Capítulo 1. Motivación y Objetivos.

Objetivos

El objetivo de este proyecto es diseñar e implementar una herramienta software que permita a los alumnos de la asignatura Arquitectura de Redes Sistemas y Servicios profundizar en sus conocimientos sobre los protocolos de nivel de enlace.

Con este propósito se ha diseñado y desarrollado un simulador del nivel de enlace de finalidad claramente formativa. Gracias a este software los alumnos pueden analizar en profundidad el comportamiento de los protocolos de este nivel, variar los parámetros que influyen en los mismos así como observar el comportamiento de forma fácil gracias a la presentación de gráficas. En definitiva, se pretende que el alumno de ingeniería tenga una visión más amplia de la que le da exclusivamente el estudio de la teoría de estos protocolos.

El presente proyecto maneja varios conceptos que se describen muy brevemente a continuación:

Simulación: Las técnicas de simulación harán posible el estudio sistemático del nivel de enlace, ya que la experimentación directa sobre un sistema real no es factible. Permitirán un estudio cuantitativo como sustitución, o como fase previa, de la implementación del sistema real, basándose en técnicas estadísticas para la obtención y el análisis de los resultados. Aunque existen lenguajes específicos para realizar programas de simulación, se ha optado por un lenguaje de propósito general como es C.

Nivel de enlace: Es el nivel 2 del modelo de capas para la interconexión de sistemas abiertos. El servicio que ofrece la capa de nivel de enlace a las capas superiores es el de llevar a cabo una comunicación fiable y eficiente entre dos máquinas adyacentes, es decir, físicamente conectadas mediante un canal de comunicación que actúa como un cable.

Fases del proyecto

Las fases que se han llevado a cabo para la realización del presente proyecto fin de carrera han sido las siguientes:

1. Estudio detallado de los protocolos de nivel de enlace y de las técnicas de simulación.
2. Estudio del simulador de partida.
3. Desarrollo del simulador de protocolos de nivel de enlace.
4. Aprendizaje del lenguaje de programación Visual Basic.
5. Desarrollo de la interfaz gráfica del simulador de protocolos de nivel de enlace.
6. Depuración del simulador y de la interfaz gráfica y realización de una versión instalable para uso de los alumnos.
7. Detección de errores y validación teórica de los resultados.
8. Corrección de los últimos errores detectados y elaboración del presente documento.

Las dos primeras fases se llevaron a cabo simultáneamente, alcanzando una duración de aproximadamente dos meses, debido sobre todo a la dificultad de la asimilación del código del simulador de partida.

La fase 3, la más compleja, se prolongó durante unos 5 meses principalmente por las numerosas variaciones realizadas al código original. Durante la realización de esta fase también se llevaron a cabo las fases 4 y 5.

La fase 6, la más sencilla y corta duró alrededor de medio mes y dio paso a la fase 7 que se prolongó durante unos 2 meses debido a las numerosas pruebas realizadas tanto al simulador como a la interfaz gráfica.

La fase 8 ha durado alrededor de 6 meses. Hay que tener en cuenta que se realizó simultáneamente con la fase 7.

Material y Métodos

Para la realización del proyecto se ha contado con un PC con microprocesador AMD K7 a 1000MHz, con 128 Mb de RAM y disco duro 6 Gb en el que se tenía instalado el sistema operativo Microsoft Windows Milenium.

Para el desarrollo y depuración del código en lenguaje C se han utilizado las herramientas Cygwin, que son parte de las utilidades y herramientas de desarrollo del proyecto GNU de software libre. Las herramientas Cygwin son un conjunto de utilidades que emplean la librería Cygwin que proporciona las llamadas al sistema y el entorno Unix que estas utilidades necesitan. Estas utilidades son:

- Gcc versión 2.95: compilador de C y C++.
- Bloodshed Dev-C++ versión 4.0: entorno de desarrollo integrado en C y C++.
- Insight versión 5.0.92: herramienta gráfica de depuración C y C++.

Hay que reseñar que para la realización del código en C se han utilizado solo funciones contenidas en el estándar ANSI C de manera que el código resultante sea portable a otras plataformas distintas de Windows, como Unix o Linux.

Para el desarrollo y depuración del código en lenguaje Visual Basic se ha empleado el programa Microsoft Visual Basic. La información necesaria relativa a este lenguaje y algunos fragmentos de código utilizados en la interfaz gráfica se han extraído de:

- Microsoft Visual Basic 6.0 Manual del Programador.
- Área Visual Basic, http://www.ciberteca.net/visualbasic/default.asp
- El Guille, http://guille.costasol.net/indice.asp

Para la realización de la memoria se han utilizado:

- Suite ofimática OpenOffice.org 1.0.1:
  - OpenOffice.org Writer
  - OpenOffice.org Draw
  - OpenOffice.org Math
  - OpenOffice.org Calc

Contenido del documento

El contenido del presente documento, memoria del proyecto fin de carrera, se ha organizado según el siguiente guión:

1. Memoria.

   Capítulo 1. Motivación y objetivos.

   Descripción de proyecto, los objetivos buscados y de los materiales utilizados para llevarlo a cabo.

   Capítulo 2. Introducción teórica.

   Introducción a la capa de nivel de enlace y a los distintos tipos de protocolos que se pueden utilizar en la misma para realizar su cometido.

   Capítulo 3. Diseño e implementación del simulador.

   Descripción del simulador de partida y de como se han diseñado e implementado el actual simulador y su interfaz gráfica.

   Capítulo 4. Validación y pruebas.

   Muestra las pruebas realizadas al simulador y comprara sus resultados con los resultados obtenidos teóricamente.

   Capítulo 5. Conclusiones y líneas de continuación.

   Conclusiones del proyecto, del funcionamiento del simulador y la interfaz gráfica y de la líneas de mejora a seguir en futuras versiones.

2. Pliego de condiciones.

   Especificación de la estructura de los ficheros de entrada y salida del simulador así como de la ventana de introducción de los parámetros de la simulación perteneciente a la interfaz gráfica.

3. Planos.

   Código fuente de los programas, tanto en C como en Visual Basic, así como el makefile para su compilación e imágenes de los formularios.

4. Apéndices.

    a. Guía del Usuario

       Documento que explica como se han de instalar y utilizar el simulador y la interfaz gráfica para la realización de simulaciones y baterías de simulaciones así como la presentación de gráficas.

    b. Guía del Programador

       Documento que explica como se han programado el simulador y la interfaz gráfica con la intención de facilitar un futuro desarrollo de ambas.

    c. Enunciados de las prácticas

       Enunciados de las prácticas realizadas por los alumnos de la asignatura Arquitectura de Redes, Sistemas y Servicios utilizando el simulador de protocolos de nivel de enlace.

Capítulo 2. Introducción teórica.

El modelo OSI del ISO

El problema de construir una red de comunicaciones es demasiado complejo para manejarlo como una unidad. Esto implica que el trabajo debe dividirse en partes. Por tanto, antes de poder desarrollar estándares, debe definirse una estructura o arquitectura que defina las tareas de comunicación. Esta línea de razonamientos condujo a la ISO en 1977, a designar al comité técnico 97 como encargado de desarrollar tal arquitectura. El resultado de sus trabajos fue el modelo de referencia para la Interconexión de Sistemas Abiertos (OSI: Open Systems Interconnection) adoptado en 1983 por el CCITT como la recomendación X.200.

La ISO, Organización Internacional de Estandarización, fue fundada en 1946. Sus miembros son las organizaciones nacionales de normalización (AENOR, ANSI, BSI, AFNOR, DIN). Su objetivo es promover el desarrollo de estándares mundiales con vistas a facilitar el intercambio internacional de bienes y servicios. Está compuesto por 7 Grupos principales que asocian temas comunes (TAG), unos 200 Comités técnicos (TC) con subcomités (SC). Cada subcomité se divide en grupos de trabajo (WG).

El modelo OSI define en siete capas los protocolos de comunicación. Cada uno de los niveles tiene funciones definidas, que se relacionan con las funciones de las capas siguientes. Los niveles inferiores se encargan de acceder al medio, mientras que los superiores definen cómo las aplicaciones acceden a los protocolos de comunicación.

La siguiente ilustración muestra la relación entre el modelo OSI y el modelo TCP/IP, que se usa en Internet.

Ilustración 1. Comparación entre el modelo OSI y el TCP/IP

Los principios aplicados para el establecimiento de las capas son:

- ✔ Una capa se creará en situaciones en donde se necesite un nivel de diferencia abstracto.

- ✔ Cada capa deberá efectuar una función bien definida.
- ✔ La función que realizará cada capa, deberá seleccionarse con la intención de definir protocolos normalizados internacionalmente.
- ✔ Los límites de las capas, deberán seleccionarse tomando en cuenta la minimización del flujo de información, a través de las interfaces.
- ✔ El número de capas debe ser lo suficientemente grande, para que cada una no realice más de una función y lo suficientemente pequeña para que la arquitectura sea manejable.

Los siete niveles en los que el modelo OSI describe una arquitectura estratificada de red son:

1. Capa Física:

Se ocupa de la transmisión de los bits por el canal de comunicación. Esta es la encargada de que si un extremo envía un bit, con valor 0 ó 1, llegue al otro extremo de la misma manera.

2. Capa de Enlace:

La función de esta capa es, a partir de un medio de transmisión común, transformarlo en una línea sin errores de transmisión para la capa de red. Fracciona la entrada en tramas de datos y las transmite en forma secuencial. Establece los límites de la trama.

Cuando una trama es totalmente destruida por una ráfaga de ruido, la capa de enlace de la computadora emisora, se encarga de retransmitirla. También se encarga de resolver la duplicidad de tramas, debido a que se puede destruir el acuse de recibo de la misma.

3. Capa de Red:

Se ocupa de controlar las operaciones de las subredes, resuelve cómo enviar los paquetes del origen al destino. Controla la congestión en la red ocasionada por la presencia de muchos paquetes, debido a que esto puede llevar a un cuello de botella.

Esta capa resuelve los problemas de comunicación, que resultan de unir redes heterogéneas, causados por uniones de redes que manejan diferentes protocolos y tienen formas diferentes de direccionamientos. Por ejemplo, una red puede no querer recibir un mensaje por ser demasiado largo, esta capa lo soluciona.

4. Capa de Transporte:

La función de esta capa es aceptar los datos de la Capa de Sesión, dividirlos si es necesario y pasarlos a la Capa de Red y asegurarse que lleguen correctamente al destino.

Esta capa crea una conexión de red, distinta para cada conexión de transporte solicitada por la capa de sesión. Si el caudal es grande puede realizar más de una conexión para mejorarlo. Debido a que estas conexiones son costosas, esta capa puede multiplexar varias conexiones de transporte sobre la misma conexión de red, para abaratarlo.

La conexión más conocida es el canal punto a punto sin error, en el cual se entregan los mensajes en el mismo orden que fueron enviados. Otra forma del servicio de transporte es el envío de mensajes aislados, que no garantizan el orden de difusión, ni la distribución de mensajes a destinos múltiples.

La capa de transporte se encarga de establecer y liberar conexiones en la red.

5. Capa de Sesión:

Permite que usuarios en distintas computadoras establezcan una sesión entre ellos, a través de la misma se puede llevar a cabo un transporte de datos, tal como lo hace la capa de transporte. La mejora de los servicios, le permite al usuario acceder a un sistema de tiempo compartido a distancia o transferir un archivo.

Servicios de esta capa:

- controlar el dialogo: las sesiones permiten que el tráfico se realice en ambas direcciones o en una sola en un momento dado. Cuando se realiza en un solo sentido, esta capa ayudará en el seguimiento de quién tiene el turno.

- administración de testigo: esto es para que en algunos protocolos los dos extremos no quieran transmitir al mismo tiempo, de esta forma sólo lo hace el que posee el testigo (token).

- sincronización: esta capa proporciona la inserción de puntos de verificación para el control de flujo. Por ejemplo, si dos computadoras desean transmitir un archivo que lleva dos horas y al cabo de una hora se interrumpen las conexiones de red, la transmisión no se debe desarrollar nuevamente desde el principio, ya que con el servicio que brinda esta capa sólo se transmite lo posterior al último punto de verificación.

6. Capa de Presentación:

Esta capa se ocupa de la sintaxis o representación de la información. Sus funciones incluyen el formateo de datos, la codificación, la traducción entre conjuntos de caracteres y códigos, así como la selección de los mecanismos de despliegue de la información en pantalla.

7. Capa de Aplicación:

Sobre la capa de transporte se encuentra esta capa. Contiene los programas de los usuarios (aplicaciones). Las aplicaciones más comunes son: transferencia de archivos (FTP), acceso de archivos remotos (TELNET) o cuando dos personas trabajan sobre computadoras distintas, para un mismo proyecto.

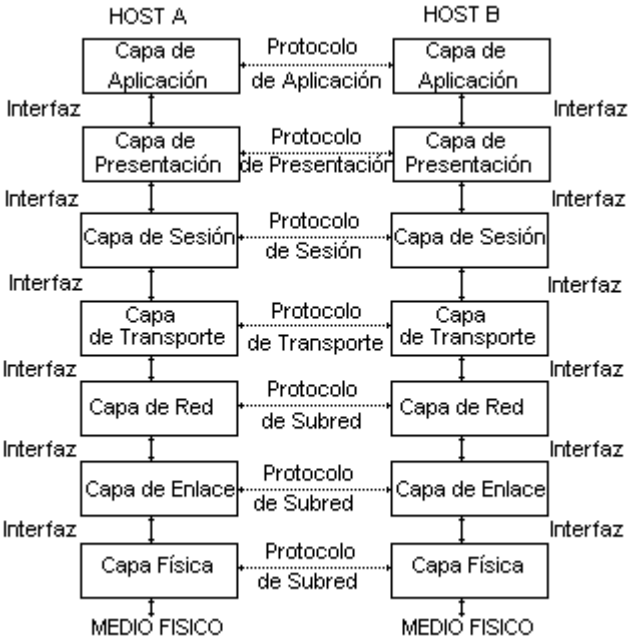

Ilustración 2. Modelo de capas OSI

Es importante observar que OSI describe las funciones que cada nivel debe ejecutar, pero no la forma en que serán llevadas a cabo dichas funciones, es decir, los detalles de implantación no se contemplan en el modelo.

La capa de enlace

El objetivo fundamental de la capa de enlace de datos es proporcionar una comunicación confiable y eficiente entre dos máquinas adyacentes.

La capa de enlace de datos tiene que desempeñar varias funciones específicas que incluyen proporcionar una interfaz de servicio bien definida con la capa de red, determinar la manera que los bits de la capa física se agrupan en tramas, manejar los errores de transmisión y regular el flujo de tramas para que los receptores no sean abrumados por los transmisores rápidos, todo ello utilizando eficientemente el ancho de banda disponible.

Principales problemas a solucionar por los protocolos de nivel de enlace de datos:

- los errores en los circuitos de comunicación,
- sus velocidades finitas de transmisión,
- el tiempo de propagación,
- uso eficiente del ancho de banda,
- uso eficiente de la capacidad de procesado y de la memoria.

La comunicación ente capas de enlace de datos de máquinas adyacentes se lleva a cabo mediante el intercambio de tramas (frames). Una trama se divide conceptualmente en dos partes claramente diferenciadas: cabecera e información.

Existen dos tipos fundamentales de tramas, de datos y de control. La tramas de datos están compuestas por la parte de cabecera y por la parte de información mientras que las tramas de control solo están compuestas por la parte de cabecera, no apareciendo la parte de información.

La parte de cabecera está compuesta por campos que contienen información de control. La cabecera de las tramas del simulador se compone de los campos kind, seq, ack y checksum, cuyo su significado se especificará más adelante.

La parte de información de una trama puede contener datos que intercambian capas de enlace pares o puede contener un paquete (packet) que es la unidad de información intercambiada ente la capa de red y la de enlace de datos en la misma máquina, o ente capas de red pares.

Relación entre paquete y trama: la capa de red construye un paquete tomando un mensaje de la capa de transporte y agregándole el encabezado de la capa de red, este paquete se pasa a la capa de enlace de datos para su inclusión en el campo info de una trama de salida. Cuando la trama llega a su destino, la capa de enlace de datos extrae el paquete de la trama y lo pasa a la capa de red.

Protocolos elementales del nivel de enlace.

A continuación se van a presentar las distintas familias de protocolos de nivel de enlace. Se va a comenzar presentando un marco inicial ideal donde actuaría el primer protocolo, el más sencillo. Posteriormente se irán añadiendo supuestos a la situación inicial y se irán presentando otras familias de protocolos se solventen los problemas de la nueva situación.

Suposiciones generales iniciales, la situación más favorable:

- ✔ En las capas física, de enlace de datos y de red hay procesos independientes que se comunican pasando mensajes de un lado a otro.
- ✔ El nodo A quiere mandar un flujo de datos a B usando un servicio confiable orientado a la conexión. Después se considerará el caso en el que B también quiere mandar datos a A simultáneamente.
- ✔ A tiene un suministro infinito de datos listos para ser enviados y nunca tiene que esperar a que se produzcan datos. Después se desechará esta suposición.
- ✔ El paquete pasado a la capa de enlace de datos a través de la interfaz desde la capa de red es de datos únicamente, que deben ser entregados bit por bit a la capa de red del destino.
- ✔ Las capas de enlace de datos están dedicadas a tiempo completo a manejar nuestro canal.
- ✔ El canal nunca pierde ni daña las tramas.
- ✔ El hardware transmisor calcula y agrega la suma de comprobación (campo checksum) y el hardware receptor la comprueba (necesario cuando eliminamos la condición precedente).
- ✔ En ninguna circunstancia se entrega un encabezado de trama de la capa de enlace a la capa de red, ya que, una interfaz rígida entre ambas capas simplifica en gran medida el software al lograr que las diferentes capas puedan evolucionar independientemente. Esta es una condición de los modelos de capas en general.
- ✔ El canal de comunicación entrega los bits con exactitud en el mismo orden en que fueron enviados.

1 Protocolo unilateral no restringido. Utopía.

Características propias:

- ✔ Es lo más sencillo posible.
- ✔ Los datos se transmiten en una sola dirección.
- ✔ Las capas de red siempre están listas.

- ✔ El tiempo de procesamiento puede ignorarse.
- ✔ Hay espacio infinito de buffer.
- ✔ El canal de comunicación nunca tiene problemas ni pierde tramas.
- ✔ No se usan número de secuencia ni acuses de recibo.
- ✔ El único evento posible es la llegada de una trama.

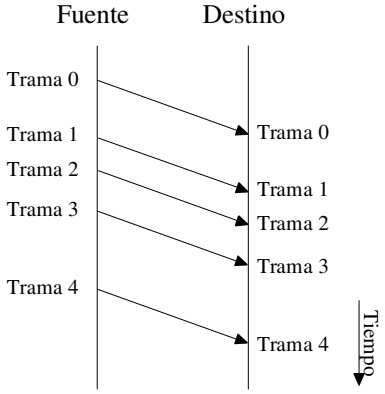

Ilustración 3. Protocolo unilateral no restringido

2 Protocolo unilateral de parada y espera.

Cambios en los supuestos:

- ✗ El receptor no posee una capacidad infinita de buffer en la cual almacenar todas las tramas de entrada. Se debe prevenir que el remitente sature al receptor.

Soluciones adoptadas:

- ✔ El receptor ha de proporcionar realimentación al transmisor, para ello manda un acuse de recibo después de cada trama recibida; el remitente tiene que esperar el acuse de recibo (un protocolo de parada y espera).
- ✔ El medio tiene que ser semi-duplex aunque el servicio todavía es simplex.

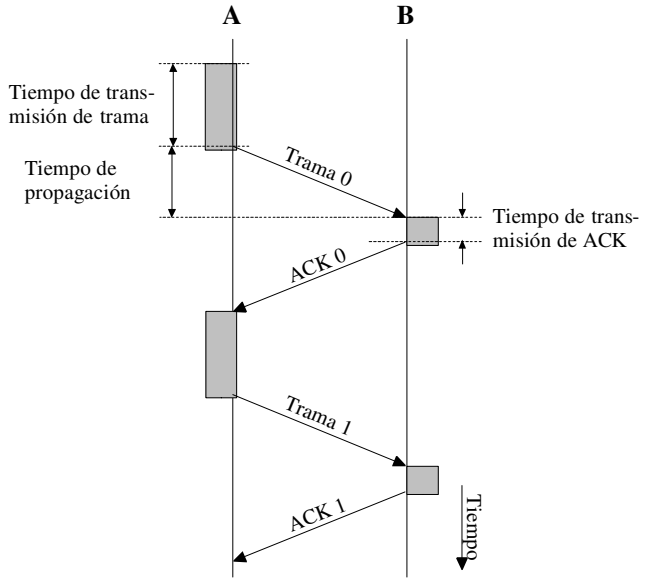

Ilustración 4. Protocolo unilateral de parada y espera

3 Protocolo unilateral para un canal ruidoso.

Cambio:

- ✗ El canal de comunicación comete errores, se pueden dañar o perder las tramas.

Soluciones adoptadas:

- ✔ Añadir un temporizador al transmisor. El transmisor manda una trama, y el receptor manda un acuse de recibo solamente si los datos fueron recibidos correctamente (se descartan las tramas dañadas). Después de algún tiempo el transmisor manda la trama de nuevo si no recibe el acuse de recibo.

- ✔ El transmisor inserta un número de secuencia en el encabezamiento de la trama. Esto permite que el receptor pueda distinguir entre tramas nuevas y duplicadas.

- ✔ Solo necesitamos distinguir entre una trama y la siguiente. Entonces se necesita solamente un bit en el campo de secuencia para distinguir entre las dos posibilidades.

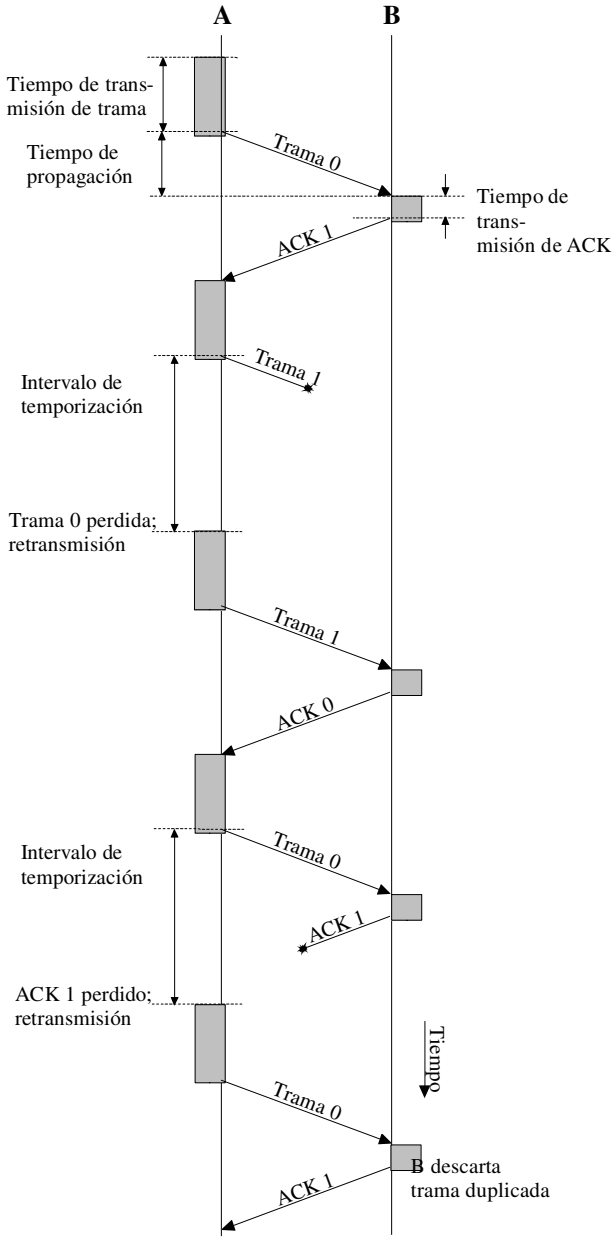

Ilustración 5. Protocolo unilateral para un canal ruidoso

4 Protocolos de ventana deslizante.

Cambios:

- ✗ El canal de transmisión es dúplex o full-dúplex. El hecho de mantener una comunicación simplex sobre este tipo de canal malgasta la mitad del ancho de banda.

- ✗ Las tramas de datos y los acuses de recibo pueden fluir en ambas direcciones. Podemos usar un campo en el encabezamiento de cada trama que especifica su clase (datos o acuse de recibo).

- ✗ Problemas con los temporizadores. ¿Por cuánto tiempo debería esperar el nivel de enlace a un paquete de datos antes de mandar un asentimiento?

Soluciones:

- ✔ Cuando una trama de datos llega, el receptor no manda inmediatamente un acuse. En vez de esto, espera hasta que su nivel de red le pase un paquete de datos. Cuando el paquete está listo, se añade el acuse para la trama recibida antes. Entonces, se combinan en una sola trama un acuse y un paquete de datos. Se llama este proceso piggybacking. Ventajas principales de piggybacking:

- ✔ Mejor uso del ancho de banda. El campo de acuse de recibo en un encabezamiento de trama de datos es solamente de unos pocos bits, mientras que una trama que es solamente un acuse consiste en esos bits, un encabezamiento, y un checksum.

- ✔ Menos tramas mandadas implican menos interrupciones de llegada de trama en el receptor, y quizás menos buffers en el receptor también.

- ✔ Para corregir los problemas surgidos con temporizadores demasiado cortos o largos, surge una nueva clase de protocolos que son más robustos, son los protocolos de ventana deslizante.

- ✔ En estos protocolos cada trama tiene un número de secuencia con un rango de 0 hasta algún n. En cualquier instante, el emisor mantiene un conjunto de números de secuencia que corresponden a las tramas que puede mandar (las tramas dentro de la ventana del emisor). El receptor mantiene una ventana del receptor que es el conjunto de tramas que puede aceptar. Las dos ventanas pueden tener tamaños distintos.

- ✔ Los números de secuencia en la ventana del emisor representan las tramas mandadas pero cuyos acuses no se han recibido. Cuando un paquete llega desde el nivel de red, se le asigna el próximo número de secuencia (modulo n)

y la ventana crece en uno. Cuando un acuse llega, la ventana disminuye en uno. De esta manera la ventana mantiene siempre la lista de tramas sin acuses.

- ✔ Ya que es posible que se pierdan o se dañen las tramas en la ventana del emisor, el emisor los tiene que guardar para una retransmisión eventual. Con una ventana de tamaño n el emisor necesita n buffers. Si la ventana crece a su tamaño máximo, el nivel de enlace de emisor no puede aceptar paquetes nuevos desde el nivel de red hasta que un buffer está vacío.

- ✔ En el receptor la ventana siempre mantiene el mismo tamaño y representa las tramas que se pueden aceptar. El receptor tiene un buffer para cada posición en la ventana; el propósito de estos buffers es permitir que se pueden entregar las tramas en orden al nivel de red aun cuando las tramas llegan en un orden distinto (debido a retransmisiones, marcos perdidos, etc.). Se descartan las tramas que llegan con números de secuencia fuera de la ventana.

5 Protocolos de ventana deslizante de un bit.

Características propias:

- ✔ El caso más sencillo es un protocolo de ventana deslizante con ventanas de tamaños máximos de 1.

- ✔ Cuando llega un marco cuyo número de secuencia es igual al número único en la ventana, se genera un acuse de recibo, se entrega el paquete a nivel de red, y se desplaza la ventana arriba en uno (es decir, invertir el número en la ventana).

- ✔ Este protocolo es de tipo parada y espera, ya que el emisor transmite un marco y espera su acuse antes de mandar el próximo.

- ✔ Ninguna combinación de marcos perdidos o terminaciones prematuras puede hacer que el protocolo entregue paquetes duplicados a cualquiera de las capas de red, ni que omita un paquete, ni que entre en un interbloqueo.

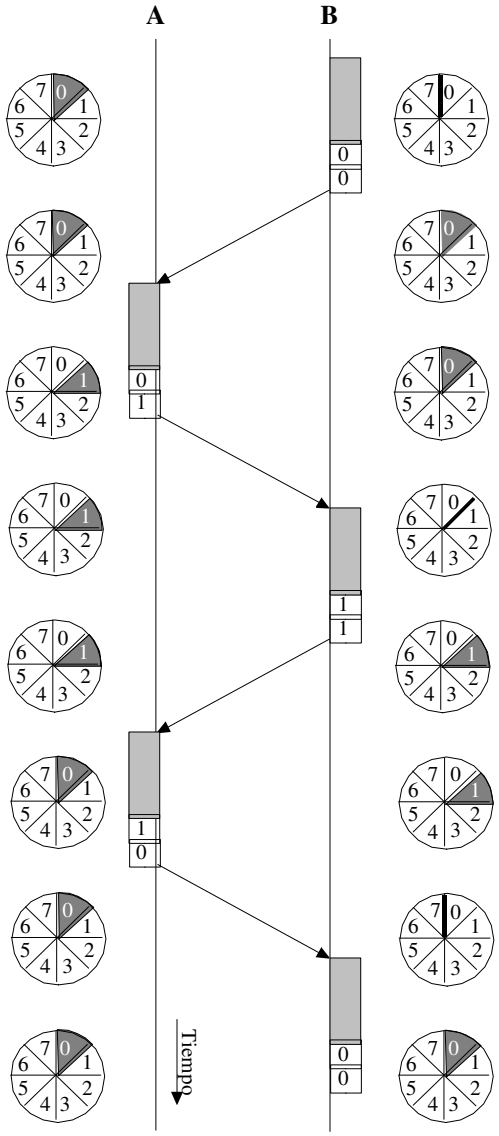

Ilustración 6. Protocolo de ventana deslizante de un bit

6 Protocolos de repetición no selectiva.

Cambios:

- El tiempo de transmisión para que una trama llegue al receptor más el necesario para que el acuse regrese no es insignificante. ¿Qué ocurre con el anterior protocolo cuando el tiempo de ida y vuelta es largo (por ejemplo, con una conexión de satélite)? El emisor malgasta ancho de banda esperando los acuses de recibo. Ejemplo:

- Canal de satélite de 50 kbps con una latencia de 250 ms. y marcos de 1000 bits.
- En t=0 mandamos un marco. En t=20 ms. es completamente mandado.
- Llega completamente en t=270. El acuse llega en t=520.
- El emisor espera 500/520 = 96% del tiempo.

✗ La capa de red no tiene un suministro infinito de paquetes que enviar.

✗ La capa de enlace de datos puede bloquear a la capa de red para que ésta no le pase mas paquetes.

Soluciones:

✔ Entubamiento (pipelining). El emisor puede mandar hasta n marcos sin esperar. Si n es suficiente grande los acuses llegarán siempre antes de que el emisor haya mandado n marcos (es decir, el tamaño de su ventana). Así, solucionamos el problema de malgastar el ancho de banda.

✔ Con el pipelining tenemos un problema si se daña o se pierde un marco en la mitad del flujo. Muchos marcos llegarán en el receptor antes de que el emisor sepa que hay un error. Cuándo el receptor nota un marco dañado o faltando, ¿qué debería hacer con los marcos correctos que siguen?

✔ La estrategia que sigue este protocolo se denomina repetir n. En esta estrategia el receptor descarta los marcos que siguen y no manda ningún acuse. Es decir, el receptor tiene una ventana de recepción de tamaño 1 y no acepta ningún marco excepto el próximo que le debe dar a nivel de red. Con tiempo el emisor notará la ausencia de acuses y retransmitirá los marcos en orden. Este enfoque malgasta ancho de banda si la tasa de errores es alta.

✔ Cuando la capa de red tiene un paquete que enviar, puede causar la ocurrencia de un evento capa de red lista.

✔ Cuando la capa de enlace de datos tiene todos los buffer de transmisión ocupados con paquetes sin acusar puede inhabilitar a la capa de red para que esta no la moleste con más trabajo.

✔ A la llegada de un acuse, la capa de enlace de datos ve si puede liberar buffer. Si esto es posible, ya puede permitir que una capa de red previamente bloqueada produzca más eventos capa de red lista, por lo que la desbloquea.

✔ Hay múltiples temporizadores, uno por cada marco pendiente de acuse.

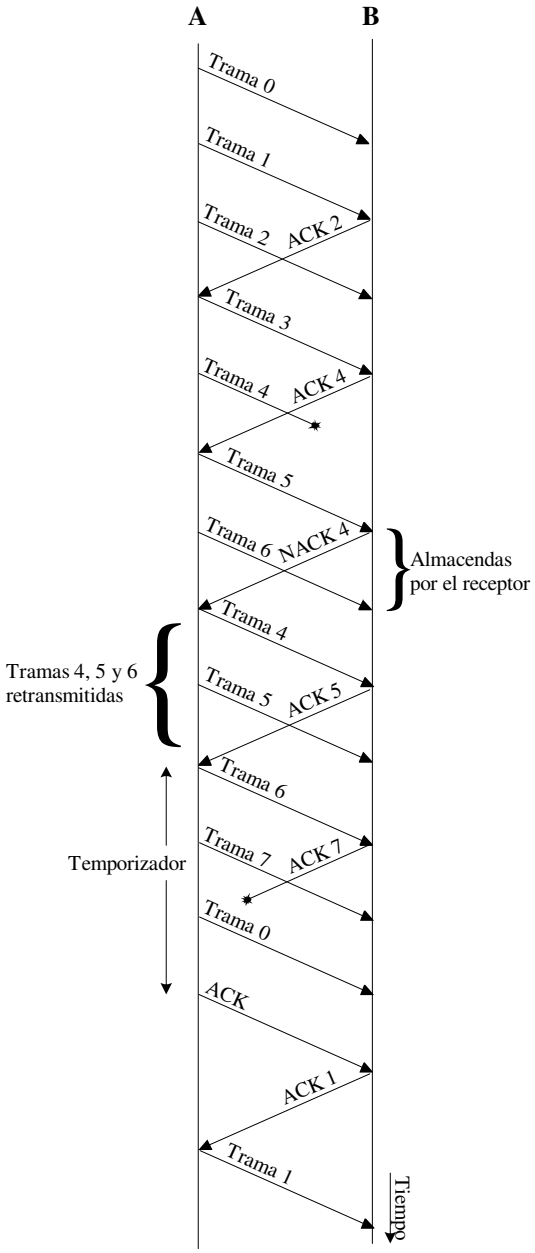

Ilustración 7. Protocolo de rechazo no selectivo

- ✔ El número de secuencia va de 0 a MAX_SEQ, es decir, hay MAX_SEQ+1 números de secuencia posibles. El número máximo de marcos pendientes de acuse debe restringirse a MAX_SEQ.

- ✔ Este protocolo funciona bien si los errores son poco frecuentes, pero si la línea es mala se desperdicia mucho ancho de banda en los marcos retransmitidos.

7 Protocolos de repetición selectiva.

Cambios:

- ✗ El canal de comunicación presenta numerosos errores.
- ✗ La ventana de recepción deja de ser de tamaño 1.
- ✗ El canal no está fuertemente cargado, por lo que no puede realizarse a menudo el piggybacking, ya que habría que esperar mucho tiempo la salida de un nuevo paquete para incorporarle el acuse, provocando que salten los temporizadores del transmisor y la consiguiente retransmisión innecesaria de paquetes.

Soluciones:

- ✔ Estrategia repetir selectivamente. En esta estrategia el receptor guarda los marcos correctos después del malo. Cuando el emisor nota que falta un acuse, solamente retransmitirá el marco malo. Si esta retransmisión tiene éxito, el receptor tendrá muchos marcos correctos en secuencia y le podrá pasar al nivel de red y mandar un acuse de recibo para el número de secuencia más alto. Este enfoque requiere una ventana del receptor con un tamaño más de 1, y su costo es el espacio para los buffers.
- ✔ El tamaño de la ventana debiera ser la mitad del número de números de secuencia. ¿Por qué? El receptor tiene que poder distinguir entre marcos duplicados y marcos nuevos.
- ✔ Asociado a cada buffer de recepción hay un bit que indica si el buffer está lleno o vacío. El número de buffers de recepción es igual al tamaño de la venta.
- ✔ Tras llegar un marco de datos en secuencia, se arranca un temporizador auxiliar. Si no se ha presentado tráfico en sentido inverso antes de terminar este temporizador, se envía un marco de acuse independiente.
- ✔ Es indispensable que el lapso asociado al temporizador auxiliar sea notablemente más corto que el del temporizador usado para temporizar marcos de datos.
- ✔ Para conseguir una mayor eficiencia en el manejo de errores se usan acuses negativos de recibo (NACKs), los cuales el receptor puede usar para acelerar la retransmisión de marcos.

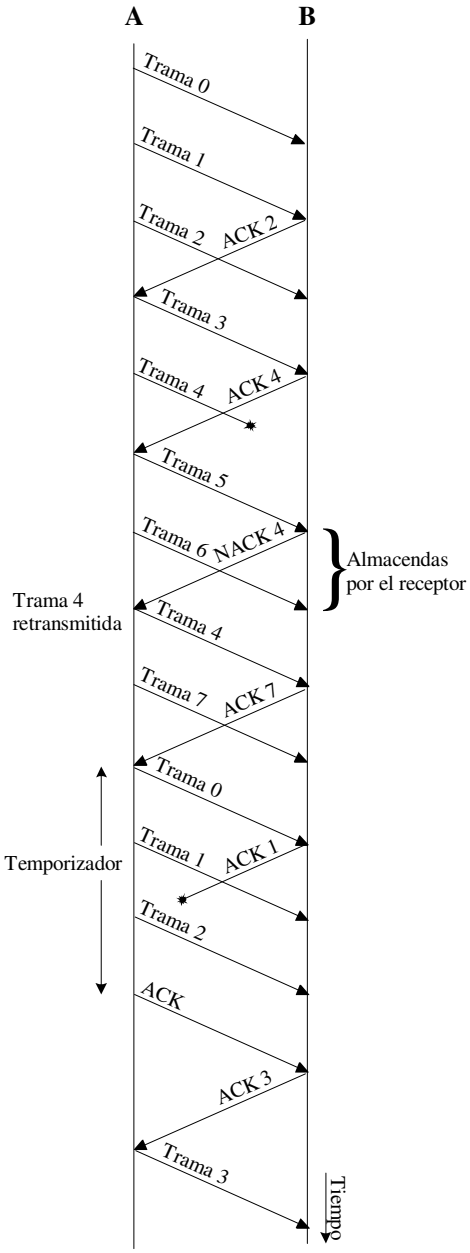

Ilustración 8. Protocolo de rechazo selectivo

Relaciones entre los protocolos.

A continuación se exponen las relaciones que se dan entre los protocolos expuestos anteriormente y como partiendo del protocolo más potente (el de ventana deslizante con repetición selectiva) pueden llegar a obtenerse los demás modificando ligeramente el funcionamiento o símplemente ajustando algunos parámetros.

La descripción de estas relaciones no se realiza desde el punto de vista del código, sino más bien se pretenden enumerar las diferencias entre los protocolos de manera que en un posterior paso se puedan plasmar estas diferencias conceptuales en diferencias en el código.

Estas relaciones se van a hacer en cascada. Es decir, una vez conseguido el protocolo de rechazo no selectivo a partir del protocolo de rechazo selectivo, para conseguir el protocolo de ventana de un bit se puede partir del protocolo de rechazo no selectivo, que ya tenemos, y que es más parecido el protocolo de ventana de un bit que el protocolo de rechazo selectivo.

1 Paso del protocolo de rechazo selectivo al de rechazo no selectivo:

- La ventana de recepción es de tamaño 1.
- No hay temporizadores auxiliares, o éstos no expiran.
- No se emplean NACKs.
- El tamaño máximo de la ventana de transmisión es MAX_SEQ y no (MAX_SEQ+1)/2.
- Cuando se produce un evento timeout se retransmiten todos los paquetes no asentidos y no sólo el más viejo.

2 Paso del protocolo de rechazo no selectivo al de ventana de un bit:

- La ventana de transmisión es 1, como la de recepción.
- Envío de un primer marco antes de entrar en el bucle infinito.
- Ahora la capa de red siempre tiene paquetes que transmitir, luego no hace falta esperar el evento network_layer_ready, sino que cada vez que llega un ACK se recoge un nuevo paquete de la capa de red.
- Para que no se produzcan eventos network_layer_ready se puede mantener la capa de red bloqueada todo el tiempo.
- El tratamiento del evento timeout es el mismo, pues solo puede haber un marco sin acusar ya que la ventana de transmisión es de 1.

3 Paso del protocolo de ventana de un bit al unilateral para canal ruidoso.

- El flujo de datos es ahora unidireccional por lo que el receptor no cogerá paquetes de su nivel de red, y sólo mandará ACKs.
- Como el receptor no manda tramas de datos, cuando le llega una trama no ha de preocuparse por el contenido del campo ack, pero sí mantiene el procesado del campo seq.
- Las tramas de acuse no contienen ninguna información, ni siquiera un número de secuencia.

4 Paso del protocolo unilateral para canal ruidoso al de parada y espera.

- El canal está libre de errores, nunca se produce el evento chksum_err.
- No se usan números de secuencia por lo que no se tienen en cuenta los campos ack y seq.
- No se usan los temporizadores o estos expiran en un tiempo infinito o suficientemente grande.

5 Paso del protocolo unilateral de parada y espera al no restringido.

- El receptor no envía acuses de recibo, tan solo espera el único evento posible, frame_arrival, lo procesa como antes y vuelve a esperar.
- El emisor tan solo coge un nuevo paquete del nivel de red, forma el nuevo marco, lo envía y vuelve a comenzar.

Parámetros de prestaciones de los protocolos

Para poder comparar unos protocolos con otros, o simplemente, para ver las prestaciones reales que nos proporcionan los protocolos de nivel de enlace, hemos de definir una serie de parámetros. Estos parámetros son:

- ➔ Caudal eficaz o cadencia eficaz (C_{ef}): la relación entre el número de bits de usuario (n) transmitidos y el tiempo de ocupación (T_{oc}), tiempo que se ha mantenido ocupado el canal para que el destinatario recibiese los datos sin error.

$$Cef = \frac{n}{T_{oc}}$$

- ➔ Rendimiento (R): la relación entre el número de bits de usuario (n) transmitidos y el número total de bits totales que se podían haber transmitido, o lo que es lo mismo, la relación entre la cadencia eficaz y el régimen binario (R_b) nominal del canal.

$$R = {}^{n}\!/_{n_{posibles}} = {}^{C_{ef}}\!/_{R_b}$$

- ➔ Utilización o uso del canal: porcentaje de tiempo durante el cual el canal ha estado ocupado. Este parámetro da una idea de si el protocolo puede obtener un mejor rendimiento ante una situación determinada.

- ➔ Retardo de ida y vuelta (RTD): mínimo tiempo transcurrido desde que comienza a transmitirse el primer bit de la trama de datos hasta que termina de recibirse el último bit de la trama de asentimiento correspondiente a esa trama de datos, se obtiene cuando no se produce ningún error en el enlace.

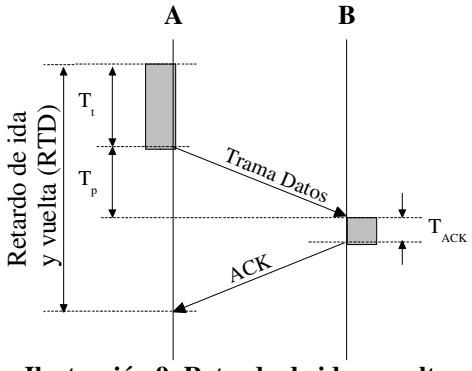

Ilustración 9. Retardo de ida y vuelta

➔ Retardo medio de ida y vuelta (RMD): tiempo medio transcurrido desde que comienza a transmitirse el primer bit de la trama de datos hasta que termina de recibirse el último bit de la trama de asentimiento correspondiente a esa trama de datos a lo largo de una simulación completa. Según el número de errores sufridos por una trama hasta su asentimiento correcto, así será su retardo de vuelta. El RMD es la media de ese retardo para todas las tramas de datos.

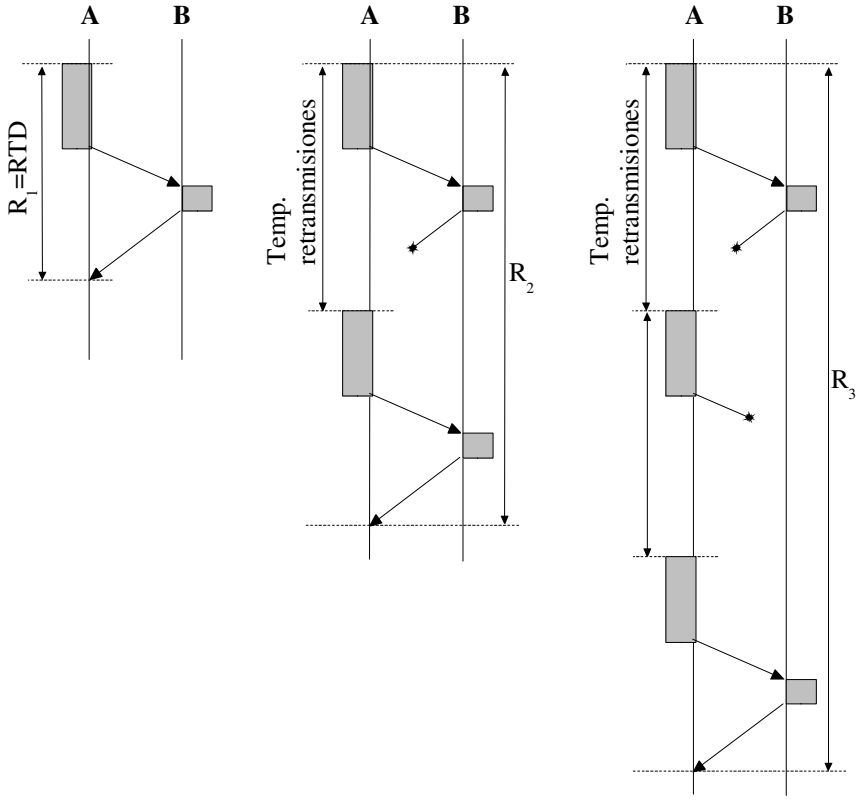

Ilustración 10. Retardo medio de ida y vuelta

Estos últimos parámetros cumplen una relación entre sí: el RMD coincide con el RTD cuando no se producen errores, es decir, el RTD es una cota mínima del RMD.

$$RMD \geqslant RTD$$

Pero hay que diferenciarlos bien, pues el RMD es un parámetro estadístico de una simulación mientras que el RTD es un valor fijo para un protocolo.

Capítulo 3. Diseño e implementación del simulador.

El simulador de partida

1 Aspectos generales

El simulador de partida sobre el que se inició el presente proyecto simula los protocolos presentes en el capítulo 3 del libro "Redes de ordenadores. Tercera edición" de Andrew S. Tanenbaum, publicado por Prentice Hall PTR en 1996. El simulador fue programado por Andrew S. Tanenbaum y es de libre distribución.

El simulador está escrito en lenguaje C y es un simulador orientado a eventos. Consiste en dos ficheros de código, sim.c y worker.c, además de un fichero de cabeceras común. También utiliza cinco protocolos de p2.c a p6.c, que son los protocolos elementales explicados en el capítulo 2 con algunos cambios hechos para que funcione el simulador. Los protocolos utilizan el archivo de cabeceras protocol.h. Todos los protocolos son compilados dentro del fichero binario, sim.exe.

El simulador utiliza tres hilos independientes:

- Main: controla la simulación
- M0: nodo 0 (nodo que transmite en los protocolos 2 y 3)
- M1: nodo 1 (nodo que recibe en los protocolos 2 y 3)

Hay seis pipes para que los tres hilos puedan comunicarse unos con otros. Los descriptores de ficheros creados se llaman como sigue:

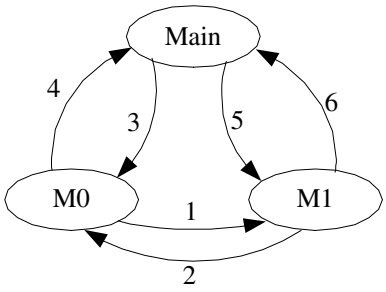

Ilustración 11. Esquema de hilos y pipes del simulador de partida

- Comunicación M0 – M1:
  - w1, r1: para tramas de M0 a M1
  - w2, r2: para tramas de M1 a M0

- Comunicación Main – M0:
    - w3, r3: señal de continuación de main a M0
    - w4, r4: señal de preparado de M0 a Main

- Comunicación Main – M1:
    - w5, r5: señal de continuación de main a M0
    - w6, r6: señal de preparado de M0 a Main

2 El hilo principal: Main

El archivo sim.c contiene el código del hilo principal. Éste primero analiza la línea de comandos y almacena los argumentos en memoria. Después de crear las pipes, el hilo principal se bifurca en los dos hijos, M0 y M1.

Tras esto, el hilo principal entra en un bucle muy sencillo que se va a ejecutar hasta llegar al final de la simulación. El hilo principal elige uno de los dos nodos y le envía un mensaje que contiene el tiempo actual al descriptor de fichero w3 o w5. Esta es la señal para continuar y consta de un integer de 4 bytes.

El hilo que recibe el mensaje lo lee y comprueba si puede continuar su ejecución (si tiene algún evento que procesar). Si es así, devuelve el código OK indicando que todo marcha bien. Si el hilo no puede continuar y no tiene ningún temporizador pendiente, devuelve el código NOTHING. Si ambos hilos devuelven NOTHING un número suficiente de veces consecutivas, se considera que se ha entrado en un bucle infinito y se termina la ejecución de la simulación. Si no se produce ningún bucle infinito, se continúa la simulación repitiendo los mismos pasos hasta que se alcance el tiempo establecido como final de la simulación.

Cuando se alcanza el tiempo estipulado como final de la simulación, el hilo principal sale del bucle y envía a cada hilo hijo un mensaje de 4 bytes todos a 0, indicando el final de la simulación. Después lee de los dos hilos los datos de la simulación y presenta un pequeño resumen con ellos.

3 Los hilos hijos: M0 y M1

Cada uno de estos hilos hijo llama al protocolo apropiado como una subrutina. Estas rutinas, contenidas en los archivos p2.c a p6.c, son las encargadas de llamar a las funciones descritas en el archivo worker.c. Ellas gestionan los temporizadores, escriben tramas en los pipes, actualizan variables, etc...

Cada protocolo realiza su propia inicialización y entra en el bucle que ejecutará hasta el final de la simulación. En la primera instrucción del bucle se llama a la función wait_for_event(). El código de esta función, y de todas las otras utilizadas por los protocolos, se encuentra en el archivo worker.c.

La función wait_for_event() actualiza algunos contadores y lee las tramas pendientes del otro nodo. Esto se hace para sacar las tramas del pipe, previniendo que este se colapse. Las tramas se almacenan en el array queue[], y se van sacando de él según sea necesario. Los punteros inp y outp apuntan al primer espacio vacío del array queue[] y la siguiente trama para ser sacada, respectivamente. La variable nframes almacena el número de tramas contenidas en el array.

Una vez que el pipe está completamente vacío, wait_for_event() envía un mensaje de 4 bytes al hilo principal para indicarle que está preparado para procesar un nuevo evento. Tras esto, espera a que el hilo principal le envíe la orden de continuar. Cuando esto ocurre, lee el mensaje del pipe y actualiza su tiempo, así todos los hilos permanecen sincronizados.

Una vez actualizado el tiempo, el nodo llama a la función pick_event() para determinar cual es el siguiente evento a procesar. La lista de potenciales eventos difiere según el protocolo simulado. La elección la realiza la función pick_event(), la cual comprueba que eventos son posibles. Por ejemplo, si no hay ningún simulador funcionando o no hay temporizadores en el protocolo simulado, no se puede producir el evento timeout. Si no hay tramas almacenadas en el array queue[] no se puede producir el evento frame_arrival.

Una vez elegido el evento, wait_for_event() lo devuelve a la función que la llamó, es decir, a uno de los hilos de los nodos que ejecutan las rutinas de los protocolos. Ahora se ejecutan las instrucciones específicas de cada protocolo y vuelve al comienzo del bucle. Y así hasta llegar al final de la simulación que se alcanza cuando se recibe del hilo principal un mensaje de 4 bytes a 0, entonces se envía al main los datos de la simulación que son el número de paquetes entregados al nivel de enlace y el número de tramas transmitidas. Tras esto se finaliza la ejecución.

4 Interfaces de entrada y salida

El simulador se ejecuta mediante una línea de comandos que contiene los parámetros de la simulación. La línea de comandos se compone de seis parámetros decimales, que son los siguientes:

sim protocol events timeout pct_loss pct_cksum debug_flags

donde:

- protocol: indica el protocolo a ejecutar, por ejemplo, el 5
- events: indica la duración de la simulación
- timeout: es el intervalo de temporización en ticks
- pct_loss: porcentaje de tramas que se pierden (0-99)
- pct_cksum: porcentaje de tramas que llegan con error

- debug_flags: habilita varios eventos para la traza
  - ➔ 1- tramas enviadas
  - ➔ 2- tramas recibidas
  - ➔ 4- expiración de un temporizador
  - ➔ 8- presentación de un resumen periódico

Estos mensajes de traza son la principal salida del simulador, según se hayan definido. Además de la traza, el simulador presenta la eficiencia que se ha obtenido con el protocolo. Para el simulador, la eficiencia es el número de paquetes entregados a los distintos niveles de red partido por el número de tramas transmitidas por los dos nodos. También se presenta el número total de ticks que se han procesado durante la simulación.

El simulador final

1 Aspectos generales

El simulador final se ha obtenido a partir del simulador de partida realizando en éste numerosos cambios. El principal cambio es que el simulador de partida no contemplaba algunos tiempos, como el tiempo de transmisión de trama, ni tenía ningún control sobre la duración de los eventos, ahora el simulador controla esos intervalos por la unidad interna de tiempo, llamada tick. El simulador final es un simulador orientado a eventos discretos.

El simulador final consta ahora de 5 hilos independientes. Se mantiene el hilo principal, el Main, y los hilos hijos M0 y M1 y se añaden dos nuevos hilos pertenecientes a los dos canales que unen ambos nodos, C0 y C1. Para comunicar estos 5 hilos independientes se crean doce pipes, sus descriptores de fichero se describen a continuación:

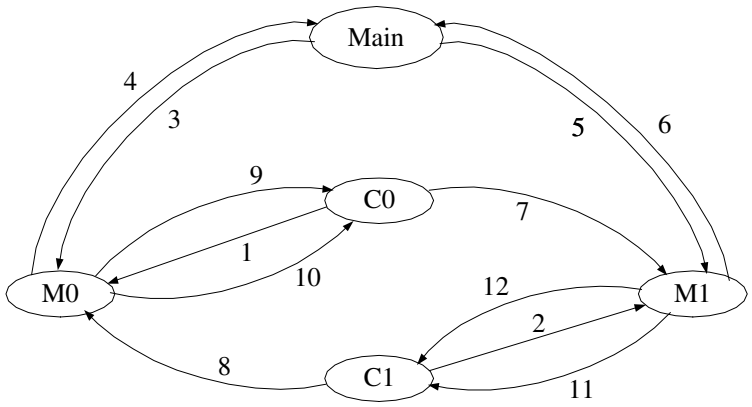

Ilustración 12. Esquema de hilos y pipes del simulador final

- Comunicación Main – M0:
  - w3, r3: señal de continuación de main a M0
  - w4, r4: señal de preparado de M0 a Main

- Comunicación Main – M1:
  - w5, r5: señal de continuación de main a M0
  - w6, r6: señal de preparado de M0 a Main

- Comunicación M0 – C0:
  - w1, r1: tramas de M0 a C0
  - w9, r9: señal de continuación de M0 a C0
  - w10, r10: estado del transmisor de C0 a M0

- Comunicación M1 – C1:
    - w2, r2: tramas de M1 a C1
    - w11, r11: señal de continuación de M1 a C1
    - w12, r12: estado del transmisor de C1 a M1
- Comunicación C0 - M1:
    - w7, r7: tramas de C0 a M1

- Comunicación C1 - M0:
    - w8, r8: tramas de C1 a M0

Otra gran diferencia entre los simuladores es que mientras en el simulador de partida había 5 protocolos a elegir con muy pocos parámetros configurables, en el simulador final disponemos de un solo protocolo pero ahora mucho más potente gracias al gran número de parámetros configurables que se proporcionan. Gracias a este elevado número de parámetros configurables, a partir del protocolo que nos proporciona el simulador final podemos simular también los demás protocolos que proporciona el simulador de partida, como ya se explicó en el apartado Relaciones entre los protocolos del capítulo 2.

2 El hilo principal: Main

El código perteneciente al hilo principal se encuentra en el fichero sim.c. El funcionamiento de este hilo es parecido al del simulador de partida.

El primer paso es analizar los parámetros que proporciona el usuario. En este caso en vez de pasar los parámetros por línea de comandos se realiza mediante un fichero de texto. Una vez analizado este fichero y extraído los parámetros de la simulación, el hilo principal crea los doce pipes necesarios y da comienzo a los otros 4 hilos hijos: M0, M1, C0 y C1.

Una vez creados todos los hilos hijos, el hilo principal entra en el bucle que ejecutará hasta el final de la simulación. Cada ejecución del bucle es un paso en el tiempo del simulador.

Al comienzo del bucle, el hilo principal lee el estado de uno de los nodos y lo procesa como ya se comentó para el simulador de partida, después le manda a ese nodo la señal de continuar y repite el mismo proceso para el otro nodo. Esto quiere decir que los dos nodos reciben la señal de continuar en todos los ticks, no como en el simulador de partida, en el que se elegía un nodo y sólo éste recibía la señal de continuar.

Este bucle se repite hasta que se alcanza el tick estipulado como final de la simulación o hasta que los dos nodos hayan mandado la señal de fin de fichero, indicando que ya han recibido del otro nodo la señal de fin de datos a transmitir, con lo cual la simulación puede finalizar.

Antes de terminar la ejecución del hilo principal se presenta un informe, mucho más extenso que el del simulador de partida, en el que se desglosan todos los datos sobre lo ocurrido durante la simulación.

3 Los hilos de los nodos: M0 y M1

El código perteneciente al protocolo que se va a ejecutar se encuentra en el archivo p.c. Pero éste código se apoya en las numerosas funciones contenidas en el archivo worker.c. Esta estructura del código se ha mantenido así por similitud con el simulador de partida.

En el simulador final los hilos de los nodos no eligen el protocolo que se va a ejecutar sino que siempre se ejecuta el mismo protocolo, pero este protocolo es mucho más potente y gracias al gran número de parámetros configurables que presenta, se puede amoldar a todos los protocolos presentados en esta memoria.

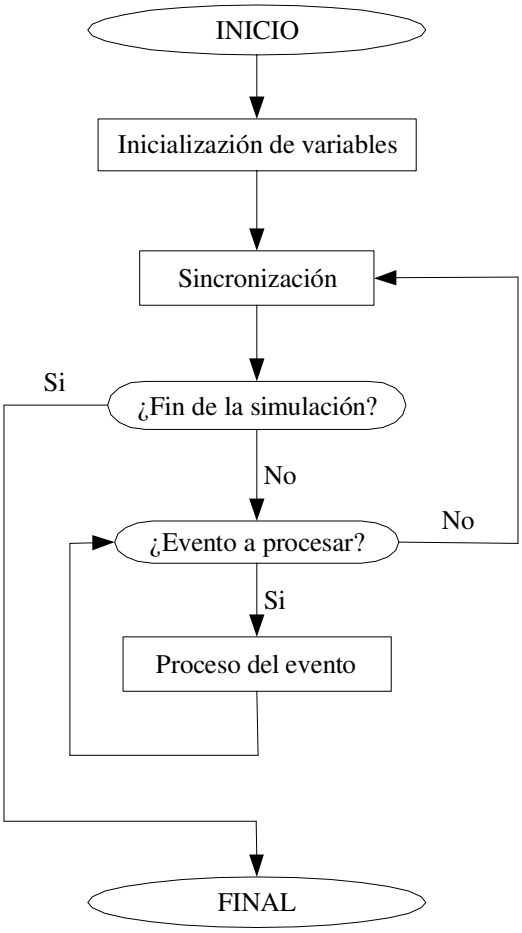

Ilustración 13. Esquema lógico del funcionamiento del protocolo

Una vez creado el hilo del nodo, inmediatamente se ejecuta el código del protocolo. Lo primero que se hace es inicializar todas las variables y estructuras. Después se comienza el bucle que se repetirá hasta el final de la simulación.

Al comienzo de éste bucle se llama a la función wait_for_event(). Esta función ha sufrido una profunda reestructuración. La finalidad de esta función es devolver el siguiente evento a procesar por el protocolo y mantener el sincronismo tanto con el hilo principal como con el canal que le corresponde. Wait_for_event() le envía al hilo principal un mensaje en el que le indica su estado y que puede ser:

- ➔ OK: el protocolo realiza su función normalmente
- ➔ NOTHING: no se ha producido ningún evento ni hay ningún temporizador en marcha durante el último tick.
- ➔ END_FILE: se ha recibido del nodo par la señal de que ha terminado de transmitir todos sus datos.

Tras enviar este mensaje lee el tiempo del hilo principal, en realidad lee un integer de 4 bytes con el tick actual. Este mismo mensaje se lo manda a su canal para mantenerlo sincronizado. También lee del canal un mensaje que le indica si el canal está libre y puede mandarle una nueva trama en caso de tener alguna a la espera de ser transmitida.

Una vez realizadas las sincronizaciones elige el siguiente evento a procesar mediante la función pick_event(). Si no hay ningún evento que procesar se repite toda la secuencia de sincronismo y se vuelve a llamar a pick_event(), todo esto sin devolver el control al protocolo, lo que no se hará hasta que pick_event() devuelve un evento no nulo. El evento devuelto es el mismo parámetro que wait_for_event() devuelve al código del protocolo para que éste continúe. También hay que resaltar que pueden procesarse varios eventos antes de que se produzca una nueva secuencia de sincronismo, es decir, en el mismo tiempo o tick de la simulación.

La función pick_event() comprueba los eventos posibles y entre ellos elige el próximo en ser procesado, según un riguroso orden. Los eventos posibles son, en orden de prioridad:

- ack_timeout: ha expirado el temporizador de asentimiento
- chksum_err: ha llegado una trama con error
- frame_arrival: ha llegado una trama sin error
- network_layer_ready: el nivel de red tiene un paquete preparado
- timeout: ha expirado uno de los temporizadores principales

Según los parámetros de la simulación y la situación instantánea de la ejecución habrá eventos que no puedan producirse.

Antes de finalizar la ejecución, se presenta un resumen bastante amplio de lo ocurrido durante la simulación. Alguno de estos datos son pasados al hilo principal mediante el correspondiente descriptor de fichero.

4 Los hilos de los canales: C0 y C1

El código perteneciente a los canales se encuentra en el archivo canal.c. El canal se encarga de la transmisión y propagación de las tramas. Recibe la señal de sincronismo (un integer de 4 bytes con el tick actual) del nodo que le corresponde y le responde a éste con el estado del transmisor. También recibe de su nodo las tramas que envía al otro nodo una vez han sido transmitidas y propagadas.

Al comienzo de su ejecución realiza la inicialización de las variables. Después entra en el bucle que ejecutará hasta que se termine la simulación. Al principio del bucle realiza la sincronización con su nodo correspondiente, recogiendo el tick actual y enviado su estado. Tras esto comprueba si hay tramas que haya enviado el nodo para ser transmitidas, si es así las mete en una lista doblemente enlazada.

El siguiente paso es examinar el estado del transmisor. Si éste está libre y hay tramas esperando su transmisión empieza su transmisión la primera trama a la espera. Si termina la transmisión de alguna trama, ésta pasa a propagarse y se introduce otra trama en el transmisor en caso de que haya alguna a la espera.

Después se comprueba si hay alguna trama propagándose. Si es así y además termina su propagación, esta trama se envía al otro nodo mediante el correspondiente descriptor de fichero. Y se repite el bucle.

Si en el proceso de sincronización, el tick recibido es 0, indica el final de la simulación. En este caso, simplemente se finaliza la ejecución.

5 Interfaces de entrada y salida

Las interfaces de entrada y salida han sufrido un gran cambio. Mientras que en el simulador de partida la interfaz de entrada era la línea de comandos ahora ésta se ha sustituido por un fichero de texto donde se reflejan todos los parámetros de la simulación. Además el número de estos parámetros ha pasado de sólo seis a treinta. La estructura de este fichero, llamado fichero de entrada se detalla en el pliego de condiciones, más exactamente, en el apartado Fichero de entrada del capítulo 6.

También ha cambiado la interfaz de salida. Si antes era un pequeño resumen de lo ocurrido durante la simulación, además de la traza, ese resumen se ha detallado mucho más y la traza se ha perfeccionado. Ahora la interfaz de salida es otro fichero cuya estructura se detalla en el apartado Fichero de salida del capítulo 6.

La interfaz gráfica InGraSE

La interfaz gráfica InGraSE se desarrolló de manera independiente al simulador. Su finalidad es facilitar la creación de los ficheros de entrada del simulador y la lectura de los ficheros de salida. Posteriormente se le sumó otra función que es la de añadir la posibilidad de realizar baterías de simulaciones y presentar sus resultados mediante gráficas.

La interfaz está programada en lenguaje Visual Basic por su potencia gráfica, su sencillez y facilidad de aprendizaje. Se basa en una interfaz de múltiples documentos (MDI) que consta de un formulario principal y de tantos formularios secundarios como ficheros de simulaciones estén abiertos, además de otros formulario adicionales empleados para realizar otras funciones.

El formulario principal consta tan solo de un menú y de una barra de herramientas. Desde este formulario solo se pueden realizar las funciones de abrir un fichero de simulación o de crear uno nuevo, personalizar el formulario principal mostrando u ocultando el menú y la barra de herramientas y mostrar la ventana Acerca de. El aspecto del formulario principal y de los demás formularios y ventanas se muestran en el capítulo 7.

Los formularios secundarios aparecen cuando se abre una simulación o se crea una nueva. En un formulario secundario se pueden especificar los parámetros de una simulación de una manera más cómoda que con el fichero de entrada. Además, desde un formulario secundario se pueden guardar estos parámetros a un fichero de entrada, s puede lanzar una simulación llamando al simulador con los parámetros que aparecen en el formulario y se pueden realizar baterías de simulaciones.

Para llevar un control de los formularios secundarios abiertos se crea una matriz dinámica. Al abrir un fichero de simulación o crear uno nuevo, se añade un nuevo elemento a esta matriz y cuando se cierra un formulario secundario se busca el elemento de la matriz que pertenece a ese formulario y se elimina. Así, cuando se vaya a realizar una gráfica con los formularios abiertos sólo hay que recorrer la matriz y obtener sus datos.

Desde un formulario secundario se puede lanzar el simulador para realizar una simulación. Al hacer esto, InGraSE llama al simulador y le pasa los parámetros de la simulación que aparecen en el formulario. Al terminar la simulación InGraSE presenta una ventana con los resultados de la simulación realizada. Estos datos se obtienen del fichero de salida que genera el simulador. La descripción de la ventana de resultados aparece también en el capítulo 7.

También desde un formulario secundario se puede lanzar una batería de simulaciones. Ésta consiste en realizar varias simulaciones sobre los mismos parámetros de simulación variando tan solo uno de ellos. Con ello podemos observar cómo se comporta el protocolo ante la variación del parámetro elegido. Para lanzar una batería de simulaciones hemos de seleccionar la opción de menú Herramientas > Batería de simulaciones. Al hacer esto se abre una nueva ventana donde podemos seleccionar el parámetro sobre el que se va a realizar la batería, su valor inicial, el final y el paso entre los distintos valores.

Una vez realizada la batería de simulaciones se abre una nueva ventana donde aparece una gráfica con los resultados de la simulación. En esta ventana se puede elegir qué datos deben aparecer en la gráfica mediante una lista desplegable de los parámetros posibles. En esta ventana también se muestra la media aritmética de los valores que toma el parámetro seleccionado.

Si en la gráfica creada aparece algún valor que nos interese eliminar tan solo tenemos que buscar a qué formulario secundario pertenece y cerrar éste. Después seleccionando la opción de menú Herramientas > Crear Gráfica se vuelve a generar la gráfica pero esta vez sin el valor que hemos eliminado.

Esto quiere decir que sean cuales sean los formularios secundarios que tengamos abiertos, al seleccionar Herramientas > Crear Gráfica se creará una gráfica tomando los valores de los formularios secundarios abiertos cuyas simulaciones se hayan realizado.

Otro aspecto a resaltar es que si tenemos abierto un formulario secundario y hemos realizado su simulación, al seleccionar Resultados en el menú accedemos a la ventana que nos muestra los resultados de la simulación.

Capítulo 4. Validación y pruebas.

Validación teórico-práctica.

En el presente apartado se realiza una introducción teórica al análisis de prestaciones de protocolos de nivel de enlace, además de presentar unas comparativas entre resultados teóricos y resultados obtenidos con el simulador de varios ejemplos realizados sobre diferentes tipos de protocolos.

1 Tráfico de datos.

A la hora de analizar las prestaciones de un protocolo, se ha de determinar antes para qué tipo de tráfico se va a analizar tal protocolo. Existen, en este caso, dos tipo de tráfico:

> Tráfico masivo: la fuente que genera el tráfico está saturada, es decir, siempre tiene tramas que transmitir. En este caso las prestaciones se calculan con la teoría de la probabilidad y el resultado obtenido se denomina *caudal eficaz* (C_{ef}).
> Tráfico interactivo: en este caso la fuente no está saturada. El problema se resuelve mediante la aplicación de la teoría de colas y el resultado obtenido se llama *retardo de tránsito* (T).

El tráfico que generan las fuentes de información de los nodos del simulador es un tráfico masivo por lo que para evaluar las prestaciones nos vamos a fijar en el *caudal eficaz* que proporcionan los protocolos. El *caudal eficaz* o *cadencia eficaz* se define como la relación entre el número de bits de información de usuario y el tiempo de ocupación del enlace (tiempo que se mantiene ocupado el canal para que el destinatario reciba los datos sin error).

2 Detección y corrección de errores.

Cuando nos encontramos ante una situación de tráfico masivo, hemos de fijarnos en el método básico de detección y corrección de errores que implemente nuestro protocolo. Estos métodos vamos a dividirlos en dos tipos:

> Métodos FEC (Forward Error Correction): la información transmitida lleva suficiente redundancia para poder corregir los errores en recepción. En este caso:
>
> P_{error} P(no poder corregir)

Métodos ARQ (Automatic Request Question): se corrige indicando que nos vuelvan a enviar el mensaje otra vez porque ha llegado mal, es decir, no hay corrección en recepción, solo detección. En este caso:

P_{error} P(no ser capaces de detectarlo)

El objetivo de estos métodos es conseguir que estas probabilidades sean lo más pequeñas posibles. En el caso de los protocolos de nivel de enlace, se envían bloques de datos, estos bloques pueden verse corrompidos por distintos tipos de errores: aleatorios, a ráfagas de error, etc... En el simulador solo se consideran errores de tipo aleatorio. Vamos a considerar:

$$P_{error\ bloque} \bullet \quad P_{eb} = 1 - (1 - p)^n$$

donde p es la probabilidad de error de bit y n el número de bits que tiene un bloque de información (incluyendo información de usuario y de control).

Podemos utilizar la siguiente aproximación:

Si $p << 1$ $P_{eb} \bullet \quad p \cdot n$

En el caso de nuestro simulador es el parámetro de probabilidad de error de bloque el que hemos de tener en cuenta. Estas fórmulas hay que tenerlas en cuenta en el caso de que el dato que tengamos sea el de probabilidad de error de bit.

En el simulador no se contempla ningún mecanismo de corrección de errores por lo cual todos los protocolos simulados serán protocolos ARQ.

3 Protocolos analizados. Datos comunes.

Los datos comunes que hemos de conocer para poder estudiar los distintos protocolos son los siguientes:

R: régimen binario del enlace físico [b/s].

Longitud de trama de datos: m + n

- m: número de bits de la cabecera de la trama [b].
- n: numero de bits de información de usuario de la trama.[b].

Tiempo de asentimiento • T_{as}: intervalo de tiempo desde que se envía el último bit de una trama de datos hasta que se recibe el último bit de una trama de control [s].

Tiempo de ocupación • T_{oc}: tiempo que se mantiene ocupado el canal para que el destinatario reciba los datos sin error [s].

Cadencia efic a z $C_{ef} = \frac{n}{T_{oc}}$

4 Unilateral de parada y espera.

4.1 Unilateral de parada y espera con Nacks.

4.1.1 Estudio teórico.

En este protocolo solo hay flujo de información en un sentido. Uno de los nodos (el que tiene paquetes que transmitir) transmite un paquete y espera a que le llegue el asentimiento del mismo, un asentimiento negativo, o a que expire el temporizador para su retransmisión. El otro nodo tan solo espera que le lleguen los paquetes y a asentirlos inmediatamente. Para simplificar, las tramas de asentimiento, tanto positivo como negativo, no van a sufrir errores.

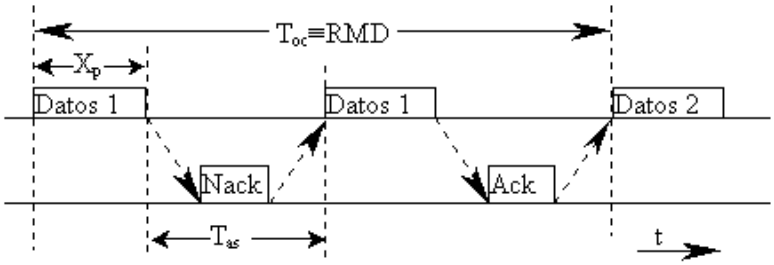

Ilustración 14. Esquema de tiempos del protocolo de parada y espera con Nacks.

El tiempo de ocupación termina precisamente donde se indica porque no vamos a poder seguir transmitiendo hasta que no sepamos si se ha recibido correctamente la trama.

En el esquema anterior sólo hemos planteado un error. Sin embargo, el número de errores dependerá de la probabilidad de error que tengamos. De este modo, el tiempo de ocupación, T_{oc}, vendrá dado por la repetición, tantas veces como errores halla, del tiempo (X_p + T_{as}) más otra vez el tiempo (X_p + T_{as}) correspondiente a la transmisión sin error. Por lo tanto:

$$T_{oc} = (X_p + T_{as})\overline{N_t}$$

donde $\overline{N_t}$ es el número medio de transmisiones que incluye las transmisiones erróneas y la correcta. El cálculo de $\overline{N_t}$ dependerá de cómo planteemos el problema, es decir, depende del protocolo que tengamos.

Vamos a definir n_t como una variable aleatoria que representa el número de transmisiones. En la siguiente tabla se muestra la relación entre el número de transmisiones, n_t, y la probabilidad de que dicho número de transmisiones se produzca, $P[n_t]$.

n_t	$P[n_t]$	
1	$1-P_{eb}$	solo hay una transmisión, la correcta
2	$P_{eb}(1-P_{eb})$	hay dos transmisiones porque solo se ha producido un error
...	...	y así sucesivamente
i	$P_{eb}^{i-1}(1-P_{eb})$	
...	...	
•	0	

Como a nosotros lo que nos interesa es el promedio, $\overline{N_t}$, tendremos que evaluar:

$$\overline{N_t}=\sum_{i=1}^{\infty} i(1-p)\,p^{i-1} \quad \text{donde } p = P_{eb}$$

$$\overline{N_t}=(1-p)\sum_{i=1}^{\infty} ip^{i-1}=(1-p)\frac{d}{dp}\sum_{i=1}^{\infty} p^i=(1-p)\frac{d}{dp}\left(\frac{p}{1-p}\right)=(1-p)\frac{p}{(1-p)^2}$$

$$\overline{N_t}=\frac{1}{1-p}$$

Y como habíamos dicho que $p = P_{eb}$, entonces: $\overline{N_t}=\frac{1}{1-P_{eb}}$

Además: $X_p=\frac{n+m}{R}$ y $T_{as}=2T_p+\frac{m}{R}$

Con lo cual finalmente:

$$C_{ef}=\frac{n}{T_{oc}}=\frac{n}{(X_p+T_{as})\overline{N_t}}=\frac{n}{\left(\frac{n+m}{R}+T_{as}\right)\frac{1}{1-P_{eb}}}=\frac{n}{n+m+T_{as}R}R(1-P_{eb})$$

Otro parámetro que podemos hallar teóricamente es el retardo medio de ida y vuelta, RMD, que es el tiempo transcurrido desde que comienza a transmitirse el primer bit de la trama de datos hasta que termina de recibirse el último bit de la trama de asentimiento correspondiente a esa trama de datos. En el caso del protocolo que nos ocupa este parámetro coincide exactamente con el tiempo de ocupación, T_{oc}. Es decir:

$$RMD \equiv T_{oc}=\left(\frac{n+m}{R}+T_{as}\right)\frac{1}{1-P_{eb}}$$

4.1.2 Simulación.

Para simular este protocolo con el simulador InGraSE lo primero que hemos de tener en cuenta es que la casilla de protocolo unilateral ha de estar marcada para que el flujo de información se dé solo en un sentido. Además todas las ventanas, tanto de transmisión como de recepción, de los dos nodos han de estar a 1 (el valor mínimo) para que en realidad el protocolo sea de parada y espera.

Como unas de las condiciones de este protocolo es que en el enlace de retorno no se producen errores, hemos de tener cuidado en que la casilla de verificación de enlaces simétricos no puede estar marcada y además el parámetro de probabilidad de error de trama del enlace 1 • 0 ha de estar a 0.

Otra cosa que se ha de tener en cuenta es que la casilla de verificación de los asentimientos negativos (Nacks) ha de estar marcada para que el simulador pueda utilizar este tipo de tramas de control durante la simulación.

El resto de parámetros del enlace que vamos a utilizar para la simulación son:

- Régimen binario: 9600 bps
- Velocidad de propagación: 100.000 m/s
- Longitud: 10 Km

Dentro de los parámetros de nodo, tiene gran importancia el tamaño máximo de trama, que vamos a poner a 100 octetos (el máximo valor permitido) para conseguir el máximo rendimiento del protocolo.

Teniendo en cuenta los anteriores valores hemos de elegir un valor para el número de ticks por segundo. Para hacernos una idea vamos a ver el tiempo que se tarda en transmitir una trama:

$$T\_tx = nº \text{ bits} / \text{régimen binario} = 8 * 100 / 9600 = 83{,}3 \text{ ms.}$$

Una vez que conocemos este dato podemos poner un valor de ticks por segundo de 1000 de manera que una trama tarde en transmitirse 83 ticks consiguiendo que la simulación sea a la vez precisa y lo suficientemente rápida.

Con estos parámetros, la pantalla del simulador quedaría del siguiente modo:

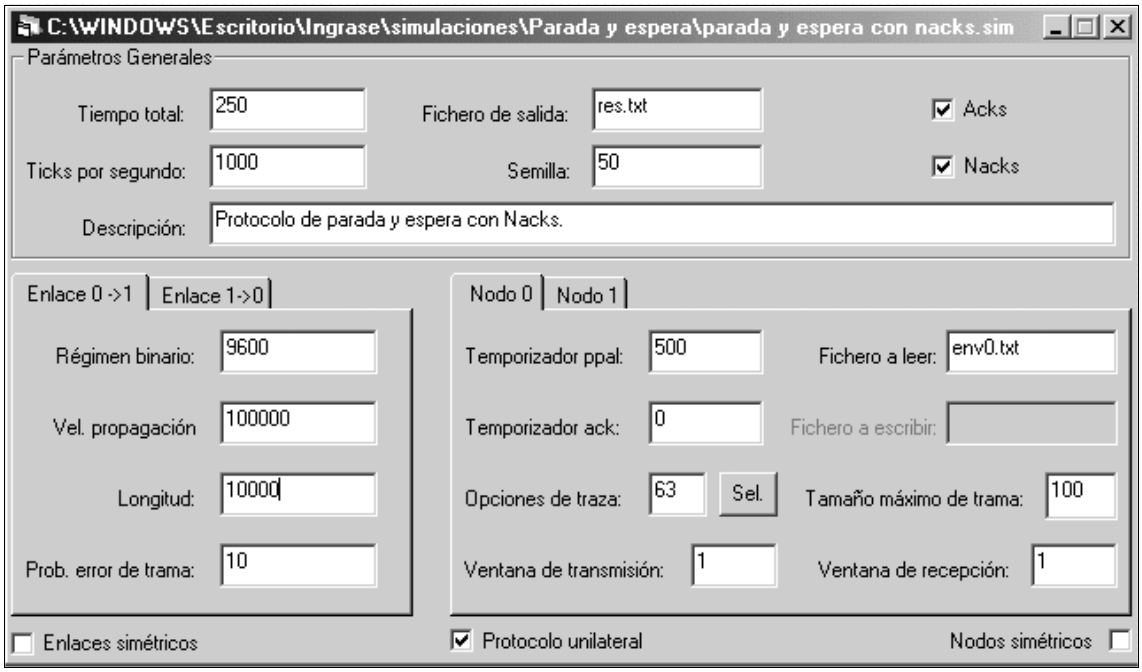

Ilustración 15. Ventana de InGraSE para el protocolo de Parada y Espera con Nacks

Una vez introducidos los parámetros ya podemos comenzar a simular. Con una simulación no basta, sino que vamos a realizar una batería de simulaciones y nos vamos a quedar con la media de los valores obtenidos como resultados de la simulación. Para ello vamos a variar la semilla de la siguiente manera:

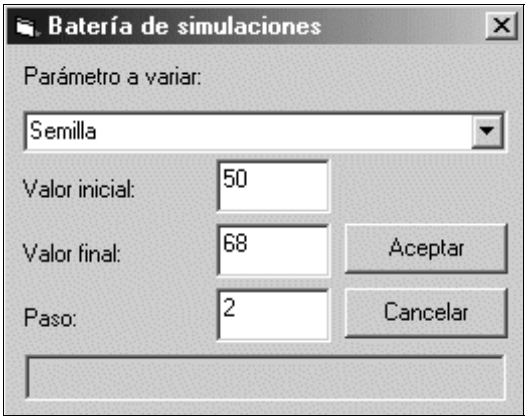

Ilustración 16. Ventana de batería de simulaciones

Además vamos a variar el parámetro de probabilidad de error de trama para comprobar como funciona este protocolo frente a este parámetro y para comparar los resultados de la simulación con resultados teóricos que vamos a obtener después. Es decir, para cada valor de la probabilidad de error de trama vamos a realizar una batería de simulaciones cambiando la semilla y quedándonos con la media. Una muestra de los resultados es lo siguiente:

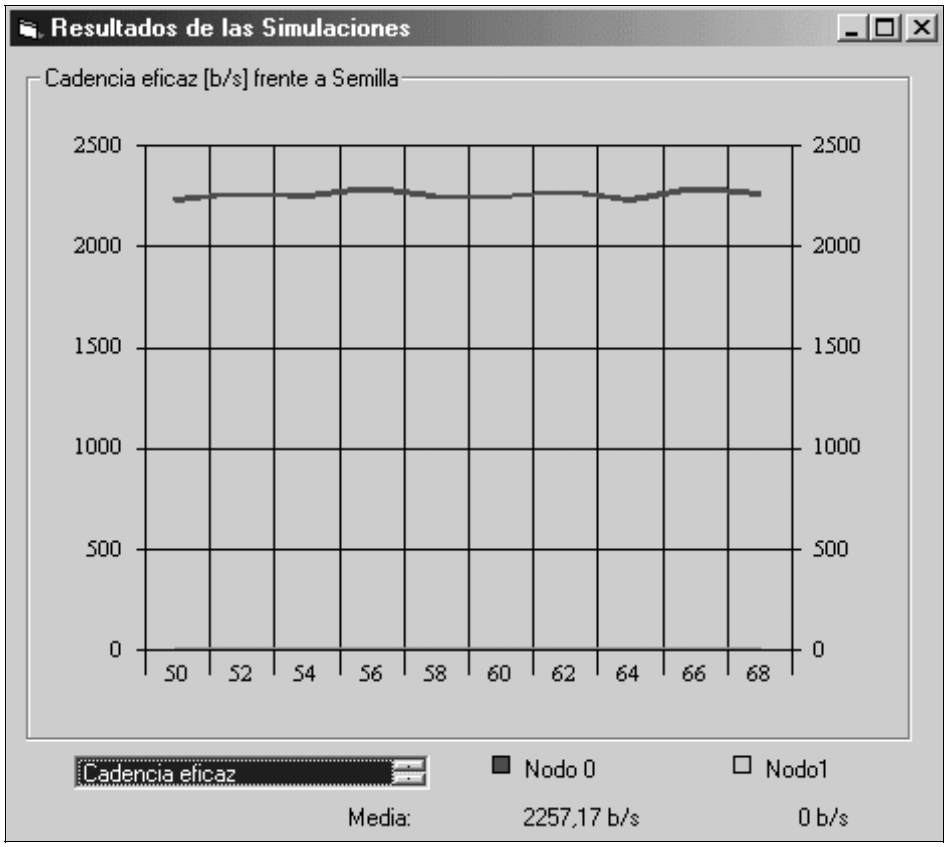

Ilustración 17. Ventana de resultados

Como puede verse las diferentes simulaciones se diferencian bastante poco unas de otras. Esto indica que el fichero que hemos utilizado para "transmitir" tiene una longitud suficiente y que el resto de parámetros de la simulación están bien diseñados (número de ticks por segundo, duración de la simulación,...).

4.1.3 Comparación de los resultados.

Ahora que podemos hallar teóricamente los parámetros de cadencia eficaz y de retardo medio de vuelta y que hemos realizado las simulaciones correspondientes a este protocolo, podemos realizar una comparativa entre los resultados teóricos y los simulados. Para ello antes hemos de conocer los parámetros en los que se basa el protocolo:

- Régimen binario: 9600 b/s
- Bits de datos por trama, n: 92 bits
- Bits de cabecera por trama, m: 8 bits
- Tiempo de asentimiento, T_{as}: 206,67 ms.
- Probabilidad de error de trama, Peb: varía entre 0 y 50 %

Ya somos capaces de obtener valores teóricos para las simulaciones que hemos realizado y presentarlos en una tabla comparando los resultados:

P_{eb} [%]	C_{ef} (teórica) [b/s]	C_{ef} (simulada) [b/s]	Error [%]
0	2537.93	2518.95	0.75
5	2411.03	2394.01	0.71
10	2284.14	2272.82	0.50
15	2157.24	2131.17	1.16
20	2030.34	2013.02	0.85
25	1903.45	1893.44	0.53
30	1776.55	1772.20	0.24
35	1649.66	1648.84	0.05
40	1522.76	1519.57	0.21
45	1395.86	1379.93	1.14
50	1268.97	1273.22	0.34

Pero para comparar estos resultados es mejor una representación gráfica como la que proporciona las siguiente figura:

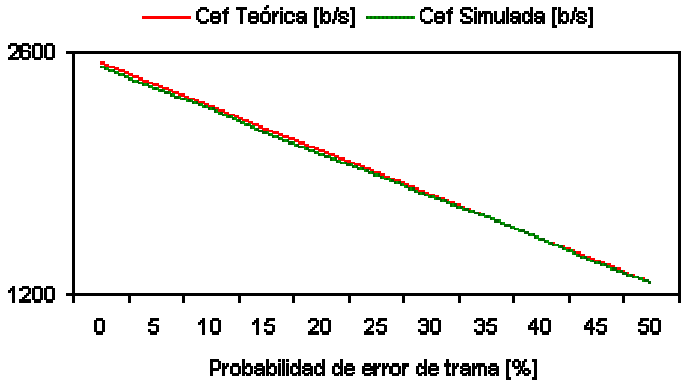

Ilustración 18. Comparación de resultados, C_{ef}

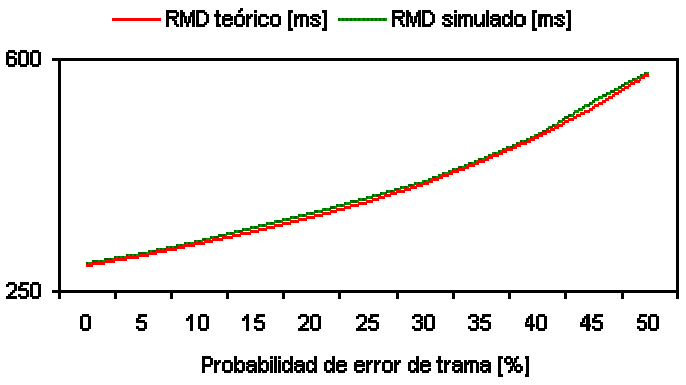

Ilustración 19. Comparación de resultados, RMD

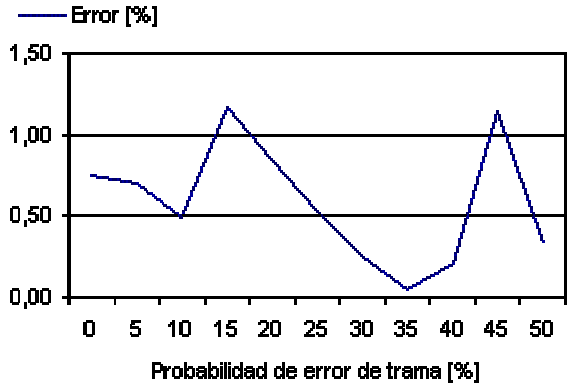

Ilustración 20. Error en los resultados

Con esto resultados por delante se comprueba como el simulador es muy fiel, para el protocolo de parada y espera con nacks, a los resultados teóricos, ya que el error relativo nunca ha llegado ni siguiera a rozar el 1,5 %, y la evolución de los datos es la misma exactamente.

En este punto hay que hacer notar que si en una simulación posterior se encontrasen errores superiores a los presentados, habría que comprobar que el fichero ha transmitir sea suficientemente largo para que la curva de resultados de simulación sea una curva suave. De esta manera podemos estar más seguros de que los resultados obtenidos son fiables.

4.2 Unilateral de parada y espera sin Nacks.

4.2.1 Estudio teórico.

Este protocolo es exactamente igual que el anterior con una sola diferencia. En este caso no se transmite una trama de asentimiento negativo cuando se recibe una trama errónea sino que se espera a que el temporizador de retransmisión del nodo que transmite salte y se retransmita la trama. El siguiente esquema aclara un poco las cosas:

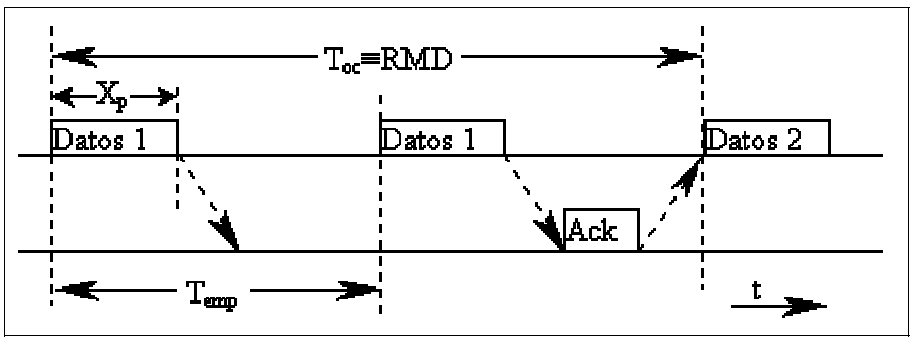

Ilustración 21. Esquema de tiempos del protocolo de parada y espera con Nacks.

Con respecto al estudio teórico del protocolo anterior lo único que cambia es la manera de obtener el tiempo de ocupación. Ahora el T_{oc} es el número medio de repeticiones por el tiempo que se pierde cuando falla una trama, es decir, el tiempo del temporizador de retransmisiones más el tiempo empleado en transmitir la trama correctamente y recibir su asentimiento. Esto es:

$$T_{oc} = T_{emp} \cdot \overline{N_r} + (X_p + T_{as})$$

El parámetro que aparece por primera vez es el número medio de retransmisiones, que se halla de la siguiente manera:

$$\overline{N_r} = \overline{N_t} = \frac{P_{eb}}{1 - P_{eb}}$$

Ya podemos hallar la cadencia eficaz:

$$C_{ef} = \frac{n}{T_{oc}} = \frac{n}{T_{emp} \cdot \overline{N_r} + (X_p + T_{as})} = \frac{n}{T_{emp} \cdot \frac{P_{eb}}{1 - P_{eb}} + \left(\frac{n+m}{R} + T_{as} \right)}$$

El otro parámetro que nos interesa es el retardo medio de vuelta, que de nuevo coincide con el tiempo de ocupación, de tal modo que:

$$RMD \equiv T_{oc} = T_{emp} \cdot \frac{P_{eb}}{1 - P_{eb}} + \left(\frac{n+m}{R} + T_{as} \right)$$

4.2.2 Simulación.

La única variación que hay que introducir a la anterior simulación para simular el actual protocolo es que la casilla de verificación de asentimientos negativos, Nacks, debe quedar desmarcada. Los otros parámetros de la simulación, tanto los pertenecientes a los nodos, como los correspondientes a los dos enlaces, han de permanecer iguales. Así se llega a la siguiente pantalla de simulación:

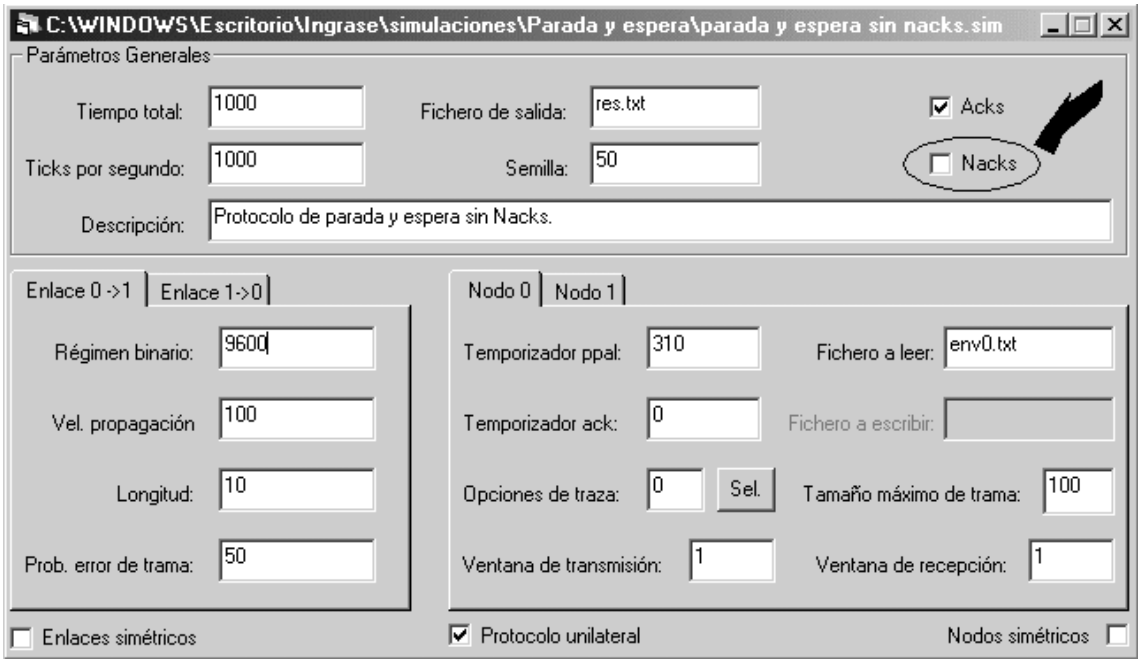

Ilustración 22. Ventana de InGraSE para el protocolo de Parada y Espera sin Nacks

De nuevo vamos a realizar un barrido de las prestaciones del protocolo para distintas probabilidades de error de trama del enlace por el que se produce el flujo de información. Para cada un de los valores de este parámetro vamos a realizar una batería de 10 simulaciones.

En esta ocasión no se va a presentar las pantallas de configuración de la batería de simulaciones por ser exactamente igual a la presentada en el protocolo anterior. Sí se muestra la pantalla de los resultados para que quede reflejada la disminución en la cadencia eficaz respecto del anterior protocolo:

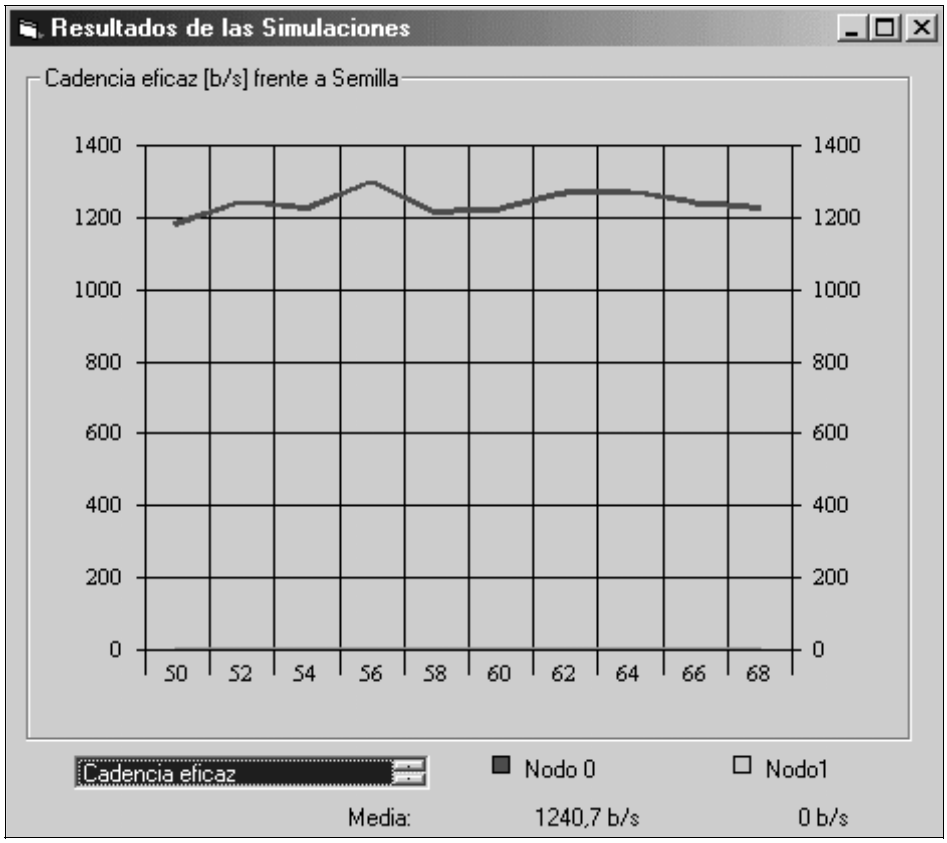

Ilustración 23. Ventana de resultados

4.2.3 Comparación de los resultados.

Los valores empleados tanto para obtener los valores teóricos como para realizar las simulaciones también son los mismos que en el protocolo anterior. Tan solo hay un parámetro cuyo valor antes no era necesario y que ahora es clave, este es el valor del temporizador de retransmisiones, que es:

- Temporizador de retransmisiones: 310 ms.

Con este nuevo valor ya disponemos de todos los datos para hallar los valores correspondientes al comportamiento teórico del protocolo y, además, podemos compararlos con los valores de las simulaciones realizadas en el apartado anterior. Si presentamos esta comparación como una tabla:

P_{eb} [%]	C_{ef} (teórica) [b/s]	C_{ef} (simulada) [b/s]	Error [%]
0	2537.93	2518.59	0.76
5	2402.75	2388.10	0.61
10	2268.49	2259.97	0.38
15	2135.15	2109.77	0.19
20	2002.72	1986.79	0.80
25	1871.19	1854.65	0.88
30	1740.54	1728.73	0.68
35	1610.77	1609.33	0.09
40	1481.88	1481.33	0.04
45	1353.85	1356.72	0.21
50	1226.67	1230.39	0.30

Presentando estos datos de manera gráfica obtenemos lo siguiente:

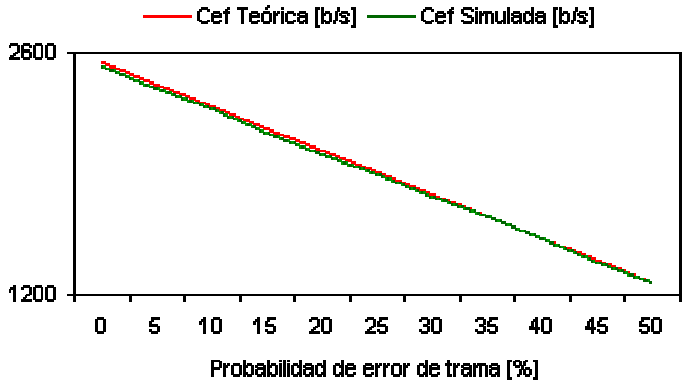

Ilustración 24. Comparación de resultados, C_{ef}

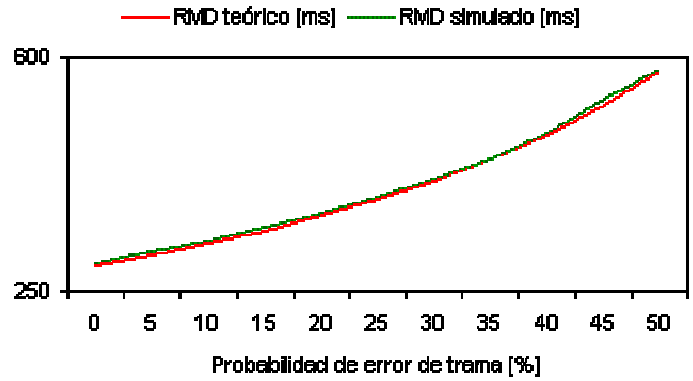

Ilustración 25. Comparación de resultados, RMD

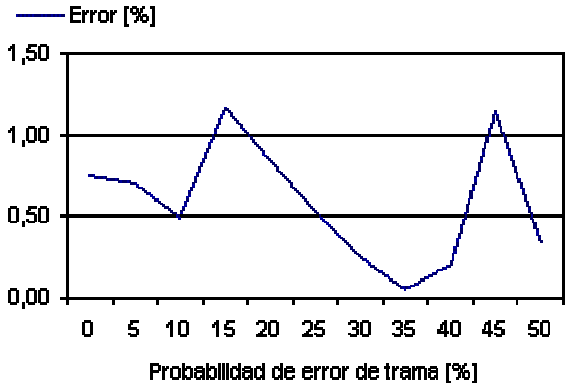

Ilustración 26. Error en los resultados

De nuevo, con estos resultados por delante, podemos comprobar como para el protocolo de parada y espera sin nacks el simulador vuelve a ser muy fiel, superando en muy pocos puntos de la simulación el error relativo el 1 %.

5 Unilateral con repetición simple.

El flujo de información continúa siendo en un solo sentido. En este protocolo la ventana de transmisión es mayor que uno mientras que la ventana de recepción se mantiene a uno. Si se eligen bien los parámetros, esto puede traducirse en que el transmisor envía tramas de manera continua mientras le van llegando los correspondientes asentimientos. El receptor no almacena los paquetes recibidos correctamente pero fuera de secuencia.

Este protocolo también puede tener dos variantes según se utilicen o no las tramas de asentimiento negativo.

5.1 Con tramas de asentimiento negativo.

En el caso de que sí se usen estas tramas, cuando se detecta en el receptor un error en una trama, éste envía un asentimiento negativo, Nack. A la recepción de este asentimiento negativo el transmisor vuelve a enviar todas las tramas a partir de la última que se recibió correctamente. El siguiente esquema explica el funcionamiento de este protocolo:

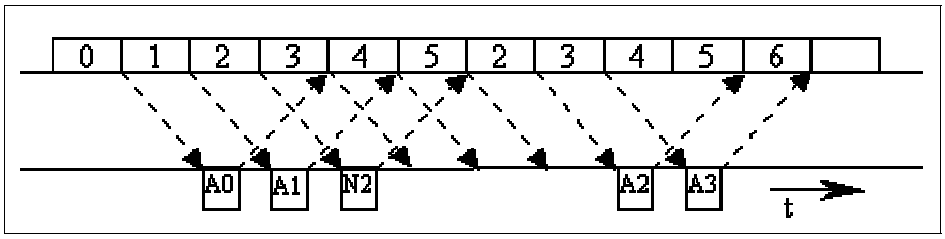

Ilustración 27. Esquema del protocolo de repetición simple con Nacks.

En la línea de arriba se representan las tramas de datos enviadas mientras que en la de abajo se representan las tramas de control de asentimiento positivo, An, o de asentimiento negativo, Nn. Cuando se recibe una trama con error se envía su trama Nack correspondiente y todas las demás tramas que lleguen distintas de la esperada son desechadas.

En este caso, el tiempo de ocupación, T_{oc}, no incluye el asentimiento final, pues ese tiempo también se emplea en enviar otras tramas. Los tiempos que entran en juego en el estudio de este protocolo se muestran a continuación:

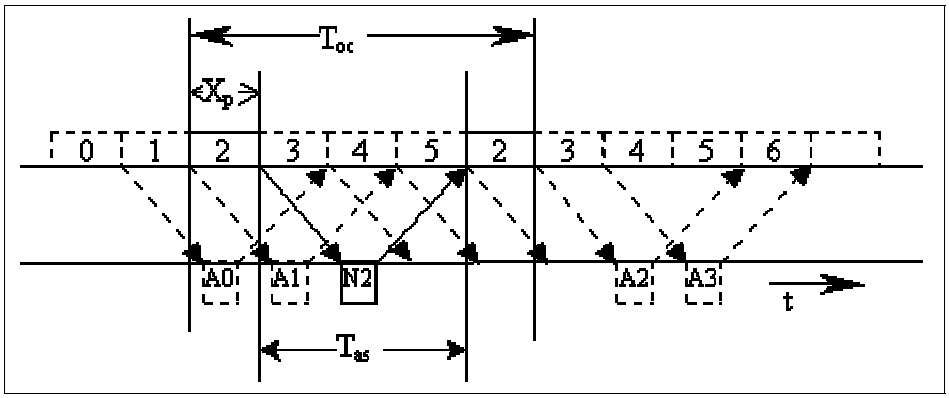

Ilustración 28. Esquema de tiempos del protocolo de repetición simple con Nacks.

Si especificamos este comportamiento mediante fórmulas:

$$T_{oc} = (X_p + T_{as})\overline{N_t} + X_p$$

donde $\overline{N_r}$ es el número medio de retransmisiones. Su relación con el número medio de transmisiones es $\overline{N_r} = \overline{N_t} - 1$. Así, ya podemos hallar la cadencia eficaz que es capaz de proporcionarnos este protocolo:

$$C_{ef} = \frac{n}{T_{oc}} = \frac{n}{\left(\frac{n+m}{R} + T_{as}\right)\overline{N_t} + X_p} = \frac{n}{\left(\frac{n+m}{R} + T_{as}\right)\frac{P_{eb}}{1-P_{eb}} + \frac{n+m}{R}}$$

$$C_{ef} = (1 - P_{eb}) \frac{n}{n + m + P_{eb} T_{as} R} R$$

Y ahora vamos a hallar el otro parámetro que vamos a obtener teóricamente, el retardo medio de vuelta. Para este protocolo el RMD es el tiempo de ocupación más el tiempo de asentimiento, según este esquema:

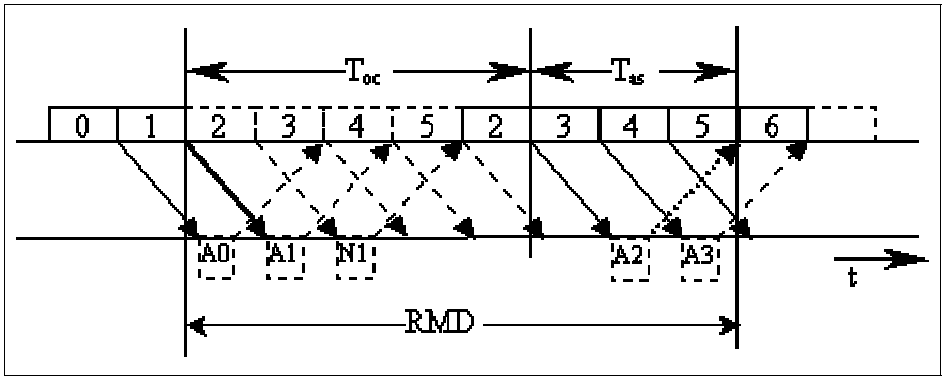

Ilustración 29. Esquema de tiempos completo del protocolo de repetición simple con Nacks.

Con fórmulas:

$$RMD = T_{oc} + T_{as} = (X_p + T_{as})\overline{N_r} + X_p + T_{as} = \left(\frac{n+m}{R} + T_{as}\right)\frac{P_{eb}}{1 - P_{eb}} + \frac{n+m}{R} + T_{as}$$

$$RMD = \left(\frac{n+m}{R} + T_{as}\right)\frac{P_{eb}}{1 - P_{eb}}$$

5.2 Sin tramas de asentimiento negativo.

El otro caso se nos presenta cuando no utilizamos tramas de asentimiento negativo en el receptor. Entonces, cuando el receptor detecta un error simplemente se queda callado y será trabajo del transmisor el darse cuenta de este error. Para ello se utiliza el temporizador principal, de duración T_{emp}.

Cuando este temporizador expira y termina de transmitirse la trama que actualmente se transmite, comienzan a retransmitirse todas las tramas ya mandadas a partir de la última trama asentida correctamente. El siguiente esquema aclara la situación:

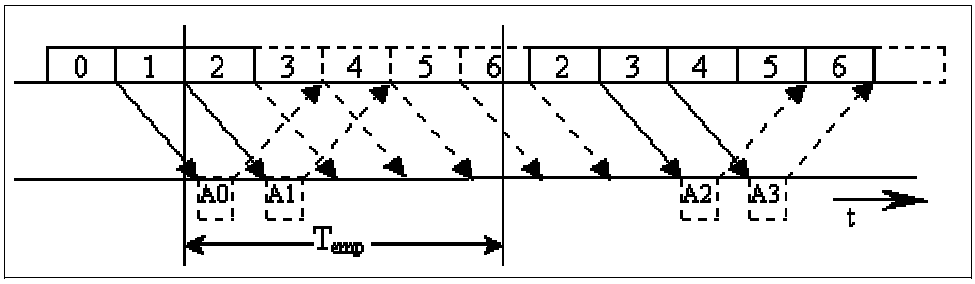

Ilustración 30. Esquema del protocolo de repetición simple sin Nacks.

No se comienza a retransmitir tramas hasta que no expira el temporizador principal y termina de transmitirse la trama actual. Así, el tiempo de ocupación consiste en el tiempo de expiración del temporizador principal más la parte de trama que queda por transmitirse por el número medio de retransmisiones, más el tiempo de transmisión de la trama correcta. El siguiente esquema refleja esta situación:

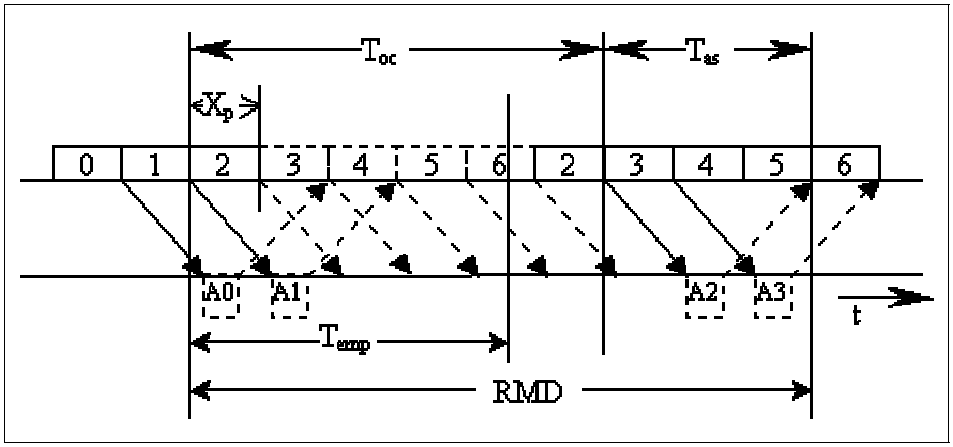

Ilustración 31. Esquema del protocolo de repetición simple sin Nacks.

Si lo expresamos con fórmulas:

$$T_{oc} = \lceil \frac{T_{emp}}{X_p} \rceil \cdot X_p \cdot \overline{N_r} + X_p$$

$$C_{ef} = \frac{n}{T_{oc}} = \frac{n}{\lceil \frac{T_{emp}}{X_p} \rceil \cdot X_p \cdot \overline{N_r} + X_p} = \frac{n}{X_p \left(\lceil \frac{T_{emp}}{X_p} \rceil \cdot \overline{N_r} + 1 \right)}$$

$$C_{ef} = \frac{n}{\frac{n+m}{R} \left(\lceil \frac{R \cdot T_{emp}}{n+m} \rceil \cdot \frac{P_{eb}}{1 - P_{eb}} + 1 \right)}$$

Y ahora el retardo medio de vuelta, que como en el anterior caso, es el tiempo de ocupación más el tiempo de asentimiento.

$$RMD = T_{oc} + T_{as} = \left(\lceil \frac{T_{emp}}{X_p} \rceil \cdot \overline{N_r} + 1 \right) X_p + T_{as} = \frac{n+m}{R} \left(\lceil \frac{R \cdot T_{emp}}{n+m} \rceil \cdot \frac{P_{eb}}{1 - P_{eb}} + 1 \right)$$

5.3 Simulación.

Esta vez no se va a simular los dos casos antes estudiados de este protocolo, con y sin nacks, sino que solo se va a simular el caso sin nacks, que presenta una mayor variación con el resto de protocolos.

Para simular un protocolo unilateral con rechazo simple sin nacks hay varias cosas importantes a tener en cuenta:

- deshabilitar la casilla de verificación de nacks.
- el tamaño de la ventana de transmisión ha de ser mayor que uno si queremos transmisión continua. En nuestro caso vamos a poner 32 como ventana de transmisión.
- el tamaño de la ventana de recepción ha de continuar a 1 para que se produzca un verdadero rechazo simple y no se acepten tramas fuera de secuencia.
- El tamaño del temporizador principal, que lo vamos a fijar a 450 ms.

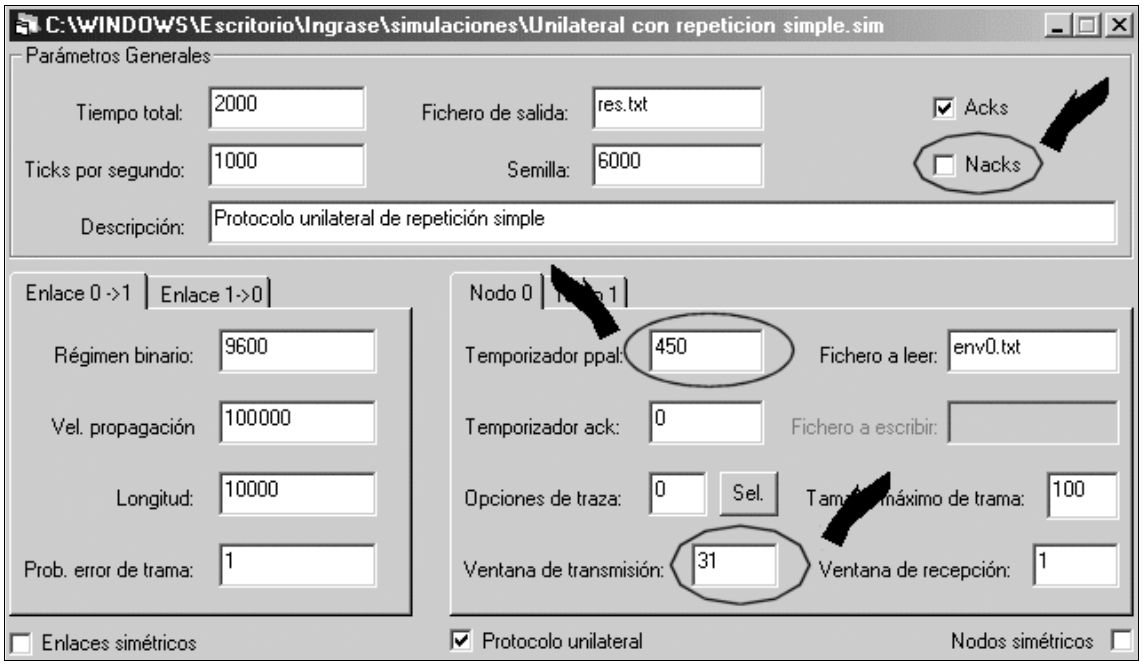

Ilustración 32. Ventana de InGraSE para el protocolo unilateral de repetición simple.

El resto de parámetros podemos verlos directamente en la ventana de simulación de la interfaz gráfica InGraSE.

Una vez que hemos definido los parámetros de la simulación, el siguiente paso es realizar las simulaciones. Para obtener los resultados buscados vamos a realizar un barrido sobre el parámetro de probabilidad de error de trama. Como ya se ha comentado antes para cada valor del parámetro se realiza una batería de simulaciones.

5.4 Comparación de los resultados.

Ya tenemos todos parámetros necesarios para la simulación y por tanto podemos obtener los valores teóricos correspondientes. Pero antes hemos de encontrar el valor de:

$$\lceil \frac{T_{emp}}{X_p} \rceil = \lceil \frac{R \cdot T_{emp}}{n+m} \rceil = \lceil \frac{9600 \cdot 0.45}{8 \cdot (92+8)} \rceil = \lceil 5.4 \rceil = 6$$

Ahora podemos confeccionar la tabla en la que podremos comparar los valores obtenidos tanto en la simulación como en el estudio teórico:

P_{eb} [%]	C_{ef} (teórica) [b/s]	C_{ef} (simulada) [b/s]	Error [%]
0	8832.00	8773.30	0.66
1	8327.31	8227.74	1.20
2	7882.84	7626.83	3.25
3	7469.08	7347.41	1.63
4	7089.23	6576.61	7.23
5	6739.28	6430.59	4.58
6	6415.83	5875.64	8.42
7	6115.98	5582.67	8.72
8	5837.24	4940.34	15.37
9	5577.46	4339.25	22.20
10	5334.77	4102.46	23.10

Presentando estos datos de manera gráfica obtenemos lo siguiente:

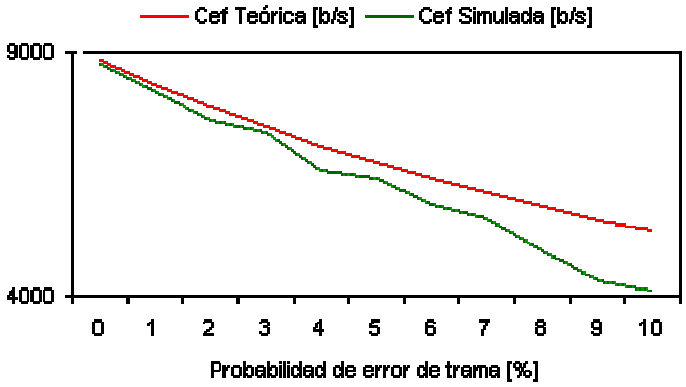

Ilustración 33. Comparación de resultados, C_{ef}

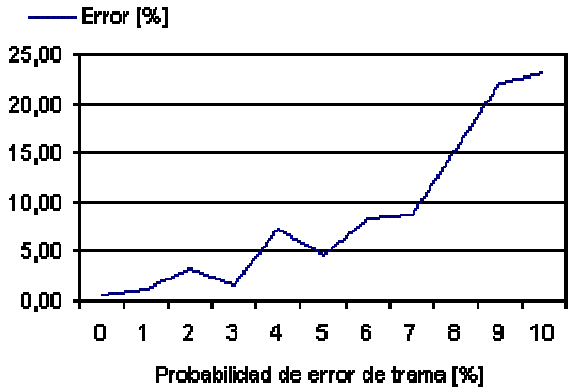

Ilustración 34. Error en los resultados, C_{ef}

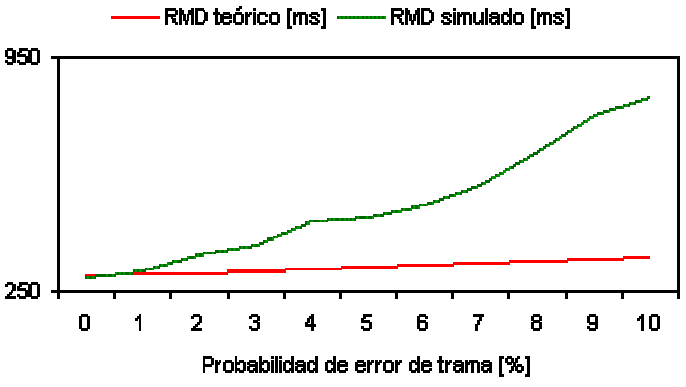

Ilustración 35. Comparación de resultados, RMD

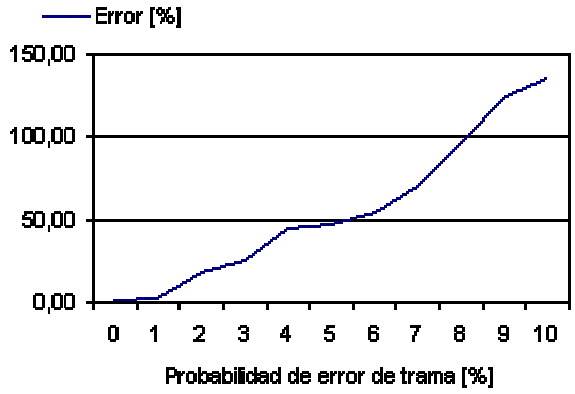

Ilustración 36. Error en los resultados, RMD

Lo primero que se aprecia al ver las comparativas es que ahora el rango de variación de la probabilidad de error es de 0 a 10% y no de 0 a 50% como antes. Esto se debe a que los errores son mucho mayores ahora que antes. Además una probabilidad de un 10% de error de trama en una trama de 100 bits es una probabilidad ya bastante grande y que no suele darse en la realidad ya que esta conlleva una probabilidad de error de bit de 10-4 que es excesivamente elevada.

Puede apreciarse también que para menores probabilidades de trama el resultado sigue siendo bastante parecido al teórico y sobre todo el comportamiento es idéntico para todo el rango de valores. Además se ve como hay una mayor diferencia entre los resultados en el RMD que en la cadencia eficaz. Este problema parece que habrá que tenerlo en cuenta para nuevas versiones del simulador.

Estudiando detenidamente uno de los ficheros de salida vemos como hay una alteración del comportamiento teórico del protocolo. Esta alteración se produce cuando se dan varios errores de trama suficientemente seguidos como para que no se vuelva la situación normal entre uno y otro. En este contexto hay veces en las que hay tramas nuevas que se adelantan a las tramas que deben retransmitirse, retardando las otras y produciendo una serie de retransmisiones posteriores innecesarias que provocan el declive en la cadencia eficaz y el aumento de RMD.

6 Unilateral con repetición selectiva.

6.1 Estudio teórico.

El flujo de información continúa siendo en un solo sentido. En este protocolo la ventana de transmisión es mayor que uno y ahora también la ventana de recepción es mayor que uno. Esto se traduce en que el transmisor envía de tramas de manera continua

mientras le van llegando los correspondientes asentimientos negativos. El receptor esta vez sí almacena los paquetes recibidos correctamente pero fuera de secuencia.

Cuando se produce error en una trama, el receptor envía un asentimiento negativo, Nack. A la recepción de este asentimiento negativo el transmisor no vuelve a enviar todas las tramas a partir de la última que se recibió correctamente, sino tan solo la trama que ha sufrido error, y continúa con la transmisión de las siguientes tramas que aún no ha transmitido. El siguiente esquema pretende explicar el funcionamiento de este protocolo:

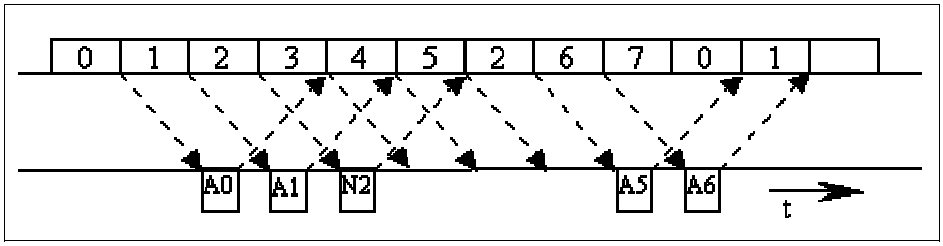

Ilustración 37. Esquema del protocolo de envío continuo y rechazo selectivo.

Con esto se consigue que el tiempo de ocupación, T_{oc}, disminuya drásticamente pues solo se compone del tiempo empleado para la primera transmisión de la trama y sus posibles siguientes retransmisiones ya que los tiempos de propagación y transmisión de los asentimientos se emplea también para transmitir las otras tramas. Los tiempos que entran en juego en el estudio de este protocolo se muestran a continuación:

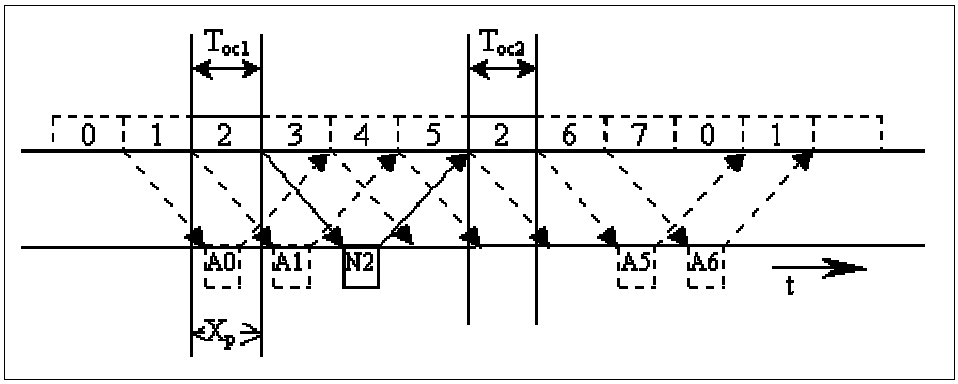

Ilustración 38. Esquema de tiempos del protocolo unilateral de repetición selectiva.

Ahora las tramas que llegan tras una trama errónea no se descartan, sino que se guardan para ser asentidas posteriormente, de modo que el tiempo de ocupación vendrá dado por:

$$T_{oc} = T_{oc1} + T_{oc2} + \ldots + T_{ocn} = X_p \cdot \overline{N_t}$$

Así, la cadencia eficaz que proporciona este protocolo viene dado por:

$$C_{ef} = \frac{n}{T_{oc}} = \frac{n}{\frac{n+m}{R} \cdot \frac{1}{1-P_{eb}}} = (1-P_{eb})\frac{n}{n+m}R$$

Sin embargo, el estudio del retardo medio de vuelta no es tan simple para este protocolo. El siguiente esquema ayuda a la comprensión de este parámetro:

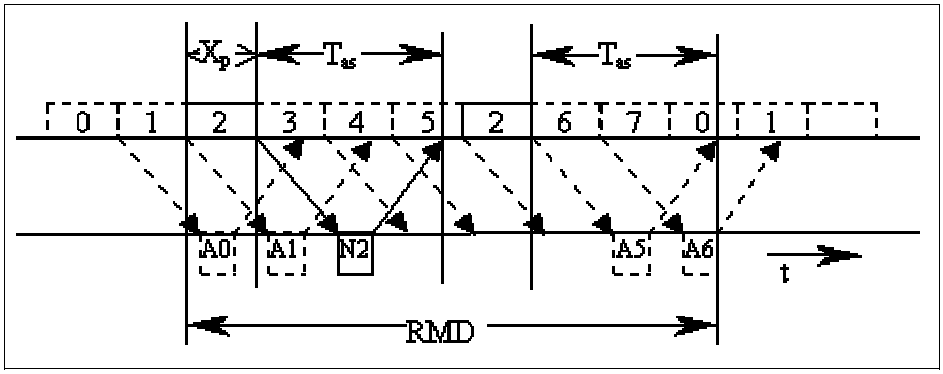

Ilustración 39. Esquema de tiempos del protocolo unilateral de repetición selectiva, RMD.

Es decir, al RMD para cada trama errónea añade el tiempo X_p y el tiempo necesario hasta que se pueda retransmitir la trama. Este tiempo es la función techo del cociente entre el tiempo de asentimiento y el tiempo de transmitir una trama de datos multiplicado por el tiempo de transmitir una trama de datos. Y la última trama que de datos que se transmite, la correcta, añade al RMD el tiempo de transmitirse más el tiempo de asentimiento. En una fórmula:

$$RMD = \left(X_p + \lceil \frac{T_{as}}{X_p} \rceil \cdot X_p \right) \cdot \overline{N_r} + X_p + T_{as} = \left(\lceil \frac{T_{as}}{X_p} \rceil + 1 \right) \cdot \overline{N_r} \cdot X_p + X_p + T_{as}$$

$$RMD = \left(\left(\lceil \frac{T_{as}}{X_p} \rceil + 1 \right) \cdot \overline{N_r} + 1 \right) \cdot X_p + T_{as}$$

$$RMD = \left(\left(\lceil \frac{R \cdot T_{as}}{n+m} \rceil + 1 \right) \cdot \frac{P_{eb}}{1-P_{eb}} + 1 \right) \cdot \frac{n+m}{R} + T_{as}$$

6.2 Simulación.

De nuevo ya tenemos el estudio realizado y podemos dedicarnos a diseñar las simulaciones. Para simular un protocolo unilateral con rechazo selectivo hay varias cosas importantes a tener en cuenta:

- habilitar la casilla de verificación de nacks, ya que de no hacerlo el protocolo seguiría comportándose como de rechazo simple.
- el tamaño de la ventana de transmisión ha de ser mayor que uno si queremos transmisión continua. En nuestro caso vamos a poner 32 como ventana de transmisión.
- el tamaño de la ventana de recepción del nodo 1 ha de ser superior a 1 para que se produzca un verdadero rechazo selectivo y sí se acepten tramas recibidas correctamente pero fuera de secuencia. En este caso vamos a poner 31 como ventana de recepción.
- puesto que el protocolo es unilateral y solo un nodo transmite tramas de datos, en este nodo es irrelevante el tamaño de la ventana de recepción.
- El tamaño del temporizador principal ha de ser lo suficientemente grande para que no expire.

El resto de parámetros podemos verlos directamente en la ventana de simulación de la interfaz gráfica InGraSE.

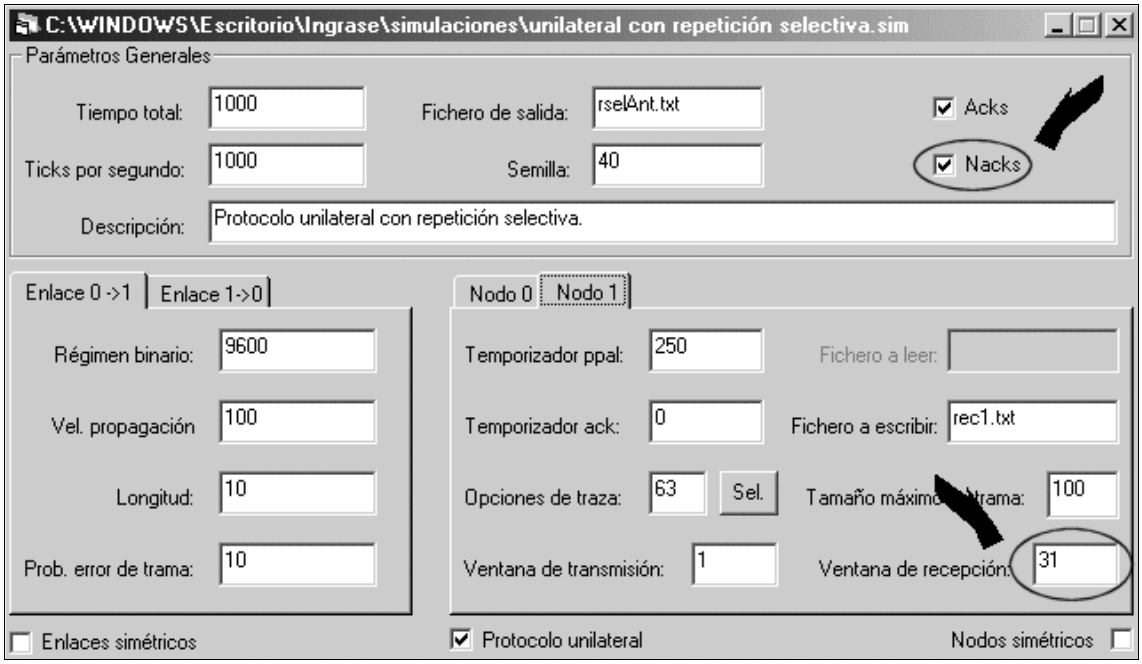

Ilustración 40. Ventana de InGraSE para el protocolo unilateral de repetición selectiva.

Una vez que hemos definido los parámetros de la simulación, el siguiente paso es realizar las simulaciones. Para obtener los resultados buscados vamos a realizar un barrido sobre el parámetro de probabilidad de error de trama. Como ya se ha comentado antes para cada valor del parámetro se realiza una batería de simulaciones.

6.1 Comparación de los resultados.

Ya tenemos todos parámetros necesarios para la simulación y por tanto podemos obtener los valores teóricos correspondientes. Pero antes hemos de realizar algunos cálculos intermedios:

$$\left\lceil \frac{T_{as}}{X_p} \right\rceil = \left\lceil \frac{R \cdot T_{as}}{n+m} \right\rceil = \left\lceil \frac{9600 \cdot 0.206}{8 \cdot (92+8)} \right\rceil = \lceil 2.47 \rceil = 3$$

El valor obtenido nos dice que antes de que pueda retransmitir la trama errónea se transmiten las 3 siguientes tramas, es decir, hay que esperar 3 * X_p = 249 ms. Ahora podemos confeccionar la tabla en la que podremos comparar los valores obtenidos tanto en la simulación como en el estudio teórico:

P_{eb} [%]	C_{ef} (teórica) [b/s]	C_{ef} (simulada) [b/s]	Error [%]
0	8832.00	8788.62	0.49
1	8743.68	8687.71	0.64
2	8655.36	8582.86	0.84
3	8567.04	8458.02	1.27
4	8478.72	8348.50	1.54
5	8390.40	8235.04	1.85
6	8302.08	8111.21	2.30
7	8213.76	7999.95	2.60
8	8125.44	7860.72	3.26
9	8037.12	7729.68	3.38
10	7948.80	7591.88	4.49

Presentando estos datos de manera gráfica obtenemos lo siguiente:

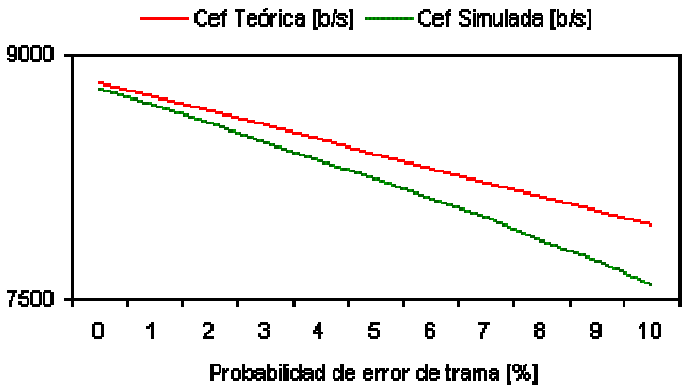

Ilustración 41. Comparación de resultados, C_{ef}

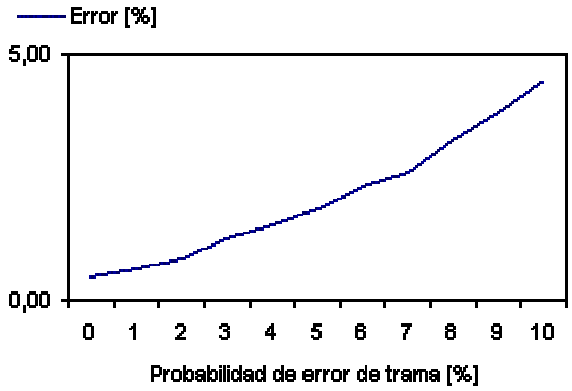

Ilustración 42. Error en los resultados, C_{ef}

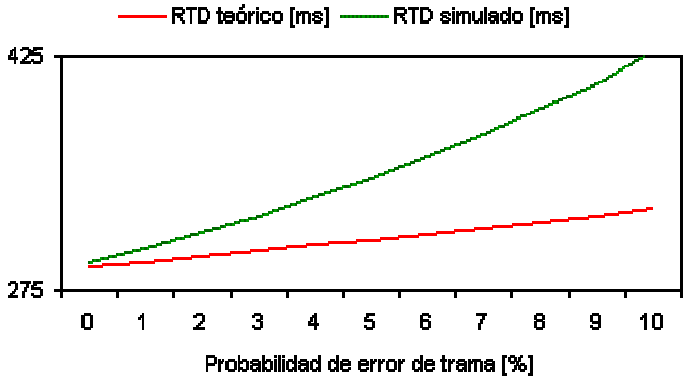

Ilustración 43. Comparación de resultados, RMD

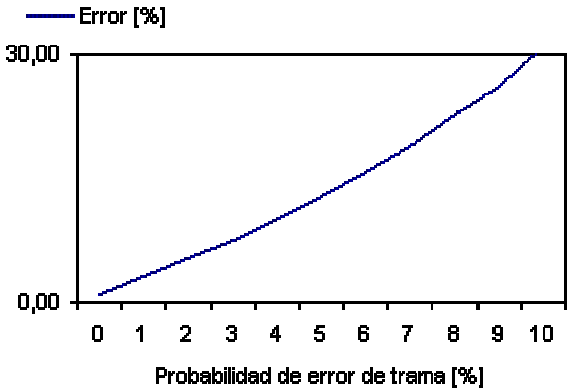

Ilustración 44. Error en los resultados, RMD

Lo primero que se aprecia al ver las comparativas es que el rango de variación de la probabilidad de error es de nuevo de 0 a 10%. Esto se debe a que los errores aunque

no llegan al nivel de los del anterior protocolo, son mayores que en el protocolo de parada y espera.

Puede apreciarse también que para menores probabilidades de trama el resultado sigue siendo bastante parecido al teórico y sobre todo el comportamiento es idéntico para todo el rango de valores. Además se ve como hay una mayor diferencia entre los resultados en el RMD que en la cadencia eficaz. Este problema parece que habrá que tenerlo en cuenta para nuevas versiones del simulador.

Estudiando detenidamente uno de los ficheros de salida vemos como no hay ninguna alteración del comportamiento teórico del protocolo, sino que el problema aparece en un caso no recogido por el estudio teórico. Este caso aparece cuando se dan dos errores de trama tan cercanos que el segundo se produce antes de que se restablezca la situación normal.

Cuando se produce un error y se envía un nack, la información de control que va en esta trama es el número de secuencia de la última trama recibida correctamente. A la recepción de este nack se reenvía la trama cuyo número de secuencia coincide con uno más del número que viene en el nack.

El problema aparece cuando se produce el segundo error en otra trama diferente a la que sufrió el primer error. La trama nack generada en este segundo error es exactamente idéntica a la que se generó en el primer error pues no ha habido tiempo a que se reciban más tramas de datos correctas. Entonces a la recepción de este nuevo nack se reenvía la misma trama que se reenvió a la recepción del primer nack, no produciéndose el efecto deseado y provocando el decaimiento de la cadencia eficaz.

Es decir, no es un error del simulador o del diseño del protocolo sino que es un problema que tiene el propio protocolo y que no se ha recogido, por complejidad, en el estudio teórico.

Validación con el alumnado.

Durante el curso académico 2001/02 en la asignatura Arquitectura de Redes, Sistemas y Servicios perteneciente a la titulación de Ingeniero de Telecomunicaciones impartida en la Escuela Superior de Ingenieros de la Universidad de Sevilla, se propuso al alumnado la realización de unas prácticas voluntarias basadas en una versión bastante avanzada del simulador de protocolos de nivel de enlace. Los enunciados de las citadas prácticas conforman el Apéndice C.

Esta versión del simulador de protocolos de nivel de enlace se componía de la versión 1.0 del simulador en C y de la versión 1.0 de la interfaz gráfica InGraSE. Esta versión del simulador era completamente funcional si bien tenía algunos errores ya subsanados y otros aún por subsanar y estaba ya dotada de un programa de instalación para facilitar a los alumnos la tarea de situar los diferentes ficheros en los directorios correspondientes y de disponer de todas las librerías necesarias.

El número de prácticas entregadas por los alumnos fue suficiente para poder recoger la impresión de los alumnos acerca del simulador, tanto de su utilización como de las mejoras que ofrece a la hora de comprender el funcionamiento de los protocolos de nivel de enlace.

Como norma general los alumnos han conseguido dominar el simulador sin demasiado esfuerzo, sin duda gracias a la facilidad que aporta la interfaz gráfica y a la entrega junto con el simulador de una guía de usuario. Surgieron algunos problemas que se han solventado introduciendo un ejemplo muy completo a la citada guía de usuario, que puede encontrarse en el Apéndice A. Los principales problemas que han tenido los alumnos en la utilización del simulador son:

- Utilización de ficheros para la comunicación demasiado pequeños, con lo que resultan simulaciones demasiado cortas para reflejar el verdadero comportamiento de los protocolos.

- No utilización de asentimientos positivos en protocolos unilaterales, con lo que la comunicación efectiva no es posible.

- Elección errónea de los rangos de variación de los parámetros a la hora de generar gráficas.

- Introducción de errores en el canal de retorno de los protocolos unilaterales cuando este canal se suele suponer libre de errores.

Además de estos problemas aparecidos a los alumnos en la utilización o comprensión del simulador y la interfaz gráfica, los alumnos se encontraron con algunos errores del simulador. La mayoría de estos errores se han suprimido en versiones posteriores, aunque aún quedan algunas posibles mejoras que realizar. Una lista de estos errores detectados, y de si han sido corregidos (✓) o no (✗) es la siguiente:

- ✔ error 91, producido al elegir el parámetro sobre el que se va a realizar la batería de simulaciones
- ✔ error 424, producido al seleccionar el menú Opciones al abrir el programa
- ✔ error al abrir el menú Ayuda
- ✔ creación de la gráfica de resultados de la batería de simulaciones sólo sobre las ventanas de simulación abiertas (poder descartar algunos valores)
- ✔ aparición de rendimientos mayores al 100%
- ✗ presentar la gráfica de resultados de la batería de simulaciones como una curva de mejor ajuste en lugar unir los puntos mediante una línea de trazos rectos
- ✗ adaptar las ventanas, controles y demás objetos a la resolución de la pantalla

A pesar de estos errores encontrados y de los problemas de utilización, la opinión generalizada de los alumnos es que el simulador de protocolos de nivel de enlace les ha sido de gran ayuda a la hora de comprender y estudiar estos protocolos.

Capítulo 5. Conclusiones y líneas de continuación.

Conclusiones

En el presente proyecto fin de carrera se ha diseñado y desarrollado un simulador de protocolos de nivel de enlace con una finalidad claramente educativa. Gracias a este software se puede analizar en profundidad el comportamiento de los protocolos de este nivel, variar los parámetros que influyen en los mismos así como observar el comportamiento de forma fácil gracias a la presentación de gráficas.

El simulador es muy flexible gracias al gran número de parámetros que presenta. Debido a ello se pueden simular distintos tipos de protocolos, desde los más simples de parada y espera hasta los más evolucionados de ventana deslizante. En el simulador se puede definir tanto el comportamiento del protocolo como las características del medio sobre el que tiene lugar la comunicación. Así, además de comprobar la influencia de un parámetro en el comportamiento del protocolo también puede estudiarse de qué manera un protocolo es más indicado para un tipo de medio determinado.

Además la utilización de ficheros de entrada y salida bien definidos facilita la introducción de los parámetros de la simulación y la lectura de los resultados de la misma.

La interfaz gráfica facilita aún más la creación de los ficheros de entrada y la lectura de los resultados. La presentación de estos resultados mediante gráficas hace que la comprensión del desarrollo de la simulación sea mucho más rápido y claro que mediante el fichero de salida.

La posibilidad añadida por la interfaz gráfica de realizar baterías de simulaciones amplía la capacidad del usuario para comprobar como la variación de un parámetro en particular influye en las prestaciones de los protocolos.

El correcto funcionamiento del simulador ha sido comprobando contrastando sus resultados con los resultados obtenidos tras un estudio teórico de diversos protocolos. El resultado ha sido completamente satisfactorio.

La utilización por parte de los alumnos tanto del simulador como de la interfaz gráfica ha permitido ratificar que el empleo del simulador de protocolos de nivel de enlace favorece la comprensión del funcionamiento de tales protocolos. También ha demostrado que el simulador de protocolos es fácil de usar y su utilización no requiere mucho tiempo de aprendizaje.

Líneas de continuación

A continuación se enumeran algunas ampliaciones que pueden mejorar el simulador de protocolos de nivel de enlace.

1 Simulador

- Inclusión de técnicas de corrección de errores como el uso de códigos cíclicos redundantes.
- Simulación de protocolos más conocidos y utilizados, como por ejemplo HDLC.
- Mejorar la eficiencia procurando que los hilos no principales solo se ejecuten cuando tengan algún evento que procesar y en caso contrario permanezcan dormidos. El hilo principal ha de ser el encargado de despertarlos cuando sea oportuno.

2 InGraSE

- Creación de una ayuda on-line a partir de la guía de usuario.
- Ampliar el número de variables del funcionamiento de la interfaz gráfica que ésta guarda en el registro, como por ejemplo la localización del simulador.
- Ampliar las opciones de presentación de gráficas en los resultados de las baterías de simulaciones, por ejemplo incluyendo la posibilidad de trazar la curva de mejor ajuste.
- Incluir en InGraSE alguna utilidad que imprima los resultados de la simulación deseados y las gráficas. O al menos que exporte las gráficas a algún fichero gráfico de formato estándar.
- Adecuar la presentación de los distintos formularios a la resolución de la pantalla.