REPRESENTACION DE MAQUINAS DE ESTADOS EN PROTOCOLOS DE COMUNICACIONES

Joaquín Luque, Ana V. Medina

RESUMEN

En este trabajo se presentan diversas técnicas que permiten pasar de la descripción de un protocolo con máquinas de estados a su representación en un computador. Se discuten aspectos como el tratamiento de eventos, la representación de las esperas, la descripción de las transiciones y la representación de los temporizadores. Las técnicas aquí presentadas han sido aplicadas con éxito en la construcción automática de convertidores de protocolos.

INTRODUCCIÓN

El crecimiento y la complejidad de las actuales redes de computadores hace que el diseño y construcción de los protocolos de comunicaciones que operan en ellas deba realizarse de forma eficaz y segura. Para ello la Ingeniería de Protocolos preconiza un amplio abanico de técnicas que pueden clasificarse, principalmente, en dos grandes escuelas: técnicas basadas en máquinas de estados y técnicas basadas en lógica temporal. De hecho la comunidad internacional reconoce estas dos grandes tendencias al normalizar 3 lenguajes de especificación de protocolos: Estelle (ISO, 1989) y SDL (CCITT, 1988) basados en máquinas de estados; y LOTOS (ISO, 1989) basado en lógica temporal.

Este trabajo se encuadra dentro de la primera de estas dos grandes líneas, que nos atreveríamos a calificar de mayoritaria. El problema que nos planteamos y al que realizamos algunas aportaciones, es el del paso de la descripción del protocolo mediante un conjunto de máquinas de estados a la representación de dichas máquinas en un código secuencial mediante un lenguaje de programación convencional. Discusiones previas sobre este problema pueden encontrarse en distintas publicaciones recientes (Svobodova, 1989; Thekkath et al., 1993; Luque y Medina, 1994; Luque et al., 1994). A este respecto son 4 las cuestiones en las que conviene fijarse: a) tratamiento de los eventos; b) representación de las esperas; c) descripción de las transiciones; y d) representación de los temporizadores (timers). A cada una de estas cuestiones dedicaremos los próximos apartados.

TRATAMIENTO DE LOS EVENTOS

Los eventos de entrada a una máquina provienen de otras máquinas o de la interfaz con el exterior. En general estos eventos son almacenados en una cola con disciplina de FIFO (First-In, First-Out: el primer evento que entra es el primer evento que sale). De esta forma al acceder a la cola se

extrae siempre el más antiguo (fig. 1). Este criterio puede ser matizado, si el protocolo lo requiere, por la existencia de un parámetro de prioridad, de tal forma que se tome siempre en consideración el evento más prioritario, sea o no el más antiguo (fig. 2).

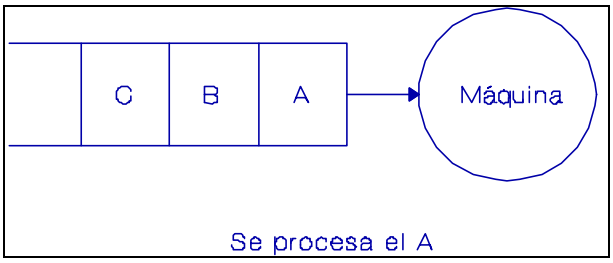

Fig. 1: Tratamiento de eventos con FIFO

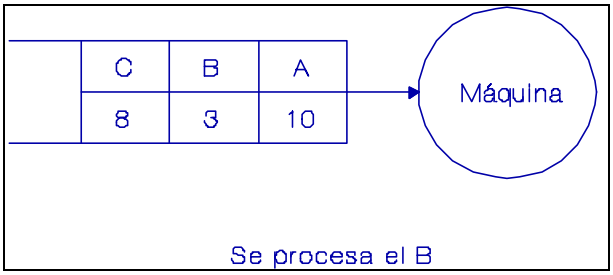

Fig. 2: Tratamiento de eventos con FIFO priorizado

A cualquiera de los dos mecanismos anteriores puede añadirse un factor de "no determinismo". Esto consiste en extraer de la cola un evento por criterio aleatorio de entre los de mayor prioridad (fig. 3). El acceder aleatoriamente a la cola tiene sobre todo interés en procesos de validación, verificación, análisis de prestaciones y pruebas, ya que permite explorar combinaciones de eventos muy diversos.

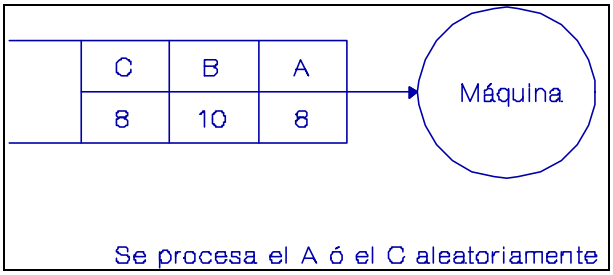

Fig. 3: Tratamiento de eventos no determinista

Otro aspecto a considerar en el tratamiento de los eventos es el efecto que tiene sobre la cola el proceso de lectura. Una solución consiste en considerar que la lectura de la cola da información del evento que hay que procesar pero deja inalterada la cola (fig. 4). De esta forma si la máquina, por encontrarse en el estado adecuado, toma en consideración el evento, debe realizarse una operación explícita de extracción del evento de la cola. Si por el contrario la máquina no pudiese procesar el evento, éste seguiría estando en la cola. La máquina debe entonces leer de nuevo la cola para ver si existe algún otro evento que sí pueda ser procesado. Por tanto, los eventos no tratados en un estado son conservados. Este sistema requiere: a) una orden explícita de extracción de eventos de la cola; b) un mecanismo más complejo de lectura de la cola; c) una especificación de los eventos que, por carecer de sentido, deban ser descartados (borrados de la cola) en cada uno de los estados.

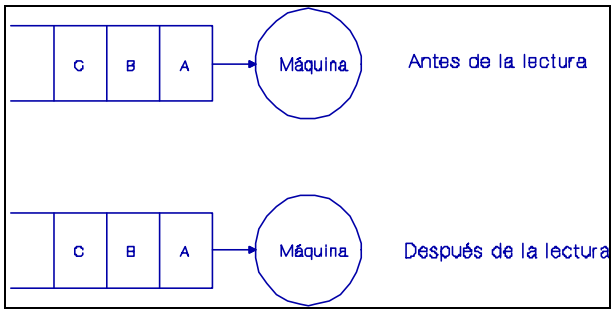

Fig. 4: Lectura de eventos sin extracción

La alternativa es que la lectura de la cola suponga automáticamente la extracción del evento (fig. 5). Si la máquina no lo puede procesar en ese estado pero el evento no debe ser rechazado es necesario que se proceda a un almacenamiento temporal interno al módulo y a una posterior consideración de los eventos almacenados. Por tanto, los eventos no tratados en un estado son descartados. Este sistema requiere: a) un mecanismo de gestión del almacenamiento interno de eventos; b) una especificación de los eventos que, por no poder ser tratados en un estado, deban ser almacenados internamente.

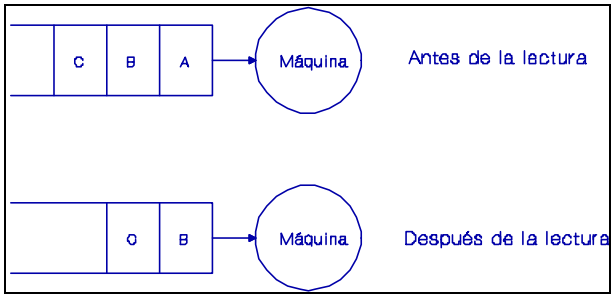

Fig. 5: Lectura de eventos con extracción

REPRESENTACIÓN DE LAS ESPERAS

Existen principalmente tres métodos de representar las situaciones de espera de las máquinas de estados y el correspondiente control del ciclo de ejecución de las mismas. La primera de estas formas es la que se utiliza cuando cada máquina de estados se representa mediante una tarea independiente, siendo el sistema operativo el

que controla la ejecución de las distintas tareas. En este caso la espera se ejecuta haciendo uso de las facilidades del sistema operativo para suspender la ejecución de tareas y reanudarlas cuando se produce algún suceso predeterminado (expiración de un timer, activación de un semáforo, etc.). En este caso la estructura, para un ejemplo con dos máquinas, sería la siguiente:

```
main() /* Máquina A */
{
    for (;;)
    {
        espera_a();
        lee_evento_a();
        procesa_evento_a();
    }
}

main() /* Máquina B */
{
    for (;;)
    {
        espera_b();
        lee_evento_b();
        procesa_evento_b();
    }
}
```

Cuando las máquinas se representan mediante rutinas de una misma tarea, y por tanto se necesita un gestor de módulos específico, la espera de una máquina puede representarse de dos formas. En primer lugar, puede utilizarse una salida de la rutina que codifica el módulo y la devolución del control al gestor de módulos. En este caso la estructura sería esquemáticamente la siguiente:

```
main() /* Gestor de módulos */
{
    for (;;)
    {
        maquina_a();
        maquina_b();
    }
}

maquina_a() /* Máquina A */
{
    lee_evento_a();
    procesa_evento_a();
}

maquina_b() /* Máquina B */
{
    lee_evento_b();
    procesa_evento_b();
}
```

La versión aquí expuesta presenta un gestor de módulos bastante simple. Algunas modificaciones comunes al gestor incluyen: a) la transferencia de control a las máquinas únicamente cuando tienen algún evento que tratar; y b) la activación de las máquinas en orden aleatorio, principalmente en simulación.

Por último, y también cuando las máquinas se representan mediante módulos de una tarea única, pueden representarse las esperas mediante transferencias de control entre corrutinas. Para ello, en primer lugar, el gestor de módulos arranca todas y cada una de las máquinas de estados. Este proceso es similar a la activación de una tarea desde el sistema operativo. Para arrancar un módulo se reserva un espacio en la pila (moviendo el Stack Pointer SP) y se llama a la rutina que representa la máquina. Dicha rutina, al igual que una tarea del sistema operativo, no debe finalizar (devolver el control con un "return") sino que cuando quiera realizar una espera, deberá transferir el control al gestor de módulos mediante la técnica de corrutinas. Una vez arrancadas todas las máquinas, el gestor de módulos les va transfiriendo control también según la técnica de corrutinas. La estructura genérica es la siguiente:

```
main() /* Gestor de módulos */
{
    arranca_maquina_a();
    arranca_maquina_b();
    for (;;)
    {
        corrutina_maquina_a();
        corrutina_maquina_b();
    }
}

maquina_a() /* Máquina A */
{
    for (;;)
    {
    espera_a();
    lee_evento_a();
    procesa_evento_a();
    }
}

maquina_b() /* Máquina B */
{
    for (;;)
    {
        espera_b();
        lee_evento_b();
        procesa_evento_b();
    }
}

arranca_maquina_a()
{
    reserva_espacio_pila();
    valor=setjmp(punto_salto_gestor_modulos);
    if (valor==0) maquina_a();
```

```
}

arranca_maquina_b()
{
    reserva_espacio_pila();
    valor=setjmp(punto_salto_gestor_modulos);
    if (valor==0) maquina_b();
}

corrutina_maquina_a()
{
    valor=setjmp(punto_salto_gestor_modulos);
    if (valor==0)
        longjmp(punto_salto_maquina_a,1);
}

corrutina_maquina_b()
{
    valor=setjmp(punto_salto_gestor_modulos);
    if (valor==0)
        longjmp(punto_salto_maquina_b,1);
}

espera_maquina_a()
{
    valor=setjmp(punto_salto_maquina_a);
    if (valor==0)
         longjmp(punto_salto_gestor_modulos,1);
}

espera_maquina_b()
{
    valor=setjmp(punto_salto_maquina_b);
    if (valor==0)
        longjmp(punto_salto_gestor_modulos,1);
}
```

La reserva de espacio en la pila es una condición intrínseca al uso de corrutinas. En efecto, puede observarse la distribución de la pila sin reserva de espacio (fig. 6), con los conflictos que acarrea, y con reserva de espacio (fig. 7). Es de notar también que las funciones "setjmp" y "longjmp" con las que se han implementado en C la técnica de corrutinas son estándares en ANSI C y en UNIX, por los que pueden encontrarse en gran número de plataformas y sistemas operativos. Otros lenguajes proporcionan mecanismos similares. En algunos, sin embargo, no es posible utilizar este recurso.

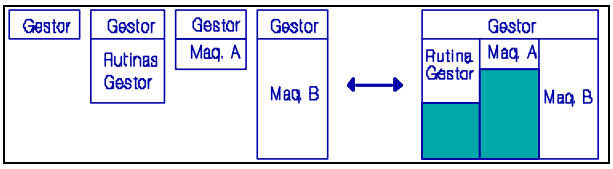

Fig. 6: Pila con corrutinas sin reserva de espacio

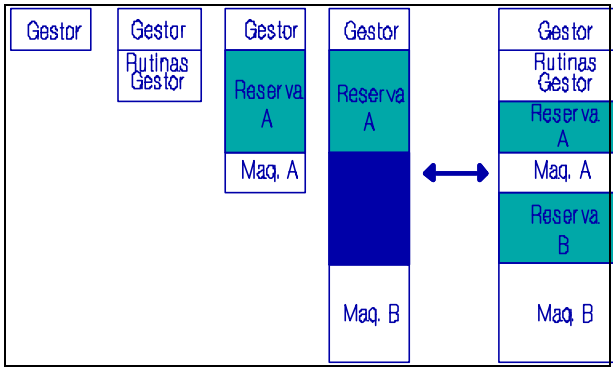

Fig. 7: Pila con corrutina y reserva de espacio

DESCRIPCIÓN DE LAS TRANSICIONES

Las transiciones pueden ser expresadas de dos formas diferentes: mediante tablas o mediante código. En muchos documentos de especificación de protocolos se expresan las máquinas de estados mediante tablas (estado, evento, condición). Para cada posible combinación de estado y evento, y para cada condición adicional que se imponga, se expresan tabularmente las acciones a realizar y el próximo estado. La otra forma de representar las transiciones es expresarlas directamente en el código usando sentencias anidadas de selección ("switch" en C o similares). Estas sentencias pueden tomar una estructura de (estado, evento, condición), o bien de (evento, estado, condición). Con este sistema el protocolo se ejecuta a mayor velocidad, ya que no hay que ir interpretando las tablas de transiciones. Por el contrario el método de tablas es muy adecuado cuando se realiza una generación automática de código. Una cuestión que debe tenerse en cuenta, sea cual sea el método de representación de transiciones elegido, es la necesidad de representar también el proceso o la transición de inicialización de la máquina de estados.

REPRESENTACIÓN DE LOS TEMPORIZADORES

Uno de los elementos presentes con gran frecuencia en las máquinas de estados que describen protocolos de comunicaciones es el de los temporizadores o timers. Si bien son similares a otros eventos, su tratamiento presenta algunas singularidades que conviene señalar. Fundamentalmente son dos las formas de abordar la representación de los temporizadores: a) mediante proceso externo al módulo; o b) mediante proceso interno al módulo. En el primero de estos enfoques (fig. 8) la activación o cancelación de un timer se traducen en una actuación de la máquina de estados hacia un proceso externo. En este proceso externo el temporizador puede ser tratado de dos formas:

1) Una posibilidad consiste en almacenar el

instante en el que expira el timer e irlo comparando periódicamente (en cada activación del proceso de gestión de timers) con el valor del reloj (real o simulado). Una vez superado el plazo impuesto por el timer se genera el evento correspondiente en la máquina original.

2) La segunda posibilidad de tratamiento del timer en el proceso externo es solicitar al sistema operativo que avise (active el proceso de gestión de timers) cuando expire el evento. Una vez que esto ocurre se genera el evento correspondiente en la máquina original.

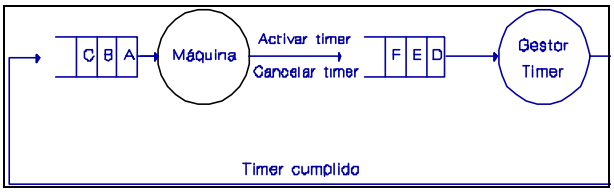

Fig. 8: Gestión externa de timers

La representación de los temporizadores mediante proceso interno al módulo (fig. 9) supone que la cola (o colas) de entrada de eventos a la máquina tiene un parámetro que indica la "hora de entrada en vigor" de cada evento. De esta forma un evento normal tendrá hora de entrada en vigor igual a cero o igual a la hora en la que se generó, con lo cual puede ser procesado inmediatamente. Por el contrario la activación de un timer supone la inclusión en la cola de un evento de expiración del timer con una hora de entrada en vigor igual a la hora actual más la duración del timer. Hasta que el reloj (real o simulado) no supera la hora de entrada en vigor de un evento, éste no es tomado en consideración. El proceso de la cancelación de un timer supone la extracción del evento de la cola.

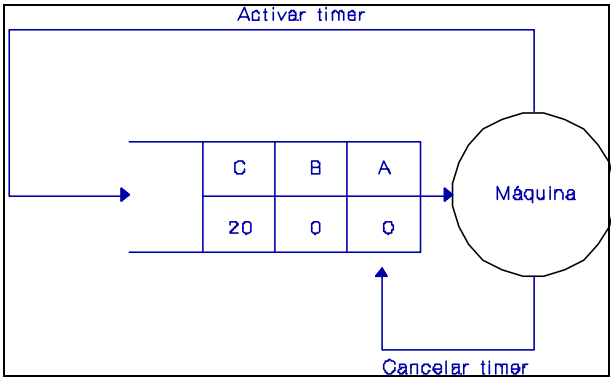

Fig. 9: Gestión interna de timers

CONCLUSIÓN

Se han presentado cuatro de los principales aspectos que conciernen a la representación de las máquinas de estados y su aplicación a la construcción de protocolos de comunicaciones. Se han aplicado las técnicas aquí descritas a situaciones reales en el desarrollo de protocolos de telecontrol de redes eléctricas. En concreto se han generado de forma automática convertidores de protocolos que se han instalado y probado en funcionamiento real, con resultados altamente satisfactorios.

Este trabajo ha sido parcialmente financiado por el Ministerio de Industria y Energía de España y galardonado por el gobierno de Andalucía (España) con el II Premio de Investigación de aplicación de la informática al control de redes eléctricas.

REFERENCIAS

ISO, ESTELLE: A formal description technique based on extended state transition model, ISO 9074, 1ª edición, International Organization for Standardization, Ginebra, (1989).

ISO, LOTOS: A formal description technique based on the temporal ordering of observational behaviour, ISO 8807, 1ª edición,International Organization for Standardization, Ginebra, (1989).

CCITT, SDL: Specification and Description Language, Z-100, Comite Consultivo Internacional Telefónico y Telegráfico, Ginebra, (1988).

Luque, J. et al., Formal Techniques Improve Connectivity in Supervisory Systems, IEEE, Computer Applications in Power, 7 (2), en prensa, (1994).

Luque, J. y Medina, A. V., Desarrollo de Protocolos. Servicio de Publicaciones del Dpto. Tecnología Electrónica. Universidad de Sevilla. Sevilla, España (1994).

Svobodova, L., Implementing OSI Systems. IEEE Journal on Selected Areas in Communications: 7 (7), 1115-1130 (1989).

Thekkath, C. A. et al., Implementing Network Protocols at User Level. IEEE/ACM Transactions on Networking: 1 (5), 554-565 (1993).