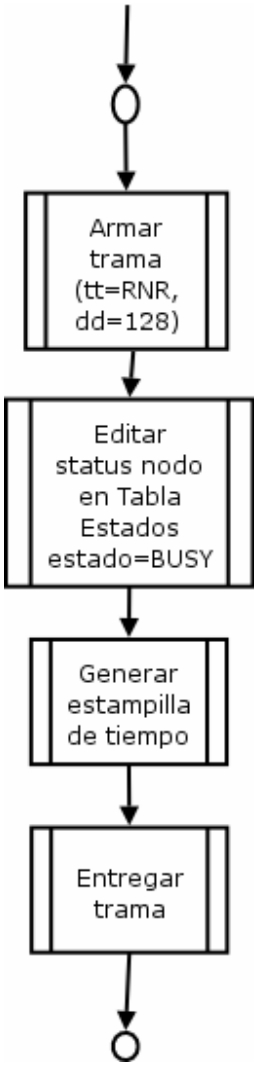


Figura 33. Subrutina Recepción Mensaje RR

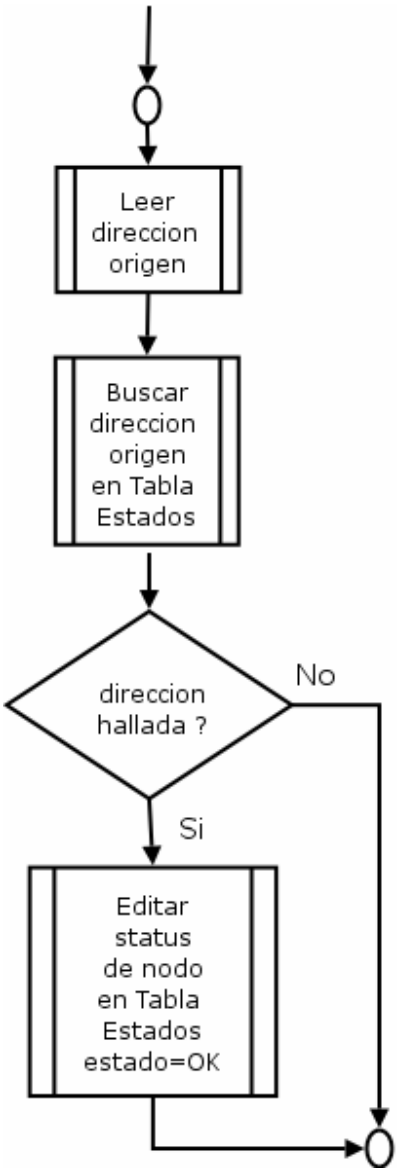
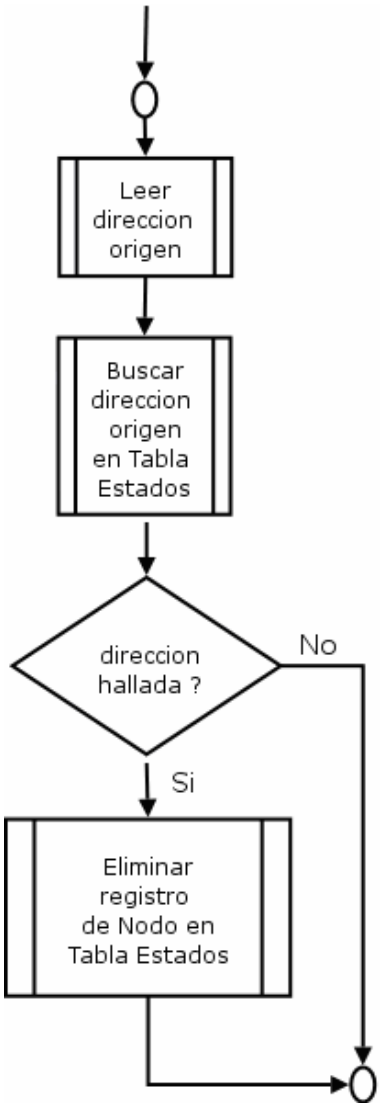


Figura 34. Subrutina Recepción Mensaje BY

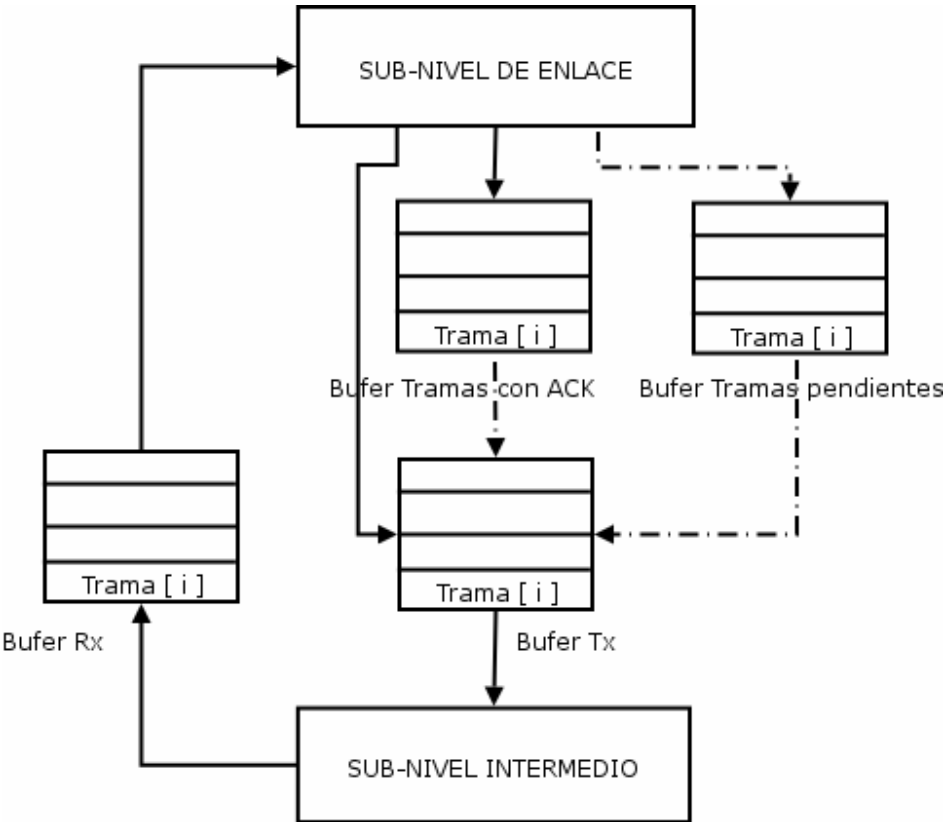


La interacción entre el sub-nivel de enlace y el sub-nivel intermedio se realiza a través del *búfer de transmisión* y el *búfer de recepción*, y en la Figura 35 se puede apreciar un esquema de la entrega de las tramas desde el sub-nivel de enlace hacia el sub-nivel intermedio (transmisión) y viceversa (recepción).

En el caso de la transmisión, existen dos búferes auxiliares, uno para las tramas pendientes por confirmación y otro para las tramas que están pendientes por ser transmitidas. Este último, se utiliza para almacenar las tramas cuando el espacio del búfer de transmisión está en el límite de su capacidad (10%).

Respecto a la recepción de tramas, éstas son colocadas por el sub-nivel intermedio en el *búfer de recepción*, de donde son "*tomadas*" por el sub-nivel de enlace de datos para ser procesadas.

Figura 35. Interacción del Sub-nivel de Enlace y el Intermedio



3.6. VALIDACIÓN DEL PROTOCOLO PDM-RING

Existen diversas herramientas para la especificación de los protocolos de comunicaciones, por ejemplo las máquinas de estado finito y las redes de Petri[33]. En el caso de los protocolos especificados mediante máquinas de estado, la verificación se puede efectuar a través del análisis de asequibilidad, que determina los estados alcanzables y los que no lo son.

Para la validación de la pila de protocolos PDM-Ring, se decidió utilizar una herramienta de software denominada **SPIN**[34] (**Simple Process Interpreter**), que fue desarrollada en Bell Labs bajo la iniciativa de G. Holzmann[35]. Este software se distribuye bajo licencia GPL, y ha sido utilizado para la verificación no solo de protocolos de comunicaciones, sino también en la validación de sistemas concurrentes, y entre sus características particulares está el que permite generar una simulación del protocolo, de tal manera que facilita las pruebas de funcionalidad.

Cabe mencionar que, durante los estudios de la Maestría en Ciencias Computacionales se tuvo la oportunidad de conocer esta herramienta y su gran poder para la simulación y validación de protocolos de comunicaciones.

3.6.1 SPIN.

Es una herramienta de software que se creó especialmente para realizar la verificación de modelos de sistemas distribuidos, y protocolos de comunicaciones. Actualmente está siendo utilizada para la validación de diversos tipos de software (embebido, concurrente).

SPIN se puede ejecutar bajo diferentes plataformas operativas y se puede usar en modo gráfico (Xspin) ó en modo texto (spin). La estructura de SPIN se presenta en la Figura 36 .

Como se aprecia en la Figura 36, uno de los componentes fundamentales en la estructura de SPIN es el lenguaje Promela (**Process Meta Language**) que, se utiliza para implementar los modelos de los sistemas concurrentes a verificar, y el cual permite describir la interacción de los procesos en un sistema.

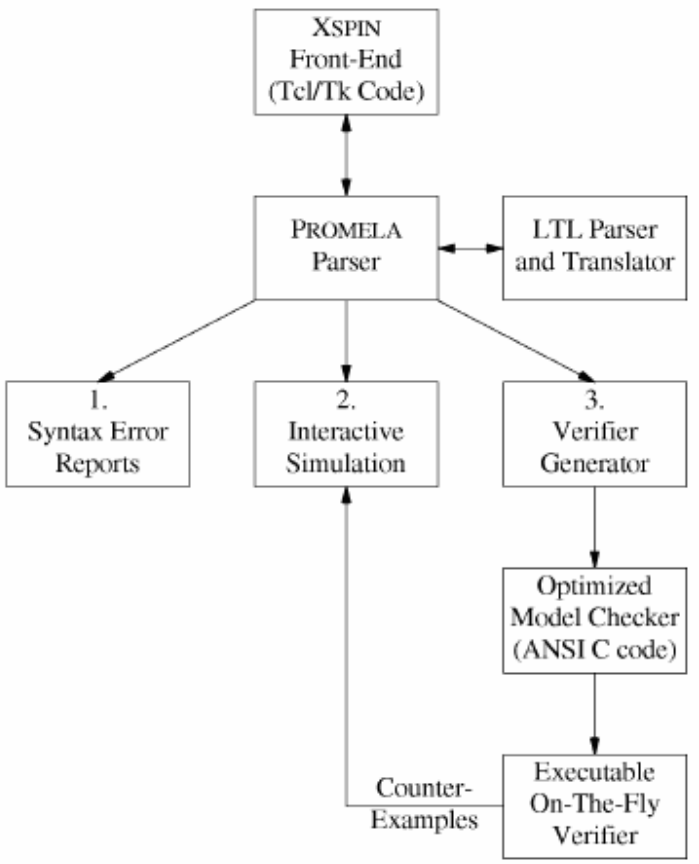
---

[33] TANENBAUM, Andrew S. Redes de Computadores. Tercera Edición. Prentice Hall, 1997. Pag. 219 - 224.

[34] La herramienta está disponible en Internet y se puede descargar del sitio web - http://spinroot.com/spin/whatispin.html.

[35] HOLZMANN, Gerard J., «The Model Checker Spin», IEEE Transactions on Software Engineering, Vol. 23, No. 5, Mayo 1997, pag. 279-295.

Figura 36. Estructura de SPIN



HOLZMANN, Gerard J., «The Model Checker Spin», IEEE Transactions on Software Engineering, Vol. 23, No. 5, Mayo 1997, pag. 279-295.

Los modelos en **Promela** son definidos utilizando tres tipos de objetos, a saber:

- Procesos, son objetos globales y obviamente todo modelo consta de por lo menos uno.
- Canales, se utilizan para modelar la transferencia de datos entre los procesos.
- Variables, son de tipo numerico entero (bit, byte, short, int) pueden ser locales o globales.

En **Promela** el comportamiento de un modelo a validar se define como el conjunto de todas las secuencias de ejecución posibles, entendiéndose secuencia de ejecución como un conjunto finito de estados ordenados y que un estado está dado por el conjunto de todas la variables locales y globales. La union de los estados incluidos en el comportamiento del sistema define el conjunto de estados alcanzables del modelo.

Los críterios de validación que pueden ser especificados en **Promela** son:

- Bloqueos (deadlocks), en el estado final de una secuencia de ejecución terminal todos los procesos instanciados deben haber terminado y todos los canales deben estar vacíos.
- Invariantes del sistema, condiciones que no pueden ser invalidadas por ningun estado del sistema.
- Ciclos erróneos,
- Aserciones, son críterios de corrección que son expresados como expresiones booleanas y que deben ser verdaderas cada vez que un proceso alcanza un estado dado.
- Afirmaciones temporales.

El otro elemento fundamental en la estructura de **SPIN** es **pan** (**Promela Analizer**), que es el programa analizador, el cual tiene incluidos los algoritmos de búsqueda a través de los cuales se verifica la validez del modelo implementado en **Promela**.

Generalmente, las herramientas de validación automatizadas como **SPIN** implementan algoritmos que intentan generar e inspeccionar todos los estados del sistema que son alcanzables a partir de un estado inicial. Existen tres tipos de algoritmos para verificación:

- *Búsqueda completa o exhaustiva* (sistemas con hasta 10 ^ 6 estados), se lleva a cabo un análisis de todos los estados alcanzables y los no alcanzables, donde cada estado alcanzable y toda secuencia de estados alcanzables pueden ser verificados por un conjunto finito de críterios de correción (bloqueos, invariantes del sistema, entre otros). Su principal desventaja es que depende de la capacidad de memoria y la velocidad de procesamiento del hardware en que se ejecute.
- *Búsqueda parcial* (sistemas hasta con 10 ^ 8 estados), se basa en la premisa de que en la mayoria de los casos prácticos, el número máximo de estados que pueden ser analizados, *A*, es solo una fracción del número total de los estados alcanzables *R*. Existen diversos críterios para la selección de los estados sucesores, entre ellos el límite de profundidad, la búsqueda dispersa, la búsqueda probabilística, el orden parcial y la selección aleatoria.
- *Simulación aleatoria* (sistemas con mas de 10 ^ 8 estados), se descartan los conjuntos *A* y *B*, y se explora el espacio de estados restante del total. Su característica principal es que, funciona independientemente del tamaño y la complejidad del sistema (inclusive para sistemas infinitos).

La diferencia fundamental entre la búsqueda exhaustiva y la búsqueda parcial es que, la primera demuestra la ausencia de errores, mientras que el propósito de la segunda es demostrar la presencia de errores en el modelo verificado[36].

En el caso de **SPIN**, para la búsqueda parcial (o controlada) se tiene implementado un algoritmo denominado **supertrace** o **Bitstate hashing**[37], que permite verificar modelos con una gran cantidad de estados debido a que utiliza solo 2 ó tres bits para almacenar la información de los estados (*vector de estados*[38]). En este caso, se utiliza un factor especial para indicar la confiabilidad de la verificacion realizada. Este factor se denomina "**hash-factor**" y se calcula diviendo la capacidad de memoria (en bits) del hardware utilizado para la validación entre el número de estados alcanzados.

El proceso para realizar la simulación y validación de un modelo en **SPIN**, se puede resumir de la siguiente manera, se inicia con la implementación del modelo dado en **Promela**, luego dependiendo del tamaño del modelo se puede continuar con la simulación aleatoria (modelos muy grandes) y/o con la búsqueda (exhaustiva o parcial) de los errores del protocolo, según los críterios de verificación definidos para el modelo.

La ilustración del proceso completo para simulación y validación usando **SPIN** se presenta en la Figura 37.

De acuerdo con lo anterior, se definieron las propiedades que rigen el comportamiento del sub-nivel intermedio y de enlace. Dichas propiedades se obtuvieron mediante una abstracción de los requerimientos y reglas de procedimiento planteadas. A través de dichas propiedades, se trató de identificar las inconsistencias o contradicciones en el diseño del protocolo, se refinó, y se verificó que el conjunto de propiedades si es válido.

A continuación se detallan las propiedades definidas para cada uno de los sub-niveles de la pila de protocolos.

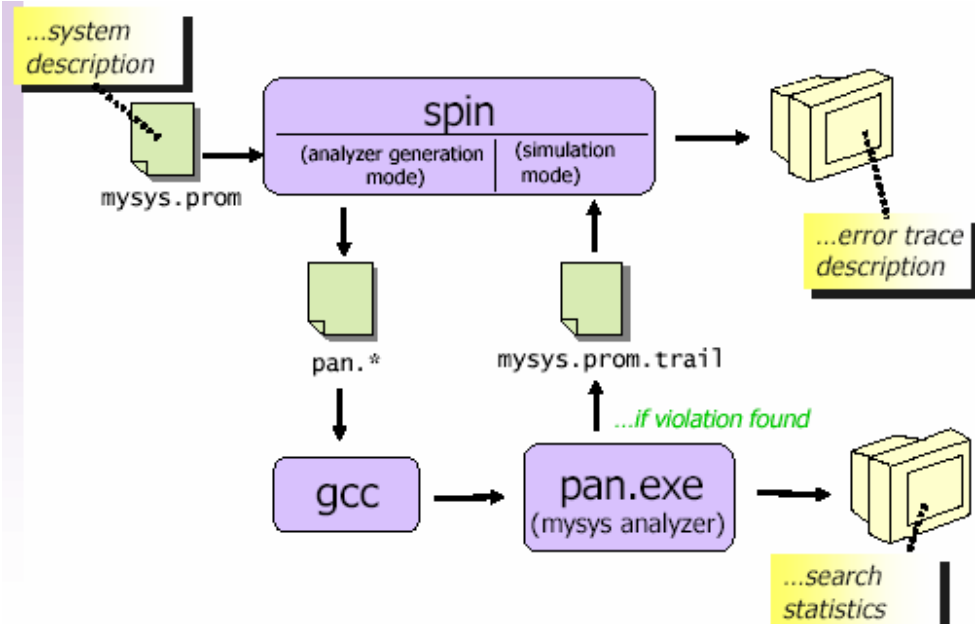
---

[36] HOLZMANN, Gerard J., Design and Validation of Computer Protocols, Prentice Hall, 1991.p. 217-243.

[37] HOLZMANN G.J., «An Analysis of Bitstate Hashing», Formal Methods in System Design, Vol. 13, No. 3, Kluwer, November 1998, pag. 287-305.

[38] Habitualmente, la información de cada estado del sistema (variables locales y globales, y el estado de los canales) es mantenida en un "vector" que habitualmente ocupa algunos bytes de memoria.

Figura 37. Proceso de Simulacion y Validación con SPIN



"*CIS 842*: *Specification and Verification of Reactive Systems*". Matt Dwyer, John Hatcliff. Kansas State University. URL - http://spinroot.com/spin/Doc/SpinIntro.pdf

3.6.2 Propiedades del sub-nivel Intermedio.

Los puntos críticos del protocolo de este sub-nivel que se verificaron fueron:

- Estado de los canales.
- Tramas pendientes por transmitir.
- Fin de la trama.
- Si un nodo posee un canal.

Propiedad - Estado del Canal

Cada canal de comunicación puede estar en los siguientes estados: vacío, ocupado con datos propios de la transmisión u ocupado con datos que identifican que el canal fue expropiado por otro nodo, ya sea por que no habían canales disponibles o por que dos nodos se apoderaron del mismo canal. Para todo canal este atributo se inicializará en "vacío" y las posibles transiciones son:

a. Si el canal está vacío se espera que en algún momento alguno de los nodos lo ocupe transmitiendo datos pasando a estado ocupado.

```
[ ] estado_canal[i] == VACIO  → <>   estado_canal[i] = OCUPADO
```

b. Si un canal está ocupado por que un nodo está transmitiendo haciendo uso de él, puede darse que termine la transmisión de forma natural.

[ ] estado_canal[i] == OCUPADO → <> estado_canal[i] = VACIO

c. Si un canal está ocupado por que un nodo está transmitiendo haciendo uso de él, puede que otro nodo expropie al nodo corriente. El nodo que expropia informa a los de de este hecho enviando un mensaje por el canal.

[ ] estado_canal[i] == OCUPADO → <> estado_canal[i] = EXPROPIADO

d. Es posible que dos nodos se hayan apoderado del mismo canal, es esta caso el nodo con mayor prioridad permanece en el canal y el nodo de menor prioridad informa a los de de este hecho enviando un mensaje por el canal.

[ ] estado_canal[i] == OCUPADO → <> estado_canal[i] = COLISIONADO

e. Una vez que un nodo informe que el canal fue expropiado o que dos nodos colisionaron, se pasará al estado de ocupado dado que se inicia el proceso de transmisión de datos y no de estados particulares.

[ ] estado_canal[i] == EXPROPIADO → <> estado_canal[i] = OCUPADO

[ ] estado_canal[i] == COLISIONADO → <> estado_canal[i] = OCUPADO

Propiedad - Nodo se apodera de un canal
Para que un nodo pueda transmitir debe apoderarse de un canal, luego cada nodo debe tener la posibilidad de saber de que canales se ha apoderado. Dado que puede poseer todos los canales, este atributo se define como un arreglo denominado *canales_posee*. Todos los elementos de este arreglo se inicializan en cero. Los posibles estados por los que este atributo puede pasar son:

a. Si el nodo no se ha apoderado de un canal, eventualmente podrá hacerlo y se podrá hacer en dos casos: si el canal está vacío o lo ha expropiado.

[ ] canales_posee[i] == NO → <> canales_posee[i] = SI

b. Si el nodo ha terminado la transmisión debe indicar que ya no posee el canal y liberarlo.

[ ] canales_posee[i] == SI → <>canales_posee[i] = NO

Propiedad - Fin de trama
Durante el proceso de transmisión un nodo debe informar si ha finalizado la transmisión de una trama por un canal en particular; para todo nodo este atributo inicia en “NA”, es decir no aplica. Las transiciones son las siguientes:

a. Si un nodo no posee tramas en transmisión entonces

fin_trama[i] = NA

b. Si un nodo requiere transmitir una trama nueva entonces se apodera de un canal, este ya no está vacío y el fin de trama está en no.

[ ] fin_trama[i] == NA → <> fin_trama[i] = NO

c. Si un nodo ha finalizado la transmisión de la trama, deberá informarlo.

[ ] fin_trama[i] == NO → <> fin_trama[i] = SI

d. Una vez que finaliza la transmisión de la trama y el nodo no se apodera nuevamente del canal el estado de fin_trama pasa a “NA”.

[ ] fin_trama[i] == SI → <> fin_trama[i] = NA

Propiedad - Tramas por Transmitir o en proceso de transmisión
Cada nodo debe poder identificar si posee tramas nuevas por transmitir o posee tramas en proceso de transmisión. Las transiciones son las siguientes:

a. Si un nodo tiene tramas nuevas pendientes por transmitir en algún momento puede dejar de tenerlas por que se ha apoderado de canales y las ha transmitido.

[ ] tramas_pendientes == SI → <> tramas_pendientes = NO

b. Si un nodo no posee tramas nuevas por transmitir, eventualmente podrá tenerlas.

[ ] tramas_pendientes == NO → <> tramas_pendientes = SI

c. Si un nodo posee tramas en proceso de transmisión debe poder identificarlo eventualmente finalizará la transmisión de todas estas tramas y cambiará este atributo.

[ ] tramas_transmitiendo == SI → <> tramas_transmitiendo = NO

d. Si un nodo no posee tramas en proceso de transmisión eventualmente las tendrá

[ ] tramas_transmitiendo == NO → <> tramas_transmitiendo = SI

3.6.3 Propiedades del sub-nivel de Enlace.
Se debió hacer una selección de los puntos críticos del protocolo de este sub-nivel a verificar, y éstos fueron:

- El estado de los nodos, se debe actualizar frecuentemente y sirve para el control de flujo.
- Verificación de errores, al momento de la recepción de cualquier trama se debe verificar si presenta errores.
- Las tramas con confirmación (**ACK**), se pueden presentar dos situaciones a saber, la confirmación de una trama llego con error o la trama se recibió bien.
- Reenvío de tramas, solo se realiza para las tramas con confirmación y se debe efectuar cuando la confirmación de una trama se recibe con error.
- La saturación del *búfer de recepción*, este búfer tiene capacidad finita y se puede llenar, por lo cual se debe detener la recepción de tramas y actualizar el estado del nodo, que pasa a ser "*ocupado*" (**BUSY**).

Propiedad - Estado del Nodo
El nodo podrá estar en tres estados: normal o activo (**OK**), ocupado (**BUSY**), e inactivo o fuera de servicio (**OUT**). El nodo podrá recibir tramas solo si está en el estado normal, y el estado *ocupado* se presenta cuando el búfer de recepción se ha llenado.

La transmisión y la recepción pueden darse de forma simultánea e independiente. Las posibles transiciones son:

a. Si un nodo está inactivo (OUT), eventualmente podrá pasar al estado activo (OK).

[ ] estado_nodo == OUT → <> estado_nodo = OK

b. Si un nodo está activo (OK), eventualmente podrá pasar al estado inactivo (OUT).

[ ] estado_nodo == OK → <> estado_nodo = OUT

c. Si un nodo está activo, eventualmente podrá pasar al estado ocupado.

[ ] (estado_nodo == OK && bufer_Rx == full) → [ ] estado_nodo = BUSY

d. Si un nodo está ocupado, eventualmente podrá pasar al estado activo.

[ ] (estado_nodo == BUSY && bufer_Rx == libre) → [ ] (estado_nodo = OK || estado_nodo = OUT)

Propiedad - Recepción de trama
Una trama pueden presentar errores (alteraciones) o no, y esto se verifica en el momento de la recepción. Si se trata de una trama con confirmación y presenta errores no es aceptada y se envía un mensaje de confirmación de trama recibida con errores, y si es una trama sin confirmación, entonces no se acepta, es decir la trama es desechada. En el caso de las tramas con confirmación aceptadas, se debe enviar la confirmación (**ACK**)

a. Si la trama que se recibe es sin confirmación y presenta errores es rechazada.

[ ] (!tipo_trama[i] == ACK && trama[i]_error == si) → [ ] trama[i] = desechar

b. Si la trama que se recibe es sin confirmación y no presenta errores es aceptada.

[ ] (!tipo_trama[i] == ACK && trama[i]_error == no) → [ ] trama[i] = aceptar

c. Si la trama que se recibe es con confirmación y presenta errores no es aceptada y se solicita su reenvió (especie de confirmación negativa).

[] (tipo_trama[i] == ACK && trama[i]_error == si) → [] (trama[i] = rechazada && trama[i]_confirmacion = negativa)

d. Si la trama que se recibe es con confirmación y no presenta errores es aceptada y se confirma su recepción (positiva).

[] (tipo_trama[i] == ACK && trama[i]_error == no) → [] (trama[i] = aceptar && trama[i]_confirmacion = positiva && contador_reenvio_trama[i] = 0 && temporizador_trama[i] = 0)

Propiedad - Saturación de Búfer de Recepción
Las tramas que se reciben son almacenadas en un búfer que se denomina búfer de recepción (*búfer_Rx*), el cual puede ser colmado (saturado). Si se presenta la saturación del *búfer de recepción*, el nodo debe cambiar su estado a "*ocupado*" y

no continuará con la recepción de tramas hasta que no haya espacio disponible en el *búfer de recepción*.

a. Si el búfer de recepción se ha llenado completamente, eventualmente podrá estar libre.

[] bufer_Rx == full → <> bufer_Rx = libre

b. Si el búfer de recepción está libre y el nodo está recibiendo tramas, eventualmente podrá pasar al estado lleno.

[] bufer_Rx == libre → <> bufer_Rx = full

Propiedad - Tramas pendientes por transmitir
Durante el proceso de transmisión, una copia de las tramas enviadas con confirmación es almacenada en un búfer especial - *bufer_Tx_ACK_pendiente* , y podrán ser utilizadas en caso de ser necesaria la retransmisión. Estas tramas, se irán dando de baja en la medida en que se reciba la confirmación de recibo sin errores por parte del destinatario o se llegue al límite del número de retransmisiones.

El *bufer_Tx_ACK_pendiente* será revisado con una cierta frecuencia de tiempo (temporizador), y si hay tramas, serán retransmitidas aquellas cuyo temporizador se haya vencido (**time out**).

Si al momento de enviar una trama con confirmación el *bufer_Tx* está lleno, la trama se almacenará en un búfer adicional - *bufer_pendientes_Tx*, que será revisado con una cierta frecuencia, y cuyas tramas serán transmitidas cuando el *bufer_Tx* esté libre.

Las posibles transiciones son las siguientes:

a. Si el búfer de transmisión está libre, eventualmente podrá pasar al estado lleno (full).

[] bufer_Tx == libre → <> bufer_Tx = full

b. Si el búfer de transmisión está lleno, el búfer de tramas pendientes por transmitir eventualmente se podrá colmar.

[] (bufer_Tx == full && tipo_trama[i] == ACK) → <> bufer_pendientes_Tx = full

c. Sí el búfer de tramas pendientes está lleno, eventualmente se podrá liberar.

```
[] (bufer_pendientes_Tx == full && !bufer_Tx == full) → <>
bufer_pendientes_Tx = libre
```

d. Si el temporizador del búfer de tramas pendientes por transmitir se vence, se descartaran las tramas vencidas , y eventualmente el espacio del búfer de tramas pendientes por transmitir quedará libre.

```
[] (temp_pendientes_Tx == 0 && !bufer_Tx == libre ) → <>
bufer_pendientes_Tx = libre
```

e. Si el temporizador del búfer de tramas pendientes por transmitir se vence y el búfer de transmisión está libre se transmitirán las tramas pendientes, y el espacio del búfer de tramas pendientes por transmitir quedará libre.

```
[] (temp_pendientes_Tx == 0 && bufer_Tx == libre ) → []
bufer_pendientes_Tx = libre
```

f. El temporizador del búfer de tramas pendientes por transmitir (*temp_pendientes_Tx*), solamente se activa cuando este búfer no está vacío.

```
[] (!bufer_pendientes_Tx == libre && temp_pendientes_Tx == 0) → []
temp_pendientes_Tx > 0
```

g. Sí el búfer de tramas pendientes por confirmación tiene tramas y se vence el temporizador (*temp_pendientes_ACK*), se verifica el estado del búfer de transmisión y si tiene espacio, se retransmiten las tramas pendientes por ACK que no hayan alcanzado el limite de reenvíos. El búfer de tramas pendientes por confirmación eventualmente quedará libre y se activa su temporizador.

```
[] (temp_pendientes_ACK == 0 && !bufer_pendientes_ACK == libre &&
!bufer_Tx == full ) → <> (bufer_pendientes_ACK = libre &&
temp_pendientes_ACK > 0)
```

```
[] (temp_pendientes_ACK == 0 && !bufer_pendientes_ACK == libre &&
!bufer_Tx == full && !contador_reenvio_trama[i] == 0 ) → []
(transmitir_trama[i] = si && contador_reenvio_trama[i]-- &&
eliminar_transmitir_trama[i] = no )
```

h. Sí el búfer de tramas pendientes por confirmación está vacío, se vence el temporizador (*temp_pendientes_ACK*), y no hay tramas para seleccionar, entonces el temporizador se desactiva.

```
[] (bufer_pendientes_ACK == libre && temp_pendientes_ACK == 0) →
[] (temp_pendientes_ACK = 0)
```

i. El temporizador de tramas pendientes (*temp_pendientes_ACK*), solamente se activa cuando el búfer de tramas pendientes no está vacío.

```
[] (!bufer_pendientes_ACK == libre && temp_pendientes_ACK == 0) →
[] temp_pendientes_ACK > 0
```

3.7. SIMULACIÓN Y VALIDACIÓN CON SPIN

La secuencia de las tareas realizadas para la simulación y validación de la pila de protocolos usando **SPIN**, fue la siguiente:

- Codificación de los modelos en **Promela**.
- Generación de simulaciones aleatorias de los modelos implementados.
- Generación automatica del programa analizador[39].
- Compilación del analizador generado utilizando los dos tipos de búsqueda (búsqueda exhaustiva y parcial) de estados alcanzables a partir de un estado inicial.
- Inspeccion de las trazas de error producidas por el analizador y depuración.
- Verificación de los resultados generados por el analizador.

A continuación, se presenta la descripción de los modelos de cada sub-nivel de la pila de protocolos que, fueron implementados en el lenguaje **Promela** para la simulacion y validación con **SPIN**.

3.7.1 Modelo del Sub-nivel Intermedio

La simulación y verificación de este sub-nivel se abstrae de los datos, esto debido a que el mismo autor, G. Holzmann, recomienda simplificar el modelo y rescatar aquellas propiedades que se consideran críticas para el protocolo. Debido a las limitaciones que posee SPIN para la verificación, incrementar las condiciones que se deben tener en cuenta, si se incluyen los datos, hace que este proceso no se pueda verificar desde el punto de vista computacional. De otra parte, las condiciones que dependen de los datos mismos de los mensajes son fácilmente implementables en un lenguajes de alto nivel, luego no requieren de verificación exhaustiva como si lo requieren por ejemplo la transición entre estados, que no se presenten **timeout** o errores.

Teniendo en cuenta esto, el modelo del sub-nivel Intermedio tiene las siguientes características:

- Número variables de canales.
- Número variable de nodos.
- La finalización de la simulación se controla por el número de ranuras de tiempo que se deseen procesar.

[39] En realidad, **SPIN** genera un conjunto de programas en lenguaje C (pan.*) que al ser ejecutados, permiten la búsqueda exhaustiva y/o parcial del modelo implementado en Promela.

- No se transmiten datos binarios (0, 1) en vez de esto, se transmite el estado en el que está el canal: **LIBRE** – el canal está disponible para ser utilizado por el nodo; **OCUPADO** – un nodo está transmitiendo datos, **EXPROPIADO** – un nodo envió un mensaje de expropiación a los de, indicando que deben ignorar lo que hayan recibido hasta ahora y el nodo que estaba apoderado del canal debe liberarlo para dar paso a los datos del nuevo nodo que se apoderó del canal.
- Número de datos a transmitir variable: lo que se traduce en que cuando un nodo se apodere de un canal, la transmisión de esta trama ocupara tantas ranuras de tiempo como número de datos especifique.
- Cada nodo posee búferes que le permiten saber por canal:
    - Si posee o no el canal y cuantos datos (ranuras) ha transmitido.
    - Si expropió el canal.
    - El estado de cada **Bucket**.
    - La cantidad de datos recibidos.
    - El estado de cada trama enviada.
- Se detecta que se ha finalizado el envío de una trama por el control de los datos que han sido enviados.
- En una ranura de tiempo un nodo se apoderará de solo un canal aunque tenga varias tramas pendientes.
- En anillo se representa por una matriz en la que se almacena por cada canal el estado del mismo.
- Una vez finaliza el proceso de recepción y transmisión para cada nodo el anillo se rota haciendo que los datos se desplacen al siguiente nodo en el anillo.
- Dado que el modelo se abstrae de los datos, una trama que sea enviada será recibida por todos los nodos y el proceso de reconocer si es para un nodo en particular se reduce a la comparación de parte de los datos que se reciben que no se incluye en el modelo.
- Por simplicidad, se expropia solo un canal, el último.
- Un nodo solo expropiará un canal mientras que finaliza completamente la transmisión de una trama.
- Los mensajes de expropiación se genera de forma automática sin tener en cuenta la prioridad de los nodos, ya que no se manejan datos en la trama.

Con esta definición se verifican las transiciones y estados que se consideran relevantes para el modelo. En el Anexo A está el modelo implementado en el lenguaje Promela.

3.7.2 Modelo del Sub-nivel de Enlace.

Al igual que en el sub-nivel intermedio, se realiza una abstraccion de los datos, buscando simplificar el modelo y rescatar aquellas propiedades que se consideran críticas para el protocolo del sub-nivel de enlace, a saber el control de flujo y la verificación de errores.

En el sub-nivel de enlace, aparte de las propiedades mencionadas anteriormente, se definieron algunas más, pero debido a que éstas dependen en su mayoría de ciertos temporizadores y que la validación en **SPIN** de un modelo tan robusto sería demasiado complejo de procesar en el equipo de computo con que se contaba para su ejecución[40], entonces se optó por simplificar el modelo y verificar únicamente los aspectos más relevantes del protocolo.

Para facilitar el modelamiento del protocolo fue necesario realizar tres modelos y estos se implementaron usando el lenguaje **Promela**. Las propiedades que se validaron fueron:

- Estado del nodo.
- Recepción de trama y verificación de errores en tramas con confirmación.
- Saturación del búfer de recepción.

Los tres modelos que se implementaron fueron:

- *Modelo 1*, se implementó el modelo de un anillo con dos nodos, donde se transmiten los diversos tipos de mensajes (datos, estado, y control) usando dos canales, uno para el envío de mensajes y otro para la recepción. Utiliza dos procesos, uno para la transmisión y otro para la recepción. Además, controla el búfer de recepción basándose en una capacidad predefinida (número de tramas que puede almacenar) y el estado del nodo se simula con una variable numérica entera. Para establecer el fin de la ejecución de la simulación del intercambio de mensajes entre los dos nodos, se definió un número máximo de tramas a transmitir. La ocupación del búfer de recepción se maneja con un contador numérico que va contando las tramas recibidas, y cuando alcanza el valor predefinido (es decir se llena el búfer !), el nodo cambia de estado (pasa a **BUSY**) y se simula la liberación del búfer, luego de lo cual cambia de estado nuevamente (pasa a **OK**), el contador toma el valor 0, se envía un mensaje de tipo **SOK** y el nodo continua recibiendo tramas. Incluso, se simula la salida del anillo por parte de uno de los nodos, para lo cual se cambia su estado (pasa a **OUT**) y se envía un mensaje tipo **BY** (salida controlada), después de lo cual el nodo "ingresa" nuevamente a la red cambiando de estado y enviando un mensaje de tipo **SOK**. Este modelo es el más completo de los tres, e implementa las propiedades relevantes del protocolo del sub-nivel de enlace. Pero, su desventaja es que al ser más complejo, demanda un mayor poder de cómputo, lo cual dificultó su verificación. La implementación en Promela de este modelo está en el Anexo B.

[40] Las pruebas se realizaron en máquinas de CPU Intel Pentium IV a 1.7 GHz con 256 MB de RAM, y bajo las plataformas operativas Linux Fedora Core 3 y Linux Red Hat 7.3.

- *Modelo 2*, se implementó un modelo simplificado que consiste en un solo nodo que envía y recibe tramas usando dos canales, uno de salida y otro de entrada. Los dos canales están unidos, y de esta manera se simula el envió y la recepción de los mensajes. Se envían y reciben mensajes de todos los tipos y se cuenta con una variable numérica entera para mantener el estado del nodo. En cuanto al búfer de recepción, éste se modela con un contador numérico que se incrementa cada vez que se envía una trama, y la cantidad límite de tramas a enviar se modela con un valor numérico predefinido. Este modelo permite verificar el intercambio de mensajes, el cambio de estado del nodo, y la saturación del búfer de recepción. La ventaja de esta implementación radica en que, es bastante "liviana" y puede ser computada fácilmente. La implementación en Promela de este modelo está en el Anexo C.

- *Modelo 3*, es el modelo más sencillo de los tres y fue implementado para verificar la recepción de tramas con error, lo cual no se hizo en los dos modelos anteriores, debido principalmente a la sobrecarga que esto ocasionaría en el modelo. Es muy similar en su implementación al *Modelo 2*, el cual se tomó como base para su construcción, es decir se utiliza un solo nodo y dos canales para el intercambio de mensajes, donde los dos canales están unidos. En este modelo, se envían y reciben tramas con o sin confirmación, mensajes de confirmación de recepción de trama (**ACK**) y mensajes de tipo **OK**, los demás de tipos de mensajes no fueron considerados relevantes para esta validación. Cuanto se reciben los mensajes de aceptación de trama (**ACK**) con error, se realiza una simulación del reenvió de trama, y se decrementa el contador de reenvíos, que al alcanzar el valor 0, es reiniciado (toma el valor 3) y la trama no se reenvía. Si el contador no ha alcanzado el limite (0), entoces se reenvía la trama y se decrementa el contador. Los tipos de mensajes a enviar/recibir son seleccionados de manera aleatoria usando las posibilidades que brinda el lenguaje **Promela**. En caso de que el proceso de envío/recepción se detenga porque no hay tramas a enviar/recibir, se utiliza la opción **timeout** de Promela que, reinicia el proceso seleccionando un tipo de mensaje aleatorio a enviar. La implementación en **Promela** de este modelo está en el Anexo D.

Se realizaron pruebas de simulación y validación usando las posibilidades que brinda **SPIN**, pero considerando las limitaciones tanto de la herramienta, como del equipo de cómputo utilizado para las pruebas. En este sentido, la principal limitación fue la enorme cantidad de estados del *Modelo 1*, lo cual a su vez se relaciona directamente con la cantidad de memoria necesaria para realizar el computo.

Para todos los modelos implementados en **Promela** se realizo la secuencia de actividades completa, pero en el caso de la verificación del *Modelo 1* del subnivel de enlace, solo fue posible realizar la búsqueda de estados alcanzables de manera parcial, ya que la búsqueda exhaustiva demanda más capacidad de computo (memoria y CPU) de la que se tuvo para las pruebas.

Los resultados de la simulacion y validación del modelo implementado para el sub-nivel de enlace se presentan de manera "*bruta*", en el Anexo A. Mientras que, en los Anexos B, C, D están los resultados de las simulación y validación de los tres modelos implementados para el sub-nivel de enlace.

En el Capítulo 3.8 se presentan los resultados de la validación organizados en tablas y por sub-nivel de la pila de protocolos, para facilitar su interpretación y análisis correspondiente.

Para la presentación de los resultados en tablas, se utilizan dos tipos, uno que contiene en sus columnas datos como, la cantidad de estados alcanzados, la cantidad de memoria usada, el tiempo de ejecución, el resultado (si es valido ó no) y el **hash-factor**[41]. Y el otro tipo de tabla que, muestra un resumen de la validación, donde se incluyen los críterios de correción evaluados, a saber bloqueos, ciclos sin progreso, código no alcanzable y saturación de búfer de recepción[42].

[41] Unicamente para la presentación de resultados del Modelo 1 en el Sub-nivel de Enlace.
[42] Solo se utilizó en la verificación del Modelo 2 del subnivel de enlace.

3.8. RESULTADOS

Los resultados de la verificación de la pila de protocolos se presentan de acuerdo al sub-nivel.

3.8.1 Resultados de Verificación del Protocolo del Sub-nivel Intermedio.

En este sub-nivel se implementó un modelo, pero se efectuaron pruebas parciales sobre él para facilitar el análisis de los resultados.

Caso1:

En la primera simulación se planeó enviar tramas del nodo 0 y del 2, de forma tal que se ocupan los dos canales. Estas tramas se envían en ranuras de tiempo diferentes dado que el inicio de la transmisión es asincróna. Las condiciones inciales para esta simulación es:

- Numero de canales : 2
- Numero de nodos: 3
- Numero de ranuras de tiempo: 16
- Tamaño del búfer que determina cuantas tramas puede transmitir un nodo se reparte proporcinalmente para el numero de nodos que se definan, en este caso cada nodo puede tener 5 tramas para transmitir:
- Cuanto bits de datos tiene la trama: 2

La secuencia de envío es:

Nodo-0:
- Envía trama en la ranura 1
- Envía trama en la ranura 5

Nodo-1:
- Envía trama en la ranura 2
- Envía trama en la ranura 11

Nodo-2:
- Envía trama en la ranura 7
- Envía trama en la ranura 12

La salida asociada a este caso se muestra en Anexo A.

Caso2:

En el caso anterior los canales estaban en algunos momentos de tiempo vacíos, por ejemplo, para la ranura de tiempo 3 el nodo 0 percibe que ambos canales están libres; en otro momento de tiempo estanas ambos ocupados, por ejemplo en una ranura de tiempo 3, donde el nodo 2 percibe que ambos canales está ocupados, por último existen momentos de tiempo en los que un canal está libre y otro ocupado, como ocurrió en la ranuta de tiempo 7, para el nodo 2.

El segundo caso pretende tipificar el caso en el que en el que en algún momento de tiempo ambos canales están ocupados y un nodo requiere transmitir. Para simplicar la salida se pondrán solo dos nodos a transmitir y será el nodo 0 el que deba expropiar a otro nodo del canal para poder transmitir su trama. Las condiciones inciales para esta simulación son:

- Número de canales: 2
- Número de nodos: 3
- Número de ranuras de tiempo: 13
- Tamaño del Búfer que determina cuantas tramas puede transmitir un nodo se reparte proporcinalmente para el numero de nodos que se definan, en este caso cada nodo puede tener 5 tramas para transmitir : 15
- Numero de bits de datos de la trama: 4

Nodo-0:
- Envía trama en la ranura 1
- Envía trama en la ranura 4

Nodo-1:
- Envía trama en la ranura 2

Nodo-2:
- No envía tramas.

La salida asociada a este caso se muestra en el Anexo A.

Para este último caso se muestran los resultados del proceso de análisis ejecutado por **SPIN (**ver Tabla 7**)**, esto se justifica porque en este caso se contemplan todos los posibles tipos de mensajes que el vocabulario del sub-nivel Intermedio tiene definidos.

Tabla 7. Pruebas de Validación del Caso 2

Estados	Memoria (MB)	Tiempo de CPU (m:s)	Resultado
2328	2.827	0:0.036	Válido

El estado "*Válido*" significa que no se hallaron errores en el modelo verificado y éste cumple con los objetivos propuestos, es decir, los tipos de mensajes, el estado de los canales y la transmisión y recepción del bucket es correcta.

Los criterios de corrección que se usaron en la verificación y su correspondiente resultado se muestran en la Tabla 8 :

Tabla 8. Resumen Validación Caso 2

Criterio	Resultado
Bloqueos	No
Ciclos sin progreso	No
Código no alcanzable	No

Como resultado de este proceso de validación, es posible afirmar por tanto, que el diseño del sub-nivel Intermedio es correcto, ya que no presenta errores, bloqueos, código no alvazable, ciclos infinitos o estados inválidos.

3.8.2 Resultados de Verificación del Protocolo del Sub-nivel de Enlace.

En este sub-nivel se implementaron tres modelos, asi que para facilitar la presentación de los resultados de la verificación realizada por el programa analizador, se hará por separado para cada modelo.

Modelo 1

En el Anexo B se puede apreciar una muestra de los resultados obtenidos en la simulación aleatoria de este modelo. En él, se observa la secuencia del envío/recepción de los diferentes tipos de mensajes del protocolo.

Asi mismo en el Anexo B, se muestran los resultados en "*bruto*" de la validación que, se obtuvieron durante algunas de las pruebas realizadas con **SPIN**.

Debido a las limitaciones del hardware y que este modelo era el más complejo de los tres, solo se pudo realizar la *búsqueda parcial* de errores, es decir se utilizo la técnica **Bitstate Hashing** implementada en **SPIN**. Ésto debido a la gran cantidad de estados del modelo y las limitaciones del hardware que se utilizo.

En la Tabla 9 se presentan los resultados obtenidos en las pruebas de verificación:

El **hash-factor** que se obtuvo no fue aceptable en ninguna de las pruebas, ya que para ello debería ser mayor que 100. Es decir, que con las pruebas de verificación realizadas no hay certeza de que el *Modelo 1* sea válido, ya que podría tener errores (bloqueos, código no ejecutado, ciclos infinitos).

Tabla 9. Pruebas de Validación del Modelo 1

Estados	Memoria (MB)	Tiempo de CPU (m:s)	hash factor	Resultado
5.52255e+06	458.095	1:53.581	3.06461	No Válido
5.48217e+06	456.354	1:42.345	3.08308	No Válido
5.23602e+06	425.327	4:18.666	3.17691	No Válido
5.79213e+06	419.081	9:23.0945	2.95953	No Válido

A pesar de que la técnica **Bitstate Hashing** es recomendable para este tipo de modelos, en los cuales el número de estados alcanzables es enorme, depende del hardware en el que se ejecuta la verificación y principalmente de la cantidad de memoria disponible[43].

Modelo 2

Una muestra de los resultados obtenidos en la simulación aleatoria y en la verificación de este modelo se presenta en el Anexo C. Donde, se puede apreciar la secuencia del envío/recepción de los diferentes tipos de mensajes del protocolo (tanto de manera sencilla, como de manera gráfica) generada por **SPIN** en modo simulación. Se realizaron varias corridas del modelo y los resultados de la simulación siempre fueron satisfactorios, es decir se logró el envío/recepción de mensajes y el control de flujo. Además, fue de gran ayuda para depurar el modelo y refinarlo.

En cuanto a la validación, este modelo se verificó usando **SPIN** en la modalidad *búsqueda exhaustiva* de errores. La salida que se obtuvo cuando se ejecutó el programa analizador se muestra en el Anexo C y en la Tabla 10 se puede apreciar el resultado obtenido:

Tabla 10. Prueba de Validación del Modelo 2

Estados	Memoria (MB)	Tiempo de CPU (m:s)	Resultado
468956	32.932	0:2.576	Válido

Cuando en la Tabla 10 se define como "*Válido*" el resultado, significa que no se hallaron errores en el modelo verificado y éste cumple con los objetivos propuestos que, son el envío/recepción de mensajes, y el control de flujo.

[43] HOLZMANN G.J., «Designing bug-free protocols with Spin» , Computer Communications Journal, Vol. 20, No. 2, 1997, pag. 97-105.

Los criterios de corrección que se usaron en la verificación y su correspondiente resultado se muestran en la Tabla 11 :

Tabla 11. Resumen Validación Modelo 2

Criterio	Resultado
Bloqueos	No
Ciclos sin progreso	No
Código no alcanzable	No
Saturación de búfer	Si

Modelo 3
Una muestra de los resultados obtenidos en la simulación aleatoria y en la verificación de este modelo se presenta en el Anexo D. Donde, se puede apreciar la secuencia del envío/recepción de mensajes del protocolo (secuencia de mensajes intercambiados en modo texto y gráficamente) generada por **SPIN** en modo simulación. Se realizaron varias corridas del modelo y los resultados de la simulación fueron satisfactorios, es decir se logró el envío/recepción de mensajes, y la retransmisión de las tramas con confirmación que presentaban errores. En las tramas con confirmación el error fue inducido de manera aleatoria y con el contador de reenvíos se controló la retransmisión. Las simulaciones fueron de gran ayuda en la depuración del modelo y su refinamiento.

Respecto a la validación, el *Modelo 3* se verificó usando **SPIN** en la modalidad *búsqueda exhaustiva*. La salida que se obtuvo cuando se ejecutó el programa analizador se muestra en el Anexo D y en la Tabla 12 se presenta el resultado obtenido:

Tabla 12. Prueba de Validación del Modelo 3

Estados	Memoria (MB)	Tiempo de CPU (m:s)	Resultado
95661	7.332	0:0.438	Válido

Cuando en la Tabla 12 se define como "*Válido*" el resultado, significa que no se hallaron errores en el modelo verificado y éste cumplió con los objetivos propuestos que, fueron el envío/recepción de mensajes y la retransmisión de tramas con confirmación que tenian errores (de manera inducida).

Los críterios de corrección que se usaron en la verificación y su correspondiente resultado se muestran en la Tabla 13 :

Tabla 13. Resumen Validación del Modelo 3

Criterio	Resultado
Bloqueos	No
Ciclos sin progreso	No
Código no alcanzable	No

En resumen, los resultados obtenidos mediante la verificación de los *Modelo2* y *3* usando **SPIN**, permiten afirmar que el protocolo del sub-nivel de enlace es correcto y cumple con los requerimientos de corrección verificados.

4. CONCLUSIONES

Para el diseño del protocolo *PDM-Ring* se definió un modelo por capas que consiste en dos sub-niveles, y para cada uno de ellos, se utilizó la metodología propuesta por G. Holzmann, la cual indica que se deben especificar los servicios que ofrece el protocolo, las asunciones ó suposiciones, el vocabulario, el formato de los mensajes, y las reglas de procedimiento.

La siguiente etapa, fue la simulación y validación del protocolo diseñado, para lo cual se seleccionó una herramienta de validación automatizada denominada **SPIN**, la cual pemite, incluso la simulación de modelos con una cantidad enorme de estados (más de 10 ^ 8). **SPIN** facilitó la validación en gran medida, gracias a su robustez y fundamento algorítmico para la detección de errores y definición de críterios de corrección.

En cuanto, a la ejecución de los modelos usando **SPIN** en modo simulación, ésta no aportó bases para la validación formal, pero en cambio si facilitó la depuración de los modelos creados y la verificación de la funcionalidad.

La realización de la simulación y validación del diseño de la pila de protocolos permitió concluir que, éste es correcto en sus dos sub-niveles. Aunque, es importante mencionar que en el sub-nivel de enlace, el modelo planteado inicialmente no pudo ser verificado en su totalidad, debido al gran número de estados del modelo y las limitaciones del hardware utilizado para la ejecución de las pruebas. Por tal razón, este modelo fue divido en dos módulos, que se implementaron mediante dos modelos, cada uno de los cuales permitió validar los críterios de corrección (bloqueos, ciclos sin progreso, código no alcanzable y saturación de búfer) planteados para el sub-nivel de enlace de datos.

5. TRABAJOS FUTUROS

- Implementar el protocolo en un lenguaje de alto nivel para así verificar su funcionalidad.

- El presente modelo fue diseñado sin considerar las redes inalámbricas, sería interesante tomar como base el diseño propuesto y redefinirlo para este tipo de redes, en el que la conexión, la seguridad y el alcance son factores determinantes.

- Al interior del protocolo en el sub-nivel Intermedio se evaluaron diferentes propuestas que no se incluyeron en el diseño y que pueden arrojar resultados interesantes de evaluar:

    - Definir diferentes niveles de estrategia para el momento en el que un nodo se apodera de un canal. En el modelo propuesto, cuando un nodo requiere transmitir una trama, primero debe verificar si hay canales disponibles, y aunque posea varias tramas por transmitir, solo se apodera de un canal por ranura de tiempo. Vale la pena evaluar el impacto si el nodo se apodera de varios canales en una misma ranura de tiempo, desde el punto de vista de la velocidad de transmisión y de las colisiones.

    - Definir diferentes niveles de estrategia para el momento en el que se decida expropiar un nodo. En el modelo propuesto, un nodo expropia solo a un nodo, sin tener en cuenta el número de tramas pendientes por transmitir, nuevamente el tema de interés a analizar sería la velocidad de transmisión de los nodos con mayor prioridad y que problemas pueden surgir para aquellos nodos que transmiten tramas sin confirmación, ya que estos serían los directamente afectados.

- Al interior del protocolo en el sub-nivel de Enlace se debe ejecutar la validación del modelo inicialmente propueto de manera exhaustiva en una máquina con alta capacidad de cómputo o cuya arquitectura permita ejecutar cómputo paralelo.

BIBLIOGRAFÍA

BAGHDADY, Elie J. «New development in FM Reception and their application to the realization of a system of “Power division multiplex” ». Communications, IEEE Transactions on [legacy, pre - 1988]. Volume: 7 , Issue: 3 . Sep 1959. pag. 147 - 161.

BANERJEA, Anindo. FERRARI, Domenico. MAAH, Bruce A., MORAN, Mark. VERMA, Dinesh C. y ZHANG, Hui. « The Tenet Real - Time Protocol Suite: Design, Implementation, and Experiences». Networking, IEEE/ACM Transactions on. Volume: 4 , Issue: 1 . Feb. 1996. pag. 1 – 10.

BLACK, Uyless. Redes de Computadoras Protocolos, normas e interfaces. Segunda edición. Ra-ma, 1997.

BRÆK, Rolv. «SDL basics ». Computer Networks and ISDN Systems. Volume 28 , Issue 12 (June 1996) Special issue on SDL and MSC. 1996. Elsevier Science Publishers B. V. Amsterdam (The Netherlands). Pag. 1585 – 1602.

CARLSON, Bruce. Communications Systems. McGraw Hill. Cuarta edición. 2001.

CONTI, M. y DONATIELLO L. « Design an Analysis of RT – Ring: a protocol for supporting real-time communications ». Industrial Electronics, IEEE Transactions on . Volume: 49 , Issue: 6 . Dec. 2002. pp 1214 - 1226.

COUCH, Leon W. Digital and analog communications systems. Prentice Hall. Sexta edición. 2001.

ERIKSSON, Christer. THANE, Henrik. GUSTAFSSON, Mikael. «A Communication Protocol for Hard and Soft Real-Systems», IEEE Real-Time Systems, 1996. Proceedings of the Eighth Euromicro Workshop on . 12-14 June 1996. pag. 187 - 192.

HOLZMANN, Gerard J., Design and Validation of Computer Protocols, Prentice Hall, 1991.

HOLZMANN, Gerard J., «The Model Checker Spin», IEEE Transactions on Software Engineering, Vol. 23, No. 5, Mayo 1997, pag. 279-295.

HOLZMANN Gerard J., «Using Spin», Plan 9 Programmer’s Manual Documents, Vita Nuova Holdings Ltd, York (England), 2nd edition, 2000. pag. 353-382.

HOLZMANN Gerard J., «An Analysis of Bitstate Hashing», Formal Methods in System Design, Vol. 13, No. 3, Kluwer, November 1998, pag. 287-305.

HOLZMANN Gerard J., «Designing bug-free protocols with Spin», Computer Communications Journal, Vol. 20, No. 2, 1997, pag. 97-105.

HOLZMANN Gerard J., «Protocol Design: Redefining the State of the Art», IEEE Software, Vol. 09, No. 1, January 1992, pag. 17-22.

LATHI, B.P. Modern digital and analog communications. Oxford Press. Tercera edición. 1998.

LIMA, M.R., LEITE J.C.B., LOQUES O. G. « A Real-Time Communication Protocol », Real Time, 1990. Proceedings. Euromicro '90 Workshop on . 6-8 June 1990. pp 129 - 134.

NICHOLLS, K. JACOBSON V. ZHANG L. « A Two-bit Differentiated Services Architecture for the Internet ». IETF RFC 2638, July 1999.
Disponible en la URL: http://irl.cs.ucla.edu/papers/twobit.pdf.

PARK, Jung Woo. PARK Hyeok Gi. KWON, WOOK Hyun. «A Real - Time Communication Protocol with Contention-resolving Algorithm for Programmable Controllers». Industrial Electronics, Control and Instrumentation, 1994. IECON '94., 20th International Conference on , Volume: 2 , 5-9 Sept. 1994. pp 1159 – 1164.

PENG, H. TAHAR, S. KHENDEK, F., «SPIN vs. VIS: a case study on the formal verification of the ATMR protocol», Formal Engineering Methods, 2000. ICFEM 2000. Third IEEE International Conference on. York, UK , 2000. pag. 79-87,

SHIMAMOTO, Shigeru. ONOZATO, Yoshikuni. TESHIGAWARA, Yoshimi. « Perfomance Evaluation of Power Level Division Multiple Access (PDMA) Scheme », IEEE , International Conference of Communications. 1992. pp. 1333 - 1337.

SKLAR, Bernard. Digital communications: Fundamentals and Application. Prentice Hall. Segunda edición. 2001.

STALLINGS, William. Comunicaciones y Redes de Computadores, Sexta Edición, Editorial Prentice Hall, Madrid (España), 2000. Pag. 182-210.

STREMLER, Ferrel G. Introduction to communication systems. Addison-Wesley. Tercera edición. 1990.

TANENBAUM, Andrew S. Redes de Computadores. Tercera Edición. Prentice Hall, 1997.

TINDELL, K. W. HANSSON, H. y WELLINGS A. J. « Analisys Real-Time Communications: Controller Area Network (CAN) ». IEEE. Real-Time Systems Symposium, 1994., Proceedings. , 7-9 Dec. 1994. pp 259 - 263.

DOCUMENTOS

ISO/DIS 11898. «Road Vehicles – Interchange of Digital Information – Controller Area Network (CAN) for High Speed Communications». Febrero 1992.

ICONTEC. «Compendio – Tesis y Otros Trabajos de Grado». Imprelibros S.A, Bogotá D.C., 2004.

REFERENCIAS EN INTERNET

Berge J., «Ethernet in Process Control», Technical article, The Industrial Ethernet Book, Issue 3, June 2000.
http://ethernet.industrial-networking.com/ieb/articles.asp
(última consulta marzo 3 de 2006).

Berge J., «FieldBus, Ethernet and the Reality of Convergence», Technical article, The Industrial Ethernet Book, Issue 23, November 2004.
http://ethernet.industrial-networking.com/ieb/articles.asp
(última consulta marzo 3 de 2006).

HART Communication Foundation. HART Protocol Overview
http://www.hartcomm.org/
(última consulta el 3 de diciembre de 2005)

LONWORKS, White Paper. «Determinism in Industrial Computer Control Network Applications». Echelon, Enero1995.
http://www.echelon.com/support/documentation/bulletin/005-0060-01A.pdf
(consultada el 11 de noviembre de 2003)

Martin Timmerman PhD. Dedicated Systems - Encyclopaedia
http://www.realtime-info.be/encyc/techno/terms/defini/def.htm
(última consulta el 2 de diciembre de 2005)

SPIN. ON -THE-FLY, LTL MODEL CHECKING with SPIN
http://spinroot.com/spin/whatispin.html
(última consulta el 3 de diciembre de 2005)

Theo Ruys & G. Holzmann . SPIN Advanced tutorial
http://spinroot.com/spin/Doc/Spin_tutorial_2004.pdf
(última consulta el 2 de diciembre de 2005)

Theo Ruys . SPIN Beginners' Tutorial.
http://spinroot.com/spin/Doc/SpinTutorial.pdf
(última consulta el 2 de diciembre de 2005)

Matt Dwyer, John Hatcliff.Kansas State University.
CIS 842: Specification and Verification of Reactive Systems
Lecture SPIN-INTRO: Introduction To SPIN
URL: http://spinroot.com/spin/Doc/SpinIntro.pdf
(última consulta el 2 de diciembre de 2005)

Gerard. J. Holzmann . Bibliography
URL: http://spinroot.com/gerard/pubs.html
(última consulta el 2 de diciembre de 2005)

ANEXO A.

Código Fuente

La implementación del Modelo del sub-nivel de enlace de datos en Promela es la siguiente:

```
#define CANALES 2 //Numero de canales
#define NODOS 3 //Numero de nodos
#define RANURAS 14 //Numero de ranuras de tiempo
#define BUFFER 15/*Tamaño del Búfer que determina cuantas tramas
puede transmitir un nodo se reparte proporcinalmente para el
numero de nodos que se definan, en este caso cada nodo puede tener
5 tramas para transmitir*/
#define TRAMA 4 /* Longitud de la trama. El primer dato indica el
tipo de mensaje LIBRE, OCUPADO, EXPROPIADO. Los 3 ultimos bits
controlan: ranura (1 - RANURAS), canal (1 - CANALES) y finalizo
transmisión (1 - SI, 0 - NO)*/
#define DATOS 2 /* Cuanto bits de datos tiene la trama, esto solo
se usa para mantener el tipo de mensaje*/

/*Tipo de dato que permite definir todos los buffer requeridos por
cada nodo*/
typedef numero_nodos {
  byte nodos [NODOS]
};

/*Tipo de dato que permite definir cuantos datos se transmiten por
trama*/
typedef trama {
  byte datos [TRAMA]
};

//Estados del canal
#define LIBRE 0
#define OCUPADO 1
#define EXPROPIADO 2

/*Matriz que almacena los estados de cada canal para cada nodo*/
numero_nodos bucket[CANALES];

/*Matriz que identifica si un nodo posee un canal y cuantos bits
se han transmitido*/
numero_nodos canales_posee[CANALES];

/*Matriz que identifica si un nodo ha expropiado un canal*/
numero_nodos canales_expropio[CANALES];
```

```
/*Matriz que identifica por nodo cuantos bits de una trama ha
recibido*/
numero_nodos bit_trama_recibe[CANALES];

/*Numero de tramas recibidas por nodo*/
byte tramas_recibidas[NODOS];

/*Matriz que almacena los datos de cada uno de los nodos de la
simulacion*/
trama nodoX[BUFFER];

/*Controla el subindice de canales vacíos*/
byte vacio=0;

/*Controla los subíndices por nodo de las tramas candidatas a ser
nuevas*/
byte indice_buffer[NODOS];

/*Variable que controla la ranura de tiempo que se esta
procesando*/
byte numero_ranuras=1;

/*Variable que determina la trama que se esta procesando*/
byte tram_proc;

/*Arreglo que contiene los subíndices de los canales vacíos por
nodo*/
byte canales_vacios[CANALES];

/*Variable que permite identificar si se ha dato una
expropiación*/
byte num_expropiados =0;

/*Proceso de comunicación*/
proctype comunicacion (){
/*Incializa estado de los canales como Libres para cada bucket*/
  byte i=0, j=0;
  do
  ::(i < NODOS) ->
    do
    ::(j < CANALES) ->
      bucket[j].nodos[i]=LIBRE;
      j++
    :: else ->
      break
    od;
    j=0;
    i++
```

```
  :: else ->
    break
  od;

  //Inicializa los buffer de transmisión de cada nodo
  nodoX[0].datos[0]=1;//Tipo de mensaje OCUPADO, LIBRE, EXPROPIADO
  nodoX[0].datos[1]=1; //ranura
  nodoX[0].datos[2]=0; //canal asignado
  nodoX[0].datos[3]=0; //fin 0:NO - 1:SI
  nodoX[1].datos[0]=1;
  nodoX[1].datos[1]=4;
  nodoX[1].datos[2]=0;
  nodoX[1].datos[3]=0;

  nodoX[5].datos[0]=1;
  nodoX[5].datos[1]=2;
  nodoX[5].datos[2]=0;
  nodoX[5].datos[3]=0;
  nodoX[6].datos[0]=1;
  nodoX[6].datos[1]=15;
  nodoX[6].datos[2]=0;
  nodoX[6].datos[3]=0;

  nodoX[10].datos[0]=1;
  nodoX[10].datos[1]=25;
  nodoX[10].datos[2]=0;
  nodoX[10].datos[3]=0;
  nodoX[11].datos[0]=0;
  nodoX[11].datos[1]=30;
  nodoX[11].datos[2]=0;
  nodoX[11].datos[3]=0;

/*Inicializa los subíndices donde se encuentran las tramas para
cada nodo*/
  indice_buffer[0]=0;
  indice_buffer[1]=5;
  indice_buffer[2]=10;

  //Procesamiento de las ranuras de tiempo
  do
  ::(numero_ranuras <= RANURAS) ->/*Inicia proceso de Lectura del
  Bucket para todos los nodos en cada ranura de tiempo*/
    printf("\n**** SE ESTA PROCESANDO LA RANURA %d
    ****\n",numero_ranuras);
    //Estado del anillo para esta ranura de tiempo
    printf("Estado del anillo para esta ranura de tiempo\n");
    i=0;
    j=0;
    do
```

```
::(i < NODOS) ->
  j=0;
  printf("NODO %d. ",i);
  do
  ::(j < CANALES) ->
    printf("Canal %d: %d ",j,bucket[j].nodos[i]);
    j++
  :: else ->
    break
  od;
  printf("\n");
  i++
:: else ->
  break
od;
i=0;
j=0;
do
::(i < NODOS) -> //Procesamiento de cada nodo
  printf("**** PROCESO DE RECEPCION DEL NODO %d ****\n",i);
  canales_vacios[0]=0;
  canales_vacios[1]=0;
  vacio=0;
  do
  ::(j < CANALES) -> /*Procesamiento de cada canal del bucket
  para el nodo i*/
    printf("**** CANAL %d ****\n",j);
    if
    ::(bucket[j].nodos[i] == OCUPADO) ->
      bit_trama_recibe[j].nodos[i]++;
      if
      ::(canales_posee[j].nodos[i] == 0)-> /*El nodo no posee
      el canal*/
        printf("Recibi un bit para el nodo %d por el canal %d
        y este nodo NO posee el canal \n",i,j);
        if
        ::  (bit_trama_recibe[j].nodos[i] == DATOS) ->
            printf("Se completo la recepción de la trama por
            el canal %d para el nodo %d\n",j,i);
            bit_trama_recibe[j].nodos[i] = 0;
            tramas_recibidas[i]++
        ::else ->
          printf("No se ha completado la recepcion de la
          trama por el canal %d para el nodo %d\n",j,i);
        fi
    ::else -> //Si posee el canal
        printf("Recibi un bit para el nodo %d por el canal %d
        y este nodo SI posee el canal, por esta razon lo
        limpia \n",i,j);
```

```
  bucket[j].nodos[i]=LIBRE;/*Se limpia el bucket de sus
  propios datos*/
  if
  ::(canales_expropio[j].nodos[i] == 1) ->
      printf("Se identifica que el canal %d fue
      expropiado y se deja vacio para que sea usado por
      el nodo %d\n",j,i);
      canales_vacios[vacio]=j+1;
      vacio++;
      canales_expropio[j].nodos[i] = 0
    ::else ->
      goto continua
    fi;
    continua:
    if
    ::(bit_trama_recibe[j].nodos[i] == DATOS) ->
      printf("Se completo recepcion de una trama por el
      canal %d para el nodo %d\n",j,i);
      bit_trama_recibe[j].nodos[i] = 0;
      canales_posee[j].nodos[i]=0; /*Se libera el canal
      como información para el nodo corriente, por que
      en realidad ya otro nodo pudo ocuparlo*/
      canales_vacios[vacio]=j+1;
      vacio++;
      tramas_recibidas[i]++
    ::else ->
      printf("No se ha completado la recepcion de la
      trama por el canal %d para el nodo %d\n",j,i);
    fi
  fi
::(bucket[j].nodos[i] == LIBRE) ->
  if
  ::(canales_posee[j].nodos[i] == 0) ->
    printf("El canal %d esta libre para el nodo %d y
    este nodo NO lo posee\n",j,i);
    canales_vacios[vacio]=j+1;
    vacio++
  ::else ->
    printf("El canal %d esta libre para el nodo %d y
    este nodo SI lo posee\n",j,i);
  fi;
::(bucket[j].nodos[i] == EXPROPIADO) ->
  bit_trama_recibe[j].nodos[i] = 0;
  if
   ::(canales_posee[j].nodos[i] != 0 && num_expropiados
   == 0) ->
    canales_posee[j].nodos[i] = 0;
    num_expropiados++;
```

```
                printf("El nodo %d posee el canal %d pero ha sido
                expropiado por otro nodo, se elimina toda
                informacion recibida y libera el canal \n",i,j);
              ::else ->
                if
                ::(canales_posee[j].nodos[i] != 0) ->
                  printf("El nodo %d posee el canal %d se elimina
                  toda informacion que haya recibido y queda
                  apoderado del canal\n",i,j);
                  num_expropiados=0;
                ::else ->
                  printf("El nodo %d NO posee el canal %d y este
                  fue expropiado por otro nodo. Elimina los datos
                  recibidos.\n",i,j);
                fi
              fi
            ::else ->
              printf("\nEstado del Canal %d invalido\n",j)
            fi;
            j++
          ::else ->
            break
          od; /* Fin del ciclo que procesa cada canal del bucket*/
          j=0; //Inicializa canal a procesar

/*Inicia proceso de transmisión*/
printf("**** PROCESO DE TRANSMISON DEL NODO %d ****\n",i);
/*Indica en que subíndice están las tramas a evaluar por cada
nodo*/
          if
          ::(i==0) ->
            tram_proc=0
          ::(i==1) ->
            tram_proc=5
          ::else ->
            tram_proc=10
          fi;
          do
          :: (tram_proc < indice_buffer[i])->
            if
            ::(nodoX[tram_proc].datos[TRAMA-1]!= 1 && canales_posee
            [nodoX[tram_proc].datos[TRAMA-2]-1].nodos[i] != 0) ->
            /*Hay tramas en proceso de transmisión*/
              printf("El nodo %d tiene la trama %d en proceso de
              transmision por el canal %d \n ",i ,tram_proc+1 ,
              nodoX [tram_proc].datos[TRAMA-2]-1);
              bucket[nodoX[tram_proc].datos[TRAMA-2]-1].nodos[i] =
              nodoX [tram_proc].datos [0]; /*ocupa el canal y envia
              el tipo de mensaje*/
```

```
            canales_posee[nodoX[tram_proc].datos[TRAMA-2]-1].
            nodos[i]++; /*Otro bit que se transmite*/
            if
            ::(canales_posee [nodoX [tram_proc]. datos[TRAMA-2]-
            1]. nodos[i] == DATOS) ->
              printf("El nodo %d termino de transmitir la trama
              %d por el canal\n",i,tram_proc+1,nodoX [tram_proc].
              datos [TRAMA-2]-1);
              nodoX[tram_proc].datos[TRAMA-1]=1;/*finalizo la
              transmisión de la trama*/
            ::else ->
              printf("El nodo %d NO ha terminado de transmitir la
              trama %d\n",i,tram_proc+1);
            fi;
          ::else ->
            goto continua1
          fi;
          continua1:
          tram_proc++
        ::else ->
          break
        od;
/*2. Verifica si hay nuevas tramas por transmitir para el nodo i*/
        if
        ::(nodoX[indice_buffer[i]].datos[TRAMA-3]!=0 && nodoX [
        indice_buffer [i]].datos [TRAMA-2] == 0) ->
        /*Se corrobora que la trama sea nueva, el indice buffer
        tiene el subíndice donde probablemente hay tramas nuevas,
        las anteriores ya deben estar en proceso de transmision o
        terminado*/
          if
          ::(nodoX[indice_buffer[i]].datos[TRAMA-3]<=numero_ranuras)
          -> /*Indica que la trama se debe enviar en esta ranura*/
            printf("El nodo %d tiene una trama nueva la numero %d y
            debe enviarse en esta ranura de tiempo  : %d \n ", i,
            indice_buffer [i]+1,numero_ranuras);
            //Se toma un canal vacio si lo hay
            if
            ::(canales_vacios[0] != 0) -> //Hay canales libres
              bucket[canales_vacios[0]-1]. nodos [i] = nodoX [
              indice_buffer [i]].datos[0];
              nodoX[indice_buffer[i]].datos[TRAMA-2]=canales_vacios
              [0]; /*asigna el canal a la trama*/
              canales_posee[canales_vacios[0]-1].nodos[i]++;
              /*Asigna el canal al nodo e incrementa el numero de
              bits transmitidos*/
              printf("El nodo %d envía el dato %d por el canal %d
              \n",i,nodoX[indice_buffer[i]].datos[0],canales_vacios
              [0]-1);
```

```
  if
  ::(canales_posee [nodoX [ indice_ buffer [i]]. datos
  [TRAMA-2]-1]. nodos[i] == DATOS) ->
    printf("El nodo %d termino de transmitir la trama
    %d por el canal \n",i,indice_buffer[i]+1, nodoX [
    indice_buffer [i] ].datos[TRAMA-2]-1);
    nodoX[indice_buffer[i]].datos[TRAMA-1]=1 /*finalizo
    la transmision de la trama*/
  ::else ->
    printf("El nodo %d NO ha terminado de transmitir la
    trama %d\n",i,indice_buffer[0]+1);
  fi;
    indice_buffer[i]++ /*mueve el indicador de tramas
    nueva por que esta trama ya ha sido ubicada*/
::else ->
    printf("No hay canales disponibles el nodo
    explorara la estrategia de expropiacion\n");
    if
    ::(i==0) ->
      byte ind=0;
      do
      :: (ind < TRAMA) ->
        nodoX [indice_buffer [i]+1].datos [ind] = nodoX
        [indice_buffer [i]].datos[ind];
        ind++
      ::else ->
        printf("Se inserta un mensaje de expropiacion
        en el buffer del nodo %d\n",i);
        break
      od;
      nodoX [indice_buffer [i]]. datos [0]=EXPROPIADO;
      /*se envía mensaje de expropiación*/
      printf("Datos de la trama de expropiacion :%d,%d
      ,%d,%d\n",nodoX[indice_buffer[i]].datos[0],nodoX
      [indice_buffer[i]].datos[1],nodoX[indice_buffer[i
      ]].datos[2],nodoX[indice_buffer[i]].datos[3]);
      printf("Datos de la trama que se desplazo:%d,%d,
      %d,%d\n",nodoX[indice_buffer[i]+1].datos[0],nodoX
      [indice_buffer[i]+1].datos[1],nodoX[indice_buffer
      [i]+1].datos[2],nodoX[indice_buffer[i]+1].datos[3
      ]);
      bucket[CANALES-1].nodos[i] = nodoX [indice_buffer
      [i]].datos[0];
      nodoX[ indice_buffer[i]].datos[TRAMA-2] = 2;
      /*asigna el canal a la trama*/
      canales_posee[CANALES-1].nodos[i] = DATOS;/*le
      indica al nodo que posee este canal y hace que
      esta trama se transmita en una ranura de tiempo*/
```

```
            bit_trama_recibe[CANALES-1].nodos[i]=DATOS-1;
            /*Forza a que en la recepcion de este nodo que
            expropia se libere el canal para que el lo
            tome.*/
            printf("El nodo %d envía el dato %d por el canal
            1\n",i,nodoX[indice_buffer[i]].datos[0]);
            canales_expropio[CANALES-1].nodos[i]=1;/*Se marca
            el canal como expropiado por este nodo*/
            if
            ::(canales_posee[CANALES-1].nodos[i] == DATOS) ->
              printf("El nodo %d termino de transmitir la
              trama que envio expropiando el canal
              %d\n",i,indice_buffer[i]+1);
              nodoX[indice_buffer[i]].datos[TRAMA-1]=1
              //finalizo la transmision de la trama
            ::else ->
              printf("El nodo %d NO ha terminado de
              transmitir la trama %d por la que expropio el
              canal 1\n",i,indice_buffer[i]+1);
            fi;
            indice_buffer[i]++ /* mueve el indicador de
            tramas nueva por que esta trama ya ha sido
            ubicada*/
          fi
        fi
      ::else -> // la trama todavÃ¬a no se debe enviar
        printf("TodavÃ¬a no se debe enviar la trama nueva\n")
      fi;
    ::else ->
      printf("El nodo %d no tiene mÃ¡s tramas nuevas, el
      indice quedo en %d\n",i,indice_buffer[0])
    fi;
    i++ //Siguiente nodo.
  :: else ->
    break
  od;// fin del ciclo que procesa cada nodo

  //Se mueven los buckets por el anillo
  j=0;
  byte temp[CANALES];
  printf("**** ROTA EL ANILLO ****\n");
  do
  ::(j < CANALES) ->
    temp[j]=bucket[j].nodos[NODOS-1];/*copia el bucket del
    ultimo nodo*/
    j++
  ::else ->
```

```
          printf("Se ha copiado el bucket del nodo %d : %d - %d\n",
          NODOS-1,bucket[0].nodos[NODOS-1],bucket[1].nodos[NODOS-1]
          );
          break
        od;
        i=NODOS-1;
        do
        ::(i > 0) ->
          j=0;
          do
          ::(j < CANALES) ->
            bucket[j].nodos[i]= bucket[j].nodos[i-1];
            j++
          :: else ->
            printf("Se ha pasado el contenido del nodo %d al nodo %d
            :%d-%d\n",i-1,i,bucket[0].nodos[i],bucket[1].nodos[i]);
            break
          od;
          i--
        :: else ->
          break
        od;
        i=0;
        j=0;
        do
        ::(j < CANALES) ->
          bucket[j].nodos[i]=temp[j];
          j++
        :: else ->
          printf("Se ha copiado el bucket del nodo %d al primer
          nodo:%d-%d\n",NODOS-1,bucket[0].nodos[0],bucket[1].nodos
          [0] );
          break
        od;
        numero_ranuras++
      ::else ->
        printf("**** TERMINA PROCESAMIENTO DE LAS %d RANURAS
        ****\n",RANURAS);
          break
      od
}

/* Se inicia la ejecución del procedimiento de comunicación*/
init {
  run comunicacion();
}
```

Simulación

El código se grabó en un archivo denominado "*Nivel_Intermedio*", y la ejecución de la simulación del modelo usando **SPIN** desde la línea de comandos bajo la plataforma Linux se realizo usando el comando

```
#spin –c Nivel_Intermedio.prom
```

Para este sub-nivel se ejecutarón dos casos de prueba:

Caso 1

En la primera simulación se planeó enviar tramas del nodo 0 y del 2, de forma tal que se ocupan los dos canales. Estas tramas se envían en ranuras de tiempo diferentes dado que el inicio de la transmisión es asincróno. Las condiciones inciales para esta simulación son:

```
#define CANALES 2 //Numero de canales
#define NODOS 3 //Numero de nodos
#define RANURAS 16 //Numero de ranuras de tiempo
#define BUFFER 15/*Tamaño del Búfer que determina cuantas tramas
puede transmitir un nodo se reparte proporcinalmente para el
numero de nodos que se definan, en este caso cada nodo puede tener
5 tramas para transmitir*/
#define DATOS 2 /* Cuanto bits de datos tiene la trama, esto solo
se usa para mantener el tipo de mensaje*/
```

Nodo-0:
- Envía trama en la ranura 1
- Envía trama en la ranura 5

Nodo-1:
- Envía trama en la ranura 2
- Envía trama en la ranura 11

Nodo-2:
- Envía trama en la ranura 7
- Envía trama en la ranura 12

La salida que arroja este caso es:

```
**** SE ESTA PROCESANDO LA RANURA 1 ****
          Estado del anillo para esta ranura de tiempo
          NODO 0.           Canal 0: 0           Canal 1: 0
          NODO 1.           Canal 0: 0           Canal 1: 0
```

```
          NODO 2.           Canal 0: 0           Canal 1: 0
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 0 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 0 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          El nodo 0 tiene una trama nueva la numero 1 y debe enviarse en
esta ranura de tiempo  : 1
          El nodo 0 envia el dato 1 por el canal 0
          El nodo 0 NO ha terminado de transmitir la trama 1
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 1 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 1 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
          Todavia no se debe enviar la trama nueva
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 2 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 2 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          Todavia no se debe enviar la trama nueva
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 0 - 0
          Se ha pasado el contenido del nodo 1 al nodo 2 :  0 - 0
          Se ha pasado el contenido del nodo 0 al nodo 1 :  1 - 0
          Se ha copiado el bucket del nodo 2 al primer nodo:  0 - 0

**** SE ESTA PROCESANDO LA RANURA 2 ****
          Estado del anillo para esta ranura de tiempo
          NODO 0.           Canal 0: 0           Canal 1: 0
          NODO 1.           Canal 0: 1           Canal 1: 0
          NODO 2.           Canal 0: 0           Canal 1: 0
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 0 y este nodo SI lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 0 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          El nodo 0 tiene la trama 1 en proceso de transmision por el
canal 0
          El nodo 0 termino de transmitir la trama 1 por el canal 0
          Todavia no se debe enviar la trama nueva
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 1 por el canal 0 y este nodo NO
posee el canal
          No se ha completado la recepcion de la trama por el canal 0
para el nodo 1
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 1 y este nodo NO lo posee
```

```
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
          El nodo 1 tiene una trama nueva la numero 6 y debe enviarse en
esta ranura de tiempo  : 2
          El nodo 1 enviÃ³ el dato 1 por el canal 1
          El nodo 1 NO ha terminado de transmitir la trama 2
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 2 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 2 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          Todavia no se debe enviar la trama nueva
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 0 - 0
          Se ha pasado el contenido del nodo 1 al nodo 2 :  1 - 1
          Se ha pasado el contenido del nodo 0 al nodo 1 :  1 - 0
          Se ha copiado el bucket del nodo 2 al primer nodo:  0 - 0

**** SE ESTA PROCESANDO LA RANURA 3 ****
          Estado del anillo para esta ranura de tiempo
          NODO 0.           Canal 0: 0           Canal 1: 0
          NODO 1.           Canal 0: 1           Canal 1: 0
          NODO 2.           Canal 0: 1           Canal 1: 1
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 0 y este nodo SI lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 0 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          Todavia no se debe enviar la trama nueva
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 1 por el canal 0 y este nodo NO
posee el canal
          Se completo la recepcion de la trama por el canal 0 para el
nodo 1
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 1 y este nodo SI lo posee
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
          El nodo 1 tiene la trama 6 en proceso de transmision por el
canal 1
          El nodo 1 termino de transmitir la trama 6 por el canal 1
          Todavia no se debe enviar la trama nueva
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 2 por el canal 0 y este nodo NO
posee el canal
          No se ha completado la recepcion de la trama por el canal 0
para el nodo 2
          **** CANAL 1 ****
          Recibi un bit para el nodo 2 por el canal 1 y este nodo NO
posee el canal
          No se ha completado la recepcion de la trama por el canal 1
para el nodo 2
```

```
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          Todavia no se debe enviar la trama nueva
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 1 - 1
          Se ha pasado el contenido del nodo 1 al nodo 2 :  1 - 1
          Se ha pasado el contenido del nodo 0 al nodo 1 :  0 - 0
          Se ha copiado el bucket del nodo 2 al primer nodo:  1 - 1

**** SE ESTA PROCESANDO LA RANURA 4 ****
          Estado del anillo para esta ranura de tiempo
          NODO 0.           Canal 0: 1           Canal 1: 1
          NODO 1.           Canal 0: 0           Canal 1: 0
          NODO 2.           Canal 0: 1           Canal 1: 1
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 0 por el canal 0 y este nodo SI
posee el canal, por esta razon lo limpia
          No se ha completado la recepcion de la trama por el canal 0
para el nodo 0
          **** CANAL 1 ****
          Recibi un bit para el nodo 0 por el canal 1 y este nodo NO
posee el canal
          No se ha completado la recepcion de la trama por el canal 1
para el nodo 0
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          Todavia no se debe enviar la trama nueva
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 1 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 1 y este nodo SI lo posee
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
          Todavia no se debe enviar la trama nueva
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 2 por el canal 0 y este nodo NO
posee el canal
          Se completo la recepcion de la trama por el canal 0 para el
nodo 2
          **** CANAL 1 ****
          Recibi un bit para el nodo 2 por el canal 1 y este nodo NO
posee el canal
          Se completo la recepcion de la trama por el canal 1 para el
nodo 2
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          Todavia no se debe enviar la trama nueva
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 1 - 1
          Se ha pasado el contenido del nodo 1 al nodo 2 :  0 - 0
          Se ha pasado el contenido del nodo 0 al nodo 1 :  0 - 1
          Se ha copiado el bucket del nodo 2 al primer nodo:  1 - 1
```

****** SE ESTA PROCESANDO LA RANURA 5 ******

Estado del anillo para esta ranura de tiempo
NODO 0. Canal 0: 1 Canal 1: 1
NODO 1. Canal 0: 0 Canal 1: 1
NODO 2. Canal 0: 0 Canal 1: 0
****** PROCESO DE RECEPCION DEL NODO 0 ******
****** CANAL 0 ******
Recibi un bit para el nodo 0 por el canal 0 y este nodo SI posee el canal, por esta razon lo limpia
Se completo recepcion de una trama por el canal 0 para el nodo 0
****** CANAL 1 ******
Recibi un bit para el nodo 0 por el canal 1 y este nodo NO posee el canal
Se completo la recepcion de la trama por el canal 1 para el nodo 0
****** PROCESO DE TRANSMISON DEL NODO 0 ******
El nodo 0 tiene una trama nueva la numero 2 y debe enviarse en esta ranura de tiempo : 5
El nodo 0 enviÃ³ el dato 1 por el canal 0
El nodo 0 NO ha terminado de transmitir la trama 2
****** PROCESO DE RECEPCION DEL NODO 1 ******
****** CANAL 0 ******
El canal 0 esta libre para el nodo 1 y este nodo NO lo posee
****** CANAL 1 ******
Recibi un bit para el nodo 1 por el canal 1 y este nodo SI posee el canal, por esta razon lo limpia
No se ha completado la recepcion de la trama por el canal 1 para el nodo 1
****** PROCESO DE TRANSMISON DEL NODO 1 ******
TodavÃa no se debe enviar la trama nueva
****** PROCESO DE RECEPCION DEL NODO 2 ******
****** CANAL 0 ******
El canal 0 esta libre para el nodo 2 y este nodo NO lo posee
****** CANAL 1 ******
El canal 1 esta libre para el nodo 2 y este nodo NO lo posee
****** PROCESO DE TRANSMISON DEL NODO 2 ******
TodavÃa no se debe enviar la trama nueva
****** ROTA EL ANILLO ******
Se ha copiado el bucket del nodo 2 : 0 - 0
Se ha pasado el contenido del nodo 1 al nodo 2 : 0 - 0
Se ha pasado el contenido del nodo 0 al nodo 1 : 1 - 1
Se ha copiado el bucket del nodo 2 al primer nodo: 0 - 0

**** SE ESTA PROCESANDO LA RANURA 6 ****
Estado del anillo para esta ranura de tiempo
NODO 0. Canal 0: 0 Canal 1: 0
NODO 1. Canal 0: 1 Canal 1: 1
NODO 2. Canal 0: 0 Canal 1: 0
**** PROCESO DE RECEPCION DEL NODO 0 ****
**** CANAL 0 ****
El canal 0 esta libre para el nodo 0 y este nodo SI lo posee
**** CANAL 1 ****
El canal 1 esta libre para el nodo 0 y este nodo NO lo posee

```
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          El nodo 0 tiene la trama 2 en proceso de transmision por el
canal 0
          El nodo 0 termino de transmitir la trama 2 por el canal 0
          El nodo 0 no tiene mas tramas nuevas, el indice quedo en 2
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 1 por el canal 0 y este nodo NO
posee el canal
          No se ha completado la recepcion de la trama por el canal 0
para el nodo 1
          **** CANAL 1 ****
          Recibi un bit para el nodo 1 por el canal 1 y este nodo SI
posee el canal, por esta razon lo limpia
          Se completo recepcion de una trama por el canal 1 para el nodo
1
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
          Todavia no se debe enviar la trama nueva
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 2 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 2 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          Todavia no se debe enviar la trama nueva
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 0 - 0
          Se ha pasado el contenido del nodo 1 al nodo 2 :  1 - 0
          Se ha pasado el contenido del nodo 0 al nodo 1 :  1 - 0
          Se ha copiado el bucket del nodo 2 al primer nodo:  0 - 0

**** SE ESTA PROCESANDO LA RANURA 7 ****
          Estado del anillo para esta ranura de tiempo
          NODO 0.           Canal 0: 0           Canal 1: 0
          NODO 1.           Canal 0: 1           Canal 1: 0
          NODO 2.           Canal 0: 1           Canal 1: 0
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 0 y este nodo SI lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 0 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          El nodo 0 no tiene mÃ¡s tramas nuevas, el indice quedo en 2
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 1 por el canal 0 y este nodo NO
posee el canal
          Se completo la recepcion de la trama por el canal 0 para el
nodo 1
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 1 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
          Todavia no se debe enviar la trama nueva
          **** PROCESO DE RECEPCION DEL NODO 2 ****
```

```
          **** CANAL 0 ****
          Recibi un bit para el nodo 2 por el canal 0 y este nodo NO
posee el canal
          No se ha completado la recepcion de la trama por el canal 0
para el nodo 2
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 2 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          El nodo 2 tiene una trama nueva la numero 11 y debe enviarse en
esta ranura de tiempo  : 7
          El nodo 2 envia el dato 1 por el canal 1
          El nodo 2 NO ha terminado de transmitir la trama 3
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 1 - 1
          Se ha pasado el contenido del nodo 1 al nodo 2 :  1 - 0
          Se ha pasado el contenido del nodo 0 al nodo 1 :  0 - 0
          Se ha copiado el bucket del nodo 2 al primer nodo:  1 - 1

**** SE ESTA PROCESANDO LA RANURA 8 ****
          Estado del anillo para esta ranura de tiempo
          NODO 0.           Canal 0: 1           Canal 1: 1
          NODO 1.           Canal 0: 0           Canal 1: 0
          NODO 2.           Canal 0: 1           Canal 1: 0
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 0 por el canal 0 y este nodo SI
posee el canal, por esta razon lo limpia
          No se ha completado la recepcion de la trama por el canal 0
para el nodo 0
          **** CANAL 1 ****
          Recibi un bit para el nodo 0 por el canal 1 y este nodo NO
posee el canal
          No se ha completado la recepcion de la trama por el canal 1
para el nodo 0
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          El nodo 0 no tiene mas tramas nuevas, el indice quedo en 2
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 1 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 1 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
          Todavia no se debe enviar la trama nueva
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 2 por el canal 0 y este nodo NO
posee el canal
          Se completo la recepcion de la trama por el canal 0 para el
nodo 2
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 2 y este nodo SI lo posee
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          El nodo 2 tiene la trama 11 en proceso de transmision por el
canal 1
```

```
          El nodo 2 termino de transmitir la trama 11 por el canal 1
          Todavia no se debe enviar la trama nueva
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 1 - 1
          Se ha pasado el contenido del nodo 1 al nodo 2 :  0 - 0
          Se ha pasado el contenido del nodo 0 al nodo 1 :  0 - 1
          Se ha copiado el bucket del nodo 2 al primer nodo:  1 - 1

**** SE ESTA PROCESANDO LA RANURA 9 ****
          Estado del anillo para esta ranura de tiempo
          NODO 0.           Canal 0: 1           Canal 1: 1
          NODO 1.           Canal 0: 0           Canal 1: 1
          NODO 2.           Canal 0: 0           Canal 1: 0
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 0 por el canal 0 y este nodo SI
posee el canal, por esta razon lo limpia
          Se completo recepcion de una trama por el canal 0 para el nodo
0
          **** CANAL 1 ****
          Recibi un bit para el nodo 0 por el canal 1 y este nodo NO
posee el canal
          Se completo la recepcion de la trama por el canal 1 para el
nodo 0
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          El nodo 0 no tiene mas tramas nuevas, el indice quedo en 2
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 1 y este nodo NO lo posee
          **** CANAL 1 ****
          Recibi un bit para el nodo 1 por el canal 1 y este nodo NO
posee el canal
          No se ha completado la recepcion de la trama por el canal 1
para el nodo 1
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
          Todavia no se debe enviar la trama nueva
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 2 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 2 y este nodo SI lo posee
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          Todavia no se debe enviar la trama nueva
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 0 - 0
          Se ha pasado el contenido del nodo 1 al nodo 2 :  0 - 1
          Se ha pasado el contenido del nodo 0 al nodo 1 :  0 - 1
          Se ha copiado el bucket del nodo 2 al primer nodo:  0 - 0

**** SE ESTA PROCESANDO LA RANURA 10 ****
          Estado del anillo para esta ranura de tiempo
          NODO 0.           Canal 0: 0           Canal 1: 0
          NODO 1.           Canal 0: 0           Canal 1: 1
          NODO 2.           Canal 0: 0           Canal 1: 1
```

```
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 0 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 0 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          El nodo 0 no tiene mas tramas nuevas, el indice quedo en 2
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 1 y este nodo NO lo posee
          **** CANAL 1 ****
          Recibi un bit para el nodo 1 por el canal 1 y este nodo NO
posee el canal
          Se completo la recepcion de la trama por el canal 1 para el
nodo 1
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
          Todavia no se debe enviar la trama nueva
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 2 y este nodo NO lo posee
          **** CANAL 1 ****
          Recibi un bit para el nodo 2 por el canal 1 y este nodo SI
posee el canal, por esta razon lo limpia
          No se ha completado la recepcion de la trama por el canal 1
para el nodo 2
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          Todavia no se debe enviar la trama nueva
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 0 - 0
          Se ha pasado el contenido del nodo 1 al nodo 2 :  0 - 1
          Se ha pasado el contenido del nodo 0 al nodo 1 :  0 - 0
          Se ha copiado el bucket del nodo 2 al primer nodo:  0 - 0

**** SE ESTA PROCESANDO LA RANURA 11 ****
          Estado del anillo para esta ranura de tiempo
          NODO 0.           Canal 0: 0           Canal 1: 0
          NODO 1.           Canal 0: 0           Canal 1: 0
          NODO 2.           Canal 0: 0           Canal 1: 1
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 0 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 0 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          El nodo 0 no tiene mas tramas nuevas, el indice quedo en 2
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 1 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 1 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
          El nodo 1 tiene una trama nueva la numero 7 y debe enviarse en
esta ranura de tiempo  : 11
          El nodo 1 enviÃ³ el dato 1 por el canal 0
```

```
          El nodo 1 NO ha terminado de transmitir la trama 3
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 2 y este nodo NO lo posee
          **** CANAL 1 ****
          Recibi un bit para el nodo 2 por el canal 1 y este nodo SI
posee el canal, por esta razon lo limpia
          Se completo recepcion de una trama por el canal 1 para el nodo
2
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          Todavia no se debe enviar la trama nueva
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 0 - 0
          Se ha pasado el contenido del nodo 1 al nodo 2 :  1 - 0
          Se ha pasado el contenido del nodo 0 al nodo 1 :  0 - 0
          Se ha copiado el bucket del nodo 2 al primer nodo:  0 - 0

**** SE ESTA PROCESANDO LA RANURA 12 ****
          Estado del anillo para esta ranura de tiempo
          NODO 0.           Canal 0: 0           Canal 1: 0
          NODO 1.           Canal 0: 0           Canal 1: 0
          NODO 2.           Canal 0: 1           Canal 1: 0
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 0 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 0 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          El nodo 0 no tiene mas tramas nuevas, el indice quedo en 2
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 1 y este nodo SI lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 1 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
          El nodo 1 tiene la trama 7 en proceso de transmision por el
canal 0
          El nodo 1 termino de transmitir la trama 7 por el canal 0
          El nodo 1 no tiene mas tramas nuevas, el indice quedo en 2
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 2 por el canal 0 y este nodo NO
posee el canal
          No se ha completado la recepcion de la trama por el canal 0
para el nodo 2
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 2 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          El nodo 2 tiene una trama nueva la numero 12 y debe enviarse en
esta ranura de tiempo  : 12
          El nodo 2 envia el dato 0 por el canal 1
          El nodo 2 NO ha terminado de transmitir la trama 3
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 1 - 0
```

```
          Se ha pasado el contenido del nodo 1 al nodo 2 :  1 - 0
          Se ha pasado el contenido del nodo 0 al nodo 1 :  0 - 0
          Se ha copiado el bucket del nodo 2 al primer nodo:  1 - 0

**** SE ESTA PROCESANDO LA RANURA 13 ****
          Estado del anillo para esta ranura de tiempo
          NODO 0.           Canal 0: 1           Canal 1: 0
          NODO 1.           Canal 0: 0           Canal 1: 0
          NODO 2.           Canal 0: 1           Canal 1: 0
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 0 por el canal 0 y este nodo NO
posee el canal
          No se ha completado la recepcion de la trama por el canal 0
para el nodo 0
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 0 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          El nodo 0 no tiene mas tramas nuevas, el indice quedo en 2
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 1 y este nodo SI lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 1 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
          El nodo 1 no tiene mÃ¡s tramas nuevas, el indice quedo en 2
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 2 por el canal 0 y este nodo NO
posee el canal
          Se completo la recepcion de la trama por el canal 0 para el
nodo 2
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 2 y este nodo SI lo posee
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          El nodo 2 tiene la trama 12 en proceso de transmision por el
canal 1
          El nodo 2 termino de transmitir la trama 12 por el canal 1
          El nodo 2 no tiene mÃ¡s tramas nuevas, el indice quedo en 2
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 1 - 0
          Se ha pasado el contenido del nodo 1 al nodo 2 :  0 - 0
          Se ha pasado el contenido del nodo 0 al nodo 1 :  1 - 0
          Se ha copiado el bucket del nodo 2 al primer nodo:  1 - 0

**** SE ESTA PROCESANDO LA RANURA 14 ****
          Estado del anillo para esta ranura de tiempo
          NODO 0.           Canal 0: 1           Canal 1: 0
          NODO 1.           Canal 0: 1           Canal 1: 0
          NODO 2.           Canal 0: 0           Canal 1: 0
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 0 por el canal 0 y este nodo NO
posee el canal
```

```
          Se completo la recepcion de la trama por el canal 0 para el
nodo 0
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 0 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          El nodo 0 no tiene mÃ¡s tramas nuevas, el indice quedo en 2
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 1 por el canal 0 y este nodo SI
posee el canal, por esta razon lo limpia
          No se ha completado la recepcion de la trama por el canal 0
para el nodo 1
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 1 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
          El nodo 1 no tiene mÃ¡s tramas nuevas, el indice quedo en 2
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 2 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 2 y este nodo SI lo posee
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          El nodo 2 no tiene mÃ¡s tramas nuevas, el indice quedo en 2
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 0 - 0
          Se ha pasado el contenido del nodo 1 al nodo 2 :  0 - 0
          Se ha pasado el contenido del nodo 0 al nodo 1 :  1 - 0
          Se ha copiado el bucket del nodo 2 al primer nodo:  0 - 0

**** SE ESTA PROCESANDO LA RANURA 15 ****
          Estado del anillo para esta ranura de tiempo
          NODO 0.           Canal 0: 0           Canal 1: 0
          NODO 1.           Canal 0: 1           Canal 1: 0
          NODO 2.           Canal 0: 0           Canal 1: 0
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 0 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 0 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          El nodo 0 no tiene mas tramas nuevas, el indice quedo en 2
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 1 por el canal 0 y este nodo SI
posee el canal, por esta razon lo limpia
          Se completo recepcion de una trama por el canal 0 para el nodo
1
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 1 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
          El nodo 1 no tiene mas tramas nuevas, el indice quedo en 2
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 2 y este nodo NO lo posee
```

```
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 2 y este nodo SI lo posee
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          El nodo 2 no tiene mas tramas nuevas, el indice quedo en 2
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 0 - 0
          Se ha pasado el contenido del nodo 1 al nodo 2 :  0 - 0
          Se ha pasado el contenido del nodo 0 al nodo 1 :  0 - 0
          Se ha copiado el bucket del nodo 2 al primer nodo:  0 - 0
          **** TERMINA PROCESAMIENTO DE LAS 15 RANURAS ****
2 processes created
```

Para aclarar la salida, se tomará como ejemplo lel procesamiento de la ranura de tiempo 5 (en negrilla). En esta ranura de tiempo la situación de los nodos es:

1. Nodo 0:
    a. Ya transmitió su primer trama, la que debió iniciar en la ranura de tiempo 1, ya que ésta solo tiene dos bits luego ocupa dos ranuras de tiempo transmitirla.
    b. Recibe, por el canal 0, el último bit de la primera trama que transmisió y por tanto lo elimina del bucket por que este dato ya dio la vuelta al anillo, luego en el bucket escribirá un 0.
    c. Recibe por el canal 1, el último bit que transmitió el Nodo 1 por el canal 1 y dado que no es un dato que el transmitió este nodo, simplemente lo toma y lo deja proseguir por el anillo.
    d. Tiene una nueva trama por transmitir y como ya se liberó el canal 0 se apodera de él escribiendo en el bucket indicando que está ocupado.
2. Nodo 1:
    a. Ya transmitió la primera trama que debía enviar en la ranura de tiempo 2.
    b. Por el canal 0 recibe el mensaje que indica que el canal está libre.
    c. Por el canal 1 recibe el primer dato que transmitió en la ranura de tiempo 2 y que ya dio la vuelta al anillo, queda a la espera del siguiente dato.
    d. No tiene tramas para transmitir en esta ranura de tiempo.

3. El nodo 2:
    a. Ve los canales libres.
    b. No tiene tramas para transmitir en esta ranura de tiempo.

Por este motivo el **Bucket** queda en el siguiente estado antes de rotar por el anillo:

N0	N1	N2
1	0	0
1	0	0

Y en el siguiente estado después de rotar el anillo:

N0	N1	N2
0	1	0
0	1	0

Caso 2

En el caso anterior los canales estaban en algunos momentos de tiempo vacíos, por ejemplo, para la ranura de tiempo 3 el nodo 0 percibe que ambos canales están libres; en otro momento de tiempo estanas ambos ocupados, por ejemplo en a ranira de tiempo 3, donde el nodo 2 percibe que ambos canales está ocupados, por último existen momentos de tiempo en los que un canal está libre y otro ocupado, como ocurrió en la ranuta de tiempo 7, para el nodo 2.

El segundo caso pretende tipificar el caso en el que en el que en algún momento de tiempo ambos canales están ocupados y un nodo requiere transmitir. Para simplicar la salida se pondrán solo dos nodos a transmitir y será el nodo 0 el que deba expropiar a otro nodo del canal para poder transmitir su trama. Las condiciones inciales para esta simulación son:

```
#define CANALES 2 //Numero de canales
#define NODOS 3 //Numero de nodos
#define RANURAS 13 //Numero de ranuras de tiempo
#define BUFFER 15/*Tamaño del Búfer que determina cuantas tramas
puede transmitir un nodo se reparte proporcinalmente para el
numero de nodos que se definan, en este caso cada nodo puede tener
5 tramas para transmitir*/
#define DATOS 4 /* Cuanto bits de datos tiene la trama, esto solo
se usa para mantener el tipo de mensaje*/
```

Nodo-0:

- Envía trama en la ranura 1
- Envía trama en la ranura 4

Nodo-1:

- Envía trama en la ranura 2

Nodo-2:

- No envía tramas.

La salida que arrojó este caso fue:

```
**** SE ESTA PROCESANDO LA RANURA 1 ****
          Estado del anillo para esta ranura de tiempo
          NODO 0.           Canal 0: 0           Canal 1: 0
          NODO 1.           Canal 0: 0           Canal 1: 0
          NODO 2.           Canal 0: 0           Canal 1: 0
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 0 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 0 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          El nodo 0 tiene una trama nueva la numero 1 y debe enviarse en
esta ranura de tiempo  : 1
          El nodo 0 envia el dato 1 por el canal 0
          El nodo 0 NO ha terminado de transmitir la trama 1
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 1 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 1 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
          TodavÃa no se debe enviar la trama nueva
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 2 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 2 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          El nodo 2 no tiene mas tramas nuevas, el indice quedo en 1
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 0 - 0
          Se ha pasado el contenido del nodo 1 al nodo 2 :  0 - 0
          Se ha pasado el contenido del nodo 0 al nodo 1 :  1 - 0
          Se ha copiado el bucket del nodo 2 al primer nodo:  0 - 0

**** SE ESTA PROCESANDO LA RANURA 2 ****
          Estado del anillo para esta ranura de tiempo
          NODO 0.           Canal 0: 0           Canal 1: 0
          NODO 1.           Canal 0: 1           Canal 1: 0
          NODO 2.           Canal 0: 0           Canal 1: 0
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 0 y este nodo SI lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 0 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          El nodo 0 tiene la trama 1 en proceso de transmision por el
canal 0
          El nodo 0 NO ha terminado de transmitir la trama 1
          TodavÃa no se debe enviar la trama nueva
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 1 por el canal 0 y este nodo NO
posee el canal
```

```
          No se ha completado la recepcion de la trama por el canal 0
para el nodo 1
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 1 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
          El nodo 1 tiene una trama nueva la numero 6 y debe enviarse en
esta ranura de tiempo  : 2
          El nodo 1 envia el dato 1 por el canal 1
          El nodo 1 NO ha terminado de transmitir la trama 2
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 2 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 2 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          El nodo 2 no tiene mas tramas nuevas, el indice quedo en 1
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 0 - 0
          Se ha pasado el contenido del nodo 1 al nodo 2 :  1 - 1
          Se ha pasado el contenido del nodo 0 al nodo 1 :  1 - 0
          Se ha copiado el bucket del nodo 2 al primer nodo:  0 - 0

**** SE ESTA PROCESANDO LA RANURA 3 ****
          Estado del anillo para esta ranura de tiempo
          NODO 0.           Canal 0: 0           Canal 1: 0
          NODO 1.           Canal 0: 1           Canal 1: 0
          NODO 2.           Canal 0: 1           Canal 1: 1
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 0 y este nodo SI lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 0 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          El nodo 0 tiene la trama 1 en proceso de transmision por el
canal 0
          El nodo 0 NO ha terminado de transmitir la trama 1
          Todavia no se debe enviar la trama nueva
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 1 por el canal 0 y este nodo NO
posee el canal
          No se ha completado la recepcion de la trama por el canal 0
para el nodo 1
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 1 y este nodo SI lo posee
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
          El nodo 1 tiene la trama 6 en proceso de transmision por el
canal 1
          El nodo 1 NO ha terminado de transmitir la trama 6
          El nodo 1 no tiene mas tramas nuevas, el indice quedo en 1
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 2 por el canal 0 y este nodo NO
posee el canal
```

```
          No se ha completado la recepcion de la trama por el canal 0
para el nodo 2
          **** CANAL 1 ****
          Recibi un bit para el nodo 2 por el canal 1 y este nodo NO
posee el canal
          No se ha completado la recepcion de la trama por el canal 1
para el nodo 2
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          El nodo 2 no tiene mas tramas nuevas, el indice quedo en 1
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 1 - 1
          Se ha pasado el contenido del nodo 1 al nodo 2 :  1 - 1
          Se ha pasado el contenido del nodo 0 al nodo 1 :  1 - 0
          Se ha copiado el bucket del nodo 2 al primer nodo:  1 - 1

**** SE ESTA PROCESANDO LA RANURA 4 ****
          Estado del anillo para esta ranura de tiempo
          NODO 0.           Canal 0: 1           Canal 1: 1
          NODO 1.           Canal 0: 1           Canal 1: 0
          NODO 2.           Canal 0: 1           Canal 1: 1
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 0 por el canal 0 y este nodo SI
posee el canal, por esta razon lo limpia
          No se ha completado la recepcion de la trama por el canal 0
para el nodo 0
          **** CANAL 1 ****
          Recibi un bit para el nodo 0 por el canal 1 y este nodo NO
posee el canal
          No se ha completado la recepcion de la trama por el canal 1
para el nodo 0
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          El nodo 0 tiene la trama 1 en proceso de transmision por el
canal 0
          El nodo 0 termino de transmitir la trama 1 por el canal 0
          El nodo 0 tiene una trama nueva la numero 2 y debe enviarse en
esta ranura de tiempo  : 4
          No hay canales disponibles el nodo explorara la estrategia de
expropiacion
          Se inserta un mensaje de expropiacion en el buffer del nodo 0
          Datos de la trama de expropiacion : 2,4,0,0
          Datos de la trama que se desplazo : 1,4,0,0
          El nodo 0 enviÃ³ el dato 2 por el canal 1
          El nodo 0 termino de transmitir la trama que envio expropiando
el canal 1
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 1 por el canal 0 y este nodo NO
posee el canal
          No se ha completado la recepcion de la trama por el canal 0
para el nodo 1
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 1 y este nodo SI lo posee
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
```

```
          El nodo 1 tiene la trama 6 en proceso de transmision por el
canal 1
          El nodo 1 NO ha terminado de transmitir la trama 6
          El nodo 1 no tiene mÃ¡s tramas nuevas, el indice quedo en 2
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 2 por el canal 0 y este nodo NO
posee el canal
          No se ha completado la recepcion de la trama por el canal 0
para el nodo 2
          **** CANAL 1 ****
          Recibi un bit para el nodo 2 por el canal 1 y este nodo NO
posee el canal
          No se ha completado la recepcion de la trama por el canal 1
para el nodo 2
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          El nodo 2 no tiene mÃ¡s tramas nuevas, el indice quedo en 2
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 1 - 1
          Se ha pasado el contenido del nodo 1 al nodo 2 :  1 - 1
          Se ha pasado el contenido del nodo 0 al nodo 1 :  1 - 2
          Se ha copiado el bucket del nodo 2 al primer nodo:  1 - 1

**** SE ESTA PROCESANDO LA RANURA 5 ****
          Estado del anillo para esta ranura de tiempo
          NODO 0.           Canal 0: 1           Canal 1: 1
          NODO 1.           Canal 0: 1           Canal 1: 2
          NODO 2.           Canal 0: 1           Canal 1: 1
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 0 por el canal 0 y este nodo SI
posee el canal, por esta razon lo limpia
          No se ha completado la recepcion de la trama por el canal 0
para el nodo 0
          **** CANAL 1 ****
          Recibi un bit para el nodo 0 por el canal 1 y este nodo SI
posee el canal, por esta razon lo limpia
          Se identifica que el canal 1 fue expropiado y se deja vacio
para que sea usado por el nodo 0
          Se completo recepcion de una trama por el canal 1 para el nodo
0
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          El nodo 0 tiene una trama nueva la numero 3 y debe enviarse en
esta ranura de tiempo  : 5
          El nodo 0 enviÃ³ el dato 1 por el canal 1
          El nodo 0 NO ha terminado de transmitir la trama 3
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 1 por el canal 0 y este nodo NO
posee el canal
          Se completo la recepcion de la trama por el canal 0 para el
nodo 1
          **** CANAL 1 ****
```

```
          El nodo 1 posee el canal 1 pero ha sido expropiado por otro
nodo, se elimina toda informacion recibida y libera el canal
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
          El nodo 1 no tiene mÃ¡s tramas nuevas, el indice quedo en 3
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 2 por el canal 0 y este nodo NO
posee el canal
          No se ha completado la recepcion de la trama por el canal 0
para el nodo 2
          **** CANAL 1 ****
          Recibi un bit para el nodo 2 por el canal 1 y este nodo NO
posee el canal
          No se ha completado la recepcion de la trama por el canal 1
para el nodo 2
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          El nodo 2 no tiene mÃ¡s tramas nuevas, el indice quedo en 3
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 1 - 1
          Se ha pasado el contenido del nodo 1 al nodo 2 :  1 - 2
          Se ha pasado el contenido del nodo 0 al nodo 1 :  0 - 1
          Se ha copiado el bucket del nodo 2 al primer nodo:  1 - 1

**** SE ESTA PROCESANDO LA RANURA 6 ****
          Estado del anillo para esta ranura de tiempo
          NODO 0.           Canal 0: 1           Canal 1: 1
          NODO 1.           Canal 0: 0           Canal 1: 1
          NODO 2.           Canal 0: 1           Canal 1: 2
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 0 por el canal 0 y este nodo SI
posee el canal, por esta razon lo limpia
          No se ha completado la recepcion de la trama por el canal 0
para el nodo 0
          **** CANAL 1 ****
          Recibi un bit para el nodo 0 por el canal 1 y este nodo SI
posee el canal, por esta razon lo limpia
          No se ha completado la recepcion de la trama por el canal 1
para el nodo 0
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          El nodo 0 tiene la trama 3 en proceso de transmision por el
canal 1
          El nodo 0 NO ha terminado de transmitir la trama 3
          El nodo 0 no tiene mÃ¡s tramas nuevas, el indice quedo en 3
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 1 y este nodo NO lo posee
          **** CANAL 1 ****
          Recibi un bit para el nodo 1 por el canal 1 y este nodo NO
posee el canal
          No se ha completado la recepcion de la trama por el canal 1
para el nodo 1
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
          El nodo 1 no tiene mÃ¡s tramas nuevas, el indice quedo en 3
```

```
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 2 por el canal 0 y este nodo NO
posee el canal
          Se completo la recepcion de la trama por el canal 0 para el
nodo 2
          **** CANAL 1 ****
          El nodo 2 NO posee el canal 1 y este fue expropiado por otro
nodo. Elimina los datos recibidos.
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          El nodo 2 no tiene mÃ¡s tramas nuevas, el indice quedo en 3
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 1 - 2
          Se ha pasado el contenido del nodo 1 al nodo 2 :  0 - 1
          Se ha pasado el contenido del nodo 0 al nodo 1 :  0 - 1
          Se ha copiado el bucket del nodo 2 al primer nodo:  1 - 2

**** SE ESTA PROCESANDO LA RANURA 7 ****
          Estado del anillo para esta ranura de tiempo
          NODO 0.           Canal 0: 1           Canal 1: 2
          NODO 1.           Canal 0: 0           Canal 1: 1
          NODO 2.           Canal 0: 0           Canal 1: 1
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          Recibi un bit para el nodo 0 por el canal 0 y este nodo SI
posee el canal, por esta razon lo limpia
          Se completo recepcion de una trama por el canal 0 para el nodo
0
          **** CANAL 1 ****
          El nodo 0 posee el canal 1 se elimina toda informacion que haya
recibido y queda apoderado del canal
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          El nodo 0 tiene la trama 3 en proceso de transmision por el
canal 1
          El nodo 0 NO ha terminado de transmitir la trama 3
          El nodo 0 no tiene mÃ¡s tramas nuevas, el indice quedo en 3
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 1 y este nodo NO lo posee
          **** CANAL 1 ****
          Recibi un bit para el nodo 1 por el canal 1 y este nodo NO
posee el canal
          No se ha completado la recepcion de la trama por el canal 1
para el nodo 1
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
          El nodo 1 no tiene mÃ¡s tramas nuevas, el indice quedo en 3
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 2 y este nodo NO lo posee
          **** CANAL 1 ****
          Recibi un bit para el nodo 2 por el canal 1 y este nodo NO
posee el canal
          No se ha completado la recepcion de la trama por el canal 1
para el nodo 2
```

****** PROCESO DE TRANSMISON DEL NODO 2 ******
El nodo 2 no tiene mÃ¡s tramas nuevas, el indice quedo en 3
****** ROTA EL ANILLO ******
Se ha copiado el bucket del nodo 2 : 0 - 1
Se ha pasado el contenido del nodo 1 al nodo 2 : 0 - 1
Se ha pasado el contenido del nodo 0 al nodo 1 : 0 - 1
Se ha copiado el bucket del nodo 2 al primer nodo: 0 - 1

**** SE ESTA PROCESANDO LA RANURA 8 ****
Estado del anillo para esta ranura de tiempo
NODO 0. Canal 0: 0 Canal 1: 1
NODO 1. Canal 0: 0 Canal 1: 1
NODO 2. Canal 0: 0 Canal 1: 1
**** PROCESO DE RECEPCION DEL NODO 0 ****
**** CANAL 0 ****
El canal 0 esta libre para el nodo 0 y este nodo NO lo posee
**** CANAL 1 ****
Recibi un bit para el nodo 0 por el canal 1 y este nodo SI posee el canal, por esta razon lo limpia
No se ha completado la recepcion de la trama por el canal 1 para el nodo 0
**** PROCESO DE TRANSMISON DEL NODO 0 ****
El nodo 0 tiene la trama 3 en proceso de transmision por el canal 1
El nodo 0 termino de transmitir la trama 3 por el canal 1
El nodo 0 no tiene mas tramas nuevas, el indice quedo en 3
**** PROCESO DE RECEPCION DEL NODO 1 ****
**** CANAL 0 ****
El canal 0 esta libre para el nodo 1 y este nodo NO lo posee
**** CANAL 1 ****
Recibi un bit para el nodo 1 por el canal 1 y este nodo NO posee el canal
No se ha completado la recepcion de la trama por el canal 1 para el nodo 1
**** PROCESO DE TRANSMISON DEL NODO 1 ****
El nodo 1 no tiene mas tramas nuevas, el indice quedo en 3
**** PROCESO DE RECEPCION DEL NODO 2 ****
**** CANAL 0 ****
El canal 0 esta libre para el nodo 2 y este nodo NO lo posee
**** CANAL 1 ****
Recibi un bit para el nodo 2 por el canal 1 y este nodo NO posee el canal
No se ha completado la recepcion de la trama por el canal 1 para el nodo 2
**** PROCESO DE TRANSMISON DEL NODO 2 ****
El nodo 2 no tiene mas tramas nuevas, el indice quedo en 3
**** ROTA EL ANILLO ****
Se ha copiado el bucket del nodo 2 : 0 - 1
Se ha pasado el contenido del nodo 1 al nodo 2 : 0 - 1
Se ha pasado el contenido del nodo 0 al nodo 1 : 0 - 1
Se ha copiado el bucket del nodo 2 al primer nodo: 0 - 1

**** SE ESTA PROCESANDO LA RANURA 9 ****
Estado del anillo para esta ranura de tiempo

```
          NODO 0.           Canal 0: 0           Canal 1: 1
          NODO 1.           Canal 0: 0           Canal 1: 1
          NODO 2.           Canal 0: 0           Canal 1: 1
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 0 y este nodo NO lo posee
          **** CANAL 1 ****
          Recibi un bit para el nodo 0 por el canal 1 y este nodo SI
posee el canal, por esta razon lo limpia
          No se ha completado la recepcion de la trama por el canal 1
para el nodo 0
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          El nodo 0 no tiene mas tramas nuevas, el indice quedo en 3
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 1 y este nodo NO lo posee
          **** CANAL 1 ****
          Recibi un bit para el nodo 1 por el canal 1 y este nodo NO
posee el canal
          Se completo la recepcion de la trama por el canal 1 para el
nodo 1
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
          El nodo 1 no tiene mas tramas nuevas, el indice quedo en 3
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 2 y este nodo NO lo posee
          **** CANAL 1 ****
          Recibi un bit para el nodo 2 por el canal 1 y este nodo NO
posee el canal
          No se ha completado la recepcion de la trama por el canal 1
para el nodo 2
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          El nodo 2 no tiene mas tramas nuevas, el indice quedo en 3
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 0 - 1
          Se ha pasado el contenido del nodo 1 al nodo 2 :  0 - 1
          Se ha pasado el contenido del nodo 0 al nodo 1 :  0 - 0
          Se ha copiado el bucket del nodo 2 al primer nodo:  0 - 1

**** SE ESTA PROCESANDO LA RANURA 10 ****
          Estado del anillo para esta ranura de tiempo
          NODO 0.           Canal 0: 0           Canal 1: 1
          NODO 1.           Canal 0: 0           Canal 1: 0
          NODO 2.           Canal 0: 0           Canal 1: 1
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 0 y este nodo NO lo posee
          **** CANAL 1 ****
          Recibi un bit para el nodo 0 por el canal 1 y este nodo SI
posee el canal, por esta razon lo limpia
          No se ha completado la recepcion de la trama por el canal 1
para el nodo 0
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          El nodo 0 no tiene mas tramas nuevas, el indice quedo en 3
```

```
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 1 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 1 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
          El nodo 1 no tiene mas tramas nuevas, el indice quedo en 3
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 2 y este nodo NO lo posee
          **** CANAL 1 ****
          Recibi un bit para el nodo 2 por el canal 1 y este nodo NO
posee el canal
          Se completo la recepcion de la trama por el canal 1 para el
nodo 2
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          El nodo 2 no tiene mas tramas nuevas, el indice quedo en 3
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 0 - 1
          Se ha pasado el contenido del nodo 1 al nodo 2 :  0 - 0
          Se ha pasado el contenido del nodo 0 al nodo 1 :  0 - 0
          Se ha copiado el bucket del nodo 2 al primer nodo:  0 - 1

**** SE ESTA PROCESANDO LA RANURA 11 ****
          Estado del anillo para esta ranura de tiempo
          NODO 0.           Canal 0: 0           Canal 1: 1
          NODO 1.           Canal 0: 0           Canal 1: 0
          NODO 2.           Canal 0: 0           Canal 1: 0
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 0 y este nodo NO lo posee
          **** CANAL 1 ****
          Recibi un bit para el nodo 0 por el canal 1 y este nodo SI
posee el canal, por esta razon lo limpia
          Se completo recepcion de una trama por el canal 1 para el nodo
0
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          El nodo 0 no tiene mas tramas nuevas, el indice quedo en 3
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 1 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 1 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
          El nodo 1 no tiene mas tramas nuevas, el indice quedo en 3
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 2 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 2 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          El nodo 2 no tiene mas tramas nuevas, el indice quedo en 3
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 0 - 0
```

```
          Se ha pasado el contenido del nodo 1 al nodo 2 :  0 - 0
          Se ha pasado el contenido del nodo 0 al nodo 1 :  0 - 0
          Se ha copiado el bucket del nodo 2 al primer nodo:  0 - 0

**** SE ESTA PROCESANDO LA RANURA 12 ****
          Estado del anillo para esta ranura de tiempo
          NODO 0.           Canal 0: 0           Canal 1: 0
          NODO 1.           Canal 0: 0           Canal 1: 0
          NODO 2.           Canal 0: 0           Canal 1: 0
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 0 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 0 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          El nodo 0 no tiene mas tramas nuevas, el indice quedo en 3
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 1 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 1 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
          El nodo 1 no tiene mas tramas nuevas, el indice quedo en 3
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 2 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 2 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          El nodo 2 no tiene mas tramas nuevas, el indice quedo en 3
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 0 - 0
          Se ha pasado el contenido del nodo 1 al nodo 2 :  0 - 0
          Se ha pasado el contenido del nodo 0 al nodo 1 :  0 - 0
          Se ha copiado el bucket del nodo 2 al primer nodo:  0 - 0

**** SE ESTA PROCESANDO LA RANURA 13 ****
          Estado del anillo para esta ranura de tiempo
          NODO 0.           Canal 0: 0           Canal 1: 0
          NODO 1.           Canal 0: 0           Canal 1: 0
          NODO 2.           Canal 0: 0           Canal 1: 0
          **** PROCESO DE RECEPCION DEL NODO 0 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 0 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 0 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 0 ****
          El nodo 0 no tiene mas tramas nuevas, el indice quedo en 3
          **** PROCESO DE RECEPCION DEL NODO 1 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 1 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 1 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 1 ****
```

```
          El nodo 1 no tiene mas tramas nuevas, el indice quedo en 3
          **** PROCESO DE RECEPCION DEL NODO 2 ****
          **** CANAL 0 ****
          El canal 0 esta libre para el nodo 2 y este nodo NO lo posee
          **** CANAL 1 ****
          El canal 1 esta libre para el nodo 2 y este nodo NO lo posee
          **** PROCESO DE TRANSMISON DEL NODO 2 ****
          El nodo 2 no tiene mas tramas nuevas, el indice quedo en 3
          **** ROTA EL ANILLO ****
          Se ha copiado el bucket del nodo 2 : 0 - 0
          Se ha pasado el contenido del nodo 1 al nodo 2 :  0 - 0
          Se ha pasado el contenido del nodo 0 al nodo 1 :  0 - 0
          Se ha copiado el bucket del nodo 2 al primer nodo:  0 - 0
          **** TERMINA PROCESAMIENTO DE LAS 13 RANURAS ****
2 processes created
```

Para este caso las ranuras de tiempo que se explican van de la 4 a la 7 (en negrilla):

1. Nodo 0:
    a. En la ranura de tiempo 4 tienen en proceso de transmisión la primera trama y requiere transmitir una segunda trama, pero ambos canales se encuentran ocupados.
    b. Por el canal 0 ubica el último dato de la primera trama.
    c. Ubica en el bucket del canal 1 el mensaje de expropiación cuya descripción es: **2,4,0,0.**
    d. Este mensaje de expropiación se transmite en solo una ranura para proceder a ubicar el primer bit de la trama que se desea transmitir en la siguiente ranura de tiempo.
    e. En la ranura de tiempo 5 en este nodo ubica el primer dato de la segunda trama que requiere transmitir y en las siguientes ranuras de tiempo hace lo mismo hasta la ranura 8 en la que transmite el último dato.
    f. En la ranura de tiempo 7 este nodo recibe su propio mensaje de expropiación y lo ignora para continuar transmitiendo sus datos por el canal 1.

2. Nodo 1:
    a. En la ranura de tiempo 5, este nodo es notificado de que ha sido expropiado el canal que poseía y sobre el que estaba transmitiendo la trama que inició su procesamiento en la ranura 2 de tiempo. Este nodo suelta el canal, limpia su búfer y deja de enviar datos.
    b. En las siguientes ranuras de tiempo recibe los datos del canal 0 asumiendo que son enviados por otro nodo y los toma.

3. Nodo 2:
    a. Este nodo se entera del mensaje de expropiación que envío el Nodo 0 en la ranura de tiempo 6, luego limpia todo lo que el Nodo 1 había transmitido.

Validación

Para la verificación del modelo usando **SPIN** se ejecutó la siguiente secuencia de comandos:

```
#spin –a Nivel_Intermedio.prom
#gcc –o pan pan.c
#./pan
```

La salida generada por el proceso de validación, se muestra a continuación:

```
(Spin Version 4.2.5 -- 2 April 2005)
  + Partial Order Reduction

Full statespace search for:
  never claim          - (none specified)
  assertion violations +
  acceptance   cycles  - (not selected)
  invalid end states +

State-vector 116 byte, depth reached 2328, errors: 0
    2329 states, stored
       0 states, matched
    2329 transitions (= stored+matched)
       0 atomic steps
hash conflicts: 0 (resolved)

2.827  memory usage (Mbyte)

unreached in proctype comunicacion
  (0 of 293 states)
unreached in proctype :init:
  (0 of 2 states)

real   0m0.036s
user   0m0.015s
sys    0m0.009s
```

ANEXO B.

Código Fuente

La implementación del *Modelo 1* del sub-nivel de enlace de datos en **Promela** es la siguiente:

```
/* Protocolo PDM-Ring
   Sub-nivel de enlace

    Descripción:
    Modelo 2 del protocolo del sub-nivel de enlace.
    Se utilizan 2 nodos que se comunican a través de 2
    canales uno de entrada y de salida.

    versión: 2.0.1
    fecha de creación: 27-11-2005
    ultima modificación: 06-12-2005
    Nombre del archivo fuente: enlace
    autores : AGD & MER.
*/

          // estados de nodo.
#define OK     0 // estado OK
#define BUSY   1 // estado ocupado
#define OUT    2 // estado fuera de servicio
#define Rx     10// capacidad búfer de Rx (tramas)
#define Tx     10// capacidad búfer de Tx (tramas)
#define MAX    100   // cantidad de tramas a enviar.
#define TAM    2 // número de nodos del anillo
                         // tamaño tabla de estados.

/*
  tipos de mensajes:
  msg - mensaje de datos sin confirmación
  msgack - mensaje de datos con confirmación
  ack - mensaje de confirmación de recepción de trama
  sok - mensaje estado de nodo OK
  by - mensaje de salida (fuera de servicio) controlada de nodo
  rnr - mensaje para informar que un nodo no se pueden recibir  tr
  mas.
  rr - mensaje para informar que un nodo ya puede recibir tramas.
*/
```

```
// Definición tipos de mensajes
mtype = {msg, msgack, ack, sok, by, rnr, rr}

/* definición de los canales - se utilizan dos elementos por
canal: el tipo de mensaje y un valor entero simulando el contenido
del mensaje */

chan toS = [1] of {mtype, byte};   // canal de salida
chan toR = [1] of {mtype, byte};   // canal de entrada

byte estados[TAM]; // tabla de estados de nodos

int cont;      // contador de tramas a enviar

// proceso de envío de tramas
proctype enviar(chan in, out)
{
  // variables de recepción y envío de datos
  byte datos_envio, datos_rec;
  // contador de tramas
     int cont_tramas;
  // contador de mensajes tipo ACK
     int cont_ack;
  // contador de tramas  con confirmación
     int cont_msg_ack;
  // centinelas para envío de mensajes RNR y RR
  bit centinela, centinela_out;
  envio: // etiqueta
  do
  ::(cont < MAX) -> // verificar contador de tramas
    inicio:
    if
    ::(estados[0] == 0) -> // verificar estado del nodo
      if
      // enviar trama sin confirmación
      ::out!msg,datos_envio ->
        cont++; // incrementar contador de tramas
        goto envio
        // enviar trama con confirmación
      ::out!msgack,datos_envio ->
        cont++; // incrementar contador de tramas
          //incrementar contador tramas con confirmación
        cont_msg_ack++;
        goto envio
      ::out!sok,0 -> // enviar mensaje estado OK
        goto envio
      // recibir mensaje RNR
      ::in?rnr,datos_rec -> goto envio
```

```
      // recibir mensaje RR
      ::in?rr,datos_rec -> goto envio
      // recibir mensaje de salida de nodo
      ::in?by,1 -> goto envio
      // recibir mensaje de confirmacion de trama
      ::in?ack,datos_rec ->
        cont_ack++; // incrementar contador
        goto envio
      ::in?sok,1 -> // recibir mensaje de estado de nodo
        if
        // verificar tramas pendientes
                    ::(cont_ack == cont_msg_ack) ->
           estados[0] = 2; // cambiar estado -> OUT
          if
          // verificar estado de canal salida
          ::(len(out) == 0) ->
            // enviar mensaje de salida
            out!by,0
          ::else ->
            centinela_out=1
          fi;
        :: else -> skip
        fi;
        goto envio
      // recibir trama con confirmación
      ::in?msgack,datos_rec ->
          // incrementar contador tramas
          cont_tramas++;
          if
          // verificar estado
          :: (estados[1] == 0) ->
            // enviar mensaje ACK
            out!ack,datos_rec;
          :: else -> skip
          fi;
        if
        // verificar búfer de recepción
        :: (cont_tramas == Tx) ->
            // cambio de estado OK  -> BUSY
estados[0] = 1;
            if
            ::(len(out) == 0) ->
              // enviar mensaje RNR
              out!rnr,0;
            :: else ->
              centinela=1
            fi;
        :: else -> skip
        fi;
```

```
    goto envio
  // recibir trama sin confirmación
  ::in?msg,datos_rec ->
    cont_tramas++; // incrementar contador
    if
    // verificar búfer de transmisión
    :: (cont_tramas == Tx) ->
      estados[0]=1; // BUSY - ocupado
      // enviar mensaje RNR
      out!rnr,0;
    :: else ->
      centinela = 1
    fi;
    goto envio
  fi;
// si estado de nodo es "ocupado"
::(estados[0] == 1) -> // cambiar estado BUSY -> OK
  cont_tramas = 0; // reiniciar contador tramas
  if
  :: (centinela == 1) ->
    if
    ::(len(out) == 0) ->
      // enviar mensaje RNR
      out!rnr,0;
    ::else -> skip
    fi;
    centinela = 0
  ::else ->
    // cambiar estado del nodo -> OK
    estados[0] = 0;
    if
    ::(len(out) == 0) ->
      // enviar mensaje RR
      out!rr,0;
    ::else -> skip
    fi;
  fi;
  goto envio;
// si estado de nodo es "fuera de servicio" - OUT
::(estados[0] == 2) ->
  cont_tramas = 0; // reiniciar contador
  if
  ::(centinela_out == 1) ->
    if
    ::(len(out) == 0) ->
      // enviar mensaje de salida de nodo
      out!by,0;
    ::else -> skip
    fi;
```

```
        centinela_out = 0;
      ::else ->
        // cambiar estado -> OK
        estados[0] = 0;
        // enviar mensaje de estado de nodo OK
        out!sok,0;
      fi;
    :: else -> goto envio
    fi;
    goto envio
  :: else ->
    estados[0] = 2; // cambiar estado de nodo -> BUSY
    if
    ::(len(in) > 0) -> // si hay mensajes pendientes por Rx
    // recibir trama pendiente en canal entrada
          if
      ::in?msg,datos_rec ->
      ::in?msgack,datos_rec ->
      ::in?ack,datos_rec -> cont_ack++
      ::in?rnr,datos_rec ->
      ::in?rr,datos_rec ->
      ::in?sok,1 ->
      ::in?by,1 ->
      fi
    ::else -> skip
    fi;
    break; // terminar ciclo de procedimiento
  od
}

// proceso de recepción de tramas
proctype recibir(chan in, out)
{
  // variables de recepción y envío de datos
  byte datos_envio, datos_rec;
  // contadores de tramas en búfer Rx
  int bufer_Rx;
  // contador de tramas con confirmación
     int cont_msgack;
  // contador de mensajes de confirmación
     int cont_ack;
  inicio:
  do
  ::(cont < MAX) -> // verificar contador de tramas
    estado:
    if
    :: (estados[1] == OK) -> // verificar estado OK
      if
      // si el búfer de Rx tiene espacio
```

```
:: bufer_Rx < Rx ->
  if
  // recibir trama tipo sin confirmación
  ::in?msg(datos_rec) ->
    bufer_Rx++; // incrementar búfer Rx
    if
    :: (estados[0] == OK) ->
      if
      ::(len(out) == 0) ->
      //enviar trama sin confirmación
        out!msg(datos_envio);
      ::else -> skip
      fi;
    :: else -> skip
    fi;
    goto inicio
  // recibir trama tipo con confirmación
              ::in?msgack(datos_rec) ->
    bufer_Rx++; // incrementar contador
    if
    // verificar estado OK
    :: (estados[0] == OK) ->
      // enviar trama de confirmación
      out!ack(datos_rec);
    :: else -> skip
    fi;
    goto inicio
  // enviar trama tipo con confirmación
  ::out!msgack(datos_rec) ->
    cont_msgack++; // incrementar contador
    goto inicio
  // recibir trama de confirmación
  ::in?ack(datos_rec) ->
    cont_ack++; // incrementar contador
    goto inicio
  // recibir mensaje tipo RNR
  ::in?rnr(datos_rec) -> goto inicio
  // recibir mensaje tipo RNR
  ::in?rr(datos_rec) -> goto inicio
  // recibir mensaje con estado OK
  ::in?sok(datos_rec) -> goto inicio
  // recibir mensaje tipo BY - salida
  ::in?by,0 -> goto inicio
  fi
:: (bufer_Rx == Rx) -> // verificar bufer Rx
  if
  // si no hay tramas pendientes por ACK
  ::(cont_ack != cont_msgack) ->
    // cambiar estado a BUSY - ocupado
```

```
        estados[1] = 1;
        if
        ::(len(out) == 0) ->
          out!rnr,1;
        :: else -> skip
        fi;
        goto inicio
      ::else ->
        // cambiar estado -> OUT
        estados[1] = 2;
        if
        ::(len(out)==0) ->
          // enviar mensaje de salida
          out!by,1
        :: else -> skip
        fi;
        goto inicio
      fi;
    fi
  // si estado de nodo es BUSY
  :: (estados[1] == 1) -> // cambiar estado BUSY -> OK
    bufer_Rx = 0; // incrementar contador
    estados[1] = 0; // cambiar estado -> OK
    if
    // si canal salida esta libre
    ::(len(out) == 0) ->
      // enviar trama tipo RR
      out!rr,1;
    ::else -> skip
    fi;
    goto inicio
  // si estado es OUT
  :: (estados[1] == 2) ->
    bufer_Rx = 0; // reiniciar búfer Rx
    // cambiar estado OUT -> OK
    estados[1] = 0;
    if
    ::(len(out) == 0) -> // si canal salida libre
      out!sok,1 // enviar mensaje tipo SOK
    ::else -> skip
    fi;
    goto inicio
  fi
::else ->
  if
  // si hay mensajes pendientes por Rx
  ::(len(in) > 0) ->
    if
    // recibir trama pendiente
```

```
       ::in?msg,datos_rec ->
       ::in?msgack,datos_rec ->
       ::in?ack,datos_rec -> cont_ack++
       ::in?rnr,datos_rec ->
       ::in?rr,datos_rec ->
       ::in?sok,1 ->
       ::in?by,1 ->
       fi
     ::else -> skip
     fi;
     break
  od
}

init{
  atomic{
      run enviar(toS, toR); // procedimiento envio de tramas
      run recibir(toR, toS); // procedimiento recepción de tramas
      toR!sok,0;       // enviar mensaje inicial, estado OK
  }
}
```

Simulación

El código se grabó en un archivo denominado "*enlace*", y la ejecución de la simulación del modelo usando **SPIN** desde la línea de comandos bajo la plataforma Linux se realizo con el comando,

```
#spin –c enlace
```

Como resultado de la ejecución del modelo en modo simulación, se obtuvo la siguiente salida:

```
proc 0 = :init:
proc 1 = enviar
proc 2 = recibir
q\p   0   1   2
  2   .   out!msg,0
  1   .   .   out!msgack,0
  1   .   in?msgack,0
  2   .   .   in?msg,0
  2   .   out!ack,0
  1   .   .   out!msg,0
  2   .   .   in?ack,0
  1   .   in?msg,0
  1   .   .   out!msgack,0
```

```
1   .   in?msgack,0
1   .   .   out!msgack,0
2   .   out!ack,0
2   .   .   in?ack,0
2   .   out!msgack,0
2   .   .   in?msgack,0
2   .   out!msgack,0
1   .   in?msgack,0
1   .   .   out!ack,0
2   .   .   in?msgack,0
2   .   out!ack,0
1   .   in?ack,0
1   .   .   out!ack,0
1   .   in?ack,0
1   .   .   out!msgack,0
1   .   in?msgack,0
1   .   .   out!msgack,0
2   .   .   in?ack,0
2   .   out!ack,0
2   .   .   in?ack,0
1   .   in?msgack,0
2   .   out!ack,0
1   .   .   out!msgack,0
2   .   .   in?ack,0
2   .   out!rnr,0
2   .   .   in?rnr,0
2   .   out!rr,0
2   .   .   in?rr,0
1   .   in?msgack,0
2   .   out!ack,0
1   .   .   out!msgack,0
1   .   in?msgack,0
1   .   .   out!msgack,0
2   .   .   in?ack,0
2   .   out!ack,0
2   .   .   in?ack,0
2   .   out!msg,0
1   .   in?msgack,0
2   .   .   in?msg,0
2   .   out!ack,0
1   .   .   out!msg,0
1   .   in?msg,0
1   .   .   out!msgack,0
1   .   in?msgack,0
1   .   .   out!msgack,0
2   .   .   in?ack,0
2   .   out!ack,0
2   .   .   in?ack,0
1   .   in?msgack,0
```

```
1   .   .   out!msgack,0
2   .   out!ack,0
2   .   .   in?ack,0
2   .   out!rnr,0
2   .   .   in?rnr,0
2   .   out!rnr,0
2   .   .   in?rnr,0
2   .   out!rr,0
2   .   .   in?rr,0
2   .   out!msg,0
2   .   .   in?msg,0
1   .   in?msgack,0
2   .   out!ack,0
1   .   .   out!msg,0
1   .   in?msg,0
1   .   .   out!msgack,0
2   .   .   in?ack,0
1   .   in?msgack,0
1   .   .   out!msgack,0
2   .   out!ack,0
2   .   .   in?ack,0
2   .   out!msg,0
2   .   .   in?msg,0
2   .   out!msg,0
1   .   in?msgack,0
1   .   .   out!rr,1
2   .   .   in?msg,0
2   .   out!ack,0
1   .   in?rr,1
1   .   .   out!msgack,0
1   .   in?msgack,0
1   .   .   out!msgack,0
2   .   .   in?ack,0
2   .   out!ack,0
2   .   .   in?ack,0
1   .   in?msgack,0
1   .   .   out!msgack,0
2   .   out!ack,0
2   .   .   in?ack,0
2   .   out!rnr,0
2   .   .   in?rnr,0
2   .   out!rr,0
2   .   .   in?rr,0
1   .   in?msgack,0
2   .   out!ack,0
1   .   .   out!msgack,0
2   .   .   in?ack,0
1   .   in?msgack,0
1   .   .   out!msgack,0
```

```
  2    .    out!ack,0
  2    .    .    in?ack,0
  2    .    out!msgack,0
  2    .    .    in?msgack,0
  2    .    out!msg,0
  1    .    in?msgack,0
  1    .    .    out!ack,0
  2    .    .    in?msg,0
  2    .    out!ack,0
  1    .    in?ack,0
  1    .    .    out!msgack,0
  1    .    in?msgack,0
  1    .    .    out!msgack,0
  2    .    .    in?ack,0
  2    .    out!ack,0
  2    .    .    in?ack,0
  1    .    in?msgack,0
  2    .    out!ack,0
  1    .    .    out!msgack,0
  2    .    .    in?ack,0
  1    .    in?msgack,0
  1    .    .    out!msgack,0
  2    .    out!ack,0
  2    .    .    in?ack,0
  2    .    out!rnr,0
  2    .    .    in?rnr,0
  2    .    out!rr,0
  2    .    .    in?rr,0
  1    .    in?msgack,0
  1    .    .    out!msgack,0
  2    .    out!ack,0
  2    .    .    in?ack,0
  1    .    in?msgack,0
  1    .    .    out!msgack,0
  2    .    out!ack,0
  2    .    .    in?ack,0
  2    .    out!msg,0
  1    .    in?msgack,0
-------------
final state:
-------------
3 processes created
```

En la salida obtenida, se puede apreciar una secuencia de los mensajes de diferentes tipos (con confirmación, sin confirmación, ACK y estados de nodo) intercambiados entre los dos procesos (enviar y recibir). Dichos mensajes son enviados/recibidos de manera concurrente y en cada nueva corrida que se realizo durante las pruebas, se obtuvo una secuencia diferente.

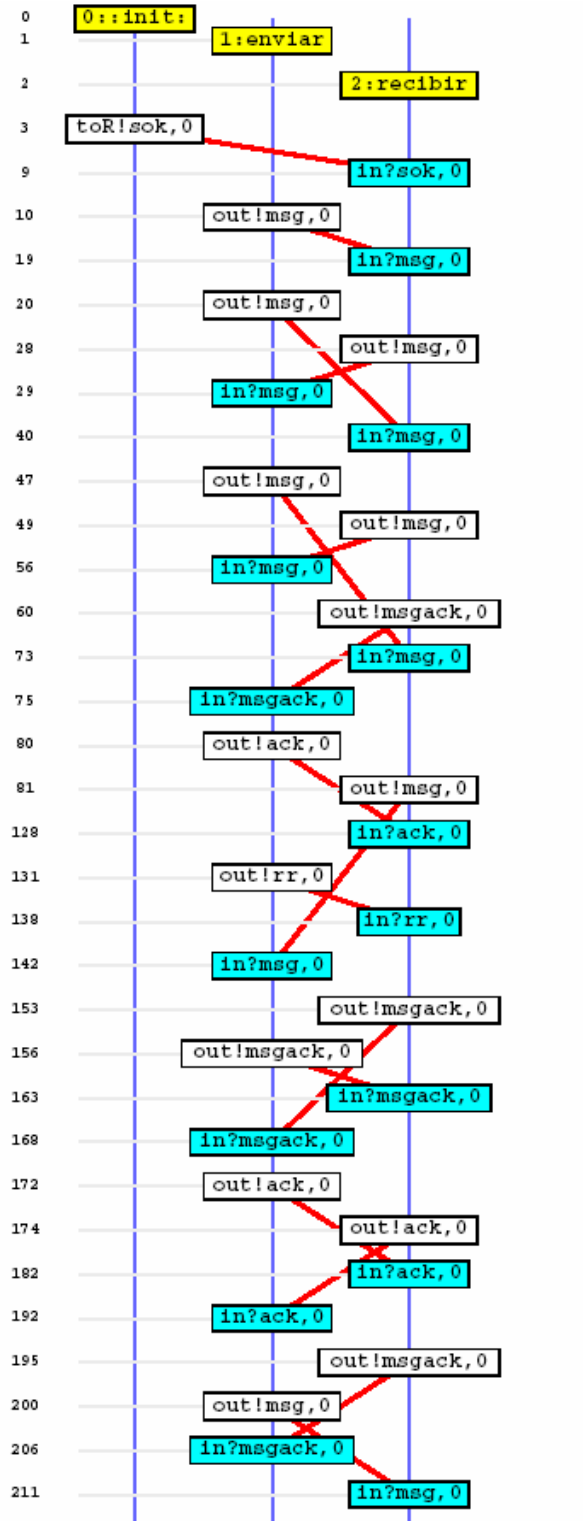
Es importante aclarar que el símbolo **?** se utiliza para indicar la recepción de mensajes a traves de un canal, en tanto que el símbolo **!** se utiliza para indicar el envio de mensajes a traves de un canal.

En las simulaciones el valor que se manipuló fue el número de tramas (mensajes), usando una constante (*MAX*) cuyo valor se cambiaba antes de cada nueva simulación. Los resultados de simulación que se presentan se realizaron usando un total de 100 tramas (*MAX* = *100*) del tipo con confirmación y sin confirmación.

Luego, usando la opción –*M* de **SPIN** en modo simulación, se obtuvo la secuencia de intercambio de mensajes de manera gráfica (ver Figura 38), la cual hace más sencilla la revisión de los resultados. Para este caso particular se ejecutó la instrucción:

```
#spin –M enlace
```

Figura 38. Simulación del Modelo 1



Validación

Para la verificación del *Modelo 1* usando **SPIN**, y considerando la gran cantidad de estados, se debio usar el método de búsqueda parcial de estados alcanzables. La secuencia de comandos utilizada fue la siguiente:

```
#spin –a enlace
#gcc –DBITSTATE –o pan pan.c
#time ./pan -m10000000 -w25
```

Donde, el primer comando genera el programa analizador; el segundo compila el programa generado en lenguaje C, y con la opción **DBITSTATE** se compila el programa analizador (**pan.***) indicando que la búsqueda de estados alcanzables será parcial; el tercer comando ejecuta el programa analizador y en este caso particular, se usaron las opciones *mN* y *wN* , para especificar la cantidad límite de estados y la cantidad máxima de memoria a utilizar.

El resultado de la corrida del programa analizador se muestra a continuación:

```
Depth=   216163 States=    1e+06 Transitions= 1.35816e+06 Memory=
381.602
Depth=   432237 States=    2e+06 Transitions= 2.71708e+06 Memory=
398.908
Depth=   647956 States=    3e+06 Transitions= 4.07862e+06 Memory=
416.214
Depth=   864363 States=    4e+06 Transitions= 5.44514e+06 Memory=
433.519
Depth= 1083828 States=    5e+06 Transitions= 6.81989e+06 Memory=
451.030
Depth= 1171845 States=    6e+06 Transitions= 8.29512e+06 Memory=
458.095
Depth= 1171845 States=    7e+06 Transitions= 9.84799e+06 Memory=
458.095
Depth= 1171845 States=    8e+06 Transitions= 1.14311e+07 Memory=
458.095
Depth= 1171845 States=    9e+06 Transitions= 1.30591e+07 Memory=
458.095
Depth= 1171845 States=     1e+07 Transitions= 1.4759e+07 Memory=
458.095
(Spin Version 4.2.5 -- 2 April 2005)
  + Partial Order Reduction

Bit statespace search for:
  never claim          - (none specified)
  assertion violations +
```

```
  acceptance   cycles   - (not selected)
  invalid end states +

State-vector 68 byte, depth reached 1171845, errors: 0
1.0949e+07 states, stored
5.52255e+06 states, matched
1.64716e+07 transitions (= stored+matched)
       2 atomic steps

hash factor: 3.06461 (best if > 100.)

bits set per state: 3 (-k3)

Stats on memory usage (in Megabytes):
875.922  equivalent memory usage for states (stored*(State-vector +
overhead))
4.194  memory used for hash array (-w25)
40.000 memory used for bit stack
320.000  memory used for DFS stack (-m10000000)
413.823  other (proc and chan stacks)
0.078  memory lost to fragmentation
458.095  total actual memory usage

unreached in proctype enviar
  (0 of 130 states)
unreached in proctype recibir
  (0 of 107 states)
unreached in proctype :init:
  (0 of 5 states)

real 1m53.581s
user 1m11.120s
sys  0m1.790s
```

Entre los datos más relevantes de la salida se observa que,

- La búsqueda parcial se detiene a una profundidad de 1171845 niveles.
- El tamaño del vector de estados del sistema es de 48 bytes.
- El **hash-factor** obtenido es muy pequeño, apenas cerca de 3, cuando un valor aceptable es mayor a 100.
- La cantidad de memoria total utilizada fue de alrededor de 458 MB.
- Luego, en la parte final de la salida, se aprecia el tiempo de computo (m:s) empleado en la ejecución 1min y 53 seg. aproximadamente.

ANEXO C.

Código Fuente

La implementación en Promela del *Modelo 2* del sub-nivel de enlace, se grabó en un archivo denominado "*nivel2*" y su contenido es:

```
/*
  Protocolo PDM-Ring : Nivel de Enlace

     Descripción: modelo del nivel de enlace que utiliza un solo
     nodo, verifica
    - la Tx y Rx de mensajes.
    - el estado del búfer de recepción (bufer_Rx).
    - los estados del nodo (OK, BUSY, OUT).

  fecha de creación: 6-12-2005
  ultima actualización: 8-12-2005
  versión: 1.0.2
  Nombre del archivo fuente: nivel2

  autor: AGD & MER.
*/

#define MAX  50// cantidad de tramas a transmitir
#define Rx   6 // capacidad del búfer de Rx
                    // estados del nodo
#define OK   0 // nodo activo y listo - OK
#define BUSY 1 // nodo ocupado - BUSY
#define OUT  2 // nodo fuera de servicio - OUT

/*
  tipos de mensajes:
  msg - mensaje de datos sin confirmación
  msgack - mensaje de datos con confirmación
  ack - mensaje de confirmación de recepción de trama
  sok - mensaje estado de nodo OK
  by - mensaje de salida (fuera de servicio) controlada de nodo
  rnr - mensaje para informar que un nodo no se pueden recibir
  tramas.
  rr - mensaje para informar que un nodo ya puede recibir tramas.
*/

// definición de tipos de mensajes (tramas)
mtype = {msg, msgack, ack, rr, rnr, by, sok};
```

```
int cont;    // contador de tramas enviadas
byte estado; // estado actual del nodo

// procedimiento para procesar tramas (envío / recepción)
proctype procesar(chan chin, chout)
{
  // variables para envío / recepción de datos
     byte o, i;
     // contador de tramas del búfer RX
     byte bufer_Rx;
  inicio:
  do
  :: (cont < MAX) -> // verificar contador de tramas
  if
  :: (estado == OK) -> // verificar estado de nodo
    if
    // recibir trama con confirmación
    ::chin?msgack(i) ->
      bufer_Rx++; // incrementar contador tramas
      if
      // verificar el búfer de Rx
      :: (bufer_Rx >= Rx) ->
        estado = BUSY; // cambiar estado
        if
        :: (len(chout) == 0) ->
          // enviar mensaje RNR
          chout!rnr,o;
        :: else -> skip
        fi
      :: else -> skip
      fi;
      if
      :: (cont < MAX) -> // verificar contador
        if
        ::(len(chout)==0)->
          // enviar mensaje de confirmación
          chout!ack(i);
          cont++ // incrementar contador
        ::else -> skip
        fi
      :: else -> goto inicio
      fi
    // recibir mensaje de confirmación
    ::chin?ack(i) ->
      if
      :: (len(chout) == 0) ->
        do
        // enviar trama sin confirmación
        ::chout!msg(o) ->
```

```
      cont++; // incrementar contador
      break
    // enviar trama con confirmación
    ::chout!msgack(o) ->
      cont++; // incrementar contador
      break
    // enviar mensaje de salida de nodo
    ::chout!by,o ->
      break;
    od;
  :: else -> skip
  fi;
// recibir trama sin confirmación
::chin?msg(i) ->
  bufer_Rx++; // incrementar contador de búfer
  if
  // verificar el bufer Rx
  :: (bufer_Rx >= Rx) ->
    estado = BUSY; // cambiar estado
    if
    :: (len(chout)==0) ->
      // enviar mensaje RNR
      chout!rnr,o;
      goto inicio
    :: else -> skip
    fi
  :: else -> skip
  fi;
  if
  // verificar contador de tramas
  :: (cont < MAX) ->
    if
    :: (len(chout) == 0) ->
      do
      // enviar trama sin confirmación
      ::chout!msg(o) ->
        cont++; // incrementar contador
        break
      // enviar trama con confirmación
      ::chout!msgack(o) ->
        cont++;
        break
      ::skip -> break
      od
    :: else -> skip
    fi
  :: else -> goto inicio
  fi
```

```
// recibir mensaje RR
::chin?rr(i) ->
  if
  :: (cont < MAX) -> // verificar contador
    if
    :: (len(chout) == 0 ) ->
      do
      ::chout!msg(o) ->
        cont++;
        break
      ::chout!msgack(o) ->
        cont++;
        break
      // enviar mensaje de salida
      ::chout!by,o ->
        break;
      od;
    :: else -> skip
    fi
  :: else -> skip
  fi;
  goto inicio
::chin?rnr(i) -> goto inicio
::chin?by(i) ->
    estado = OUT;
    goto inicio
// recibir mensaje de estado de nodo OK
::chin?sok(i) ->
  if
  :: (cont < MAX) ->
    if
    ::(len(chout)==0)->
      do
      ::chout!msg(o) ->
        cont++;
        break
      ::chout!msgack(o) ->
        cont++;
        break
      ::skip -> break
      od
    ::else -> skip
    fi
  :: else -> skip
  fi;
  goto inicio
:: timeout -> // si no se cumple ninguna guarda
  if
```

```
      :: (cont < MAX) ->
        chout!sok,o; // enviar estado de nodo
      :: else -> goto inicio
      fi
    fi
  ::(estado == BUSY) -> // si el estado es "ocupado"
    bufer_Rx = 0; // reinicia bufer Rx
    estado = OK; // cambiar estado de nodo
    if
    :: (len(chout) == 0) ->
      chout!rr,o;
    :: else -> skip
    fi;
    goto inicio
  // si el estado es "fuera de servicio"
  ::(estado == OUT) ->
    estado = OK; // cambiar estado
    if
    :: (len(chout) == 0) ->
      chout!sok,o; // enviar estado de nodo
    :: else -> skip
    fi;
    goto inicio
  fi;
  :: else ->
    if
    // si hay tramas en el canal entrada
    :: (len(chin) > 0) ->
      if
      // recibir trama
      ::chin?by,(i) ->
      ::chin?msg,(i) ->
      ::chin?msgack,(i) ->
      ::chin?ack,(i) ->
      ::chin?rnr,(i) ->
      ::chin?rr,(i) ->
      ::chin?sok,(i) ->
      fi
    :: else -> skip
    fi;
    estado = OUT;  // cambiar estado de nodo -> salida
    break
  od;
}
```

```
// proceso inicial
init{
  /* definición de canales – el primer campo representa el tipo de
  mensaje y el segundo es un valor entero que simula el contenido
  del mensaje */

  chan Ain = [1] of {mtype, byte}; // canal entrada
  chan Aout = [1] of {mtype, byte}; // canal salida

  atomic{
    run procesar(Ain,Aout); // ejecucion de procesos
    run procesar(Aout,Ain);
  }

  Ain!sok,0; // enviar primer mensaje -> estado OK

}
```

Simulación

Para las pruebas de simulación se realizaron varias corridas cambiando cada vez el número de tramas a transmitir (*constante MAX*). Por ejemplo para una corrida con un total de 50 tramas a transmitir, se obtuvo la siguiente secuencia de intercambio de mensajes.

```
proc 0 = :init:
proc 1 = procesar
proc 2 = procesar
q\p   0   1   2
  1   Ain!sok,0
  1   .   chin?sok,0
  2   .   chout!msg,0
  2   .   .   chin?msg,0
  1   .   .   chout!msg,0
  1   .   chin?msg,0
  2   .   chout!msgack,0
  2   .   .   chin?msgack,0
  1   .   .   chout!ack,0
  1   .   chin?ack,0
  2   .   chout!by,0
  2   .   .   chin?by,0
  1   .   .   chout!sok,0
  1   .   chin?sok,0
      timeout
  2   .   chout!sok,0
```

```
2    .    .    chin?sok,0
1    .    .    chout!msg,0
1    .    chin?msg,0
     timeout
1    .    .    chout!sok,0
1    .    chin?sok,0
2    .    chout!msgack,0
2    .    .    chin?msgack,0
1    .    .    chout!ack,0
1    .    chin?ack,0
2    .    chout!msg,0
2    .    .    chin?msg,0
     timeout
2    .    chout!sok,0
2    .    .    chin?sok,0
     timeout
2    .    chout!sok,0
2    .    .    chin?sok,0
1    .    .    chout!msgack,0
1    .    chin?msgack,0
2    .    chout!ack,0
2    .    .    chin?ack,0
1    .    .    chout!by,0
1    .    chin?by,0
2    .    chout!sok,0
2    .    .    chin?sok,0
1    .    .    chout!msgack,0
1    .    chin?msgack,0
2    .    chout!ack,0
2    .    .    chin?ack,0
1    .    .    chout!msgack,0
1    .    chin?msgack,0
2    .    chout!ack,0
2    .    .    chin?ack,0
1    .    .    chout!msg,0
1    .    chin?msg,0
2    .    chout!msg,0
2    .    .    chin?msg,0
1    .    .    chout!msgack,0
1    .    chin?msgack,0
2    .    chout!ack,0
2    .    .    chin?ack,0
1    .    .    chout!msg,0
1    .    chin?msg,0
2    .    chout!msgack,0
2    .    .    chin?msgack,0
1    .    .    chout!ack,0
1    .    chin?ack,0
2    .    chout!msgack,0
```

```
2    .    .    chin?msgack,0
1    .    .    chout!ack,0
1    .    chin?ack,0
2    .    chout!msg,0
2    .    .    chin?msg,0
1    .    .    chout!msgack,0
1    .    chin?msgack,0
2    .    chout!ack,0
2    .    .    chin?ack,0
1    .    .    chout!msg,0
1    .    chin?msg,0
2    .    chout!rnr,0
2    .    .    chin?rnr,0
     timeout
2    .    chout!sok,0
2    .    .    chin?sok,0
1    .    .    chout!msg,0
1    .    chin?msg,0
     timeout
2    .    chout!sok,0
2    .    .    chin?sok,0
1    .    .    chout!msgack,0
1    .    chin?msgack,0
2    .    chout!ack,0
2    .    .    chin?ack,0
1    .    .    chout!by,0
1    .    chin?by,0
2    .    chout!sok,0
2    .    .    chin?sok,0
     timeout
1    .    .    chout!sok,0
1    .    chin?sok,0
2    .    chout!msg,0
2    .    .    chin?msg,0
     timeout
2    .    chout!sok,0
2    .    .    chin?sok,0
     timeout
2    .    chout!sok,0
2    .    .    chin?sok,0
1    .    .    chout!msg,0
1    .    chin?msg,0
     timeout
2    .    chout!sok,0
2    .    .    chin?sok,0
1    .    .    chout!msgack,0
1    .    chin?msgack,0
2    .    chout!ack,0
2    .    .    chin?ack,0
```

```
1    .    .    chout!msgack,0
1    .    chin?msgack,0
2    .    chout!ack,0
2    .    .    chin?ack,0
1    .    .    chout!msg,0
1    .    chin?msg,0
2    .    chout!msg,0
2    .    .    chin?msg,0
1    .    .    chout!rnr,0
1    .    chin?rnr,0
     timeout
2    .    chout!sok,0
2    .    .    chin?sok,0
1    .    .    chout!msgack,0
1    .    chin?msgack,0
2    .    chout!ack,0
2    .    .    chin?ack,0
1    .    .    chout!msg,0
1    .    chin?msg,0
     timeout
1    .    .    chout!sok,0
1    .    chin?sok,0
2    .    chout!msg,0
2    .    .    chin?msg,0
     timeout
1    .    .    chout!sok,0
1    .    chin?sok,0
2    .    chout!msg,0
2    .    .    chin?msg,0
     timeout
2    .    chout!sok,0
2    .    .    chin?sok,0
1    .    .    chout!msgack,0
1    .    chin?msgack,0
2    .    chout!ack,0
2    .    .    chin?ack,0
1    .    .    chout!by,0
1    .    chin?by,0
2    .    chout!sok,0
2    .    .    chin?sok,0
     timeout
1    .    .    chout!sok,0
1    .    chin?sok,0
     timeout
1    .    .    chout!sok,0
1    .    chin?sok,0
2    .    chout!msg,0
2    .    .    chin?msg,0
1    .    .    chout!msg,0
```

```
  1    .    chin?msg,0
  2    .    chout!rnr,0
  2    .    .    chin?rnr,0
       timeout
  2    .    chout!sok,0
  2    .    .    chin?sok,0
  1    .    .    chout!msg,0
  1    .    chin?msg,0
  2    .    chout!msgack,0
  2    .    .    chin?msgack,0
  1    .    .    chout!ack,0
  1    .    chin?ack,0
  2    .    chout!msgack,0
  2    .    .    chin?msgack,0
-------------
final state:
-------------
3 processes created
```

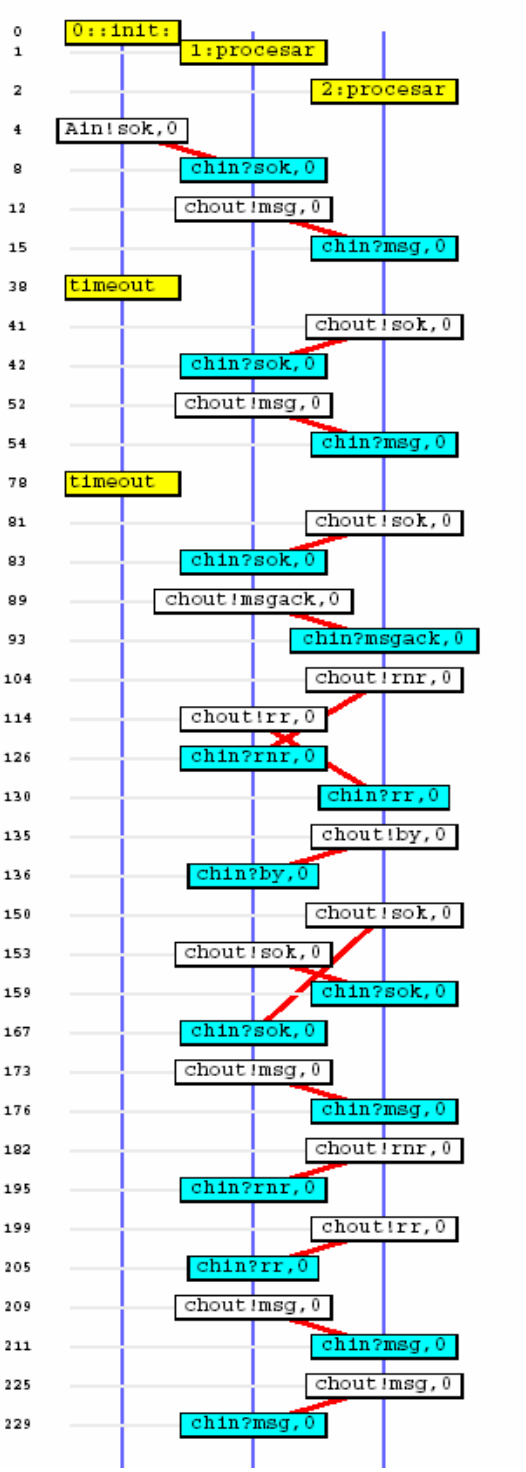
En la salida se observa que se enviaron/recibieron mensajes de todos los tipos, y se simuló incluso aquellos instantes donde el nodo no recibe/envía nada, es decir cuando los canales están vacíos (**timeout**).

Ejecutando **SPIN** en modo simulación y usando la opcion *–M*, se generó la secuencia de transferencia/recepción de los mensajes de manera gráfica. El comando ejecutado fue:

```
#spin –M nivel2
```

La salida obtenida de la simulación está en la Figura 39.

Figura 39. Simulación del Modelo 2



Validación

Para la verificación del *Modelo 2* usando **SPIN** en modo búsqueda exhaustiva, la secuencia de comandos utilizada fue:

```
#spin –a nivel2
#gcc –o pan pan.c
#time ./pan
```

Donde, el primer comando genera el programa analizador; el segundo compila el programa analizador (pan.*) indicando que la búsqueda de estados alcanzables será de manera exhaustiva; el tercer comando ejecuta el programa analizador utilizando la opción **time** para que, se genere el tiempo de ejecución junto con los resultado de la verificación.

El resultado de la corrida del programa analizador se muestra a continuación:

```
(Spin Version 4.2.5 -- 2 April 2005)
  + Partial Order Reduction

Full statespace search for:
  never claim          - (none specified)
  assertion violations +
  acceptance   cycles  - (not selected)
  invalid end states +

State-vector 48 byte, depth reached 285, errors: 0
  226827 states, stored
  165976 states, matched
  392803 transitions (= stored+matched)
       1 atomic steps
hash conflicts: 22460 (resolved)

Stats on memory usage (in Megabytes):
12.702 equivalent memory usage for states (stored*(State-vector +
overhead))
11.145 actual memory usage for states (compression: 87.74%)
  State-vector as stored = 41 byte + 8 byte overhead
2.097  memory used for hash table (-w19)
0.320  memory used for DFS stack (-m10000)
0.149  other (proc and chan stacks)
0.086  memory lost to fragmentation
13.476 total actual memory usage

unreached in proctype procesar
```

```
  (0 of 183 states)
unreached in proctype :init:
  (0 of 5 states)

real 0m0.9998s
user 0m0.820s
sys  0m0.70s
```

Entre los datos más relevantes de la salida se observan los siguientes:

- La búsqueda exhaustiva se detiene a una profundidad de 285 niveles.
- El tamaño del vector de estados del sistema es de 48 bytes.
- La cantidad de estados alcanzados es de 226827.
- La cantidad de memoria total utilizada fue de alrededor de 13.476 MB.
- En la parte final de la salida, se aprecia el tiempo de computo (m:s) empleado en la ejecución 0min: 0.9998s seg.

ANEXO D.

Código Fuente

La implementación en Promela del Modelo 3 del sub-nivel de enlace, se grabó en un archivo denominado "*nivel2a*" y su contenido se muestra a continuación:

```
/*
  Protocolo PDM-Ring : Nivel de Enlace

     Descripción: modelo del nivel de enlace que utiliza un solo
     nodo, verifica
          - la Tx y Rx de mensajes en un canal con ruido
          (distorsión de mensajes).
    - errores en las tramas con ACK (msgack).
    - retransmisión de tramas con ACK .

  fecha de creación: 10-12-2005
  ultima actualización: 11-12-2005
  nombre del archivo: nivel2a
  versión: 2.0.1

  autor: AGD & MER
*/

#define MAX  100 // cantidad de tramas a transmitir
#define MAX_RV 3 // cantidad de reenvíos
#define OK   0 // nodo OK
#define BUSY 1 // nodo ocupado
#define OUT  2 // nodo "fuera de servicio"

/*
  tipos de mensajes:
  msg - mensaje de datos sin confirmación
  msgack - mensaje de datos con confirmación
  ack - mensaje de confirmación de recepción de trama
  sok - mensaje estado de nodo OK
  by - mensaje de salida (fuera de servicio) controlada de nodo
     rnr - mensaje para informar que un nodo no se pueden recibir
     tramas.
     rr - mensaje para informar que un nodo ya puede recibir
     tramas.
*/
```

```
// definición de tipos de mensajes
mtype = {msg, msgack, ack, sok};

byte cont;     // contador de tramas enviadas
byte cont_reenvios;// contador de reenvíos
byte estado;   // estado actual del nodo

proctype procesar(chan chin, chout)
{
  byte o, i; // variables de envoi/recepcion de datos
  inicio:
  do
  ::(cont < MAX) -> // verificar contador de tramas
    if
    // recibir trama con confirmación
    ::chin?msgack,i ->
      do
      ::chout!ack,0 -> // trama con errores
        printf("cont_reenvios=%d\n",cont_reenvios);
        break
      ::chout!ack,11 -> break // trama sin errores
      od
    // recibir trama de confirmación
    ::chin?ack,i ->
      if
      :: (i == 0) ->
        do
        ::(cont_reenvios == 0) ->
          // reiniciar contador de reenvíos
          cont_reenvios = MAX_RV;
          printf("limite de reenvío de trama\n");
          goto inicio
        ::else ->
          // enviar trama con confirmación
          chout!msgack,11; // reenvío de trama
          // incrementar contador de reenvíos.
          cont_reenvios--; // incrementar conta
          break
        od
      :: else ->
        // enviar mensajes
        do
        ::chout!msgack,11 ->
          cont++;
          break
        ::chout!msg,MAX ->
          cont++;
          break
        ::chout!msgack,0 ->
```

```
      cont++;
      break
    od;
  fi
// recibir trama tipo sin confirmación
::chin?msg,i ->
  // enviar mensajes
  do
  ::chout!msgack,11 ->
    cont++;
    break
  ::chout!msg,MAX ->
    cont++;
    break
  ::chout!msgack,0 ->
    cont++;
    break
  od;
// si es un mensaje de estado OK
::chin?sok,i ->
  // enviar mensajes
  do
  ::chout!msg,o ->
    cont++; // incrementar contador tramas
    break
  ::chout!msgack,11 ->
    cont++;
    break
  ::chout!sok,o -> break
  ::chout!msgack,0 ->
    cont++;
    break
  od
// si nohay mensajes
:: timeout ->
  // enviar mensajes
  do
  ::chout!msg,o ->
    cont++;
    break
  ::chout!msgack,11 ->
    cont++;
    break
  ::chout!sok,o -> break
  ::chout!msgack,0 ->
    cont++;
    break
  ::skip -> goto inicio
  od
```

```
    fi
  :: else ->
    if
    ::(len(chin)>0) -> // mensajes pendientes por Rx
      if // recibir mensaje
      ::chin?msg,i ->
      ::chin?msgack,i ->
      ::chin?ack,i ->
      ::chin?sok,i ->
      fi
    ::else -> skip
    fi;
    break
  od
}

init{
  // definición de canales
  chan Ain = [1] of {mtype, byte}; // canal de entrada
  chan Aout = [1] of {mtype, byte}; // canal de salida

  atomic{
    run procesar(Ain,Aout);
    run procesar(Aout,Ain);
  }
  Ain!sok,0; // enviar primer mensaje – estado nodo
}
```

Simulación

Se realizaron varias pruebas de simulación de la recepción de tramas de confirmación con error, y se probó el reenvió de tramas usando un contador (*cont_reenvios*) con decremento. El error se indujo, usando la posibilidad que tienen los ciclos en Promela de seguir un flujo aleatorio, es decir a pesar que se recibía un mensaje de confirmación correcto, se simuló el hecho de recibir una confirmación con error. La salida de una de las pruebas de simulación se puede observar en la siguiente secuencia, donde se simuló el envió de 50 tramas, algunas de ellas con error.

```
proc 0 = :init:
proc 1 = procesar
proc 2 = procesar
q\p   0   1   2
  1   Ain!sok,0
  1   .   chin?sok,0
  2   .   chout!sok,0
```

```
2   .   .   chin?sok,0
1   .   .   chout!sok,0
1   .   chin?sok,0
2   .   chout!msg,0
2   .   .   chin?msg,0
1   .   .   chout!msgack,0
1   .   chin?msgack,0
2   .   chout!ack,11
2   .   .   chin?ack,11
1   .   .   chout!msgack,11
1   .   chin?msgack,11
2   .   chout!ack,0
2   .   .   chin?ack,0
        cont_reenvios=0
1   .   .   chout!msgack,11
1   .   chin?msgack,11
2   .   chout!ack,11
2   .   .   chin?ack,11
1   .   .   chout!msg,50
1   .   chin?msg,50
2   .   chout!msg,50
2   .   .   chin?msg,50
1   .   .   chout!msg,50
1   .   chin?msg,50
2   .   chout!msg,50
2   .   .   chin?msg,50
1   .   .   chout!msgack,0
1   .   chin?msgack,0
2   .   chout!ack,0
        cont_reenvios=1
2   .   .   chin?ack,0
1   .   .   chout!msgack,11
1   .   chin?msgack,11
2   .   chout!ack,0
        cont_reenvios=2
2   .   .   chin?ack,0
1   .   .   chout!msgack,11
1   .   chin?msgack,11
2   .   chout!ack,0
        cont_reenvios=3
2   .   .   chin?ack,0
            limite de reenvió de trama
    timeout
1   .   .   chout!msgack,11
1   .   chin?msgack,11
2   .   chout!ack,11
2   .   .   chin?ack,11
1   .   .   chout!msg,50
1   .   chin?msg,50
```

```
2   .   chout!msgack,0
2   .   .   chin?msgack,0
1   .   .   chout!ack,0
1   .   chin?ack,0
            cont_reenvios=0
2   .   chout!msgack,11
2   .   .   chin?msgack,11
1   .   .   chout!ack,0
            cont_reenvios=1
1   .   chin?ack,0
2   .   chout!msgack,11
2   .   .   chin?msgack,11
1   .   .   chout!ack,0
            cont_reenvios=2
1   .   chin?ack,0
2   .   chout!msgack,11
2   .   .   chin?msgack,11
1   .   .   chout!ack,11
1   .   chin?ack,11
2   .   chout!msg,50
2   .   .   chin?msg,50
1   .   .   chout!msg,50
1   .   chin?msg,50
2   .   chout!msgack,11
2   .   .   chin?msgack,11
1   .   .   chout!ack,11
1   .   chin?ack,11
2   .   chout!msg,50
2   .   .   chin?msg,50
1   .   .   chout!msgack,11
1   .   chin?msgack,11
2   .   chout!ack,11
2   .   .   chin?ack,11
1   .   .   chout!msg,50
1   .   chin?msg,50
2   .   chout!msg,50
2   .   .   chin?msg,50
1   .   .   chout!msgack,11
1   .   chin?msgack,11
2   .   chout!ack,11
2   .   .   chin?ack,11
1   .   .   chout!msg,50
1   .   chin?msg,50
2   .   chout!msg,50
2   .   .   chin?msg,50
1   .   .   chout!msg,50
1   .   chin?msg,50
2   .   chout!msgack,0
2   .   .   chin?msgack,0
```

```
1   .   .   chout!ack,11
1   .   chin?ack,11
2   .   chout!msgack,0
2   .   .   chin?msgack,0
1   .   .   chout!ack,0
1   .   chin?ack,0
            cont_reenvios=3
        limite de reenvio de trama
    timeout
    timeout
    timeout
    timeout
2   .   chout!msgack,11
2   .   .   chin?msgack,11
1   .   .   chout!ack,11
1   .   chin?ack,11
2   .   chout!msg,50
2   .   .   chin?msg,50
1   .   .   chout!msg,50
1   .   chin?msg,50
2   .   chout!msg,50
2   .   .   chin?msg,50
1   .   .   chout!msgack,0
1   .   chin?msgack,0
2   .   chout!ack,11
2   .   .   chin?ack,11
1   .   .   chout!msgack,0
1   .   chin?msgack,0
2   .   chout!ack,11
2   .   .   chin?ack,11
1   .   .   chout!msg,50
1   .   chin?msg,50
2   .   chout!msgack,0
2   .   .   chin?msgack,0
1   .   .   chout!ack,0
            cont_reenvios=0
1   .   chin?ack,0
2   .   chout!msgack,11
2   .   .   chin?msgack,11
1   .   .   chout!ack,0
1   .   chin?ack,0
            cont_reenvios=1
2   .   chout!msgack,11
2   .   .   chin?msgack,11
1   .   .   chout!ack,0
            cont_reenvios=2
1   .   chin?ack,0
2   .   chout!msgack,11
2   .   .   chin?msgack,11
```

```
1    .    .    chout!ack,11
1    .    chin?ack,11
2    .    chout!msgack,0
2    .    .    chin?msgack,0
1    .    .    chout!ack,11
1    .    chin?ack,11
2    .    chout!msg,50
2    .    .    chin?msg,50
1    .    .    chout!msgack,0
1    .    chin?msgack,0
2    .    chout!ack,0
2    .    .    chin?ack,0
          cont_reenvios=3
               limite de reenvio de trama
     timeout
2    .    chout!msgack,11
2    .    .    chin?msgack,11
1    .    .    chout!ack,11
1    .    chin?ack,11
2    .    chout!msgack,11
2    .    .    chin?msgack,11
1    .    .    chout!ack,11
1    .    chin?ack,11
2    .    chout!msg,50
2    .    .    chin?msg,50
1    .    .    chout!msg,50
1    .    chin?msg,50
2    .    chout!msgack,0
2    .    .    chin?msgack,0
1    .    .    chout!ack,11
1    .    chin?ack,11
2    .    chout!msgack,0
2    .    .    chin?msgack,0
1    .    .    chout!ack,0
               cont_reenvios=0
1    .    chin?ack,0
2    .    chout!msgack,11
2    .    .    chin?msgack,11
1    .    .    chout!ack,0
               cont_reenvios=1
1    .    chin?ack,0
2    .    chout!msgack,11
2    .    .    chin?msgack,11
1    .    .    chout!ack,0
               cont_reenvios=1
1    .    chin?ack,0
2    .    chout!msgack,11
2    .    .    chin?msgack,11
1    .    .    chout!ack,11
```

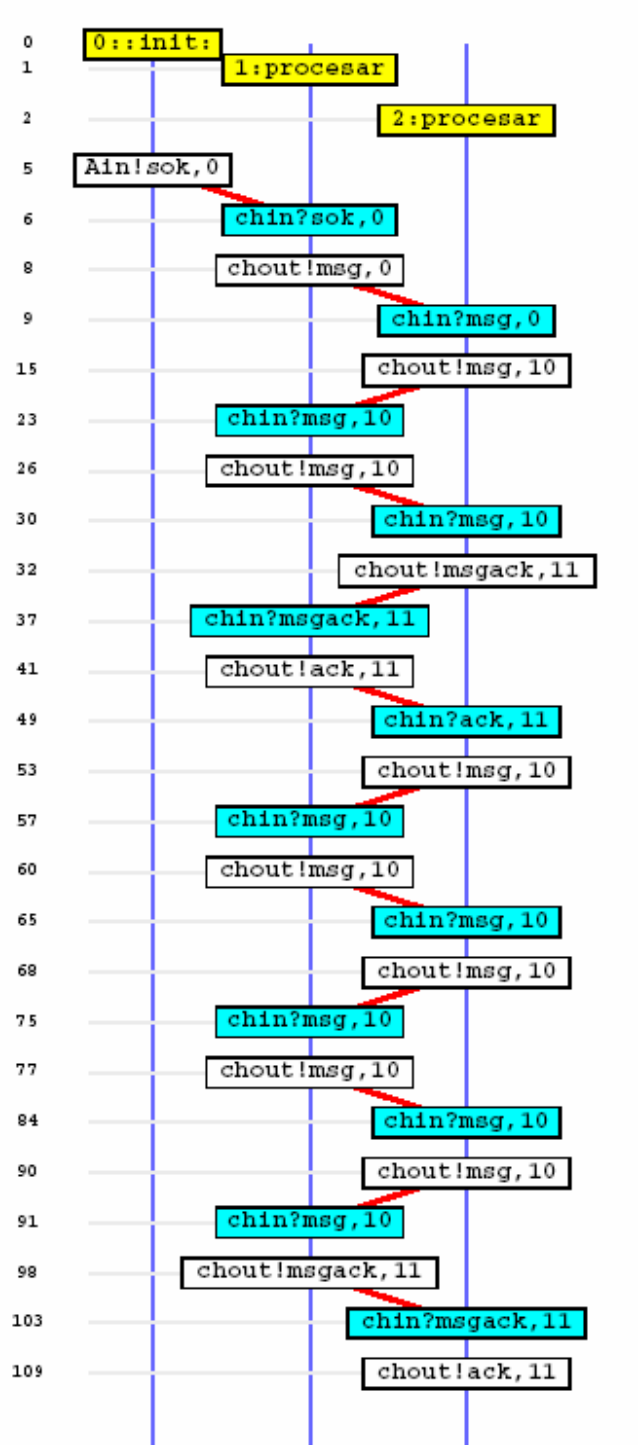
```
  1    .    chin?ack,11
  2    .    chout!msg,50
  2    .    .    chin?msg,50
  1    .    .    chout!msg,50
  1    .    chin?msg,50
  2    .    chout!msg,50
  2    .    .    chin?msg,50
  1    .    .    chout!msg,50
  1    .    chin?msg,50
  2    .    chout!msgack,11
  2    .    .    chin?msgack,11
  1    .    .    chout!ack,0
                 cont_reenvios=3
  1    .    chin?ack,0
            limite de reenvío de trama
       timeout
       timeout
  1    .    .    chout!msg,0
  1    .    chin?msg,0
  2    .    chout!msg,50
  2    .    .    chin?msg,50
  1    .    .    chout!msg,50
  1    .    chin?msg,50
  2    .    chout!msgack,11
  2    .    .    chin?msgack,11
-------------
final state:
-------------
3 processes created
```

Luego, se ejecutó **SPIN** en modo simulación con la opción *–M*, y se obtuvo la secuencia del intercambio de mensajes de manera gráfica:

```
#spin –M nivel2a
```

En la Figura 40 se puede apreciar la secuencia de intercambio de mensajes generada.

Figura 40. Simulación del Modelo 3



Validación

Para la verificación del *Modelo 3* usando **SPIN** en modo búsqueda exhaustiva, la secuencia de comandos utilizada fue:

```
#spin –a nivel2a
#gcc –o pan pan.c
#time ./pan
```

Donde, el primer comando genera el programa analizador; el segundo compila el programa analizador (pan.*) indicando que la búsqueda de estados alcanzables será de manera exhaustiva; el tercer comando ejecuta el programa analizador utilizando la opción **time** para que, se genere el tiempo de ejecución junto con los resultado de la verificacion.

El resultado de la corrida del programa analizador se muestra a continuación:

```
(Spin Version 4.2.5 -- 2 April 2005)
  + Partial Order Reduction

Full statespace search for:
  never claim          - (none specified)
  assertion violations +
  acceptance   cycles  - (not selected)
  invalid end states +

State-vector 44 byte, depth reached 2140, errors: 0
  120079 states, stored
   95661 states, matched
  215740 transitions (= stored+matched)
       1 atomic steps
hash conflicts: 11519 (resolved)

Stats on memory usage (in Megabytes):
6.244  equivalent memory usage for states (stored*(State-vector +
overhead))
5.007  actual memory usage for states (compression: 80.19%)
  State-vector as stored = 34 byte + 8 byte overhead
2.097  memory used for hash table (-w19)
0.320  memory used for DFS stack (-m10000)
0.136  other (proc and chan stacks)
0.092  memory lost to fragmentation
7.332  total actual memory usage

unreached in proctype procesar
```

```
   (0 of 100 states)
unreached in proctype :init:
   (0 of 5 states)

real 0m0.438s
user 0m0.400s
sys  0m0.040s
```

Entre los datos más relevantes de la salida se observan los siguientes:

- La búsqueda exhaustiva se detiene a una profundidad de 2140 niveles.
- El tamaño del vector de estados del sistema es de 44 bytes.
- La cantidad de estados alcanzados es de 120079.
- La cantidad de memoria total utilizada fue de alrededor de 7.332 MB.
- En la parte final de la salida, se aprecia el tiempo de computo (m:s) empleado en la ejecución 0min: 0.438 seg.