UNA METODOLOGIA Y AMBIENTE DE PROGRAMACION DE SISTEMAS DISTRIBUIDOS A PARTIR DE REDES DE NUTT

M. Consuelo Franky de Toro
Universidad de los Andes
Bogotá - *Colombia*

1. INTRODUCCION

La programación de procesos de un sistema distribuido que va a operar sobre una red de computadores exige una metodología diferente a la que se utiliza para un sistema centralizado. En efecto, a diferencia de los procesos de un sistema centralizado, los procesos de un sistema distribuido se conciben como servidores permanentes y su sincronización se realiza por intercambio de mensajes y no por memoria compartida [Franky 85].

En la Universidad de los Andes se ha venido desarrollando y uniformizando una metodología de programación de sistemas distribuidos. En este artículo se presenta dicha metodología, la cual ya ha sido aplicada en la implementación de prototipos de investigación tales como los siguientes : sistema distribuido de transacciones bancarias, sistema de bibliotecas heterogéneas interconectadas, sistema manejador general de bases de datos distribuidas, etc.

La metodología de programación de sistemas distribuidos descrita en este artículo se basa en la utilización de la técnica gráfica de **"Redes de Nutt"**. Programar un sistema distribuido con esta metodología implica dos etapas :

- Especificar en redes de Nutt el conjunto de procesos que hacen parte del sistema, los cuales deben comunicarse únicamente a través de mensajes.

- Transformar las redes de Nutt de los procesos a un programa único que pueda ejecutarse en cada uno de los sitios (nodos) del sistema. Este programa debe simular el paralelismo de los procesos que van a ejecutarse en un mismo sitio.

Correspondiente a esta metodología se ha desarrollado en la Universidad de los Andes un software que facilita la realización de

las dos etapas mencionadas. Este software llamado "**Ambiente de Programación de Sistemas Distribuidos**" consta de dos módulos principales :

- Un Editor gráfico que permite describir y editar las redes de Nutt de los procesos del sistema distribuido que se quiere diseñar.

- Un Compilador que transforma las redes de Nutt previamente descritas con el Editor en un programa PASCAL que puede ejecutarse en cada uno de los microcomputadores de una red local particular. Este programa está estructurado de forma tal que permite la simulación de paralelismo de los procesos en computadores monoproceso (MS-DOS, CP/M).

A continuación se describe cada una de las etapas que implica la metodología de programación de sistemas distribuidos basada en Redes de Nutt.

2. ESPECIFICACION EN REDES DE NUTT DE LOS PROCESOS DE UN SISTEMA DISTRIBUIDO

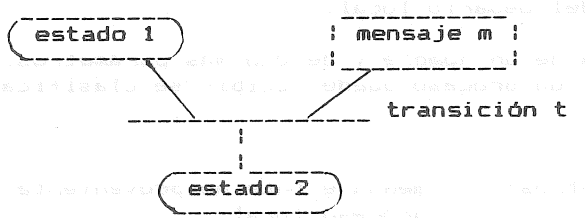
Las redes de Nutt [Nutt 73, Rolin 80, Franky 82] son una técnica gráfica derivada de las Redes de Petri [Peterson 77] que permite describir para cada estado de un proceso:

- los mensajes que puede recibir de otros procesos (locales o remotos),
- las acciones asociadas (acceso a datos locales, evaluación de condiciones o emisión de nuevos mensajes),
- el nuevo estado al cual pasa.

Esta técnica es muy adecuada para describir procesos distribuidos cuya sincronización no se realiza por memoria compartida sino por intercambio de mensajes. En este caso los procesos se consideran permanentes, a la espera de mensajes que los obligan a realizar acciones y pasar a esperar nuevos mensajes.

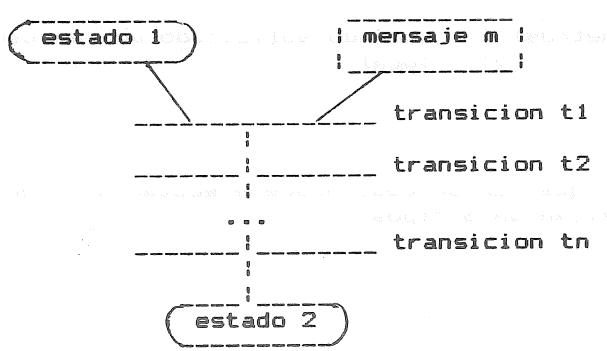
2.1 CONVENCIONES

Una Red de Nutt básica se representa con el siguiente grafo:

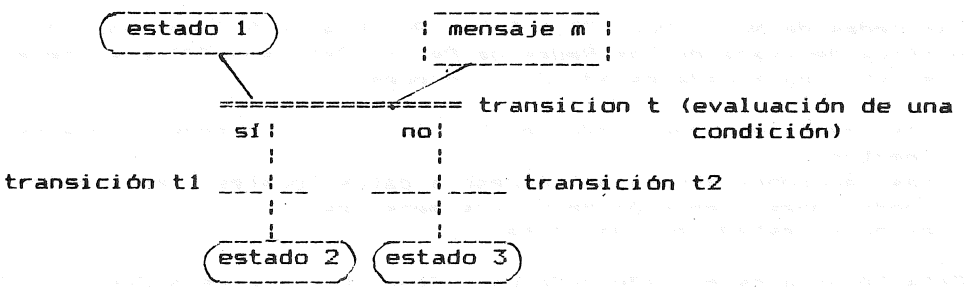


Cuando el proceso descrito se encuentra en el estado 1 y recibe el mensaje m, realiza las acciones asociadas a la transición t y pasa al estado 2.

En lugar de una sola transición t se puede considerar una serie de transiciones t1,...,tn que el proceso debe realizar al recibir el mensaje m :



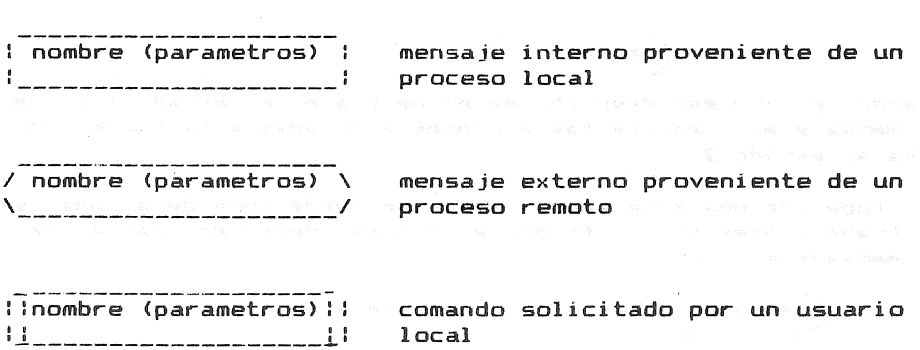
Además se pueden tener ramificaciones en una red de Nutt dependiendo de la evaluación de variables. Por ejemplo:



Un **proceso** se describe entonces a través de **un conjunto de redes de Nutt** que cubran todos los estados, mensajes que puede recibir y transiciones que debe realizar el proceso en cuestión.

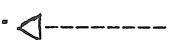
Un proceso se ejecuta en un sitio específico dentro de un sistema distribuido y puede recibir mensajes de procesos locales (que se ejecutan en el mismo sitio) o de procesos remotos (que se ejecutan en sitios remotos). Si se trata de un proceso que interactúa con el usuario, puede recibir también mensajes (llamados comandos) provenientes del usuario local.

Cada mensaje consta de un nombre y de 0 o más parámetros.
Los mensajes que un proceso puede recibir se clasifican en 3 tipos:

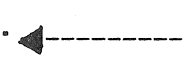


Las transiciones que un proceso puede ejecutar al recibir un mensaje se clasifican en 6 tipos:

acción local a (asignación, procedimiento o función que no involucra emisión de mensajes)



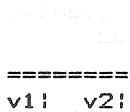
envio de un mensaje a un proceso local p



envio de un mensaje a un sitio externo s

difusión de un mensaje a un conjunto cp de procesos locales (si no se especifica cp se asume una difusión total a todos los procesos locales)

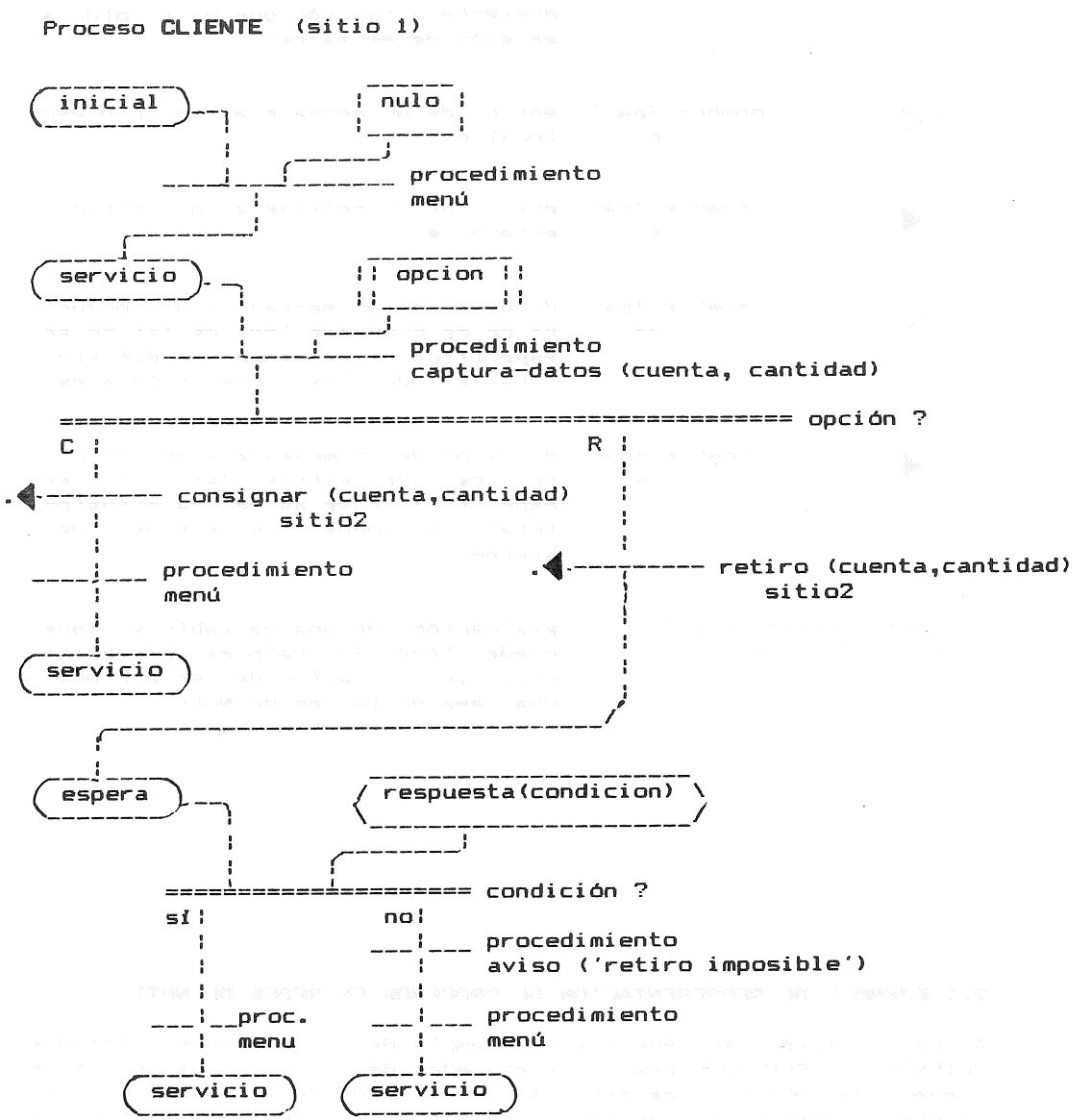
difusión de un mensaje a un conjunto cs de sitios (si no se especifica cs se asume una difusión total a todos los sitios del sistema)



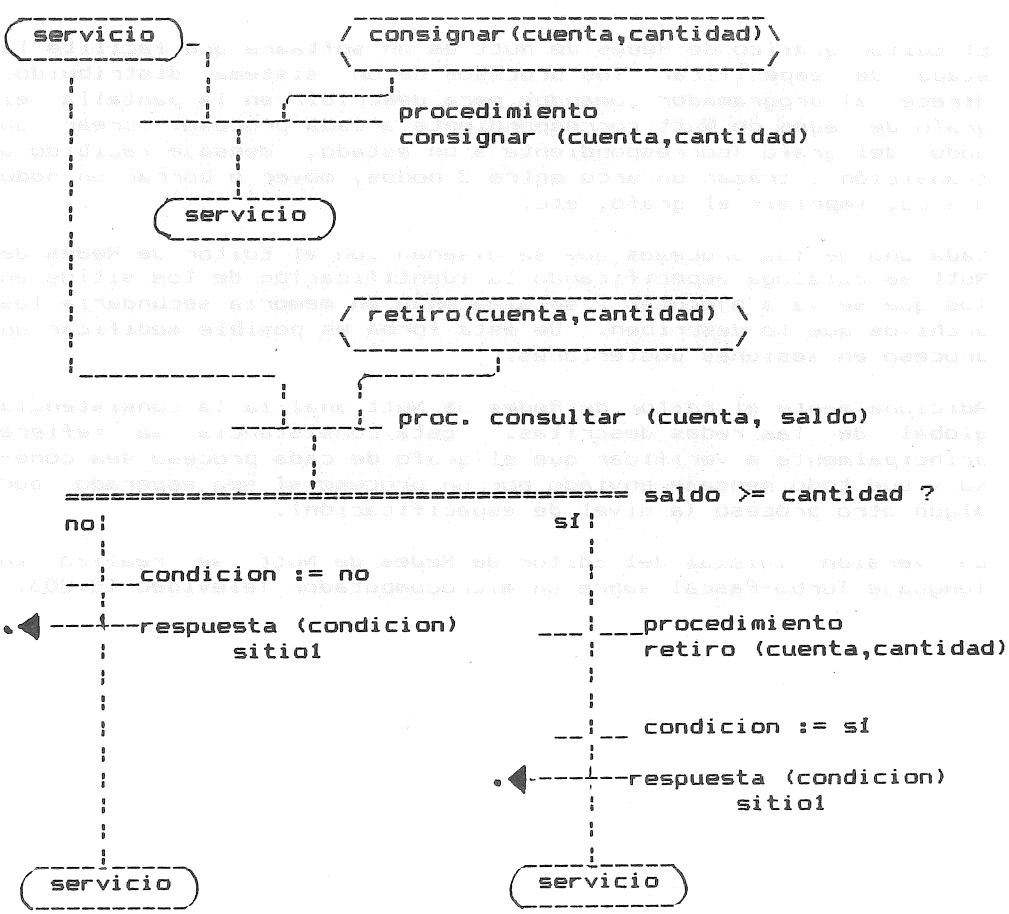
evaluación de una variable v (que puede tener n valores posibles: v1,...,vn) a fin de seleccionar una rama de la red de Nutt

2.2 EJEMPLO DE REPRESENTACION DE PROCESOS EN REDES DE NUTT

A continuación se muestra un ejemplo de 2 procesos llamados CLIENTE y SERVIDOR descritos en redes de Nutt. Los procesos tienen como objetivo permitir a los clientes de un banco realizar consignaciones y retiros sobre cuentas corrientes: CLIENTE (en un sitio 1) recibe los pedidos e interactúa con los usuarios y SERVIDOR (en un sitio 2) actualiza las cuentas corrientes. Los 2 procesos se sincronizan intercambiándose mensajes :


Proceso CLIENTE (sitio 1)
inicial
nulo
procedimiento
menú
servicio
opcion
procedimiento
captura-datos (cuenta, cantidad)
opción ?
C
R
consignar (cuenta,cantidad)
sitio2
procedimiento
menú
retiro (cuenta,cantidad)
sitio2
servicio
espera
respuesta(condicion)
condición ?
sí
no
procedimiento
aviso ('retiro imposible')
proc.
menú
procedimiento
menú
servicio
servicio

Proceso **SERVIDOR** (sitio 2)



Explicación: Inicialmente el proceso CLIENTE muestra un menú por pantalla para poder recibir las demandas del usuario (el mensaje especial "nulo" significa que el proceso no necesita recibir ningún mensaje para realizar esta transición inicial). Cuando el proceso CLIENTE recibe una demanda de consignación (opción C), la envía al sitio 2 (para que sea procesada por el proceso SERVIDOR) y sigue atendiendo nuevas demandas; cuando el proceso CLIENTE recibe una demanda de retiro (opción R) la envía al sitio 2, como en el caso anterior, pero antes de recibir nuevas demandas espera la respuesta correspondiente. Si el proceso SERVIDOR determina que un retiro no puede efectuarse (saldo insuficiente) envía al sitio 1 una respuesta negativa para que el proceso CLIENTE avise al usuario.

2.3 EL EDITOR DE REDES DE NUTT [Pérez 85]

El Editor gráfico de Redes de Nutt es un software que facilita la etapa de especificar los procesos de un sistema distribuido. Ofrece al programador comandos para describir en la pantalla el grafo de redes de Nutt correspondiente a cada proceso: crear un nodo del grafo (correspondiente a un estado, mensaje recibido o transición), trazar un arco entre 2 nodos, mover o borrar un nodo o arco, imprimir el grafo, etc.

Cada uno de los procesos que se diseñan con el Editor de Redes de Nutt se cataloga especificando la identificación de los sitios en los que se va a ejecutar y se almacenan en memoria secundaria los archivos que lo describen. De esta forma es posible modificar un proceso en sesiones posteriores.

Adicionalmente el Editor de Redes de Nutt analiza la consistencia global de las redes descritas. Esta consistencia se refiere principalmente a verificar que el grafo de cada proceso sea conexo y que todo mensaje enviado por un proceso sí sea esperado por algún otro proceso (a nivel de especificación).

La versión inicial del Editor de Redes de Nutt se realizó en lenguaje Turbo-Pascal sobre un microcomputador Televideo TS-803.

3. TRANSFORMACION DE LAS REDES NUTT DE UN CONJUNTO DE PROCESOS A UN PROGRAMA UNICO

La metodología de programación de sistemas distribuidos basada en Redes de Nutt implica, en una segunda etapa, transformar las redes de Nutt que describen los procesos del sistema diseñado a un programa único que pueda ser ejecutado en cada uno de los sitios del sistema.

El programa a generar debe simular la ejecución paralela de los procesos en cada uno de los sitios del sistema y debe incluir el código correspondiente a la interfaz con el nivel de comunicación de la red donde se va a implantar el sistema.

3.1 ESTRUCTURA DEL PROGRAMA A GENERAR

El programa se estructura en 3 partes o niveles:

- **nivel de Aplicación:** es el código correspondiente a los procesos descritos en redes de Nutt

- **nivel de Sistema Operacional Distribuido:** corresponde al núcleo principal que simula la ejecución paralela de los procesos del nivel de aplicación en un sitio del sistema

- **nivel de Comunicación:** es el código que permite la comunicación entre los procesos del nivel de aplicación ya sea dentro de un mismo sitio o entre sitios diferentes; este código depende de la red de comunicación específica sobre la cual se quiere implantar el sistema distribuido.

3.2 GENERACION DEL NIVEL DE APLICACION

Esta parte del programa contiene el código de ejecución que corresponde a cada proceso del sistema distribuido diseñado. Para ello se debe recorrer el grafo de redes de Nutt correspondiente a cada proceso y generar el siguiente esquema de código (suponiendo un lenguaje imperativo estilo PASCAL) :

```
Dependiendo-de estado haga:
¦
¦    e1 : Dependiendo-de mensaje-recibido haga:
¦         ¦    m11 : transiciones asociadas
¦         ¦          actualización de estado
¦         ¦    m12 : transiciones asociadas
¦         ¦          actualización de estado
¦         ¦    ...
¦         ¦    m1n : transiciones asociadas
¦         ¦          actualización de estado
¦         fin-dd
¦
¦    e2 : Dependiendo-de mensaje-recibido haga:
¦         ¦    m21 : transiciones asociadas
¦         ¦          actualización de estado
¦         ¦    m22 : transiciones asociadas
¦         ¦          actualización de estado
¦         ¦    ...
¦         ¦    m2n : transiciones asociadas
¦         ¦          actualización de estado
¦         fin-dd
¦   ....
¦
¦     em :Dependiendo-de mensaje-recibido haga:
¦         ¦   ....
¦         fin-dd
¦
fin-dd
```

Los estados del proceso están representados por **e1, e2,..., em.** Los mensajes que puede recibir el proceso en un estado **ei** están representados por **mi1, mi2,..., min.** En ejecución, cada vez que el proceso reciba un mensaje y obtenga turno examinará el estado en el cual está, ejecutará las transiciones asociadas y actualizará su estado.

La transformación general a código de cada uno de los tipos de transiciones de una red de Nutt es la siguiente:

- acción local: se debe transformar en una asignación o en una invocación del procedimiento asociado a la acción

- envío interno : se debe transformar en instrucciones para depositar el mensaje con la identificación del proceso destinatario en un área local llamada "Buzón" que es compartida por todos los procesos del sitio

- envío externo : se debe transformar en instrucciones para depositar el mensaje en una "Cola de Emisión" que maneja el nivel de comunicación; se indica el sitio destinatario

- difusión interna : se debe transformar en instrucciones para depositar en el Buzón un mensaje destinado a cada proceso local concernido

- difusión externa : se debe transformar en instrucciones para depositar el mensaje en la Cola de Emisión con la indicación de sus sitios destinatarios

- evaluación de una variable : se debe transformar en un "Dependiendo-de" (CASE en PASCAL).

Para ilustrar con un ejemplo, tomemos el caso del proceso CLIENTE presentado en la sección 2.2. Siguiendo el recorrido por profundidad del grafo del proceso se debe generar el siguiente código en PASCAL (el cual se muestra parcialmente):

```
.....

VAR EST-cliente : INTEGER ;        % variable que representa el
                                   % estado actual del proceso
.....
PROCEDURE cliente ;
BEGIN
  CASE EST-cliente OF
  ¦ 1 : BEGIN                      % estado inicial
  ¦     ¦ CASE MENSAJE-REC OF      % mensaje recibido
  ¦     ¦   0 : BEGIN              % 0 es el mensaje nulo
  ¦     ¦       ¦ menu ;
  ¦     ¦       ¦ EST-cliente := 2 ;
  ¦     ¦       END ;
  ¦     ¦ END ;
  ¦     END ;
  ¦
  ¦ 2 : BEGIN                      % estado servicio
  ¦     ¦ % en este estado se leen comandos del usuario
  ¦     ¦ .........                % lee un caracter en opcion
  ¦     ¦ IF KEYPRESSED THEN       % indica si se tecleo algo
  ¦     ¦   BEGIN
  ¦     ¦     ¦ captura-datos (cuenta, cantidad) ;
  ¦     ¦     ¦ CASE opcion OF
  ¦     ¦     ¦   'C':BEGIN
  ¦     ¦     ¦           ........ % meter mensaje de consig-
  ¦     ¦     ¦                    % nacion en cola de emision
  ¦     ¦     ¦         EST-cliente := 2 ;
  ¦     ¦     ¦       END ;
  ¦     ¦     ¦   'R': BEGIN
  ¦     ¦     ¦           ........ % meter mensaje de retiro en
  ¦     ¦     ¦                    % cola de emision
  ¦     ¦     ¦         EST-cliente := 3 ;
  ¦     ¦     ¦       END ;
  ¦     ¦     ¦ END ;
  ¦     ¦   END ;
  ¦     END ;
```

```
        ¦ 3 : BEGIN                          % estado espera
        ¦     ¦ CASE MENSAJE-REC OF
        ¦     ¦    1 : BEGIN
        ¦     ¦        ¦ condicion := COMPONENTE (1) ;
        ¦     ¦        ¦ % la funcion COMPONENTE extrae parametros
        ¦     ¦        ¦ % del mensaje recibido
        ¦     ¦        ¦ IF condicion = 'SI' THEN
        ¦     ¦        ¦    CONTROL := 1 ;
        ¦     ¦        ¦ IF condicion := 'NO' THEN
        ¦     ¦        ¦    CONTROL := 2 ;
        ¦     ¦        ¦ CASE CONTROL OF
        ¦     ¦        ¦   1 : BEGIN
        ¦     ¦        ¦          menu ;
        ¦     ¦        ¦          EST-cliente := 2 ;
        ¦     ¦        ¦        END ;
        ¦     ¦        ¦   2 : BEGIN
        ¦     ¦        ¦          aviso ('retiro imposible') ;
        ¦     ¦        ¦          menu ;
        ¦     ¦        ¦          EST-cliente := 2 ;
        ¦     ¦        ¦        END ;
        ¦     ¦        ¦ END ;
        ¦     ¦       END ;
        ¦     ¦ END ;
        ¦     END ;
        ¦ END ;
      END ;
      .........
```

Es de anotar que el tratamiento de los comandos provenientes del usuario en el proceso encargado de recibirlos (por ejemplo, el proceso CLIENTE) se debe hacer evitando la espera inactiva para no retardar indefinidamente el turno para los demás procesos. Para ello se debe dar turnos frecuentes a dicho proceso, y en cada turno el proceso esperará un tiempo prudencial máximo a que el usuario teclee un caracter para proceder o no a ejecutar las transiciones asociadas (por ejemplo, captura del comando completo con sus parámetros).

Adicionalmente, en este nivel de Aplicación se deben especificar los procedimientos y funciones nombrados en los grafos de redes de Nutt de los procesos.

Así por ejemplo para los procesos CLIENTE y SERVIDOR mostrados en la sección 2.2, el programador deberá especificar el texto de los procedimientos Consignar(cuenta, cantidad), Retiro(cuenta, cantidad), Menú, Captura-datos (cuenta, cantidad), Aviso(texto) y Consultar(cuenta, saldo).

3.3 GENERACION DEL NIVEL DE SISTEMA OPERACIONAL DISTRIBUIDO

Esta parte del programa debe simular la ejecución simultánea de los procesos descritos en redes de Nutt, encargándose entonces de de la activación y sincronización de dichos procesos.

Considerando solamente el caso en el que cada sitio del sistema distribuido es una máquina monoproceso, el nivel de Sistema Operacional Distribuido se constituye en el núcleo principal del programa que da turnos repetidos a los procesos del nivel de aplicación (invocando los procedimientos correspondientes) para simular su ejecución paralela.

En ejecución, el nivel de Sistema Operacional Distribuido trabaja examinando una **tabla de Estados-Mensajes** que especifica los estados de cada proceso y los mensajes que puede recibir un proceso en cada estado. Para el ejemplo del sistema distribuido de procesos CLIENTE y SERVIDOR presentados en la sección 2.2, la tabla de Estados-Mensajes tendría una información equivalente a la que se muestra a continuación :

PROCESOS	ESTADOS	MENSAJES
cliente	inicial	nulo
	servicio	opcion
	espera	respuesta(condicion)
servidor	servicio	consignar(cuenta,cantidad)
		retiro(cuenta,cantidad)

Teniendo en cuenta esta tabla, el estado actual de cada proceso **activo** y el Buzón donde los procesos depositan los mensajes que quieren intercambiarse, se decidirá en un momento dado a qué proceso se le puede dar turno de ejecución (será aquél que esté en el estado adecuado para recibir un mensaje del Buzón destinado a él). El Buzón se examina en el orden en que son depositados los mensajes. La unidad de turno de ejecución para un proceso le permitirá realizar las transiciones comprendidas entre la recepción de un mensaje para su estado actual y la actualización de ese estado.

Es importante anotar que el código que activa los procesos debe tener en cuenta los sitios en los que se va a ejecutar cada proceso; de esta forma, la ejecución del programa en un sitio dado debe activar solamente los procesos asignados a ese sitio.

Teniendo en cuenta las explicaciones anteriores, se puede presentar ahora el esquema general del programa a generar :

```
Programa
¦
¦ Declaración de variables usadas en las redes de Nutt
¦
¦ Declaración de variables internas usadas en el programa
¦
¦ Declaración de una variable estado para cada proceso
¦
¦ Declaración de procedimientos nombrados en las redes de Nutt
¦
¦ Declaración  de procedimientos del nivel de comunicación
¦
¦ Declaración  de un procedimiento por  proceso  conteniendo
¦ el código que se genera para cada proceso
¦
¦ Núcleo principal :
¦ ¦
¦ ¦  Cargar tabla de Estados-Mensajes en memoria principal
¦ ¦
¦ ¦  Inicializar variables internas
¦ ¦
¦ ¦  Inhibir procesos no asignados al sitio
¦ ¦
¦ ¦   Activar procesos que tienen un estado inicial asociado
¦ ¦   al mensaje "nulo" (la activación se hace mediante  in-
¦ ¦   vocación al procedimiento correspondiente al proceso)
¦ ¦
¦ ¦  Ciclo
¦ ¦    ¦
¦ ¦    ¦ Dar turno al proceso encargado de recibir los coman-
¦ ¦    ¦ dos del usuario
¦ ¦    ¦
¦ ¦    ¦ Dar  turno  al  proceso destinatario de uno  de  los
¦ ¦    ¦ mensajes del buzón teniendo en cuenta la tabla Esta-
¦ ¦    ¦ dos-Mensajes y el estado actual de cada proceso
¦ ¦    ¦
¦ ¦   fin-ciclo
¦ fin-nucleo-principal
fin-programa
```

3.4 GENERACION DEL NIVEL DE COMUNICACION

El programa debe incluir en su parte de nivel de comunicación los procedimientos que permiten el intercambio de mensajes entre sitios en la red sobre la cual va a operar el sistema distribuido.

Estos procedimientos deben encargarse de extraer cada mensaje de la Cola de Emisión de un sitio para conformar paquetes que puedan transmitirse por la red a otro sitio. Así mismo deben recibir los paquetes que llegan al sitio, y deducir de ellos los mensajes que deben depositarse en el Buzón local.

Los procedimientos del nivel de comunicación deben corresponder a

los protocolos de comunicación vigentes para la red específica con la cual se está trabajando.

3.5 COMPILADOR DE REDES DE NUTT [Del Risco 85]

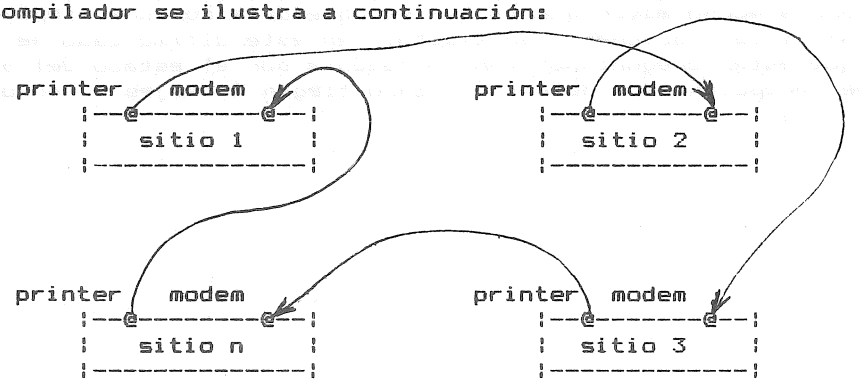
El Compilador de Redes de Nutt es un software que automatiza la etapa de transformar a un programa las redes de Nutt de un conjunto de procesos, especificadas previamente con el Editor de Redes de Nutt.

En su funcionamiento el Compilador de Redes de Nutt recorre en forma recursiva las redes de Nutt de los procesos generando el código en Turbo-Pascal correspondiente a cada uno de ellos. El programador debe completar la especificación en Turbo-Pascal del tipo de variables utilizadas y de los procedimientos y funciones incluidas en las redes de Nutt. El Compilador produce entonces un programa global que da turnos a cada uno de los procesos para simular su ejecución paralela.

Las partes del programa llamadas nivel de Aplicación y nivel de Sistema Operacional Distribuido corresponden a los esquemas generales expuestos en las secciones precedentes.

Como nivel de Comunicación, el Compilador incluye en el programa los procedimientos que permiten la comunicación de microcomputadores en una red local simple con topología de anillo unidireccional. La versión inicial de este Compilador fué desarrollada para microcomputadores Televideo TS-803 (sistema CP/M).

La topología de la red local asumida por el nivel de Comunicación del Compilador se ilustra a continuación:



Cada micro se conecta a otros dos (sucesor y predecesor) a través de sus puertos seriales MODEM y PRINTER; por el PRINTER un micro envía sus mensajes a su sucesor y por el MODEM los recibe de su predecesor.

El nivel de Comunicación funciona en base a un mecanismo de interrupción aprovechando las facilidades que para ello provee el STI

[STI] encargado de controlar el puerto MODEM. El funcionamiento de este nivel se puede describir así:

Constantemente circula por la red un mensaje de control llamado "Ficha" (es generado la primera vez por el sitio 1 cuando se pone en funcionamiento la red). Un sitio solo puede transmitir los mensajes de su Cola de Emisión cuando tiene la Ficha en su poder (los mensajes colocados en esta cola corresponden a los envíos externos que desean hacer los procesos locales). La llegada de la Ficha a un sitio genera una interrupción con las siguientes acciones:

- recepción de todos los mensajes que vengan después de la Ficha provenientes del sitio predecesor en el anillo (un primer mensaje de control indica cúantos mensajes se deben recibir); estos mensajes son colocados en la Cola de Emisión del sitio

- verificación del destino de cada mensaje de la Cola de Emisión: si el destino coincide con el sitio, el mensaje es colocado en el Buzón local (este es el buzón que examina el nivel de Sistema Operacional Distribuido para dar turnos de ejecución a los procesos locales); si el destino no es el sitio, el mensaje se deja en la Cola de Emisión

- emisión al sitio sucesor de la ficha y de los mensajes de la Cola de Emisión (dicha cola queda finalmente vacía).

Es importante resaltar que la implantación del Nivel de Comunicación en base a interrupciones lo hace totalmente transparente a los demás niveles del programa generado y además, asegura una eficiencia mucho mayor que la que se obtendría con una implantación sin interrupciones: en efecto, en este último caso se tendría que estar preguntando constantemente por el estado del puerto de recepción para detectar cuando llegan mensajes y entonces recibirlos.

4. CONCLUSIONES

La motivación principal para usar la técnica gráfica de las Redes de Nutt es que permite programar procesos distribuidos de forma mucho más natural que lo que permite la algorítmica convencional (la sincronización de estos procesos se hace por mensajes y no por memoria compartida). Su simplicidad, la visión general que presentan de todo el sistema diseñado, de cada uno de los procesos que lo constituyen y de la comunicación entre ellos, son algunas de las ventajas que ofrecen las Redes de Nutt.

La metodología de programación de sistemas distribuidos expuesta en este artículo indica cómo representar procesos en redes de Nutt y cómo transformar esas redes a un programa que pueda ejecutarse en cada uno de los sitios del sistema. La metodología indica cómo generar código correspondiente a las partes del programa llamadas niveles de Aplicación y de Sistema Operacional Distribuido dejando abierta únicamente la parte del nivel de Ccomunicación, la cual depende estrictamente de la red sobre la cual se está trabajando.

Adicionalmente, el ambiente de programación constituido por el Editor y el Compilador de Redes de Nutt constituye una herramienta poderosa para la programación de sistemas distribuidos. Con ella se pueden programar en principio protocolos de comunicación para niveles superiores, protocolos de ejecución de transacciones para aplicaciones distribuidas, o sistemas distribuidos completos contemplando varios niveles (por ejemplo, un sistema de Base de Datos Distribuida).

Con este ambiente de programación distribuida el programador se concentra en el diseño de cada uno de los procesos de su sistema, despreocupándose de los aspectos de comunicación entre sitios y de la simulación de la ejecución paralela de los procesos en cada sitio : estos aspectos son resueltos de forma automática por los niveles de Comunicación y de Sistema Operacional Distribuido incluidos en el programa generado por el Compilador de Redes de Nutt.

Actualmente el ambiente de programación distribuida tiene limitaciones por memoria ya que fue implantado en un ambiente de microcomputadores CP/M de 64K. Se proyecta en un futuro transportarlo a un ambiente de equipos más potentes para eliminar dichas limitaciones.

Eventualmente el ambiente de Programación Distribuida descrito en este artículo puede aprovecharse para simular el funcionamiento de sistemas distribuidos que vayan a ser implantados no solo en redes locales sino también en redes de gran alcance, reduciendo así los costos de las pruebas iniciales del sistema.

BIBLIOGRAFIA

[Del Risco 85] : José M. Del Risco, Antonio J. Del Risco
"Simulador e Interpretador de Redes de Nutt para aplicaciones distribuidas", Proyecto Dirigido, Dpto. de Ing. de Sistemas, Universidad de los Andes, Colombia, Julio de 1985.

[Franky 82] : M. Consuelo Franky de Toro
"Contribution au controôle de cohérence des systèmes transactionnels répartis en vue de favoriser l'utilisateur intéractif", Tesis de Doctorado de Tercer Ciclo en Informática, Univeridad de Lille I, Francia, Enero de 1982.

[Franky 85] : M. Consuelo Franky de Toro
"Notas del Curso de Sistemas Distribuidos", Dpto. de Ing. de Sistemas, Universidad de los Andes, Colombia, 1985.

[Nutt 73] : J. D. Noe, G. J. Nutt
"Macro E-Nets for representation of parallel systems", IEEE Transactions on Computers, Volume C-22, No. 8.

[Pérez 85] : Daniel Pérez M.
"Editor y Verificador de consistencia para Redes de Nutt", Tesis de Postgrado, Dpto. de Ing. de Sistemas, Universidad de los Andes, Colombia, Agosto de 1985.

[Peterson 77] : J. L. Peterson
"Petri Nets", ACM Computing Surveys, Vol. 9, No.3, Sept. 1977.

[Rolin 80] : P. Rolin
"Exploitation d'un modèle d'evaluation de Nutt au cours de la vie d'un produit", INRIA Projet SIRIUS, 1980.

[STI] :
"Z80 Microcomputer Peripherals - S.T.I. Controller", MK3801.