Soporte de programación para protocolos del nivel 2 OSI/ISO

Guillermo Rigotti
UNICEN – Fac. de Ciencias Exactas
ISISTAN – Grupo de Objetos y Visualización
Pje. Arroyo Seco, (7000) Tandil, Bs. As. Argentina
TE: +54-2293-440363 FAX: +54-2293-440362
Email: grigotti@exa.unicen.edu.ar

Resumen

En este trabajo se presenta parte de un soporte desarrollado con el objetivo de facilitar la enseñanza de las materias del área de comunicación de datos. Dicho desarrollo fue motivado por la experiencia adquirida a través del dictado de dichas materias y apunta a: 1- facilitar a los alumnos la asimilación de conceptos básicos de arquitecturas de niveles, tales como interfaz, protocolo, punto de acceso al servicio, etc, que se dificulta por su abstracción, y 2- proveer un medioambiente de programación simple para la comunicación de procesos remotos, la que se dificulta debido a la naturaleza asincrónica de los mismos y a las características del medio de comunicación subyacente.

Los desarrollos presentados en este trabajo conciernen a arquitecturas de niveles y a protocolos de nivel 2 del modelo OSI/ISO. Estos constituyen la primera etapa de un proyecto más ambicioso en el marco del cual se pretende proveer un conjunto de herramientas desarrolladas de manera homogénea que faciliten la asimilación de los conceptos más importantes del área por parte de los alumnos.

Palabras clave: arquitectura de niveles, protocolos.

1. Introducción

El presente trabajo surge como consecuencia de las dificultades experimentadas por los alumnos para la asimilación de conceptos básicos de comunicación de datos, sobre todo en sus pasos iniciales en el área.

El objetivo perseguido es el desarrollo de un ambiente homogéneo de rápida asimilación por parte de los alumnos, que posibilite la comprensión y el manejo de los conceptos de mayor importancia relacionados con las materias correspondientes, a través de la programación, simulación y/o visualización de interacciones entre los procesos que componen los protocolos. Este ambiente, integrado por diferentes módulos, debe ser lo suficientemente flexible para que puedan plasmarse conceptos de diferente complejidad, desde aquellos abstractos y relativamente simples de las arquitecturas de niveles hasta los más complejos y dinámicos que representan interacciones entre procesos remotos asincrónicos comunicándose a través de medios que introducen errores y demoras, en ciertos casos variables.

En nuestro caso en particular, los conceptos generales de comunicación de datos se imparten divididos en dos materias, una del segundo curso de la carrera[1], en la que se desarrollan los conceptos básicos de arquitecturas de niveles y los dos primeros niveles del modelo OSI/ISO [Rose, 1990] [Hebrawi, 1993], y un segundo curso perteneciente al cuarto año de la carrera, que se refiere a los niveles superiores de dicho modelo. Estos conceptos generales son completados con arquitecturas especificas tales como TCP/IP en materias optativas de la carrera.

[1] Se hace referencia a la carrera Ingenieria en Sistemas, Dpto, Computación y Sistemas de la Fac de Ciencias Exactas de la UNICEN.

A través de varios años de experiencia en el dictado de la materia, se detectaron en el primer curso de Comunicación de Datos dos problemas de importancia referidos a los resultados logrados con los alumnos.

El primero de ellos fue la dificultad para que los alumnos asociaran los conceptos abstractos propios de las arquitecturas de niveles, como interfaz, primitiva, etc. con su correspondiente implementación en sistemas reales. Esto se debe a que en casi todas las implementaciones, la división estricta en niveles no es respetada por razones de eficiencia. Por otro lado, se dificulta la comprensión de la implementación como consecuencia de la complejidad que presenta, ya que incluye detalle de manejo de dispositivos de transmisión como placas de red y debido a que el software de comunicaciones se encuentra generalmente a este nivel embebido en el sistema operativo, un entorno complejo de entender para los alumnos en esta etapa de la carrera, y que nos aleja de los conceptos relativos a la comunicación, sobre los cuales se desea enfatizar en la materia.

El segundo problema detectado fue la dificultad de los alumnos para asimilar las interacciones que se producen y las consideraciones a tener en cuenta en la programación de procesos asincrónicos que se comunican, en este caso punto a punto, en el contexto del nivel 2 del modelo OSI/ISO, a través de un medio que introduce demoras, posibles errores y/o pérdidas de bloques.

Para atacar estos problemas se desarrolló un sistema que permite experimentar en estos aspectos a través de la programación simple de dos procesos remotos que se comunican. En una primera etapa de su uso, el objetivo es resaltar la interacción entre entidades locales entre sí pertenecientes a niveles adyacentes, permitiendo la asociación de conceptos tales como interfaz y primitiva, punto de acceso al servicio, etc, con su correspondiente implementación. En una segunda etapa, dicho módulo posibilita la experimentación en la programación de protocolos de nivel 2 vía programación de las entidades pares correspondientes, simulación de sus interacciones y visualización de las mismas, por ejemplo, intercambio de frames, errores y vencimiento de timers.

Debido a que el desarrollo realizado constituirá un módulo del sistema final que se tiene previsto desarrollar, sus características se ajustan a las que el ambiente integrado deberá tener:

- Rapidez de aprendizaje del lenguaje a utilizar por parte de los alumnos
- Sencillez del soporte provisto

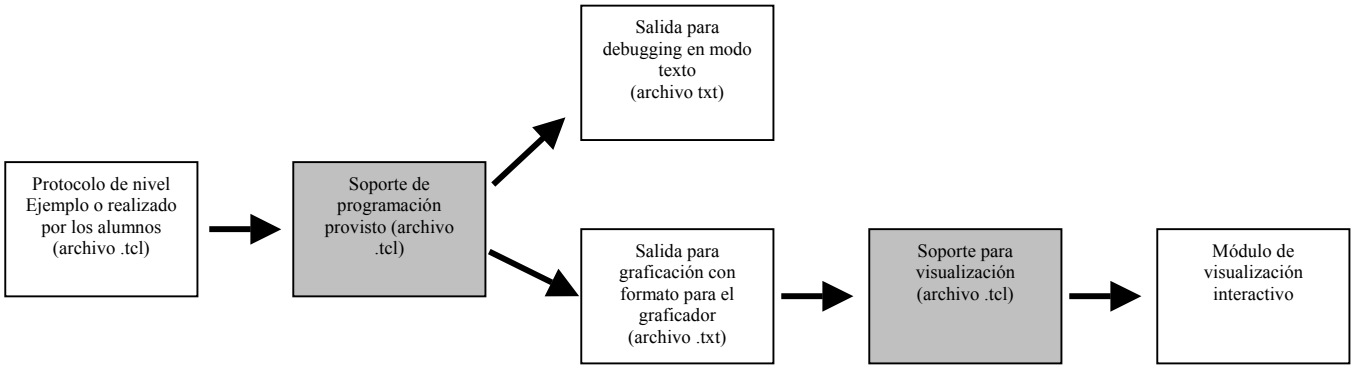

Fig. 1. Módulos que componen el sistema desarrollado (sombreados)

- Visualización de las interacciones entre los procesos involucrados

El trabajo realizado hasta el momento, que permite la experimentación con conceptos de arquitecturas de niveles y con la programación de protocolos de nivel 2, se compone de un módulo que ofrece un soporte para la programación de procesos que se comunican a nivel 2 - representa dos equipos conectados por un vínculo de comunicación -, y de un módulo que permite la visualización de las interacciones - intercambio de frames y gestión de timers empleados por los procesos – por parte de los usuarios. Ambos módulos se desarrollaron en lenguaje Tcl [Osterhout,1994], el mismo lenguaje en el que el usuario debe programar sus protocolos. La vinculación entre estos módulos se reduce a un archivo de texto generado por el módulo de soporte de programación, que es procesado luego por el soporte de visualización.

El sistema permite la elaboración de ejemplos representativos de situaciones particulares en los protocolos, para que sean analizadas por los usuarios, y además posibilita el desarrollo de protocolos por parte de dichos usuarios. En ambos casos se cuenta con una salida en modo texto, que registra los eventos de importancia para el protocolo, y una salida gráfica a través de la cual el usuario puede visualizar las interacciones entre los procesos que se comunican.

En la figura 1 se muestran los diferentes módulos y su relación. Los módulos sombreados son los que integran el sistema. El módulo de soporte de programación es el que emula los equipos y el vínculo de comunicación, y provee un soporte simple de programación que consta de un conjunto reducido de interacciones con el protocolo de nivel 2. Dicho protocolo esta representado por código incluido en el módulo de la izquierda. Este código, hará uso del soporte de programación, por ejemplo invocando a la primitiva enviar o recibir. El soporte de programación está desarrollado en lenguaje TCL, aumentado con la funcionalidad del simulador Ns [Fall, 2000]. La ejecución de este módulo teniendo como entrada al código del protocolo dará lugar a la simulación del mismo, generando los dos archivos de salida que se muestran en la figura. Uno de ellos es un archivo legible para el usuario, que contiene un detalle de los eventos que han ocurrido durante la simulación ordenados cronológicamente. Estos eventos pueden ser relativos al uso del vínculo de comunicación, por ejemplo comienzo de transmisión de un frame, fin de transmisión o pérdida del frame, o relativos a los procesos, por ejemplo vencimiento de un timer, ocupación de un buffer, solicitud de transmisión de un frame al hardware de comunicaciones, etc.. El otro archivo de salida contiene en un formato especial, legible para el programa de graficación, todos los eventos de interés para ser visualizados.

2 - Módulo para soporte de programación

Como fue mencionado, este módulo permite la emulación de dos equipos con una funcionalidad muy simple, y un vínculo de comunicación que los conecta. En la figura 2 se muestra la estructura del soporte implementado, sus componentes y su relación con una arquitectura de niveles.

Cada equipo posee un hardware y un software ideales y muy simples, que en conjunto proveen al programador un soporte de programación sencillo, orientado a comunicaciones. Incluida en el hardware del equipo, se encuentra una placa de comunicaciones que es capaz de comunicarse con su par en el otro equipo utilizando el vínculo de transmisión, y con el software de comunicaciones a desarrollar por el usuario, a través de interacciones locales simples y bien definidas que son interceptadas por el software de soporte. En una implementación real, estas interacciones se producirían a traves de librerías de comunicaciones o llamadas al sistema operativo.

El soporte de software que ofrece el equipo, representado en la parte inferior de cada módulo, posibilita al proceso de nivel 2 (en la parte superior), independizarse de las funciones de comunicación de bajo nivel. Para ello, es provisto con un conjunto de funciones las cuales se representan de manera simplificada en la figura 2.

Las funciones relativas al envío y recepción de datos son las siguientes:
Enviar: es una función invocada por el proceso de nivel 2. Esta función pertenece al software de soporte incluido en el equipo y su función es desencadenar el envío del frame que le es pasado como parámetro. Esto implica chequear si la placa de comunicaciones se encuentra libre y en este caso pasar la información a su buffer y solicitar el envío, o, si la placa esta enviando, tomar las previsiones para que al fin del envío corriente se envíe la información recibida.
Recibir: esta función es invocada por el software del equipo, y pertenece al proceso de nivel 2, es decir, debe ser provista por el programador de este nivel. Dependiendo de la programación realizada, esta indicación desencadenara acciones de administración de los buffers de recepción y gestión de timers.
Fin envío: es una indicación provista por el software de soporte al proceso de nivel 2. Indica que el frame del cual se ha solicitado el envío ya ha sido enviado. El proceso de nivel 2 podrá entonces tomar decisiones tales como enviar el siguiente bloque y/o detener ciertos timers.

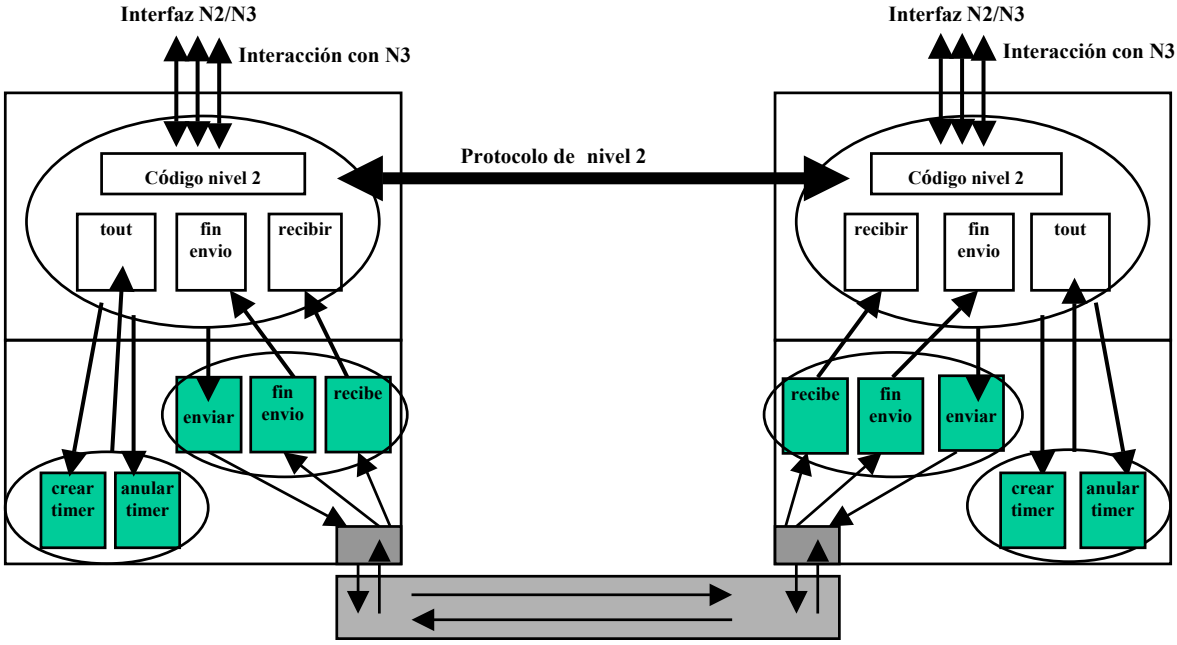

Fig. 2. Soporte para la programación relacionada con los conceptos básicos de arquitecturas de niveles.

Las funciones relativas al manejo de timers permiten crear un evento, cancelar dicho evento o dejar que ocurra, en este caso se invocara a un procedimiento especificado al configurar el timer.

Debe tenerse en cuenta que en una implementación real no es posible tener una delimitación a nivel software entre niveles 1 y 2 desde un punto de vista estricto, ya que el nivel 1 se encuentra integrado con el hardware de comunicaciones, resultando además ineficiente pasar bit por bit al hardware de comunicaciones la información a ser enviada o siendo recibida. De manera similar ha sido desarrollado este soporte de comunicaciones, no permitiendo por lo tanto una delimitación clara desde el punto de vista estricto de una arquitectura de niveles, entre los niveles 1 y 2. Sin embargo, desde el nivel 2 a superiores es posible delimitar en forma estricta la funcionalidad de cada nivel. Esta aparente limitación de la implementación es motivada por la intención de que el soporte provisto sea similar a los soportes reales.. La abstracción en la cual esta basado es un equipo con la funcionalidad indicada por las flechas que cruzan de la parte inferior a superior (y en sentido inverso) en la figura 2. Cabe aclarar que el objetivo de esta implementación es que sobre este

soporte provisto los alumnos implementen el nivel 2 de la arquitectura, y sobre este, niveles superiores.

A continuación se describen los elementos provistos por el soporte de comunicaciones que son accesibles a los procesos de nivel 2 y por lo tanto de interes para el programador de los mismos.

2.1 Buffers

Los buffers son capaces de almacenar bloques, que constituyen las PDUs de nivel 2, y que, desde el punto de vista de uno de los procesos, serán enviadas al otro proceso o bien serán recibidos provenientes del lado remoto.

La información que almacena un buffer esta compuesta de

Datos recibidos o a enviar: estos constituyen la verdadera información que intercambian las entidades de nivel 2, es decir, las PDUs de nivel 2. Estos datos consisten de un string en el cual los campos se separan por el caracter "/", elegido arbitrariamente como delimitador de campos.

Longitud del bloque: la longitud del bloque almacenado en el buffer se considera sólo a efectos de que el soporte que simula la transmisión pueda calcular el tiempo de transmisión. Esta longitud se especifica en bits.

Rótulo asignado al bloque: el rótulo es un elemento adicional que se incorpora al bloque, sólo a efectos de que aparezca identificando al mismo en el programa de visualización. Por ejemplo si el emisor está enviando bloques con ciertos números de secuencia al receptor, los rótulos podrían ser BLOQUE-i, donde i es el número de secuencia de cada bloque.

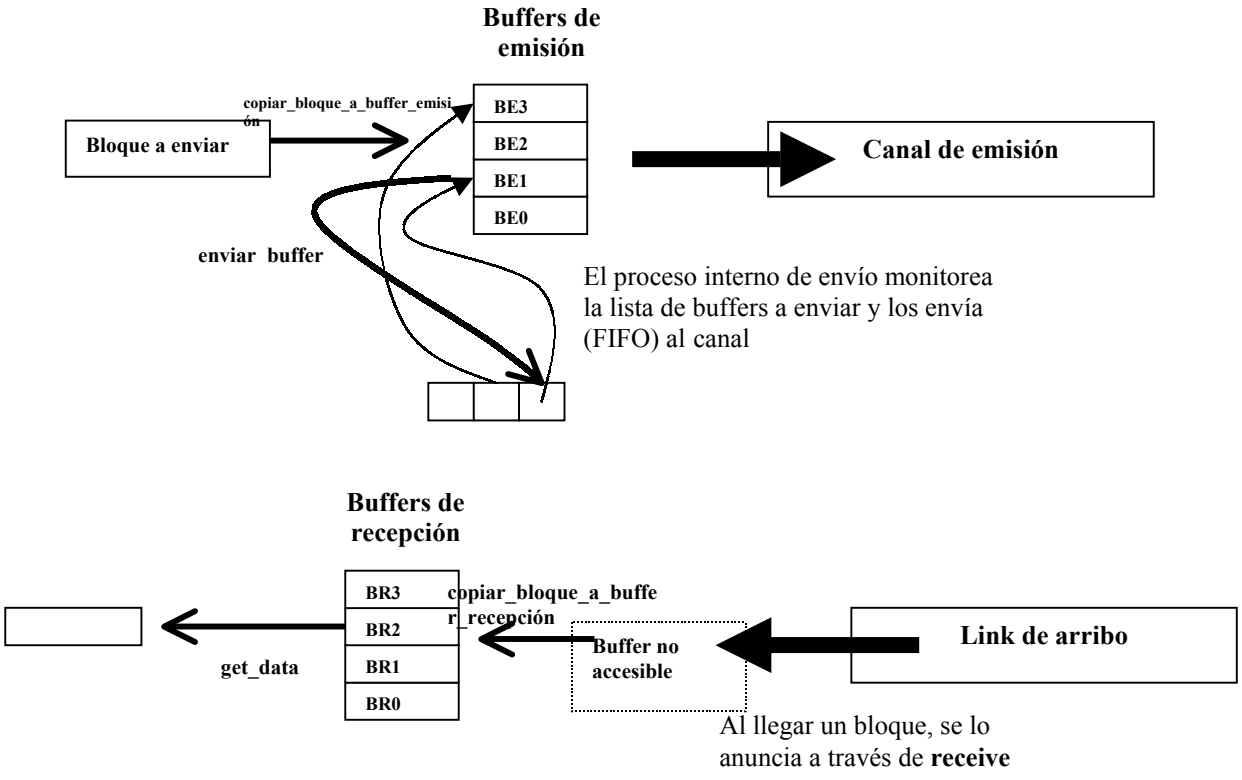

Fig. 3. Administración de los buffers por parte del soporte de comunicaciones. En la parte superior se muestran el proceso de envío y las funciones utilizadas por el proceso de nivel 2. En la parte inferior se muestra el proceso de recepción.

La cantidad de buffers provista por el soporte es configurable por el programador, y puede variar entre 1 y 16 buffers de emisión y entre 1 y 16 buffers de recepción.

Los buffers de emisión se identifican como BEi, donde i es el número de buffer, y los de recepción como BRi.

En la figura 3 se muestran los procesos de envío y de recepción de PDUs y el uso de buffers. El proceso de nivel 2 puede acceder a un buffer de envío a través de la función copiar_bloque-a_buffer_emision. Esta copia de la información al buffer no desencadena el envío por la línea, sino que es necesaria la invocación a la función enviar_buffer, ya que esto permite controlar el funcionamiento de mecanismos de control de flujo, tales como ventanas deslizantes [Tanenbaum, 1996]. El soporte provisto crea una lista de vínculos a los buffers en el orden en que se ha realizado la función enviar_buffer, y procede a interactuar con la placa de comunicaciones para llevar a cabo el envío de la información correspondiente.

En el caso de la recepción de información, el soporte del equipo la almacena en un buffer no accesible al proceso de nivel 2, e inmediatamente le anuncia de la llegada de la información a través de la función receive, que debe ser provista por el nivel 2. El proceso debe entonces leer la información y colocarla en el buffer que corresponda según el protocolo que este siendo utilizado. Esto se lleva a cabo invocando a la función copiar_bloque_a_buffer_recepcion. Posteriormente, el proceso accederá a la información almacenada en los buffers utilizando la función get_data. Debe tenerse en cuenta que la función receive debe ser ejecutada inmediatamente para que la información recibida no sea destruida por el próximo bloque que arribe. El código de la función receive debe ser mínimo para evitar demoras innecesarias.

2.2 Timers

Los timers son elementos que permiten al proceso de nivel 2 tomar ciertas acciones luego de transcurrido un tiempo a partir de la producción de cierto evento. Por ejemplo, retransmitir un bloque si no ha llegado su confirmación de recepción en un cierto tiempo preestablecido.

El programador puede arrancar y detener los timers, y cuando éstos producen el timeout, se invoca automáticamente a un procedimiento específico del proceso de nivel 2, que deberá ser escrito por el programador de acuerdo a la funcionalidad del protocolo que se esté desarrollando. Se provee, de manera automática, un timer por cada buffer. Estos timers se identifican como TEi y TRi según el buffer al que estén asociados.

2.3. Canal de comunicaciones

En el nivel del que se ocupa esta implementación, el vínculo de comunicación que une a dos equipos responde a un canal punto a punto entre ellos. No se trata en este caso de una conexión punto a punto a nivel 4 soportada en general por una red, como ocurre en el caso de dos procesos conectados a través de TCP.

Esto implica que el vínculo simulado produce errores de transmisión, pérdida de frames completos, demoras fijas de transmisión y demora fija de propagación de la señal. Debe tenerse en cuenta que si se tratara de un vínculo soportado por una red, podría ocurrir el fenómeno de duplicación de paquetes, inversión del orden de llegada de los mismos, y demoras de transmisión y propagación variables.

Este vínculo es bidirecional simultáneo, y está representado por dos vínculos unidireccionales, para que sus parámetros puedan ser definidos de manera independiente y obtener así vínculos de comunicación asimétricos tanto respecto a demoras, velocidades o producción de errores.

Los parámetros de cada uno de los vínculos son los siguientes:

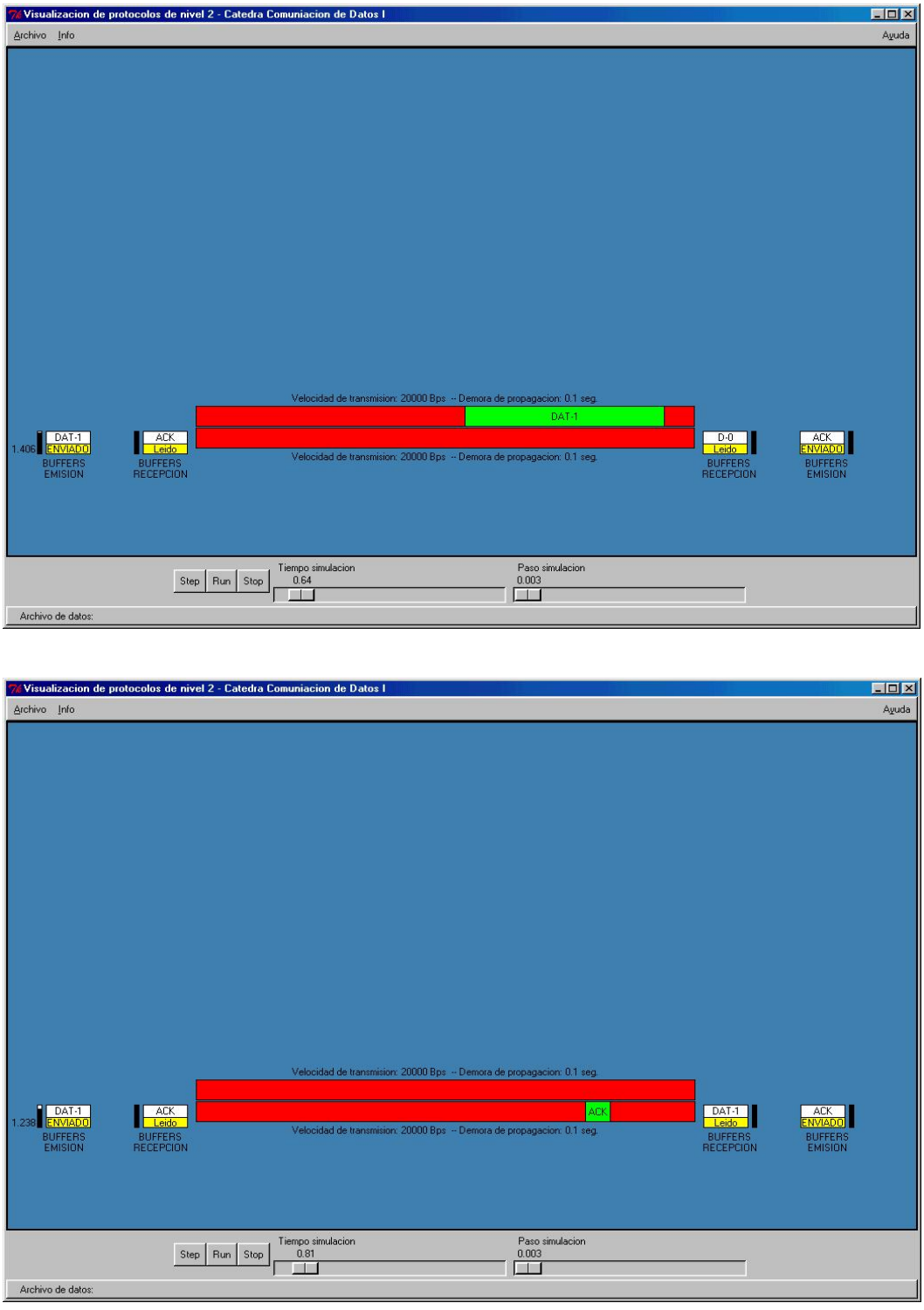

Fig. 4. Vista de la inetraz ofrecida por el modulo de visualización. Se puede observar los equipos que se comunican, el canal bidireccional, buffers y timers.

- Velocidad de transmisión: se especifica en bits por segundo
- Demora de propagación: se especifica en segundos
- Probabilidad de pérdida de bloque: se especifica como un número entre 0 y 1. Una pérdida de bloque significa que el bloque no llegará a destino.

- Probabilidad de error en bloque: se especifica como un número entre 0 y 1. Un error en un bloque significa que el bloque llega a destino, pero es detectado como erróneo por el receptor. Esta característica aún no está implementada.

Además de poder especificar la probabilidad de que los bloques se pierdan en forma aleatoria, es posible configurar la pérdida de un bloque en particular, especificando de manera explícita, antes de proceder a su transmisión, el buffer en el cual se encuentra. Esta característica es útil para producir y visualizar situaciones particulares en el comportamiento de los procesos.

3. Módulo de visualización

El objetivo de este módulo, desarrollado en Incr Tcl [Ulferts, 1995] es permitir el seguimiento de las interacciones entre los procesos que se comunican. Este seguimiento resulta un complemento útil a la salida de debugging en modo texto, y en muchos casos permite entender rápidamente el funcionamiento de un protocolo o detectar las causas de situaciones anormales producidas en el mismo, por ejemplo la duplicación de bloques de datos a causa de configurar timers de retransmisión demasiado pequeños.

Como se puede ver en la figura 4, la visualización permite observar los elementos relevantes para la comunicación en cada equipo (buffers y timers) y el tráfico en el canal bidireccional.

Las opciones ofrecidas al usuario permiten iniciar una visualización y correrla de modo continuo o paso a paso. La velocidad de la visualización y el detalle de las interacciones se obtiene configurando el paso de simulación. Debido a que el módulo de visualización trabaja fuera de línea con la verdadera simulación, es posible desde cualquier punto de la misma, volver el tiempo atrás para observar con más detalle ciertas interacciones.

Los elementos representados son los siguientes:

- Bloques enviados a través del canal de comunicaciones: se representan en el, desplazándose de un equipo a otro los correspondientes frames intercambiados por los procesos. La velocidad del desplazamiento depende de la demora de propagación definida para el canal y el tiempo de transmisión esta en relación directa con la velocidad de transmisión definida y con el tamaño de cada bloque. Para identificar cada uno de los bloques intercambiados, el módulo de soporte de programación permite definir un rótulo para cada uno de ellos. De esta manera, por ejemplo, puede definirse el rotulo DATOS i para cada bloque de datos emitido y ACK i para cada asentimiento enviado.
- Bloques almacenados en los equipos, esperando para ser enviados o para ser leídos en caso de haber sido recibidos desde el canal: Se representan, en cada equipo, la cantidad de buffers definida; cada buffer puede estar vacío, ocupado con un frame esperando ser enviado, con un frame en proceso de transmisión. Estos estados se indican con un rotulo asociado a cada buffer.
- Timers: los timers se representan en cada equipo, asociados al buffer correspondiente. Se indica el tiempo faltante para que se produzca el timeout.

Como se observa en la figura 1, el módulo de visualización opera sobre los archivos de texto generados para ese fin por el módulo de soporte de programación. Este registra información sobre eventos específicos de interés para la visualización, como por ejemplo ocupación de un buffer, comienzo y fin de transmisión de un frame, creación o vencimiento de un timer, etc. Cada uno de estos eventos es almacenado en el archivo mencionado, registrándose el tipo de evento, de acuerdo a él los componentes de la visualización involucrados (canal, buffer, etc), el tiempo de comienzo del evento y el tiempo de fin del evento. Con esta información, el módulo de visualización mostrará en pantalla el estado de la simulación en cada posible instante de la misma, de acuerdo a los parámetros configurados.

4. Escribiendo los procesos de nivel 2

Como ya fue mencionado, el objetivo del soporte desarrollado es proveer un medioambiente de programación para procesos de nivel 2 y superiores que resulte simple de utilizar, conservando las características más importantes que deben tenerse en cuenta al desarrollar aplicaciones sobre plataformas reales. Adicionalmente, se proveen salidas textual y grafica interactiva para la comprobación y análisis del funcionamiento de los protocolos desarrollados.

El escenario de operación del protocolo de nivel 2 consiste en dos equipos conectados a través de un canal bidireccional simultáneo. Los programas a desarrollar, se ejecutarán en dichos equipos, los que les proveen cierta funcionalidad que, en un medioambiente real, podría, entre otras opciones, ser provista por librerías de comunicaciones y por el sistema operativo.

Estas funciones que permiten a los procesos de nivel 2 interactuar con las funciones de más bajo nivel de manera clara y resumida incluyen, entre otras, el envío de bloques, arranque de timers, etc., y son explicadas a continuación. Para su descripción, se las ha agrupado en relación a la emisión de bloques, a la recepción, y al manejo de timers.

4.1 Funciones asociadas a la emisión de bloques

4.1.1 copiar_bloque_a_buffer_emision {longitud rotulo datos id_buffer}

Copia un bloque de datos que se desea enviar a un buffer de emisión. Esta copia no implica que los datos que se almacenen en el buffer sean efectivamente enviados, esto debe indicarse explícitamente a través de la función enviar_buffer.

Los parámetros que recibe son los siguientes:

- longitud: es la longitud en bits de la información a enviar. Se especifica a efectos de determinar el tiempo de transmisión.
- rotulo: es una cadena de caracteres que identificara al frame en la fase de visualización. No tiene ninguna influencia en cuanto al comportamiento de los procesos.
- datos: Son los datos que componen la PDU que será enviada. Estos datos serán interpretados por el proceso par en el equipo remoto. Esta PDU contendrá campos específicos de nivel 2 y podría también contener información propia del nivel superior, no analizada por el nivel 2. Este parámetro es un cadena de caracteres que no debe contener espacios en blanco. Para facilitar la comunicación entre los niveles 2, se conviene en que los diferentes campos de la PDU estén separados por una barra /.
- id_buffer: identifica al buffer de emisión en el cual se desea copiar la información a enviar. Estos buffers se indican como BEi.

4.1.2 enviar_buffer {id_buffer}

Solicita el envío de la información contenida en uno de los buffers de emisión. El efecto es que este buffer sea incorporado a la cola de transmisión, y que sea enviado por el hardware de comunicaciones cuando el canal se desocupe. El tratamiento de los buffers en la cola es FIFO. La separación del proceso de copia de bloque en buffer y el envío, permite que, por ejemplo en caso de retransmisión, no sea necesario generar el bloque nuevamente, sino solicitar su reenvío; de esta manera el soporte responde a conceptos tales como ventana de emisión.El parámetro que recibe es la identificación del buffer de emisión cuyo contenido se desea enviar al proceso remoto.

4.1.3 fin_envio {buffer_id}

Este procedimiento es invocado por el soporte en el equipo cuando el hardware de comunicaciones termina de enviar un bloque. El parámetro recibido es el nombre del buffer de emisión en el cual se halla el bloque enviado. Las acciones que tomará el proceso de nivel 2 dependerán del protocolo en particular, por ejemplo, al recibir esta indicación, se podría enviar el siguiente bloque.

4.1.4 perder_bloque {id_buffer}
Esta función, si bien está relacionada con el envío de información, no es propia de la programación de estos procesos. Se invoca para forzar la perdida del contenido de un buffer cuando sea enviado. Se utiliza para comprobar el funcionamiento del protocolo ante posibles pérdidas de bloques en la línea. Al invocarse esta función, se pierde sólo el próximo envío del contenido del buffer, es decir, si en el buffer se copió el bloque 1 y luego (o antes) se invoca a esta función, la transmisión de bloque 1 se perderá en el canal, pero si luego se retransmite bloque 1 o se envía otro bloque con enviar_bloque, esta transmisión será correcta. El parámetro que recibe es la identificación del buffer de emisión cuyo contenido se desea perder en el proceso de envío.

4.2 Funciones asociadas a la recepción de bloques

4.2.1 receive {}
Esta función debe ser escrita por el programador del proceso de nivel 2. Es invocada cuando se recibe un bloque a través de la línea. En general, aquí deberá copiarse el bloque recibido a uno de los buffers de recepción.

4.2.2 copiar_bloque_a_buffer_recepcion {id_buffer}
Copia el bloque recibido por la línea y puesto a disposición del proceso de nivel 2 a un buffer de recepción. Esta copia permitirá luego al proceso de nivel 2 procesar el bloque de la manera que corresponda. Por ejemplo, analizar sus campos para determinar si se trata de información a ser entregada al nivel superior o información de control intercambiada con su proceso par. Esta función deberá ser invocada inmediatamente después de que es invocada la función receive del proceso de nivel 2 por parte del soporte provisto. Debe observarse que es posible que al invocarse la función receive el proceso de nivel 2 decida almacenar la información recibida en un buffer de recepción y además procesar esta información. Si bien esta manera de programar el nivel 2 es correcta, se hace énfasis en el hecho de que en un sistema con múltiples procesos, la operación de recepción (extraer la información del buffer de la placa de red y almacenarla en un buffer del sistema operativo), debe insumir el menor tiempo posible para no bloquear otras interrupciones que podrían producirse. Es por este motivo que al ser invocada la función receive, solo deberá copiarse la información a un buffer, y luego, de manera asincrónica se procesara la información recibida. Una manera de producir este efecto se muestra en el ejemplo. El parámetro que recibe es la identificación del buffer de recepción en el cual se desea copiar la información recibida.

4.2.3 get_data {id_buffer}
Permite obtener los datos (es decir, la PDU de nivel 2) contenidos en un buffer de recepción. El parámetro que recibe es la identificación del buffer de recepción al cual se desea acceder al contenido.

4.3 Funciones asociadas al manejo de timers
Se proveen dos tipos de timers, aquellos provistos automáticamente por el soporte de programación y aquellos que deben ser creados explícitamente por el programador.
Los primeros son los timers asociados a cada uno de los buffers de emisión y recepción, los otros no tienen función especifica, pudiendo ser utilizados para funciones tales como piggybacking y otras dependientes de cada protocolo en particular. La cantidad máxima prevista de este ultimo tipo de timers es 5, y deben ser creados antes del comienzo de la simulación

4.3.1 create_timer {timer_id}

A través de la invocación a este método se crea un nuevo timer. Este timer es uno de los cinco adicionales a los asociados a los buffers que son provistos por defecto. El parámetro que se envía es el nombre dado al timer. La función retorna el objeto timer instancia do.

4.3.2 start_timer {timer_id tiempo}
Esta función produce el arranque del timer identificado como timer_id. Dicho timer se configura con un valor inicial dado por el parámetro tiempo. Los parametros son el timer que se desea arrancar y la duración del mismo.

4.3.3 cancelar_timer {timer_id}
La invocación a este procedimiento provoca la cancelación del timer cuyo nombre se envía como parámetro. Esta cancelación implica que no se realizará la invocación a la función timeout; por otro lado, el timer queda en estado inactivo.

4.3.4 timeout {timer id}
Invocado por el soporte de programación cuando vence un timer. Este procedimiento debe ser escrito por el usuario, debiendo contener la funcionalidad necesaria en cada caso; por ejemplo, si se trata de un timer asociado a un buffer de emisión, debería reenviarse el contenido del buffer e incrementar el contador de reintentos de envío. La función timeout es invocada cualquiera que sea el timer que ha vencido, debiendo identificar el proceso de nivel 2 de que timer se trata en función del parámetro id. El parámetro recibido es la identificacion del timer que venció.

4.3.5 get_buffer_timer {buffer_id}
Permite obtener el objeto correspondiente al timer asociado al buffer de emisión o recepción cuyo nombre se envía como parámetro.

5. Conclusiones y trabajos relacionados

El soporte desarrollado ha cumplido satisfactoriamente con los objetivos fijados. Se experimentó con él durante el año 2002, trabajando con aproximadamente 50 grupos de 4 alumnos cada uno. En primer término se puso énfasis en la asimilación del soporte por parte de los alumnos a través de la programación de procesos muy simples que permitieron el intercambio de datos, pero privilegiando la definición de interfaces y primitivas entre el nivel 2 y el superior y haciendo un paralelo entre las facilidades provistas y las que realmente provee un sistema operativo.

Posteriormente y por cuestiones de tiempo y organización, se trabajo con un numero más reducido de alumnos haciendo énfasis en la programación de protocolos simples y la detección de anomalías por fallas en su diseño.

Se planea desarrollar protocolos ejemplo cuyo objetivo sea que los alumnos comprendan el funcionamiento de los mecanismos básicos del nivel 2 (por ejemplo mecanismos de ventana deslizante) y los errores de diseño que pueden ocurrir (números de secuencia insuficientes, timers demasiado pequeños, etc.), a través del módulo de visualización. Es necesario por otra parte, el diseño de trabajos prácticos que integren el aspecto interfaces y el aspecto programación del nivel 2, a efectos de reducir el tiempo empleado por los alumnos en su realización.

En un contexto más amplio, se está trabajando en el desarrollo de soportes similares relativos a diferentes temas del area comunicación de datos. Todos ellos están siendo desarrollados en Tcl en su parte que provee el soporte de programación, y en Itcl para la parte grafica.

Tanto el codigo correspondiente al soporte desarrollado como los ejemplos provistos a los alumnos pueden solicitarse al autor via email.

6. Bibliografía

[Fall, 2000] K. Fall (ed), "Ns Notes and Documentation" VINT Project, UC Berkeley, LBL, USC/ISI, Xerox PARC, March 2000.

[Halsall, 1992], "Data Communications, Computer Networks and Open Systems", Halsall, F., Addison-Wesley, 1992.

[Hebrawi, 1993] "OSI Upper Layer Standards and Practices", B. Hebrawi, McGraw-Hill, 1993

[Osterhout,1994] "Tcl and the Tk Toolkit", John K. Osterhout, Addison Wesley, 1994.

[Rose, 1990] "The Open Book. A Practical Perspective on OSI", Marshall Rose, Prentice Hall, 1990.

[Tanenbaum, 1996] "Computer Networks" 3rd edition, Tanenbaum, A., Prentice Hall, 1996.

[Ulferts, 1995] "[incr Widgets] An Object Oriented Mega-Widget Set", Mark L. Ulferts, Usenix Tcl Workshop 95, Toronto, Canada, July 8, 1995.