UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Unidad Académica de los Ciclos Profesional y de Posgrado del Colegio de Ciencias y Humanidades

"DISEÑO DE UN SISTEMA DE TRANSACCIONES MULTIUSUARIO E IMPLANTACION DE SU SUB-SISTEMA LOCAL"

T E S I S

QUE PRESENTA

FELIPE LOPEZ GAMINO

PARA OBTENER EL GRADO DE MAESTRO EN CIENCIAS DE LA C O M P U T A C I O N

MEXICO, D. F.

1987.

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

México, D.F. a 17 de septiembre de 1987

Lic. Manuel Márquez Fuentes,
Director de la Unidad Académica
de los Ciclos Profesional y de
Posgrado del C.C.H. de la U.N.A.M.

Presente

Por la presente hago de su conocimiento que después de haber leído la tesis titulada "Diseño de un sistema de transacciones multiusuario e implantación de su subsistema local", elaborada por el alumno Felipe López Camino, considero que llena los requisitos necesarios para ser presentada en el examen para obtener el grado de Maestro en Ciencias de la Computación.

Asimismo, le notifico que yo fungí como director de esta tesis.

A t e n t a m e n t e

M. en C. Luis Hugo Poñarrieta Echenique

México, D.F. a 17 de septiembre de 1987

Lic. Manuel Márquez Fuentes,
Director de la Unidad Académica
de los Ciclos Profesional y de
Posgrado del C.C.H. de la U.N.A.M.

Presente

Por la presente hago de su conocimiento que después de haber leído la tesis titulada "Diseño de un sistema de transacciones multiusuario e implantación de su subsistema local", presentada por el alumno Felipe López Gamino, considero que llena los requisitos necesarios para ser presentada en el examen para obtener el grado de Maestro en Ciencias de la Computación.

A t e n t a m e n t e

Dr. Miguel Tomasena Fagoaga

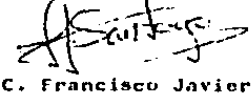
México, D.F. a 7 de octubre de 1987

Lic. Manuel Márquez Fuentes.
Director de la Unidad Académica
de los Ciclos Profesional y de
Posgrado del C.C.H. de la U.N.A.M.

Presente

Por la presente hago de su conocimiento que después de haber leído la tesis titulada "Diseño de un sistema de transacciones multiusuario e implantación de su subsistema local", elaborada por el alumno Felipe López Gamino, considero que llena los requisitos necesarios para ser presentada en el examen para obtener el grado de Maestro en Ciencias de la Computación.

Atentamente



M. en C. Francisco Javier Santoyo Vázquez

México, D.F. a 30 de octubre de 1987

Lic. Manuel Márquez Fuentes,
Director de la Unidad Académica
de los Ciclos Profesional y de
Posgrado del C.C.H. de la U.N.A.M.

Presente

Por la presente hago de su conocimiento que después de haber leído la tesis titulada "Diseño de un sistema de transacciones multiusuario e implantación de su sub-sistema local", presentada por el alumno Felipe López Gamino, considero que llena los requisitos necesarios para ser presentada en el examen para obtener el grado de Maestro en Ciencias de la Computación.

A t e n t a m e n t e

Dr. Renato Barrera Rivera

México, D.F., 5 de noviembre de 1987

Lic. Manuel Márquez Fuentes.
Director de la Unidad Académica
de los Ciclos Profesional y de
Posgrado del C.C.H. de la U.N.A.M.

Presente

Por la presente hago de su conocimiento que después de haber leído la tesis titulada "Diseño de un sistema de transacciones multiusuario e implantación de su subsistema local", elaborada por el alumno Felipe López Gamino, considero que llena los requisitos necesarios para ser presentada en el examen para obtener el grado de Maestro en Ciencias de la Computación.

A t e n t a m e n t e

M. en C. Raymundo Segovia Navarro

C O N T E N I D O

CAPITULO 1

INTRODUCCION

En los años 70's y principios de los 80's el mundo de las computadoras y las comunicaciones vieron emerger nuevas tecnologías que cambiaron radicalmente los patrones del procesamiento de datos. Las tecnologías que dieron lugar a este cambio fueron: las redes de computadoras, el procesamiento distribuido y las microcomputadoras.

El crecimiento y abaratamiento de los sistemas de comunicación permiten en la actualidad que esta tecnología sea económicamente más accesible que antaño. El empleo masivo de microcomputadoras ocasiona que los costos de sistemas de cómputo basados en este tipo de máquinas sean cada vez más bajos y que los tiempos de respuesta sean menores. La combinación de estos desarrollos ha traído como consecuencia cambios en el diseño e implantación de los sistemas de cómputo de la actualidad. Muchos trabajos que anteriormente se hacían en computadoras grandes pueden ahora efectuarse en pequeñas computadoras. Datos almacenados en grandes unidades centralizadas pueden ahora ser compartidos por un gran número de pequeñas unidades dispersas, las cuales con una computadora chica pueden procesar dichos datos o proporcionar un diálogo terminal que simplifique el acceso a los mismos.

Este trabajo presenta el diseño de un sistema multiusuario usado para procesar, en línea, transacciones bancarias de consulta de saldo, depósito y retiro de dinero, utilizando como soporte una red de microcomputadoras para su manejo y operación y distribuyendo la información de los clientes entre todas las sucursales del sistema.

El sistema mantiene distribuida la información de los clientes que tiene registrados -nombre, dirección, cuenta, saldo, transacciones efectuadas en el mes, etc.-, entre todas las sucursales de la empresa. Cada sucursal almacena, por medio de una microcomputadora (micro), la información perteneciente a un cierto conjunto de clientes (p. ej., los que viven o trabajan en los alrededores de la sucursal). La red permite que todas las sucursales se comuniquen entre sí de manera que puedan intercambiar información entre ellas. La figura 1.1 muestra un esquema general de esta idea.

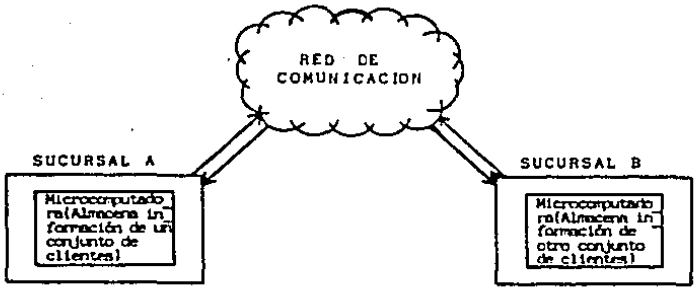


Fig. 1.1 Panorama global del sistema.

Cuando un cliente llega a una sucursal, la A por ejemplo, a efectuar transacciones -saldo, depósito o retiro-, éstas son procesadas localmente si la información del cliente se encuentra registrada en esa sucursal. Si esta información se encuentra en otra sucursal, la B por ejemplo, entonces A debe establecer un procedimiento para solicitar (o enviar) a B la información requerida para satisfacer dichas transacciones. En este caso diremos que la transacción se procesa globalmente. Esto último implica la utilización de la red para que ésta efectúe toda la gestión necesaria a fin de que A y B puedan intercambiar la información que necesitan para procesar las transacciones. En el primer caso, la red no se emplea dado que las transacciones se satisfacen con la información que almacena localmente la micro. El sistema procesa las transacciones en línea, dando respuestas en tiempo real y permitiendo, a la vez, que los clientes puedan llevarlas a cabo simultáneamente en las diferentes sucursales que lo componen.

El sistema está dividido en tres grandes subsistemas, cada uno de los cuales cumple con un cierto objetivo y que, en conjunto, permiten el procesamiento de las transacciones tanto local como globalmente. Estos subsistemas son:

- la red de comunicación, que permite conectar a todas las sucursales a fin de que puedan intercambiar información entre sí;
- el software de control de la red, necesario para tener en funcionamiento correcto a la red, para proporcionar un medio seguro y confiable por el cual transite la información enviada por las sucursales; y

- el software de control de las sucursales, con el cual se satisfacen las transacciones tanto locales como globales y a la vez se conecta la sucursal a la red.

La figura 1.2 muestra el esquema un poco más detallado de la organización del sistema.

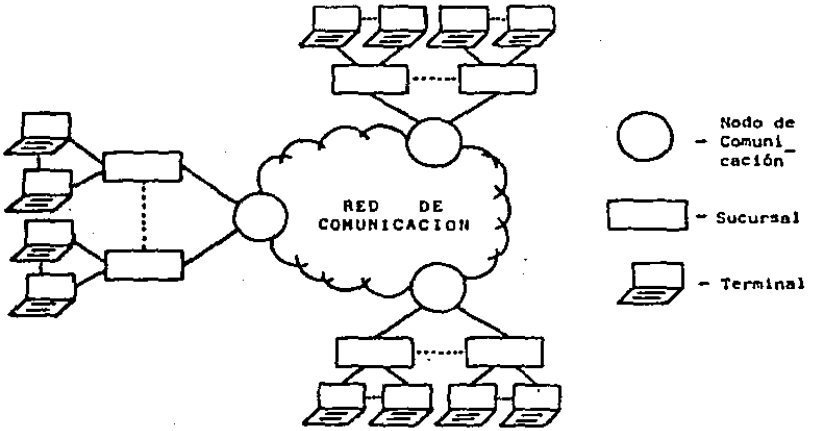


Fig. 1.2 Organización general del sistema.

La RED DE COMUNICACION permite que todas las sucursales puedan comunicarse entre sí. En cada sucursal existe una micro, una de cuyas funciones consiste en conectarla a la red. Esta micro además almacena la información de un conjunto de clientes del sistema y procesa las transacciones tanto locales como globales.

En la red existen además nodos de comunicación, los cuales consisten en micros que sirven para conectar a un conjunto de sucursales a la red. Cuando una sucursal requiere información que se encuentra en otra, los nodos se encargan de efectuar las operaciones necesarias para establecer el intercambio de información entre ambas sucursales. Los nodos también controlan el tráfico de información en toda la red.

En el capitulo 2 se detallan las diferentes cuestiones involucradas en su diseño, como:

- la definición de su configuración física;
- el establecimiento de las interfaces entre las sucursales y los nodos y entre los nodos mismos, tanto en hardware como en software; y
- la especificación de los protocolos de comunicación, necesarios para el intercambio de información entre dos micros cualesquiera de la red.

En este capitulo también se describen los elementos de hardware que se requieren para las micros, tanto en nodos como en sucursales, para establecer las configuraciones físicas que se necesitan en estos puntos, así como para conformar a la red. No se hace mucho énfasis en este aspecto ya que este trabajo está orientado mas bien hacia la parte del software del sistema que hacia la del hardware.

Al tratar los aspectos del diseño de la red tambien se contempla lo siguiente:

- modularidad de la misma, para poder crecer llevando al mínimo los cambios que deban hacerse en la configuración, permitiendo con esto que se puedan agregar más sucursales a la red, o más terminales a una sucursal, fácilmente; y
- disponibilidad del sistema, de tal manera que la caida de una micro, linea de comunicación o falla en general, afecte al mínimo la operación del sistema.

El SOFTWARE DE CONTROL DE LA RED permite utilizar la estructura física de la misma. En la figura 1.3 se muestran las partes de la red que éste controla.

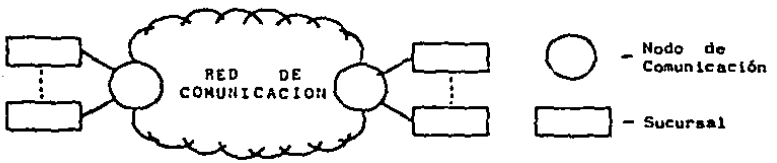


Fig. 1.3 Partes manejadas con el software de control de la red.

Este software tiene como objetivos principales:

1. asegurar que la información intercambiada entre sucursales se entregue correctamente; y
2. supervisar el funcionamiento adecuado de la red.

Para satisfacer el primer objetivo debe:

- establecer procedimientos para conectar las sucursales a la red;
- establecer políticas y procedimientos para dar atención a las sucursales conectadas al nodo;
- controlar el intercambio de información entre nodos y entre sucursales y nodos;
- definir rutas que debe seguir la información enviada por una sucursal;
- controlar el tráfico de información en la red; y
- asegurar que los datos enviados por una sucursal "fuente" lleguen a la sucursal "destino".

Estos aspectos son tratados en el capítulo 3, especificando además el procedimiento que se sigue para procesar una transacción tanto local como globalmente.

Para cumplir con el segundo objetivo se establecen los procedimientos a realizar cuando se presentan situaciones de emergencia como:

- falla de la micro que atiende localmente a una sucursal;
- falla de la micro de un nodo;
- falla de una línea de comunicación entre dos micros; y
- procesamiento incompleto de una transacción.

Aquí también se consideran los aspectos relacionados con la seguridad del sistema como son:

- protección contra almacenamiento de información no válida;
- protección contra accesos no autorizados al sistema, para evitar que alguna persona ajena entre al mismo y realice transacciones fraudulentas con las cuentas que éste maneja.

Estas cuestiones se describen detalladamente en el capítulo 5.

El SOFTWARE DE CONTROL DE LAS SUCURSALES es el que se utiliza para la operación local de una sucursal. La figura 1.4 esquematiza la parte del sistema controlada con éste.

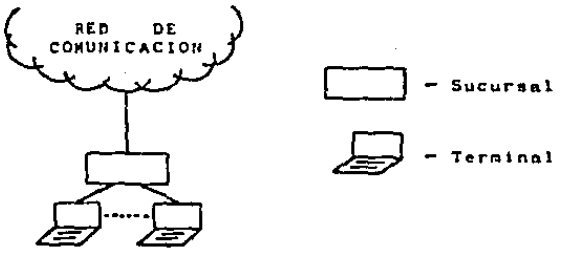


Fig. 1.4 Partes manejadas con el software de control de la sucursal.

Este software, que llamaremos sistema local, está formado por dos partes principales:

1. un supervisor, que permite controlar en tiempo real las operaciones efectuadas por la micro de la sucursal; y
2. un sistema de transacciones, encargado de llevar el registro de la información de los clientes, así como del manejo apropiado de las cuentas que éstos poseen.

El supervisor y el sistema de transacciones permiten manejar en línea las operaciones de: consulta de saldo, depósito y retiro de dinero, para las diferentes cuentas de un cliente. El sistema está pensado originalmente para manejar, por cliente: cuenta de cheques, cuenta de ahorros, tarjeta de crédito y cuenta de valores retirables en días pre-establecidos.

El supervisor además:

- lleva a cabo labores de tiempo compartido para dar atención a los clientes que efectúan trámites en la sucursal, a través de las terminales controladas con la micro;
- controla la ejecución de los diferentes programas requeridos para el procesamiento de una transacción; y
- controla las comunicaciones que se deben establecer con otras micros cuando esto sea requerido.

El sistema de transacciones:

- almacena la información de los clientes registrados en la sucursal;

- procesa todas las solicitudes de transacción recibidas en la sucursal tanto internas como externas; y
- registra todas las operaciones de depósito y retiro realizadas con las cuentas de la sucursal, tanto por transacciones locales como por globales.

El software local de una sucursal también contempla:

- que un cliente -dentro de la misma sucursal- realice varias transacciones con diferentes cuentas; éstas pueden estar algunas dentro de la misma sucursal y otras en sucursales diferentes;
- que una cuenta -dentro de la misma sucursal- sea accesada desde otras sucursales al mismo tiempo. En este caso establece mecanismos de protección y seguridad para evitar manejar cantidades incorrectas o que se envíe a una sucursal información que no le corresponde.

El capítulo 4 describe en detalle todos los aspectos relacionados con el diseño de este sistema local.

Finalmente, la implantación completa de un sistema de esta naturaleza representa un trabajo de gran envergadura, que además de implicar el empleo de una gran cantidad de tiempo y trabajo, requiere de la participación de varias personas. Por ser éste un trabajo de titulación, sólo se llevó a cabo la implantación de la parte local del sistema, manejando únicamente cuenta de cheques. En el capítulo 6 se dan los detalles de esta implantación, mostrando algunos resultados experimentales que se obtuvieron con respecto a los tiempos de respuesta que da el sistema local, cuando se efectúan las transacciones de consulta de saldo, depósito y retiro de dinero con una cuenta de cheques.

CAPITULO 2

RED DE COMUNICACION

La red de comunicación tiene como objetivo primario el permitir que cualquier sucursal del sistema pueda comunicarse (intercambiar información) con cualquier otra sucursal. La figura 2.1 ilustra el concepto.

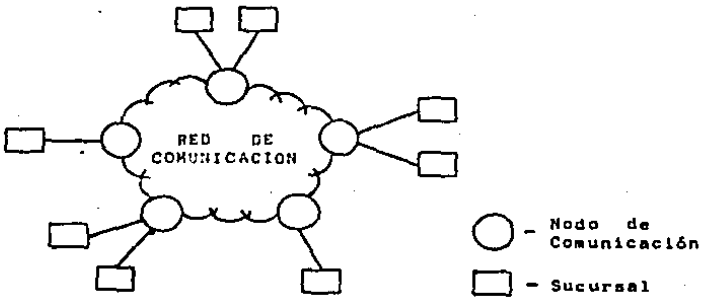


Fig. 2.1 Interconexión de las sucursales por medio de la red de comunicación.

Cada sucursal del sistema es atendida y controlada por una microcomputadora para satisfacer las necesidades locales de la misma. Cada sucursal está ligada a un nodo de comunicación. Un nodo también es una micro la cual se encarga de atender a un conjunto de sucursales, controlando las comunicaciones que éstas efectúen con sucursales conectadas a otro nodo. Los nodos sirven entonces de enlace entre todas las sucursales del sistema.

El conjunto de nodos define los límites de la red de comunicación. La red es capaz de transferir información entre dos sucursales cualesquiera, sin importar el contenido de dicha información; su propósito simplemente es mover información entre una sucursal fuente y una sucursal destino. De esta manera la red es un recurso compartido que evita conectar directamente las

sucursales, punto a punto, lo cual produciría un sistema excesivamente caro para su implantación; y a la vez permite que todas las sucursales puedan comunicarse entre sí. Lo importante de esta red es que está basada totalmente en el empleo de microcomputadoras, tanto en sucursales como en nodos.

Además de esto, la red permite que una transacción global sea procesada en línea. Esto trae como consecuencia que la información de todas las cuentas manejadas por el sistema esté actualizada "al momento" con la consiguiente elevación de la calidad del servicio ofrecido a los clientes del sistema bancario. El tener la información de las cuentas actualizada "al instante" justifica por sí solo el uso de la red ya que, por ejemplo, cualquier transacción de depósito o retiro se registra de inmediato con lo cual se puede evitar, en un momento dado, que una cuenta sea sobre-girada en forma excesiva por la realización de varios retiros de dinero sin fondos. Este es sólo un pequeño ejemplo de los múltiples beneficios que se obtienen al manejar la información en línea.

2.1 TOPOLOGIA DE LA RED

La topología seleccionada para la red (es decir, para sus nodos) es la de una Configuración Parcialmente Conectada de Punto a Punto [WEIT80]. En este tipo de red existe una colección de nodos inter-conectados parcialmente, tal que la transmisión de datos de una computadora fuente a una destino se hace a través de trayectorias entre los nodos, las cuales se seleccionan (buscan) en el momento que se hace la transmisión [STAL85]. En nuestro caso, para intercambiar información entre sucursales, los nodos se utilizan como puntos intermedios que controlan el tráfico y seleccionan las rutas de información entre dichas sucursales.

La figura 2.2 muestra un ejemplo de esta configuración. Para el ejemplo se considera que el sistema bancario posee 40 sucursales. Si utilizamos cuatro micros de comunicaciones (en principio el número es arbitrario, más adelante se justifica) como enlace entre estas sucursales, se puede tener una configuración como la que se muestra en la figura.

El esquema muestra también la forma en la cual se distribuyen las sucursales entre los nodos. En general se busca que los nodos estén balanceados en cuanto a la cantidad de sucursales que están ligadas a ellos (también se justifica más adelante). Es por eso que a cada nodo se le asocian 10 sucursales ya que se distribuyen uniformemente entre los mismos.

Por la forma en que los nodos transmiten la información a través de la red desde una sucursal fuente a una destino (forma que se describe con más detalle en el capítulo 3), esta red

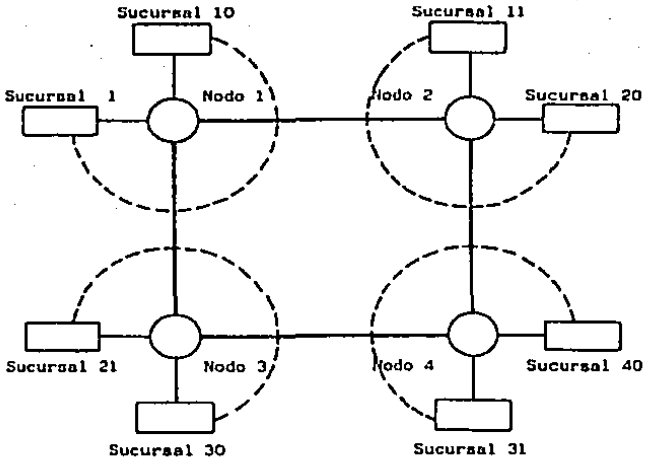


Fig. 2.2 Configuración parcialmente conectada de punto a punto.

pertenece a la clasificación conocida como: Paquetes Conmutados [STAL85]. En este tipo de red, las transferencias de información se hacen en unidades de datos conocidas como paquetes, las cuales son de un tamaño fijo. Cuando un nodo recibe un paquete de información de una sucursal fuente, simplemente lo toma y lo envía a un nodo que esté próximo a la sucursal destino. Por ejemplo, si se desea enviar un paquete de la sucursal 1 a la 31 (en la fig. 2.2), la trayectoria que seguiría podría ser: sucursal 1 a nodo 1, nodo 1 a nodo 3, nodo 3 a nodo 4, nodo 4 a sucursal 31. Los nodos actúan simplemente como conmutadores de paquetes: reciben un paquete, determinan la trayectoria que deben seguir y lo envían para acercarlo más a su destino.

Para hacer la transferencia de información existen dos técnicas que se emplean usualmente en redes de paquetes conmutados: Datagramas (1) y Circuitos Virtuales [HART81]. En la técnica de Datagramas un paquete de información que se envía de un

(1) El término en Inglés es DATAGRAM. Lo traduje como DATAGRAMA y así lo uso en el resto del trabajo.

punto a otro no sigue una ruta pre-establecida; ésta se va determinando conforme el paquete pasa de nodo en nodo en el camino a su destino. Con los Circuitos Virtuales, en cambio, antes de enviar el paquete se determina la ruta que éste va a seguir; una vez establecida, el paquete se envía.

Las principales ventajas y desventajas de ambas técnicas se citan a continuación [STAL85]:

a) Ventajas de datagramas:

- no es necesario pre-establecer una ruta determinada entre dos puntos de la red para el intercambio de información; esto trae como consecuencia que cuando se intercambian pocos paquetes entre ambos, la entrega de los mismos sea más rápida que con ciruitos virtuales;
- si en la trayectoria que sigue un datagrama surge congestionamiento, es más fácil evitarlo enviando el datagrama por rutas alternas ya que no existe una trayectoria pre-definida para el mismo;
- hay más flexibilidad para recuperar fallas. Si en la ruta de un datagrama falla un nodo, siempre se puede encontrar una ruta alterna por la cual enviarlo, cosa que es más difícil realizar con circuitos virtuales.

b) Desventajas de datagramas:

- los paquetes de información requieren de más bytes de "overhead" debido a que necesitan más información de control para poder llegar a su destino;
- cuando un datagrama llega a un nodo, hay que emplear tiempo para determinar la ruta que debe seguir en su camino al punto destino. Con circuitos virtuales este tiempo es menor ya que una vez que el circuito se ha establecido los paquetes siempre siguen la misma ruta, sin que haya pérdida de tiempo al tratar de decidir por dónde deben continuar éstos.

c) Ventajas de circuitos virtuales:

- una vez que el circuito se ha establecido entre dos puntos de la red, se puede intercambiar una gran cantidad de paquetes de información los cuales llegan a su destino rápidamente;
- si un mensaje tiene que transmitirse segmentado en varios paquetes, es más fácil controlar la secuencia del envío y recepción de los mismos, ya que como todos deben seguir la misma ruta, se puede determinar fácilmente si los paquetes se están recibiendo en el mismo orden en el cual fueron transmitidos.

d) Desventajas de circuitos virtuales:

- el establecimiento previo de una ruta ocasiona que el tiempo empleado para el intercambio de paquetes sea mayor que con los datagramas, cuando la cantidad es reducida;

- si un nodo falla en la red, todos los circuitos virtuales que pasan por el mismo se pierden. Recuperar errores de este tipo es más difícil que con los datagramas ya que hay que restablecer todos estos circuitos usando otro nodo diferente.

Tomando en cuenta estas consideraciones, para este diseño se escoge la técnica de datagramas por las siguientes razones:

- para procesar una transacción global basta con intercambiar tres paquetes de información entre las sucursales involucradas, según se verá en las secciones 3.2 y 3.4.3. Al ser poca la cantidad de paquetes intercambiados, hay ventaja de los datagramas sobre los circuitos virtuales;
- el tiempo de respuesta al procesamiento de una transacción global es menor debido, precisamente, a que no es necesario establecer el circuito virtual. Tan solo el tiempo requerido para establecer este circuito sería equivalente al tiempo empleado para el intercambio de dos datagramas;
- hay más flexibilidad en los datagramas para la recuperación de fallas y congestionamientos en la red. Los mecanismos y procedimientos de recuperación de errores son menos complejos que en el caso de circuitos virtuales.

La principal desventaja en la utilización de los datagramas se tiene en los bytes de "overhead" que se deben agregar a cada paquete de información transmitido en la red. Según se describe en la sección 2.3.3, en cada paquete el 50% de los bits (aproximadamente) son de control. Esto ocasiona una reducción, a casi la mitad, del ancho de banda de la información útil transmitida en cada paquete intercambiado entre dos sucursales. Sin embargo, se prefiere pagar este costo en aras de los beneficios obtenidos por la simplicidad de los procedimientos usados para el manejo de los datagramas y por la reducción del tiempo de respuesta en el procesamiento de una transacción global.

Características generales de la configuración seleccionada

a) Funciones generales de un nodo - En cada nodo existe una micro que sirve de enlace entre las sucursales conectadas al nodo y la red. Sus funciones principales son:

- atender a las micros de sucursales conectadas al nodo, tanto para recibir información de éstas, como para enviarles información recibida por él con destino a dichas sucursales;
- determinar la(s) ruta(s) por la(s) cual(es) enviar información que tiene como destino a sucursales conectadas a otro nodo;
- establecer protocolos de comunicación tanto con las sucursales conectadas al nodo, como con los nodos que son sus vecinos en la red;

- controlar el tráfico de información que pasa por el nodo;
- controlar el flujo del intercambio de información entre el nodo y una sucursal, así como entre el nodo y un nodo vecino;
- verificar constantemente que las sucursales conectadas al nodo estén funcionando. Cuando se detecta una sucursal caída, debe tomar medidas de emergencia para manejar este problema;
- registrar las líneas de comunicación, con otros nodos, que estén fuera de servicio, a fin de establecer rutas alternas para el envío de paquetes;
- eliminar de la red los paquetes de información recibidos que no cumplan con los formatos establecidos para su tránsito por la misma.

b) Funciones generales de una sucursal - En cada una de las sucursales del sistema se tiene también una microcomputadora que controla localmente las operaciones efectuadas dentro de la sucursal. Las principales funciones de esta micro son:

- atender a las terminales conectadas a la micro, a través de las cuales se proporciona atención a los clientes en la sucursal;
- efectuar el procesamiento de las transacciones realizadas por los clientes en la sucursal, ya sea con cuentas que se encuentren ahí o en otras sucursales;
- registrar todas las operaciones de depósito y retiro realizadas con las cuentas pertenecientes a la sucursal, ya sea con transacciones efectuadas ahí mismo o por transacciones llevadas a cabo desde otras sucursales;
- satisfacer solicitudes de transacciones (p. ej., saldos) que se reciban desde otras sucursales;
- establecer protocolos de comunicación con el nodo que atiende a la sucursal;
- efectuar medidas correctivas cuando se presenta alguna falla o error; por ejemplo, cuando se corta la comunicación con el nodo;
- proteger las cuentas registradas contra accesos no autorizados.

La operación local de una sucursal se describe con detalle en el capítulo 4. Por el momento consideraremos que a una micro de sucursal se pueden conectar máximo 15 terminales. Las razones que justifican este número se exponen en la siguiente sección. En caso de que en una sucursal (p. ej., la matriz) se tenga que manejar una mayor cantidad de terminales, bastará con agregar una o más micros para poder atender a 15 o más terminales adicionales, como si se tratara de otra sucursal más.

c) Tolerancia a fallas - Una configuración de esta naturaleza permite adoptar medidas de emergencia adecuadas cuando se presentan fallas en la red como: caída (falla) de una línea de transmisión, caída de un nodo, bloqueo de un nodo por saturación de mensajes, etc.; o sea fallas que en un momento dado pueden afectar el funcionamiento de toda la red. De la configuración se

puede anotar lo siguiente:

- en cada nodo se deben tener dos microcomputadoras, una principal y una de reserva. La principal estará siempre en funcionamiento realizando las actividades descritas en a). En caso de falla, entraría en operación la de respaldo a fin de que el nodo no quede sin funcionar, ya que esto dejaría incomunicadas a todas las sucursales conectadas al nodo. Lo anterior garantiza una disponibilidad casi completa de todos los nodos de la red (este aspecto se trata con más detalle en la sección 5.1.2);
- cada nodo tiene un mínimo de dos líneas de comunicación y un máximo de cuatro (según se observará en el inciso d). Al tener por lo menos dos líneas alternas, se garantiza que cuando una quede fuera de servicio, se tenga otra ruta por la cual el nodo siga conectado a la red. La probabilidad de que ambas líneas fallen al mismo tiempo es demasiada baja, por lo que esta cantidad mínima es suficiente para conectar a cada nodo.
  Se propone un máximo de 4 por dos razones: una es para brindar un crecimiento modular de esta configuración; cuando se agreguen más nodos a la red se permite la posibilidad de que un nodo tenga comunicación hasta con cuatro nodos vecinos.
  La otra razón es económica: a fin de no tener una configuración totalmente conectada de punto a punto - que resulta excesivamente cara porque cada que se agrega un nuevo nodo, se debe aumentar el número de líneas de comunicación en una cantidad igual al número de nodos existentes-1 - se establece un máximo de 4 líneas que son suficientes para conectar cualquier nodo a la red;
- lo anterior ocasiona que para cada nodo existan trayectorias alternas para enviar información a otros nodos. Cuando una línea deja de funcionar -por falla eléctrica, descompostura, etc.- o cuando el nodo que está en el otro extremo se satura por exceso de tráfico, siempre se podrá encontrar una trayectoria alterna que permita intercambiar información entre cualquier nodo de la red. Esto brinda una redundancia tal, que permite asegurar que en un momento de la operación de la red, no habrá un nodo que quede aislado por falla de las líneas que lo conectan con otros nodos.

d) Modularidad - En esta parte se exponen los criterios a seguir para permitir un crecimiento modular en la configuración de la red. El objetivo que se persigue es facilitar el crecimiento del sistema de tal manera que al agregar sucursales al mismo la configuración que ya existe en ese momento sea afectada al mínimo, tanto en hardware como en software.

En principio consideraremos que a un nodo pueden asociarse 10 sucursales como máximo (en la siguiente sección se justifica este número). Conforme se vayan agregando sucursales será necesario incrementar la cantidad de nodos en la red. En la figura 2.3 se presentan esquemas del crecimiento que puede tener la red de acuerdo a la cantidad de nodos que existan en la configuración.

Se parte de un esquema basico (1) de 4 nodos. Con esto se tienen 40 sucursales como máximo en el sistema. El siguiente esquema (2) muestra un configuración con 8 nodos. Esta red permite un máximo de 80 sucursales. Se observa que al agregar los nuevos nodos, la estructura anterior se conserva. Asi se sigue con los demás esquemas.

La idea central de este crecimiento es que al agregar nuevos nodos se cumplan los siguientes objetivos:

- evitar modificar la estructura fisica que ya existe en ese momento;
- dar, por lo menos, dos lineas de comunicación a cada nuevo nodo para cumplir con lo expuesto en c);
- tener un crecimiento sistemático que permita agregar nodos siguiendo un orden determinado.

El primer objetivo se cumple ya que de los esquemas se observa que al agregar nodos se usa la estructura que ya existe en ese momento, sin hacer cambio alguno. De ahi también se observa que cada nuevo nodo tiene dos lineas de comunicación con lo cual se está acorde con el segundo objetivo.

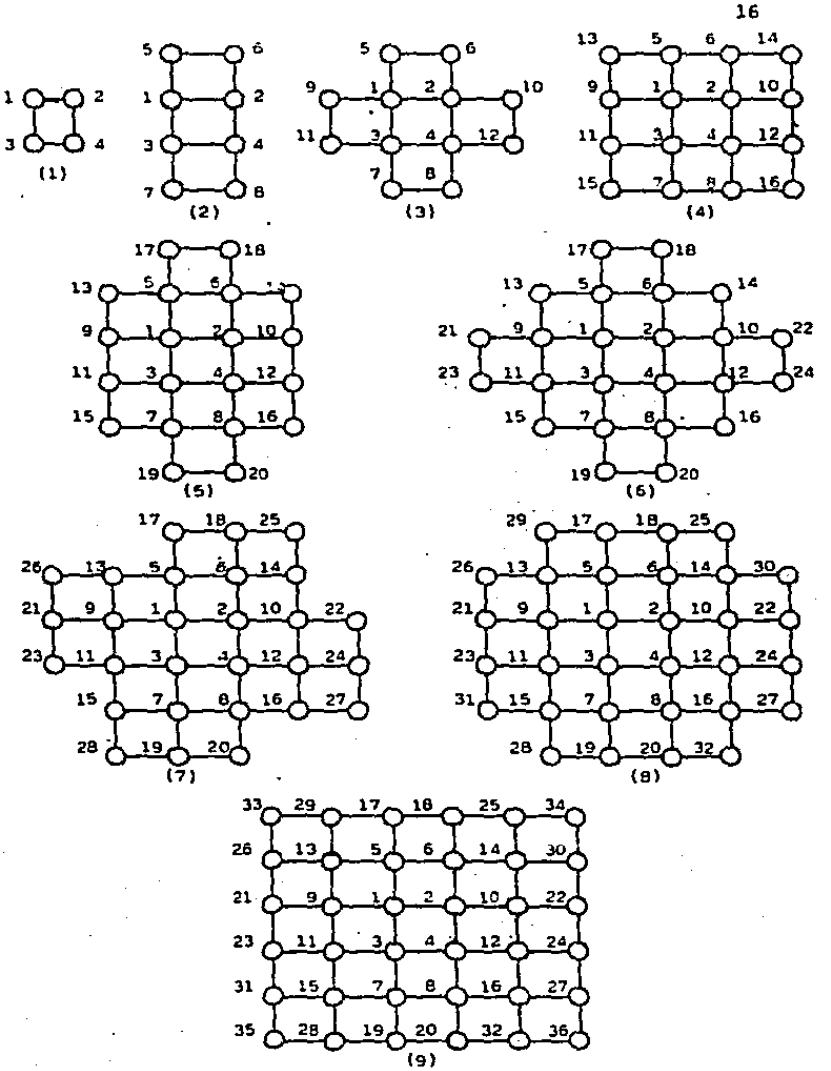
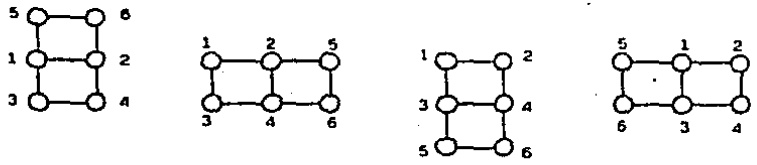
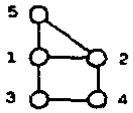


Fig. 2.3 Configuraciones posibles de la red en base al número de nodos que existan en la misma.

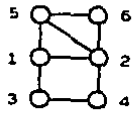
El crecimiento sistemático se tiene de la siguiente manera: partiendo del esquema (1) el objetivo es ir desarrollando una "cruz" alrededor de estos 4 nodos, que podemos llamar centrales. Por ejemplo, si agregamos dos nodos la configuración resultante podría ser una de las siguientes:



La colocación de los nuevos nodos dependerá de la posición geográfica de las nuevas sucursales. Los esquemas (2), (3), (5) y (6) de la figura 2.3 son ejemplos de como se desarrollan los extremos de esta cruz. Aquí cabe mencionar que cada que se extienda uno de estos extremos, será necesario agregar siempre dos nodos aunque la cantidad de sucursales nuevas sólo requiera uno. Esto debe ser así ya que de otra manera se altera el esquema de crecimiento. Veamos: si tenemos el esquema inicial de 4 nodos, al agregar uno nuevo la configuración resultante podría ser:



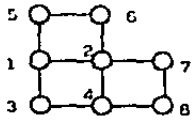
Esto es así porque todo nodo debe tener por lo menos dos líneas de transmisión para asegurar redundancia en las trayectorias. Al agregar un nuevo nodo, la resultante podría ser:



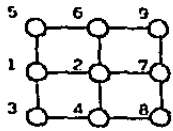
lo cual rompe el esquema de crecimiento.

El agregar siempre dos nodos en los extremos de la cruz tiene un costo extra el cual se compensa con la sistematización que se obtiene con este tipo de crecimiento. De cualquier manera este costo puede no tener relevancia, si en un momento dado el nodo extra necesita ser utilizado al agregar más sucursales al sistema.

Volviendo al esquema de crecimiento, una vez que dos extremos han sido desarrollados, el objetivo es formar ahora un cuadro. Por ejemplo, en un esquema como el siguiente:



Al agregar un nuevo nodo se podría formar el siguiente cuadro:



Los esquemas (4), (7), (8) y (9) de la figura 2.3 son ejemplos de como se desarrolla este cuadro. A partir de un cuadro, cuando se agregan más nodos, se desarrollan de nuevo los extremos de la cruz central y así sucesivamente. Cada que se agreguen nodos, éstos se numerarán consecutivamente de arriba hacia abajo y de izquierda a derecha, tal como se muestra en los esquemas de la figura 2.3. Los nodos que ya existan en la red conservan su identificación. En realidad lo importante de esto es que cada nodo tenga un número que lo identifique en forma única dentro de la red y que no cambie su número cuando se agreguen nuevos nodos.

Aunque en los esquemas de la figura 2.3 los nodos se van agrupando alrededor de la configuración inicial de 4 nodos, en realidad puede ser que alguno(s) de los extremos se desarrolle(n) más que otro(s), dependiendo de la colocación geográfica de las sucursales; de cualquier manera, lo expuesto anteriormente sigue vigente. Lo importante aquí es que se crezca de a dos nodos o de a uno, según sea el caso.

2.2 ELEMENTOS DE HARDWARE REQUERIDOS PARA FORMAR LA RED

2.2.1 Configuración de un Nodo

En la figura 2.4 se muestra un esquema detallado de la configuración de un nodo. En el esquema se observa que a un nodo máximo se le pueden conectar 10 sucursales; esto se justifica al final de la presente sub-sección. Cada uno de los elementos de esta configuración, y su función, se detallan a continuación.

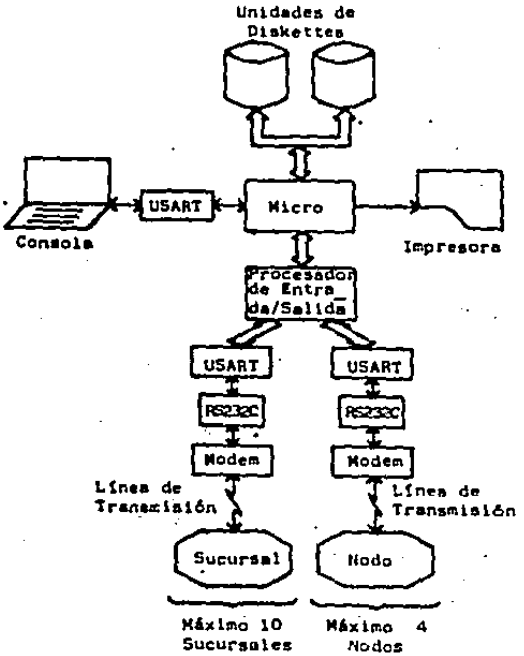


Fig. 2.4 Configuración física de un nodo.

a) Micro - Es la micro de comunicaciones que se encuentra en un nodo. Sus funciones son las que se describen en el inciso a) de

la sección anterior. Los programas que debe ejecutar son descritos en el capitulo siguiente.

b) Procesador de Entrada/Salida - Es un pequeño procesador frontal que se va a encargar de controlar la transferencia de información entre la micro y las lineas que la comunican con las sucursales u otros nodos. Este procesador es capaz de:

- efectuar transmisión full-duplex con las sucursales y con otros nodos;
- recibir, por cada sucursal y nodo asociados, hasta 7 paquetes de información cada uno de 24 bytes de longitud (secciones 2.3.2 y 3.1) guardándolos en su memoria interna, hasta que el procesador central los tome;
- recibir del procesador central, por cada sucursal y nodo asociados, hasta 7 paquetes también de 24 bytes, para que los transmita a la sucursal o nodo que corresponda;
- controlar el envio/recepción de bytes con los USART conectados a él.

La actividad principal de este procesador es la de controlar el intercambio de información entre la micro y las lineas de comunicación, cuidando no perder bytes mientras el procesador central realiza otras actividades. En el apéndice B se describen las caracteristicas principales de un procesador de esta naturaleza, el cual se empleó para la implantación del sistema local. Aquí, el procesador se usa para controlar comunicaciones con terminales, aunque también se puede utilizar para controlar comunicaciones con modems tal como lo requiere la configuración de un nodo.

Entre la micro y este procesador deben manejar un total de 14 puertos de entrada/salida; 10, para establecer comunicación con sucursales; 4, para comunicarse con nodos vecinos.

c) USART - Es la unidad de hardware que se encarga de controlar la transmisión/recepción sincrona de bytes entre el nodo y las sucursales u otros nodos. El USART debe tener la capacidad para transmitir y recibir paquetes HDLC (descritos en la sección 2.3.2), rechazando aquellos que son erróneos. El dispositivo QUADART, descrito en el apéndice B, cumple con estas especificaciones por lo que también puede utilizarse para realizar estas funciones.

d) RS-232C - Es la interfaz eléctrica estándar que debe existir entre un USART y un modem o entre un USART y una terminal. Sus caracteristicas son las estándares. El Quadart mencionado anteriormente ya maneja esta interfaz en cada uno de los USART que contiene.

e) Modem - Usado para convertir las señales digitales a analógicas para su transmisión sobre la linea de comunicación y viceversa, en la recepción. Debe permitir la transmisión en full-duplex.

Con respecto a la velocidad de transmisión del modem se tiene lo siguiente: en un momento dado de la operación del sistema, se van a transmitir hasta 7 paquetes de información de 24 bytes cada uno (ver sección 3.1), lo cual equivale a 1,344 bits. Las líneas telefónicas en general permiten operar a velocidades en el rango de 300 bps a 9,600 bps, con un margen tolerable de errores en la transmisión.

Con base en esto, se propone que los modems operen a una razón de 9,600 bps, con lo cual un paquete de 24 bytes se transmite en menos de un segundo y el margen de errores es bajo (aproximadamente de 15 a 18 bytes erróneos por cada 10,000 transmitidos [NICH82]). Esta velocidad se usa para transmitir información entre nodo y sucursal o entre nodo y nodo, indistintamente.

Cualquier modem de velocidad media(1,200 bps a 9,600 bps [NICH82]) de los que existen actualmente en el mercado, puede cumplir satisfactoriamente con esta función.

f) Línea de Transmisión - Debe ser una línea telefónica dedicada full-duplex. Sobre ella se transmitirá a 9,600 bps. Se usan dedicadas por dos razones:

1 - Privatización de la información que se transmite.
2 - Velocidad de transmisión (en líneas dedicadas se puede llegar a 9,600 bps).

g) Consola - Se usa para: iniciar la ejecución de los programas requeridos en una sesión de trabajo, correr diagnósticos, monitorear la actividad del nodo, ejecutar programas de emergencia, etc. Debe tener asociada una teleimpresora para que en ella se escriban mensajes que en un momento dado se puedan usar como traza de las actividades que está realizando el nodo.

h) Unidades de diskette - Se emplean para almacenar los programas requeridos para la operación del nodo, archivos con información que necesitan los programas, etc. Aunque para esta configuración, como para la de una sucursal, se propone emplear diskettes para almacenar la información que maneja el sistema, obviamente es mucho mejor usar discos fijos de mayor capacidad y velocidad. En ambos esquemas aparecen diskettes porque en la implantación del sistema local, éstos fueron utilizados para guardar la información manejada experimentalmente con el mismo. Para efectos de una implantación real de todo el sistema es indiscutible que en lugar de usar diskettes se deben emplear discos fijos.

i) Impresora de Líneas - Se usa para imprimir reportes, programas, etc.

En el esquema de la figura 2.4 se observa que a un nodo máximo se pueden conectar 10 sucursales. Las razones que justifican esto son las siguientes:

1) Cada que la micro del nodo atiende una línea de sucursal,

puede enviar/recibir a través de ella un máximo de 7 paquetes de 24 bytes cada uno. Considerando tanto transmisión como recepción se tiene que por cada sucursal va a manejar 2,688 bits. Si la velocidad de transmisión/recepción es de 9,600 bps entre nodo y sucursal, se requieren (aproximadamente) 280 mseg para enviarlos/recibirlos. Manejando un máximo de 10 sucursales, el tiempo máximo de atención para todas ellas será de 2.8 segundos.

2) En el caso de la comunicación nodo-nodo, la cantidad de bits a enviar/recibir es la misma que en nodo-sucursal. La velocidad de transmisión es la misma por lo que se requerirán también 280 mseg para enviar/recibir bits cada que se da atención a una línea de nodo. Si se conectan 4 nodos como máximo, el tiempo de atención será de 1.12 segundos.

3) Cuando un nodo recibe un paquete de información, realiza varias actividades con ello. ej., encontrar la ruta por la cual enviarlo para que llegue a su destino). Considerando que el tiempo empleado para efectuarlas es despreciable con respecto a los tiempos de transmisión/recepción (microseg. vs. miliseg.), se tiene que al conectar al nodo el máximo de sucursales(10) y de nodos(4), se emplearía un tiempo máximo de 3.92 segundos para dar atención a todos, considerando que se transmiten/reciben 7 paquetes de 24 bytes con cada uno de ellos. Este es el peor caso en el tiempo de respuesta de un nodo. El tiempo promedio, obviamente, es menor.

Esta cifra es bastante aproximada a las que se muestran en la sección 6.2 para los tiempos medidos experimentalmente en la ejecución del sistema local. Esto es así, ya que aunque los objetivos e información manejados en éste son diferentes a los de un nodo, se realizan actividades similares como son: atención en tiempo compartido a usuarios (terminales en un caso; sucursales y nodos vecinos, en el otro), establecimiento de comunicaciones, registro de información, etc. De hecho el diseño presentado en este trabajo está inspirado en la implantación del sistema local, ya que de esa experiencia se partió a una generalización de las actividades ahí efectuadas, concluyendo con el diseño global de este sistema. En dicha implantación se utilizó un microcomputadora multi-usuario Cromemco como soporte para efectuar las operaciones del sistema local (los detalles se dan en los capítulos 4 y 6).

Cuando se agregan sucursales al sistema se busca que éstas se distribuyan uniformemente entre los nodos para que, al menos por probabilidad, la carga sea equiparable entre ellos y así el tráfico en los mismos esté balanceado (o sea, que la cantidad de paquetes que tengan que manejar sea aproximadamente la misma). Con esto se persigue evitar que algún nodo se convierta en cuello de botella por exceso de carga -lo cual ocasionaría tiempos de respuesta mayores-, con la consiguiente saturación por manejo excesivo de paquetes de información.

2.2.2 Configuración de una Sucursal

En la figura 2.5 se muestra el esquema detallado de la configuración de una sucursal.

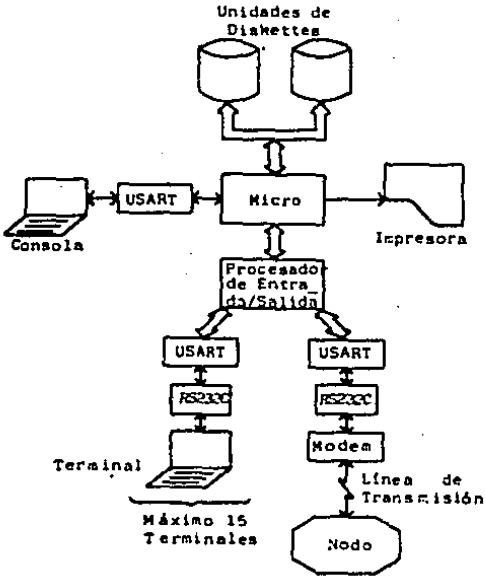


Fig. 2.5 Configuración física de una sucursal.

El esquema de esta configuración es semejante al de un nodo. La diferencia principal estriba en que el procesador frontal controla ahora comunicaciones con terminales en lugar de modems -excepto uno, el usado para establecer comunicación con el nodo asociado-. Las terminales en este caso, se emplean para dar atención a los clientes y, a través de ellas, procesar las transacciones que éstos efectúan. La micro de la sucursal lleva a cabo ahora las funciones descritas en el inciso b) de la sección 2.1. Los programas que ejecuta son los que se describen en el capítulo 4.

Como se observa de este esquema, la idea central de este diseño es que tanto en sucursales como en nodos se tengan configuraciones parecidas. En el caso de sucursales, el sistema sirve para atender terminales y procesar transacciones; en el caso de nodos, para atender modems y establecer comunicaciones. El software en ambos casos efectúa funciones similares: labores de tiempo compartido, atención a varios usuarios (terminales o modems), control de la ejecución de varios programas, etc.

En suma, a partir de la idea primaria de un sistema local, se pasa a una generalización que contempla la operación de una red que conecta, comunica y coopera al funcionamiento de muchos sistemas locales que procesan entre todos información descentralizada, representada por las cuentas de los usuarios.

2.3 PROTOCOLOS DE COMUNICACION

Una vez definida la configuración física de la red, se describe en detalle el conjunto de protocolos de comunicación requeridos para establecer y controlar la transmisión de información en la red.

Existe una serie de estándares establecidos por diversas organizaciones mundiales para definir los diferentes niveles (o capas) funcionales que se deben utilizar para intercambiar información en una red. Entre estas organizaciones se pueden mencionar dos principalmente: la Organización de Estándares Internacionales (ISO) y la Unión de Telecomunicaciones Internacionales(ITU) en su sección Comité Consultivo sobre Telefonía y Telegrafía Internacionales(CCITT).

ISO ha propuesto un conjunto de 7 capas para el control, operación y funcionamiento de una red de computadoras. Las diferentes capas propuestas controlan:

- la interfaz eléctrica de la computadora con la red;
- el intercambio de información entre dos puntos adyacentes de la red;
- el intercambio entre dos puntos cualesquiera;
- la integridad de la información transmitida entre dos puntos de la red;
- el inicio de una sesión;
- la presentación ante una computadora remota de la red; y
- la realización de procesos en esa computadora remota.

La idea central de esta propuesta es permitir conectar cualquier tipo de computadora (idealmente) a una red, para que pueda accesar los recursos (p. ej. información) de otra computadora en la red o, inclusive, de computadoras conectadas a

otra red.

CCITT propone sólo un subconjunto de 3 niveles usados básicamente para controlar el intercambio de información en la red. Estos son equivalentes a los 4 primeros de ISO.

Para este diseño se selecciona la segunda alternativa, ya que la red propuesta no es de uso general, sino dedicada exclusivamente al procesamiento de las transacciones bancarias. Para efectos de operación y control de la red, basta con utilizar estos 3 niveles para intercambiar información entre las diferentes sucursales del sistema. Esta propuesta se conoce como Recomendación X.25 de CCITT [MART81].

Cada nivel (o capa funcional), idealmente, es independiente de los otros niveles, o sea, los trabajos internos realizados en un nivel son transparentes a los otros niveles, por lo que en un momento dado se puede cambiar una capa por otra nueva (para mejorarla, obviamente) sin necesidad de hacer cambios a los otros niveles.

Las tres capas en cuestión son:

a) Capa 1 - que consiste en el control eléctrico de la línea de comunicaciones.

b) Capa 2 - que es el protocolo para establecer comunicación sobre un canal apropiado que puede ser una línea telefónica, un cable coaxial, un satélite, etc. En nuestro caso es una línea telefónica dedicada.

c) Capa 3 - constituido, principalmente, por el protocolo para el intercambio de información entre dos usuarios cualesquiera (p. ej. dos sucursales), el establecimiento de rutas para la información, el control del tráfico en la red y la integridad de la información.

Sobre este tercer nivel se encuentra la aplicación que se controla con la red, que en este caso es el sistema de transacciones bancarias. En la figura 2.6 se muestra un esquema de estas capas.

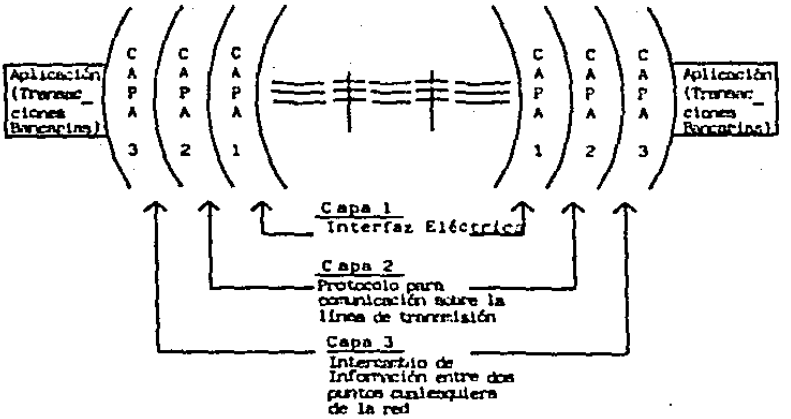


Fig. 2.6 Capas de control para intercambiar información en la red.

El detalle de cada uno de los tres niveles se hace a continuación, enmarcándolo dentro del diseño de la red.

2.3.1 Capa Funcional 1

Se refiere al control eléctrico de la línea de comunicaciones. Está constituido por la conexión física de las micros con el equipo de comunicación (modems, acopladores de líneas, etc.). Se relaciona con las interfaces (p. ej. RS232C), conexiones de las patas de las interfaces, regulación de los niveles de voltaje, acoplamiento de impedancias, etc. En general las conexiones e interfaces se ajustan a los estándares establecidos[CHOR80] y es lo que se comentó en las secciones 2.2.1 y 2.2.2.

2.3.2 Capa Funcional 2

Es el protocolo de comunicación sobre la línea de transmisión. Está relacionado con la transmisión confiable de caracteres (datos, o sea, información) de una micro a otra sobre una línea de transmisión. A este nivel se maneja la detección de errores de transmisión, control de paridad, transmisión síncrona, etc. También se conoce como protocolo de bajo nivel.

El protocolo a utilizar será el conocido como: Control de Línea de Datos de Alto Nivel (HDLC), que es el que la Recomendación X.25 define para utilizar en este nivel. Las principales características de este protocolo son:

- permite transmitir información en serie en forma síncrona;
- la información transmitida puede contener cualquier patrón de bits y puede contener cualquier cantidad de bits. Es un protocolo orientado a bits;
- permite transmisión de punto a punto en una red;
- puede operar sobre líneas full-duplex; o sea, puede haber intercambio de información entre dos puntos al mismo tiempo;
- opera eficientemente a altas velocidades de transmisión.

La transmisión de mensajes con este protocolo se hace bajo tres formatos básicos:

- formato de transferencia de información (Formato I);
- formato de supervisión (Formato S); y
- formato no-numerado (Formato U).

Bajo este protocolo la máquina que transmite se conoce como "Estación Primaria" y la que recibe como "Estación Secundaria". En base a esto, a continuación se detallan las características de los dos primeros formatos que son los usados en este diseño.

a) Formato de Transferencia de Información (Formato I).- Es el formato básico que permite la transferencia de información entre dos micros adyacentes. En la figura 2.7 se muestra el esquema de este formato.

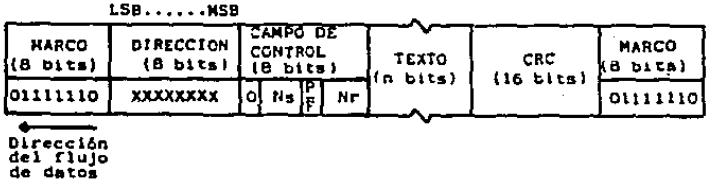


Fig. 2.7 Formato de Transferencia de Información.

El byte marcado como MARCO contiene el patrón de bits único: 01111110, que indica el inicio y fin de un paquete (cuando este patrón se presenta entre los bytes de MARCO no existe problema, ya que la máquina que transmite siempre inserta un cero después de cinco unos y la que recibe siempre borra un cero después de recibir cinco unos, con lo cual dicho patrón sigue siendo único). El campo de DIRECCION sirve para identificar a la estación secundaria a la cual se envía el paquete. El campo permite especificar hasta 255 direcciones diferentes.

El campo CONTROL contiene dos subcampos Ns y Nr los cuales almacenan los números de secuencia del paquete enviado a la máquina adyacente y del que se espera recibir desde ella, respectivamente. Estos subcampos permiten que con un solo paquete se puedan dar por aceptados (correctamente) hasta 7 paquetes transmitidos previamente. En la sección 3.1 se describe más concretamente el uso de estos subcampos. El bit P/F tiene las siguientes funciones:

- P=1, cuando la estación primaria autoriza a la secundaria a transmitir;
- P=0, cuando la estación secundaria sólo puede recibir información de la primaria; esto es, no debe transmitir;
- F=0, cuando la estación secundaria está enviando una serie de paquetes a la primaria;
- F=1, cuando la estación secundaria envía el último paquete a la primaria.

El campo TEXTO es el que almacena la información transmitida con el paquete. Con respecto a este diseño el tamaño del campo es de 18 bytes, según se observará en la siguiente sección.

El campo CRC contiene el valor que checa que todos los bits anteriores del paquete hayan sido transmitidos correctamente. Se utiliza el estándar CRC-CCITT para la detección de errores.

b) Formato de Supervisión (Formato S).- Sirve para llevar a cabo ciertas funciones de supervisión con la transmisión. Este formato es semejante al anterior con la diferencia de que no tiene campo de TEXTO. Su esquema es:

MARCO (8 bits)	DIRECCION (8 bits)	CAMPO DE CONTROL (8 bits)				CRC (16 bits)	MARCO (8 bits)
01111110	XXXXXXXX	10	SS	P/F	Nr		01111110

Dirección del flujo de datos

Fig. 2.8 Formato de Supervisión.

Este tipo de formato es usado para: dar por aceptados una serie de paquetes recibidos -tanto por estación primaria, como por secundaria-, solicitar envíos adicionales, solicitar retransmisión de uno o más paquetes o solicitar una pausa en las transmisiones. En el campo de CONTROL los bits 3 y 4 se utilizan para indicar los siguientes tipos de supervisión:

00:	RR	Listo para Recibir
01:	REJ	Paquete Rechazado
10:	RNR	No está Listo para Recibir
11:	SREJ	Paquete Rechazado Selectivamente

Los tipos más comunes son: RR, enviado por una estación -primaria o secundaria- para indicar que se recibieron correctamente los paquetes numerados hasta el Nr-1. Indica además que está listo para recibir el paquete número Nr; y RNR, enviado por una estación -primaria o secundaria- para indicar que temporalmente está ocupada y no puede recibir más paquetes. El fin de esta condición puede ser indicado con otro paquete de supervisión de tipo RR.

En ambos formatos, el USART es el encargado de procesar los bytes MARCO y CRC. En el caso de transmisión, inserta los bytes MARCO y checa todos los bits que forman parte del mensaje para agregar el CRC correspondiente al final del paquete. En la recepción detecta la ocurrencia de los bytes MARCO para reconocer donde empieza y termina un paquete. Además verifica los bits del mensaje para determinar si el CRC obtenido es el mismo que viene al final del paquete o no. En caso de que no concuerden descarta el paquete recibido. El dispositivo Quadart descrito en el apéndice B, ya efectúa por hardware estas funciones por lo que para transmitir sólo hay que especificar, por software, los campos: DIRECCION, CONTROL y TEXTO. En la recepción, sólo entrega al procesador frontal el contenido de los campos CONTROL y TEXTO, descartando los campos de MARCO, DIRECCION y CRC. También por hardware maneja la inserción y eliminación de un cero después de cinco unos cuando éstos se presentan entre los bytes MARCO, tanto en la transmisión como en la recepción.

En el apéndice A se describe en forma detallada la manera en la cual se lleva a cabo el intercambio de información entre dos estaciones por medio de este protocolo. En nuestro caso, este proceso es el que gobierna el intercambio entre dos micros adyacentes en la red. Las acciones descritas en ese apéndice son las que efectúan los módulos L2S y L2NN descritos en las sección 3.1.

2.3.3 Capa Funcional 3

Está constituida por el protocolo de comunicación entre dos puntos cualesquiera de la red. Se refiere al conjunto de acciones que se deben efectuar para que una micro establezca comunicación con cualquier otra micro de la red. Las principales funciones que se llevan a cabo en esta capa son:

- enviar mensajes de una micro a otra ya sea en uno o varios paquetes de información. En nuestro caso, este envío se simplifica ya que cualquier intercambio de información entre micros se hará con un solo paquete, el datagrama, según se describe más adelante;
- aplicar controles de comunicación para prevenir la pérdida o doble procesamiento de mensajes, control del flujo de mensajes entre micros, así como el direccionamiento de las micros origen y destino;
- asegurar la integridad de la información, la cual consiste, principalmente, en que un paquete llegue a la micro a la cual fue enviado y que la información que éste contiene llegue sin alteración;
- establecer rutas para la transmisión de la información, esto es, encontrar las rutas adecuadas para enviar mensajes de una micro a otra, evitando congestionamientos de tráfico o líneas físicas dañadas, y al mismo tiempo tratando de optimizar el viaje de la información de tal manera que el mensaje llegue en el menor tiempo posible a su destino;
- controlar y avisar de fallas en la red como son: falla de una línea de comunicación entre dos micros, caída de una micro de sucursal, caída temporal de una micro de comunicación, etc.

Las anteriores son las funciones principales de esta capa; junto con las otras dos proporcionan lo que se conoce como "Sistema de Transporte de la Red". Los módulos de software que lo integran se describen en el siguiente capítulo.

En esta sección se describe solamente el protocolo que se debe utilizar para intercambiar mensajes entre dos sucursales de la red. Este protocolo se conoce con el nombre de DATAGRAMA y es uno de los propuestos en la Recomendación X.25 para este nivel.

Un DATAGRAMA (según la definición del CCITT [MART81]) consiste en: "un mensaje el cual está contenido en el campo de datos de un solo paquete, que es entregado en la máquina de destino indicada en su campo de dirección. Ninguna referencia es registrada en la red acerca de cualquier datagrama enviado antes o después del actual hacia esa máquina; esto es, no existe relación alguna entre el datagrama actual y los enviados, previa o posteriormente a éste, hacia dicha máquina".

Entre las principales propiedades de los datagramas se encuentran las siguientes:

- tienen formato y protocolos de manejo simples;
- pueden contener hasta 128 bytes de datos (cantidad recomendable);
- no es necesario establecer un circuito virtual [MART81] en forma previa al envío del datagrama. Este viaja a su destino sin ningún preámbulo;
- si la dirección destino no puede ser alcanzada por el datagrama debido a situaciones tales como: dirección no existente, falla de la máquina destino o falla en la red; entonces el datagrama será regresado al que lo envió con el código apropiado en el encabezado.

El formato general del datagrama es el propuesto para redes que se manejan de acuerdo al estándar CCITT X.25 y es el de la fig. 2.9.

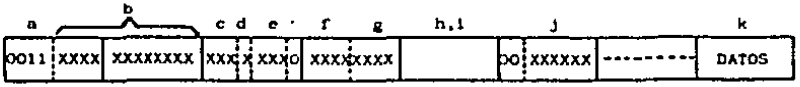


Fig. 2.9 Formato de un DATAGRAMA.

Cada uno de los campos de este formato se utiliza de la siguiente manera:

a) Indica que este paquete es un datagrama.

b) Contiene un número asignado por la micro (sucursal) emisora para identificar a cada datagrama. Tiene como propósito el que ésta lo identifique como suyo y lo retire de la red, cuando por algún motivo no se puede entregar y le es devuelto. El número se generará consecutivamente de 0 a 4095 y otra vez desde 0.

c) Contiene un número entre 0 y 7 que indica el número del próximo paquete que puede recibir esta máquina.

d) Este bit será igual a 0 cuando se quiera indicar que se van a enviar datagramas adicionales a la micro (sucursal) destino. Será igual a 1 cuando el datagrama actual sea el último que se envía. Para nuestro caso, en general será igual a 1 porque los intercambios entre una micro y otra normalmente constan de un solo paquete.

e) Contiene un valor entre 0 y 7 que indica el número del paquete que está enviando esta máquina.

f y g) Indican la cantidad de bytes ocupados por las direcciones de las máquinas emisora y receptora, respectivamente.

h e i) Constituyen el campo de direcciones de las micros emisora(fuente) y receptora(destino). Todas las micros de la red tienen una dirección que ocupa dos bytes. Cuando se trate de un nodo, en el primer byte estará el número de nodo (1 a 99) y en el segundo habrá un cero. Cuando se trate de una sucursal, en el primer byte estará el número del nodo al cual está conectada y en el segundo el número asignado a la sucursal (1 a 10, máximo), dentro de todas las sucursales de ese nodo. Por ejemplo, si se tiene el par de bytes:

2	0

estarán direccionando la micro que constituye el nodo 2 de la red. Si se tiene:

2	1

harán referencia a la micro de la sucursal 1 conectada al nodo 2.

j) Contiene un número que indica la cantidad de bytes (a continuación) que constituyen el campo de facilidades. En este campo normalmente habrá códigos para control de la red. Los que se usan aquí están relacionados principalmente con la recuperación de errores y fallas y se describen en la sección 5.1.

k) Constituye el campo de datos, o sea la información a enviar de una máquina a otra.

Para este diseño, la información exacta que almacena el datagrama, así como su longitud, se describen en las secciones 3.2 y 3.4.2. Por el momento consideraremos que todo el datagrama tiene una longitud fija de 18 bytes.

Para transmitir un datagrama de una sucursal a otra, toda la información descrita anteriormente deberá encontrarse dentro de un paquete HDLC, según se muestra en la figura 2.10.

MARCO	Dirección de la micro adyacente a la que se envía el paquete	Byte de control del paquete HDLC	DATAGRAMA (18 bytes)	CRC (2 bytes)	MARCO
01111110					01111110

Fig. 2.10 Paquete HDLC.

La longitud total de los paquetes HDLC intercambiados entre las micros del sistema, en nodos o sucursales, es de 24 bytes en el caso de los paquetes con formato I, y de 6 bytes en el caso de paquetes con formato S.

CAPITULO 3

SOFTWARE DE CONTROL DE LA RED

En base a los protocolos definidos para las capas 2 y 3 de la red, en este capitulo se definen las caracteristicas y forma de operación de lo que denominaremos Sistema de Transporte de la red. Este sistema está constituido por todos los programas, estructuras de datos, archivos, mecanismos y procedimientos de operación usados para que un paquete de información transmitido desde una sucursal fuente llegue a una sucursal destino, utilizando a los nodos como puntos intermedios para la transmisión de dicho paquete.

El esquema general de este sistema es el mostrado en la figura 3.1.

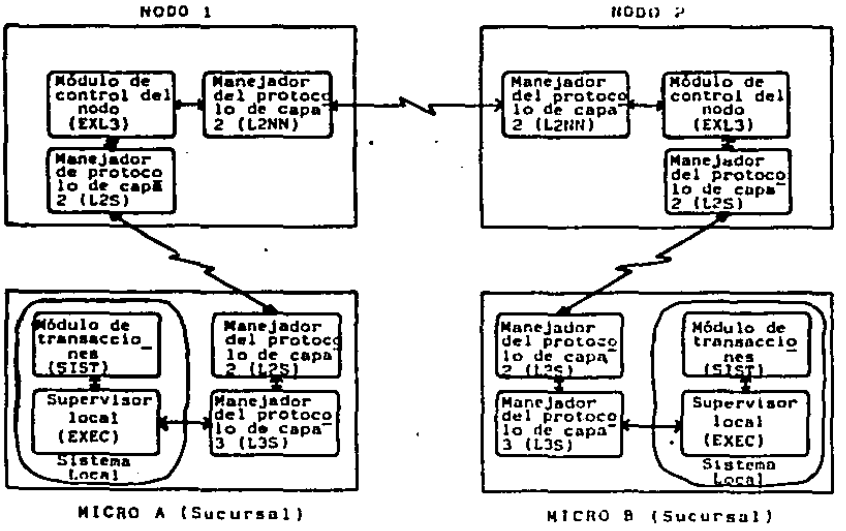


Fig. 3.1 Sistema de transporte de la red.

En este esquema se muestran todos los módulos (programas) involucrados en el sistema de transporte cuando una micro, de una sucursal, establece comunicación con otra micro de una sucursal diferente.

En el esquema estas micros se identifican como A y B para facilitar la explicación posterior que se hace del sistema. Los cuadros marcados como NODO 1 y NODO 2 se refieren a las micros que están en los nodos con los cuales se encuentran conectadas estas sucursales. Aunque en el esquema sólo aparecen dos nodos, en realidad puede haber más en la trayectoria de comunicación, ya que dependiendo del tráfico en la red, lineas caidas u otros inconvenientes, la información entre A y B puede pasar por más de dos nodos. Cuando ambas sucursales están conectadas al mismo nodo, éste sólo actúa como intermediario entre los paquetes transmitidos entre A y B.

En esta capitulo sólo se describen las funciones de los módulos que permiten establecer esta comunicación, así como las estructuras de información que utilizan. Estos módulos son los marcados como: Manejador del protocolo de la capa 2, Manejador del protocolo de la capa 3 y Módulo de control del nodo. A grandes rasgos sus principales funciones son las siguientes:

a) Manejador del protocolo de la capa 2 - Es el encargado de efectuar una transmisión confiable de información entre dos micros adyacentes, sean nodos o sucursales, por medio de los paquetes HDLC. Recibe el nombre de L2S cuando está como interfaz (de software) entre nodo y sucursales. Recibe el nombre de L2NN cuando está entre nodos. Los módulos son esencialmente los mismos, aunque los L2NN tienen estructuras de información de mayor tamaño que los L2S.

En cada nodo habrá un L2S por cada sucursal conectada a él. En cada sucursal existirá un L2S para el intercambio de información con el nodo asociado. También en cada nodo habrá un L2NN por cada nodo vecino que tenga en la red.

b) Manejador del protocolo de la capa 3 - Es el encargado de efectuar un intercambio confiable de paquetes de información entre dos sucursales cualesquiera de la red, por medio de los datagramas. A través de éstos, las sucursales intercambian información cuando tienen que procesar transacciones globales (o sea, transacciones efectuadas en la sucursal A con cuentas registradas en la sucursal B). Existe uno de estos módulos en cada sucursal de la red. Recibe el nombre de L3S.

c) Módulo de control del nodo - Es el encargado de supervisar la atención y ejecución de los módulos L2S y L2NN que se encuentran en un nodo. Además de esto realiza algunas funciones adicionales como: determinar la ruta que deben seguir los datagramas para acercarlos más a su destino, controlar el tráfico de los paquetes que pasan por el nodo, registrar sucursales o nodos fuera de servicio , comunicar estas fallas a los demás nodos y sucursales de la red, etc. Existe un módulo de este tipo en cada nodo.

Tiene el nombre de EXL3.

Los módulos marcados como Supervisor local (conocido como EXEC) y módulo de transacciones (denominado SIST), forman parte del sistema local de una sucursal. EXEC es el supervisor local que atiende a todos los programas que ejecuta la micro de una sucursal. SIST es el módulo que procesa especificamente las transacciones de consulta de saldo, depósito y retiro de dinero efectuadas por los clientes. Las funciones de estos módulos se describen en el capítulo siguiente. Por el momento consideraremos que tanto EXEC como SIST intervienen (parcialmente) en este sistema de transporte y que EXEC necesita dar atención (tiempo de ejecución) a SIST y L3S para que éstos puedan llevar a cabo las funciones que tienen asignadas.

En este punto cabe recordar la forma en la cual se procesa una transacción: si el cliente la efectúa con una cuenta que está registrada en la sucursal en que se encuentra, ésta es satisfecha localmente dentro de la sucursal; en caso contrario, será necesario realizar un procedimiento para accesar la cuenta del cliente en la sucursal en la que está registrada a fin de satisfacer la transacción. Con esto en mente, a continuación se describen las caracteristicas de estos módulos.

3.1 MANEJADOR DEL PROTOCOLO DE LA CAPA 2: L2S

Es el encargado de efectuar una transmisión confiable de información entre dos micros adyacentes. Recibe el nombre de L2S.

El objetivo principal del módulo es establecer un intercambio de información full-duplex tal como el que se describe en el apéndice A, utilizando para ello el protocolo HDLC descrito en la sección 2.3.2. Las estructuras y algoritmo descritos a continuación, están orientados a satisfacer estos requerimientos.

Como se mencionó anteriormente, el módulo recibe el nombre de L2NN cuando controla las comunicaciones entre nodos vecinos. L2S y L2NN son esencialmente el mismo módulo. La única diferencia es que L2NN tiene estructuras de mayor tamaño que las de L2S, debido a que el tráfico entre nodos es mayor que entre nodo y sucursal. Estos cambios se anotan al final de la descripción de estas estructuras.

Todos los paquetes que el módulo deba enviar a la micro adyacente, debe entregárselos al procesador frontal para que éste los transmita fisicamente, con formato HDLC, a dicha micro. Cuando el procesador frontal reciba paquetes HDLC, L2S debe requerirselos para que éste se los entregue.

El formato de los paquetes que L2S puede enviar es el mostrado en la figura 3.2.

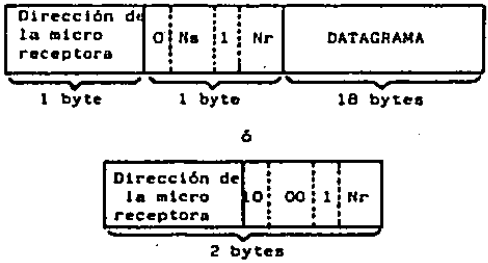


Fig. 3.2 Tipos de los Paquetes Recibidos por L2S.

El formato de los paquetes que recibe es semejante aunque sin el byte de dirección en ambos casos.

Como se mencionó en la sección 2.3.2, el USART se encarga de manejar por hardware los bytes de MARCO y CRC en la transmisión/recepción de los paquetes HDLC; de esta forma, el módulo L2S no debe preocuparse por los mismos. Asimismo, el USART descarta todos aquellos paquetes que son detectados con errores (p. ej., los que no concuerdan en el CRC). El procesador frontal, a su vez, controla, junto con el USART, que no se pierdan los bytes que se reciben desde la micro adyacente. Cuando los recibe, los va guardando hasta que L2S los toma para procesarlos. En la transmisión controlan el envío de bytes sin que intervenga L2S.

ESTRUCTURAS DE INFORMACION

Las siguientes son las principales estructuras usadas por este módulo:

a) Buffer BTRANS.- Es un arreglo de 640 bytes el cual almacena un máximo de 32 paquetes de 20 bytes cada uno. El arreglo se divide en 32 áreas numeradas del 0 al 31 (índices del área). El buffer se usa para almacenar temporalmente un paquete que está en espera de ser transmitido o que ya fue transmitido, pero todavía no se recibe la confirmación de que llegó bien. Cada paquete está constituido por:

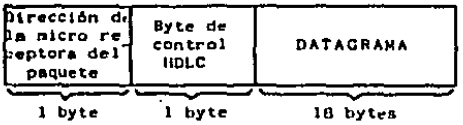


Fig. 3.3 Paquete HDLC almacenado en BTRANS.

De la figura anterior se observa que para cada paquete no se guardan los bytes de MARCO y CRC (sección 2.3.2) que conforman un paquete HDLC completo. No se hace esto porque el USART se los agrega automáticamente, por hardware, cuando efectúa la transmisión de paquetes.

En cuanto llega la confirmación de que llegó bien, el área ocupada por el paquete se libera.

b) Cola ESPTRA.- Es una cola circular de 25 elementos. Se emplea para almacenar los índices de los paquetes (1) que están en espera de ser transmitidos. Cada elemento tiene dos campos: uno es el apuntador al siguiente elemento; el otro contiene el índice (0 a 31) del área ocupada en BTRANS por el paquete a transmitir.

c) Cola ESPCT.- Es una cola circular de 7 elementos. Sirve para almacenar los índices de los paquetes que ya fueron transmitidos y que están en espera de que se reciba la confirmación de que fueron aceptados correctamente. Cada elemento contiene 4 campos:

- 1o., índice del paquete transmitido;
- 2o., hora en que se envió el paquete (al nodo o a la sucursal). Cada que se da atención a L2S se checa contra la hora actual. Si ya pasó un cierto tiempo pre-establecido (time-out) y no se ha recibido la aceptación, entonces se retransmite el paquete;
- 3o., número de retransmisiones del paquete. Si después de haber efectuado un determinado número de intentos de enviar el paquete no se recibe su aceptación, se considerará que la línea de comunicación o la micro receptora están fuera de servicio. En este caso se suspende el envío de paquetes y se toman acciones correctivas de emergencia; estas acciones se describen en el algoritmo de este módulo y en la sección 5.1.2;
- 4o., apuntador al siguiente elemento.

Como se observa, por el tamaño, en un momento dado sólo

(1) El término: índice del paquete, se refiere al índice del área ocupada por el paquete en BTRANS.

pueden estar siete paquetes en tránsito hacia una micro. Esto se hace tanto por la capacidad del buffer en la estación receptora, como para controlar el flujo de la transmisión de datos, tal como lo establece el protocolo HDLC.

Cuando un paquete puede ser transmitido se inserta al final de esta cola y se borra su índice de ESPTRA. Se forma el paquete HDLC y se envía a la micro receptora (nodo o sucursal adyacente). Cuando se recibe la confirmación de su aceptación , se borra su índice de la cola y se libera el área correspondiente de BTRANS.

d) Arreglo SR.- Es un arreglo lineal de cuatro elementos. Cada elemento tiene la siguiente función:

- SR[0]: almacena el número del siguiente paquete que va a enviar a la micro adyacente (este valor se coloca en el subcampo Ns del byte de control del paquete). El número varía de 0 a 7 (módulo 8);
- SR[1]: almacena el número del siguiente paquete que se espera recibir desde la micro adyacente (el valor se coloca en el subcampo Nr del byte de control). El número varía de 0 a 7 (módulo 8);
- SR[2]: contiene la cantidad de paquetes transmitidos a la micro adyacente que están en espera de recibir la confirmación de su aceptación por parte de ésta. Máximo pueden ser 7 paquetes;
- SR[3]: contiene un valor que indica la cantidad de paquetes que se pueden recibir simultáneamente desde la micro adyacente. Máximo pueden ser 7 paquetes.

e) Buffer BREC.- Almacena temporalmente los paquetes recibidos desde la micro adyacente y que están en espera de ser pasados al módulo L3S o EXL3, según se esté en sucursal o nodo. El buffer es de 608 bytes y está dividido en 32 áreas de 19 bytes cada una. En cada área sólo se guarda el byte de control y el datagrama (18 bytes) de cada paquete HDLC recibido desde la micro adyacente.

Como se mencionó anteriormente, las estructuras y algoritmo que usan L2S y L2NN son idénticos, aunque el tamaño de las estructuras de L2NN es mayor que el de las de L2S, dado que en un momento de la operación un nodo puede estar en comunicación con 10 sucursales y 4 nodos, como máximo, con 7 paquetes en tránsito con cada uno de ellos. De esta manera los tamaños de las estructuras de L2NN son:

- BTRANS: 3000 bytes, para almacenar hasta 150 paquetes de 20 bytes cada uno;
- ESPTRA: con capacidad para almacenar hasta 143 índices;
- ESPCT: permanece igual ya que sólo pueden estar en tránsito 7 paquetes como máximo;
- BREC: 2850 bytes, para almacenar hasta 150 paquetes de 19 bytes cada uno.

ALGORITMO

Para que este módulo lleve a cabo sus funciones, necesita que L3S, o EXL3 según corresponda, le den atención. Por lo tanto, todo lo descrito para el mismo se efectúa sólo hasta que el control es cedido a L2S.

Tanto en éste como en el resto de los algoritmos del presente capítulo y el siguiente, se hace mención a "datagramas de aviso". Estos datagramas son especiales y sirven para comunicar fallas (y también su recuperación) que pueden presentarse en la operación del sistema (p. ej., sucursal fuera de servicio). Estos problemas son descritos con detalle en el capítulo 5, junto con los datagramas de aviso empleados en cada caso. De cualquier forma, todos los diagramas contienen las acciones de emergencia que deben efectuarse cuando ocurren estas situaciones anormales.

También, todos los algoritmos tienen rutinas de INICIO y FIN, que sirven para establecer las condiciones iniciales de operación y para efectuar las acciones finales del módulo, respectivamente. Se describen aparte del algoritmo central debido a que sólo se usan cuando se inicia y termina la ejecución del módulo. Por lo tanto, en los diagramas de bloques de los algoritmos estas rutinas no aparecen. A continuación se describe el algoritmo de L2S.

Rutina INICIO

1. Limpia BTRANS, BREC, ESPTRA y ESPCT.
2. Hace: SR[0]=0; SR[1]=0; SR[2]=0; SR[3]=7.
3. Hace: LOCAL=false (LOCAL es una variable que se pone en "verdadero" cuando la sucursal queda trabajando sólo localmente; está en "falso", en caso contrario).

Rutina FIN

No se efectúan acciones adicionales.

Rutina PRINCIPAL

Diagrama de la figura 3.4.

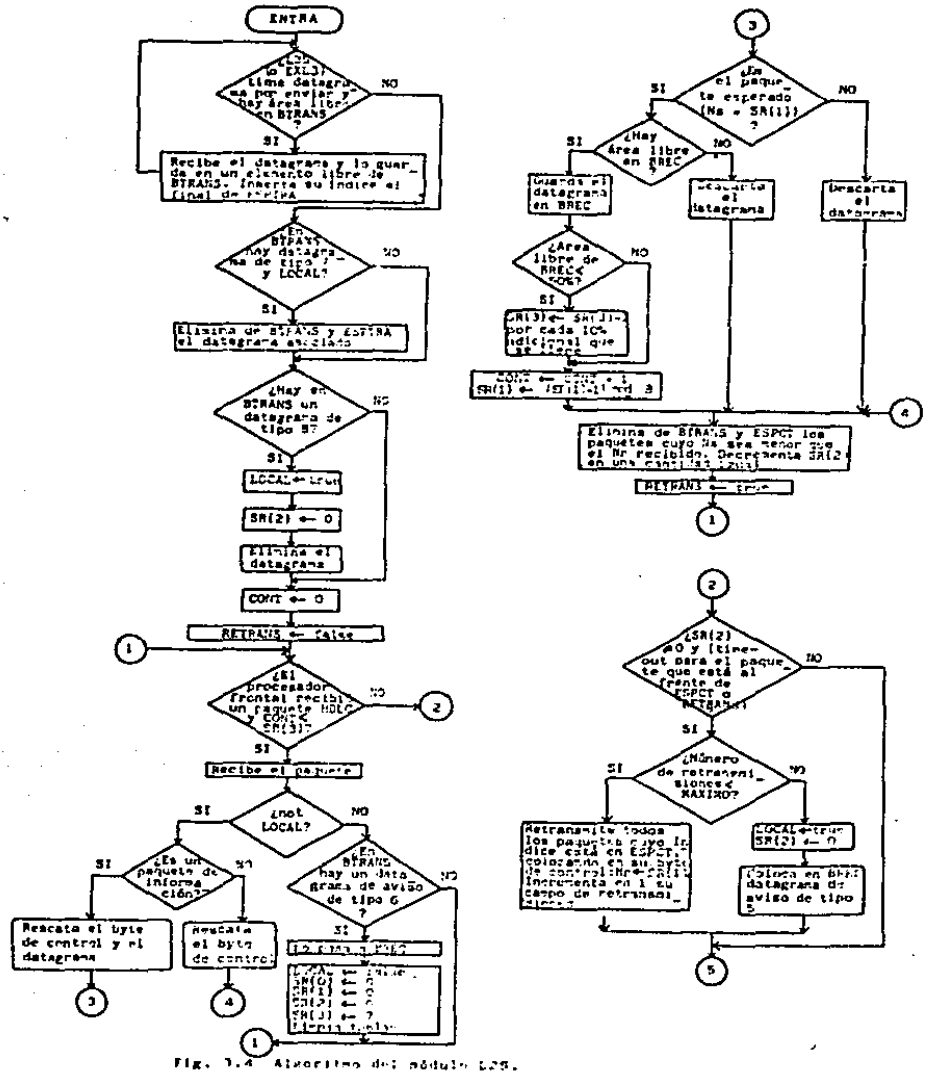


Fig. 3.4 Algoritmo del módulo LDS.

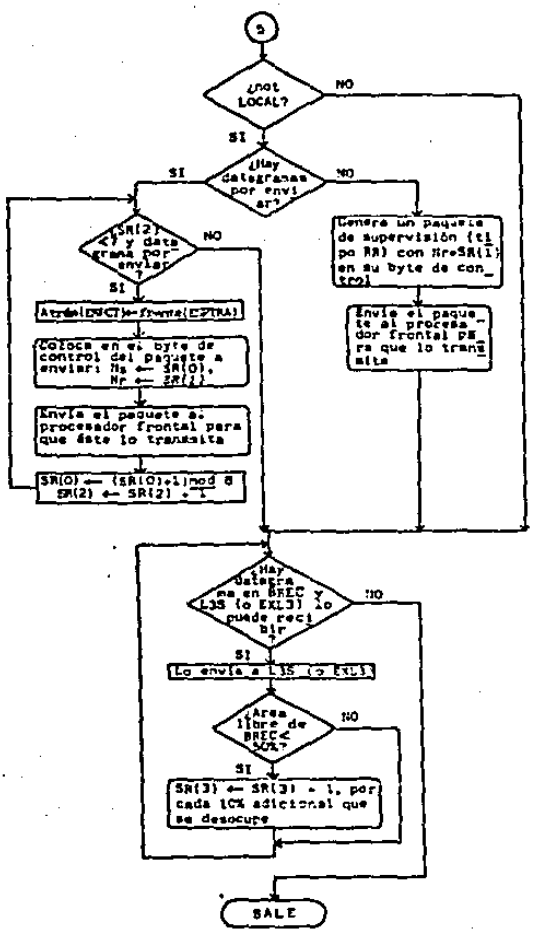


Fig. 3.4 Algoritmo del módulo L2S (Cont.).

3.2 MANEJADOR DEL PROTOCOLO DE LA CAPA 3: L3S

Es el encargado de efectuar un intercambio confiable de paquetes de información entre dos sucursales de la red. Para intercambiar estos paquetes utiliza los Datagramas, descritos en la sección 2.3.3. Recibe el nombre de L3S. Se tiene uno de estos módulos en la micro de cada sucursal.

Existen tres clases de datagramas que una sucursal (p. ej. A) puede emitir:

1 - SOLICITUD
2 - RESPUESTA
3 - CONFIRMACION

El datagrama pertenece a la clase SOLICITUD cuando contiene una petición para usar los datos que están en otra sucursal (p. ej. B). Es de la clase RESPUESTA, cuando contiene información que A manda como respuesta a una solicitud recibida desde B. Y pertenece a la clase CONFIRMACION cuando A envía un datagrama como confirmación de que una respuesta enviada desde B se recibió correctamente en A.

La sucursal B también puede emitir la misma clase de datagramas con respecto a A.

Lo anterior significa que para procesar una transacción global se requiere efectuar los siguientes pasos (suponiendo que la transacción se origina en A y la cuenta a manejar está en B):

a) La sucursal A emite un datagrama de SOLICITUD hacia B, enviándole los datos necesarios para que procese la transacción.

b) La sucursal B recibe el datagrama y procesa la transacción con la cuenta indicada en el mismo. Con el resultado producido genera un datagrama de RESPUESTA el cual envía a A.

c) La sucursal A lo recibe, toma la información y contesta con un datagrama de CONFIRMACION hacia B. A da por terminada la transacción.

d) B lo recibe y con esto da por terminada la transacción.

Cuando la transacción es local (o sea, la transacción se inicia en A y la cuenta está ahí mismo), no es necesario recurrir a lo anterior, ya que ésta se procesa con los recursos propios de la sucursal. En la sección 3.4.3 se describe con más detalle el procedimiento seguido para procesar cualquier tipo de transacción.

En la descripción anterior no se considera que haya problemas en el procesamiento de la transacción global. Cuando estos se presentan (p. ej., que la sucursal A quede fuera de servicio antes de recibir el datagrama de RESPUESTA), será necesario

efectuar acciones de emergencia. Para estos casos se usa una cuarta clase de datagramas que denominaremos de AVISO. Por medio de éstos se comunican a nodos y sucursales las diversas fallas que se presentan durante la operación del sistema. Como se mencionó en la sección anterior, estas cuestiones se tratan con más detalle en el capitulo 5.

Los datagramas que emiten las sucursales tienen una longitud de 18 bytes. En la figura 3.5 se muestra un ejemplo de ellos.

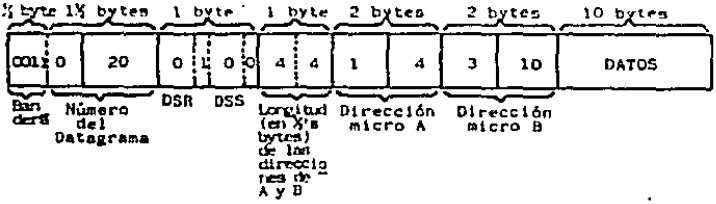


Fig. 3.5 Ejemplo de un Datagrama Construido por L3S.

En este ejemplo el datagrama a enviar es el número 20 generado por A. DSR y DSS sirven para indicar la clase de datagrama que se está enviando a B. Los valores que pueden tomar DSR y DSS son:

Datagrama	DSR	DSS
SOLICITUD	0	0
RESPUESTA	0	1
CONFIRMACION	1	0
AVISO	1	1

(Un datagrama de aviso también puede ser generado por el Supervisor de un nodo (módulo EXL3)).

En base a la figura, la dirección de A indica que está conectada al nodo 1 y es la número 4 de las sucursales de ese nodo. B está en el nodo 3 y es la número 10 de dicho nodo. El campo de DATOS que se envía dentro del datagrama siempre tiene una longitud de 10 bytes y su formato se explica en la sección 3.4.2.

En base a lo anterior, las principales estructuras de información que utiliza este módulo, y su algoritmo, son los descritos a continuación.

ESTRUCTURAS DE INFORMACION

a) Buffer BT.- Es un arreglo lineal de 864 bytes usado para almacenar un máximo de 48 datagramas de 18 bytes cada uno. Se divide en 48 áreas numeradas del 0 al 47 (índice del área). Esta

estructura almacena temporalmente datagramas que ya fueron enviados a otra sucursal. Cuando se recibe la confirmación de su aceptación, el área ocupada por éstos se libera.

b) Tabla TRANS.- Se utiliza para guardar algunos datos relacionados con los datagramas que fueron enviados por la sucursal. Su estructura es:

Número del datagrama	Dirección de la micro destino	Clase del datagrama	Hora de envío	Area ocupada en el buffer BT	Número de cuenta	Número de terminal

Tabla 3.1 Estructura de la tabla TRANS.

El título de las cinco primeras columnas indica la información que se guarda en ellas.

La columna NUMERO DE CUENTA sirve para guardar el número de la cuenta objeto de la transacción.

En la última columna habrá -1 en caso de que el datagrama enviado corresponda a la RESPUESTA a una SOLICITUD recibida desde B. Si se trata de una SOLICITUD o una CONFIRMACION enviada por A, esa columna guardará el número de la terminal en la cual se está manejando la transacción correspondiente.

Para las dos primeras clases de datagramas, en esta tabla se guardan los datos mencionados anteriormente, hasta que se recibe la confirmación de su aceptación por parte de la micro destino. En cuanto ésta se recibe, se desocupa la fila correspondiente y se libera el area ocupada en BT. Si el datagrama es una confirmación, éste se elimina de la tabla, y se libera el área que corresponda de BT,sólo después de un periodo de tiempo pre-determinado.

c) Buffer BR.- Tiene la misma finalidad que BT sólo que para los datagramas recibidos. Tiene capacidad para almacenar 48 datagramas. Por cada datagrama existe un byte adicional (que llamaremos BAND) el cual estará en TRUE cuando un datagrama se reciba por primera vez. Se pondrá en FALSE cuando el mismo datagrama se reciba por segunda o más veces. Este byte tiene la finalidad de detectar si un datagrama nuevo ya fue recibido anteriormente. Si así sucede, el datagrama nuevo ya no se procesa con lo cual se evita doble procesamiento de una misma transacción.

d) Cola INFOD.- Se usa para almacenar temporalmente el contenido

del campo de DATOS que viene en los datagramas recibidos por L3S. Antes que L3S regrese el control a EXEC, le da toda la información almacenada en esta cola para que SIST la procese. Cada elemento de la cola tiene 4 campos:

- 1o., para almacenar los DATOS;
- 2o., para almacenar la dirección de la micro origen;
- 3o., para almacenar el número de la terminal (o -1, en su caso); y
- 4o., para apuntar al siguiente elemento de la cola. Esta estructura tiene 48 elementos como máximo.

ALGORITMO

Las acciones descritas a continuación se llevan a cabo hasta que EXEC le da atención a L3S. Esto sucede en dos casos: cuando SIST (o el propio EXEC), de la sucursal A (ver figura 3.1), necesita enviar información a la sucursal B, ya sea una solicitud, una respuesta o una confirmación; o cuando EXEC le da atención debido a que ya le toca su "turno" de acuerdo con la política de "Round-robin" [HABE76] manejada por aquel (descrita en la siguiente sección). Las acciones que efectúa en cada ocasión son las especificadas en el diagrama de la figura 3.6.

Rutina INICIO

1. Limpia estructuras.
2. Ordena la ejecución de L2S.

Rutina FIN

1. Termina ejecución de L2S.

Rutina PRINCIPAL

Diagrama de la figura 3.6 (en él aparece el término "paquete de aviso", el cual corresponde al campo de DATOS de un datagrama de aviso).

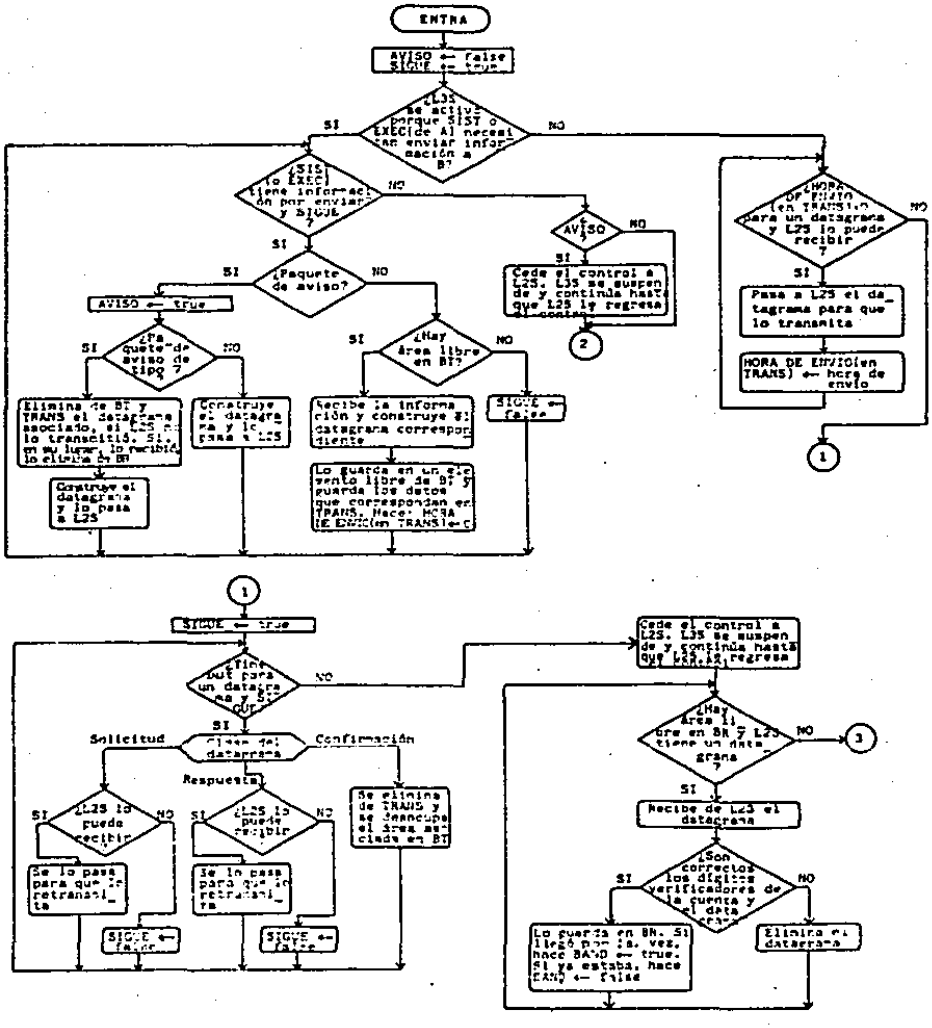


Fig. 3.6 Algoritmo del módulo L35.

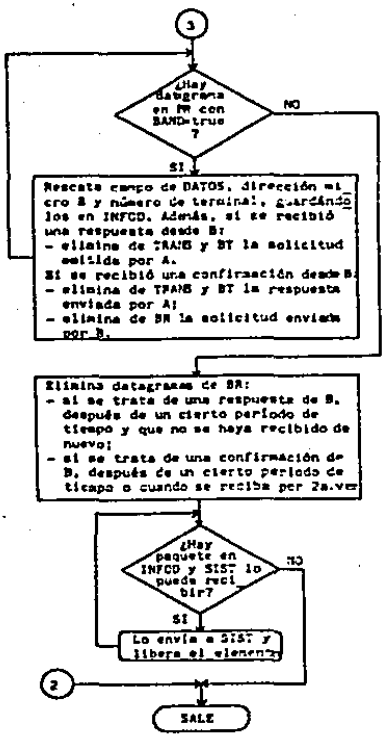


Fig. 3.6 Algoritmo del módulo L3S (Cont.).

3.3 MODULO DE CONTROL DE NODOS: EXL3

Es un módulo que sólo se localiza en las micros de todos los nodos de la red. Recibe el nombre de EXL3. Entre las principales funciones de este módulo se encuentran las siguientes:

- atender y ejecutar a los módulos L2S y L2NN del nodo para transmitir los datagramas recibidos por éstos, a otras sucursales u otros nodos;
- determinar la ruta que debe seguir un datagrama para acercarlo más a su destino;
- registrar lineas de comunicación caidas entre nodo y nodo o entre nodo y sucursal;
- registrar sucursales caidas conectadas a ese nodo;
- comunicar a los demás nodos y sucursales de la red la ocurrencia de estas fallas para que tomen medidas preventivas, a fin de que no traten de utilizarlas para la transmisión de paquetes;
- realizar acciones correctivas adicionales cuando se presenten esos tipos de falla;
- eliminar temporalmente datagramas para sucursales o nodos, cuando las colas de recepción y envio de los módulos correspondientes estén saturadas; o sea, controlar el tráfico de paquetes;
- retirar de la red los datagramas que estén circulando sin ningún destino (éstos pueden ser producto de errores en la transmisión que afectan el contenido del campo de dirección de la micro destino);
- retirar de la red a todos aquellos datagramas que no cumplan con el formato establecido;

Este módulo auna sus funciones a las de L3S y juntos definen los mecanismos de operación que se utilizan para la capa 3 de la red.

Para describir las estructuras y el algoritmo usados por EXL3 consideraremos que en un nodo existen los módulos de software (tomando como base el nodo 2 de la figura 2.2, con 10 sucursales) indicados en la figura 3.7.

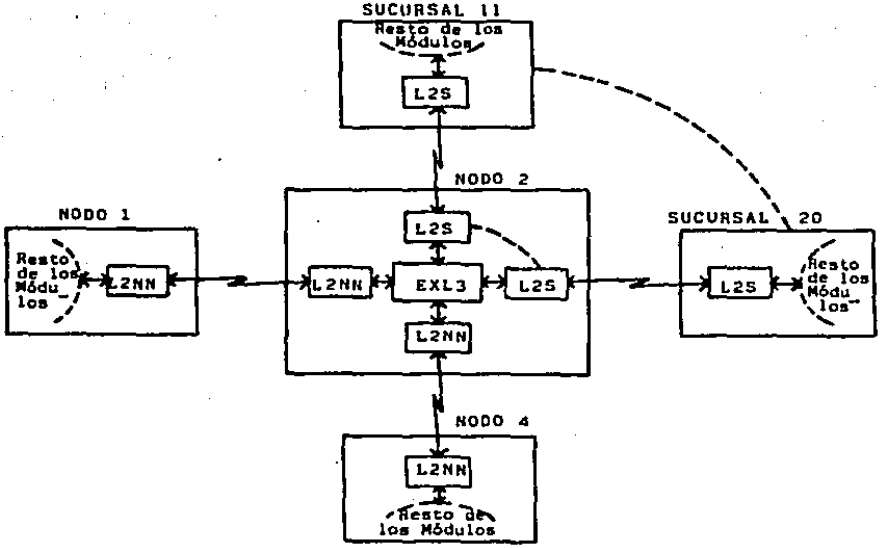


Fig. 3.7 Módulos de software de un nodo.

ESTRUCTURAS DE INFORMACION

Las principales estructuras utilizadas son:

a) Tabla ASIGS - Esta tabla es usada por EXL3 para determinar que módulo L2S debe utilizar cuando quiere intercambiar información con una determinada sucursal. Su estructura es la mostrada en la tabla 3.2.

La tabla tiene un máximo de 10 filas para almacenar datos de un máximo de 10 sucursales. Internamente EXL3 asigna un índice a cada L2S para identificarlo. La relación entre un L2S y la sucursal con la cual se comunica se establece con las dos primeras columnas de esta tabla. La columna de status

Indice asignado a cada L2S (dentro de EXL3)	Número de sucursal (dentro de ese nodo)	Status

Tabla 3.2 Estructura de la Tabla ASIGS.

ocupa un byte por sucursal y cada bit indica lo siguiente:

Bit	Función
0	Sucursal fuera de servicio, ya sea por falla en la línea de comunicación o por falla en la micro de la sucursal.
1	Saturación en la cola de recepción de paquetes de L2S.
2	Saturación en la cola de transmisión de paquetes de L2S.
3 a 7	Sin uso.

Si ocurre el evento indicado, el bit correspondiente estará en 1; si no, en 0.

El Status de una sucursal puede cambiar dinámicamente durante el transcurso de una sesión de trabajo, ya sea para suspender o rehabilitar la atención a la misma.

La tabla está siempre en un archivo. Cuando se inicia una sesión de trabajo, EXL3 lee el archivo y carga la tabla para su uso en la sesión. Se tiene en archivo para que pueda ser modificada por programa (fuera de sesión) si es que cambia la cantidad de sucursales conectadas al nodo, o la asignación de módulos L2S con respecto a las sucursales.

b) Tabla ASIGN - Es una tabla semejante a la anterior pero usada para establecer la relación que existe entre los módulos L2NN y el número del nodo al que atienden. Esta tabla tiene 4 filas para permitir que a un nodo se le conecten máximo otros 4 nodos. Esta tabla está en el mismo archivo que la anterior.

c) Tabla RUTA - Contiene las direcciones de los nodos vecinos hacia los cuales se debe enviar un datagrama para acercarlo más a

su nodo destino. La estructura de la tabla es:

Nodos Destino \ Rutas	1a.	2a.	3a.	4a.
1				
2				
⋮				
99				

Tabla 3.3 Estructura de la Tabla RUTA.

Cada elemento de la tabla es un registro de 4 campos. Cada campo ocupa un byte por lo que el tamaño máximo de la tabla es de 396 bytes. El tamaño de la tabla permite manejar hasta 99 nodos en la red. Aunque en principio la red puede crecer modularmente hasta cualquier tamaño, es necesario establecer un límite en la cantidad de nodos que ésta puede tener ya que hay algunas estructuras de información, como la presente, que se usan en los módulos de software las cuales requieren tener un tamaño definido dentro de los mismos. Con 99 nodos como máximo, el sistema puede manejar hasta 990 sucursales que es un número mas que suficiente para satisfacer las necesidades de cualquier sistema bancario nacional. Por ejemplo, Banamex y Bancomer, que son las dos instituciones principales del país, manejan actualmente de 600 a 700 sucursales en toda la república. Así, 99 nodos es un buen límite para el crecimiento de este sistema.

Volviendo a la tabla, tenemos que para cada nodo destino se registra la dirección del siguiente nodo vecino hacia el cual se debe enviar el datagrama para acercarlo más a dicho nodo. Para cada nodo destino en realidad se tienen cuatro rutas alternas, para prevenir el caso de falla en alguna de ellas. La primera alternativa representa la ruta más corta entre este nodo y el destino; la última, la más larga. En cada campo existe un valor que representa lo siguiente:

Valor	Significado
+n	Indica que el nodo n (1<=n<=99) está en servicio.
-n	Indica que el nodo n (o la línea de comunicación con éste) está fuera de servicio.
0	Indica que no hay conexión por esa ruta alterna.

Por ejemplo, de acuerdo con la configuración mostrada en la figura 2.2, los valores que pueden aparecer en la tabla RUTA del nodo 1 son:

1	0	0	0	0
2	2	3	0	0
3	3	2	0	0
4	3	2	0	0
5	0	0	0	0
6	0	0	0	0
		⋮	⋮	
99	0	0	0	0

Para llegar de un nodo a otro existen por lo menos dos rutas alternas, ya que cada nodo se conecta por lo menos a otros dos. Máximo existen cuatro rutas alternas.

El contenido de esta tabla cambia de un nodo a otro. Por ejemplo los valores que tendría para el nodo 2, según figura 2.2, serían:

1	1	4	0	0
2	0	0	0	0
3	1	4	0	0
4	4	1	0	0
5	0	0	0	0
6	0	0	0	0
		⋮	⋮	
99	0	0	0	0

Cuando ocurre una falla en la línea de comunicación entre dos nodos vecinos, cada nodo debe registrar en su tabla este hecho (colocando un signo negativo en el campo que corresponda). Cuando falla un nodo, los vecinos deben anotar este suceso en sus tablas.

En ambos casos, la finalidad es evitar enviar datagramas por las líneas o nodos que estén fuera de servicio.

Cuando un datagrama recibido se debe enviar a una sucursal del mismo nodo, no es necesario consultar la tabla, ya que ésta sólo se usa cuando la sucursal destino está en otro nodo.

Esta tabla se guarda en un archivo para que pueda ser modificada por programa (fuera de sesión) cuando ocurra un cambio en la misma, p. ej. con la inclusión de un nuevo nodo en la red.

d) Tabla INTERS - Sirve para almacenar temporalmente datagramas que deben ser pasados a módulos L2S del nodo, ya sea desde otros L2S o desde módulos L2NN. Su estructura se muestra en la tabla 3.4.

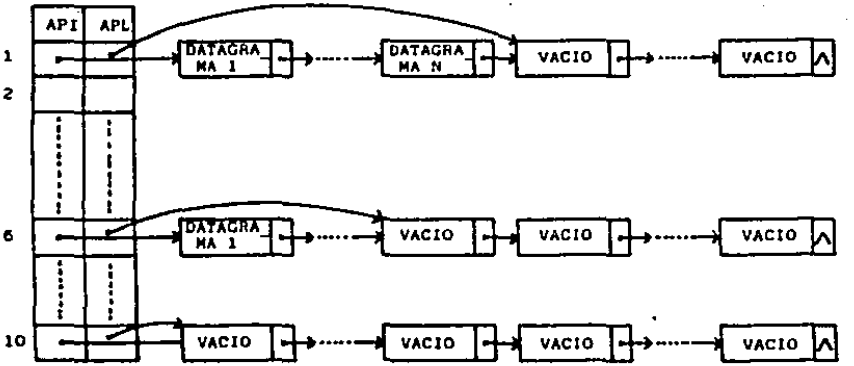


Tabla 3.4 Estructura de la Tabla INTERS.

Cada elemento de la tabla contiene dos apuntadores a una lista de 32 elementos (máximo). API es el apuntador al 1er. elemento de la lista. APL el apuntador al 1er. elemento desocupado. Cada elemento de la lista tiene dos campos: uno de 18 bytes para almacenar un datagrama y otro para apuntar al siguiente elemento de la lista.

Cada que el nodo reciba un datagrama para una sucursal conectada a él, éste será insertado temporalmente en la lista que corresponde a dicha sucursal (numeradas del 1 al 10). La inserción se hace en el elemento apuntado por el APL de esa sucursal. En un momento dado la tabla puede presentar un aspecto como el de la tabla 3.4. Cuando EXL3 le da atención al L2S de la sucursal (p. ej. 1), entonces le envía todos los datagramas que están en su lista para que L2S, a su vez, los envíe a la sucursal.

En ese momento la lista queda como se observa para la sucursal 10.

La tabla se define a su tamaño máximo, para evitar pérdida de tiempo en la asignación y liberación del espacio ocupado por los elementos. El espacio por lista es:
32(elementos) x (18 bytes(datagrama) + 2 bytes(apunt.)) = 640 bytes

El espacio de la tabla es:
10(sucursales) x 4 bytes(apuntadores) = 40 bytes

El tamaño total de la estructura sería:
10(listas) x 640 bytes(cada lista) + 40 bytes(tabla) = 6440 bytes

e) Tabla INTERN.- Es una tabla semejante a la anterior, pero que almacena, temporalmente, datagramas que deben ser enviados a módulos L2NN, para que éstos los transmitan a los nodos. El tamaño varía un poco: hay 96 elementos por lista y un máximo de 4 filas en la tabla. El tamaño máximo de ésta es de 7696 bytes.

f) Colas NORMAL y ALTAP - Se usan para colocar ahí los índices de los módulos L2S y L2NN controlados por EXL3. Un módulo colocado en la cola ALTAP (alta prioridad) recibe el doble de atención por parte de EXL3, que uno colocado en la NORMAL (prioridad normal). Las colas se accesan secuencialmente de tal manera que los módulos se ejecutan conforme se van encontrando.

g) Cola TEMP - Almacena temporalmente datagramas recibidos desde L2S o L2NN. Consta de 48 elementos.

h) Tabla TOTSUC - Registra a todos los nodos y sucursales existentes en el sistema. Es un arreglo de 99 enteros.

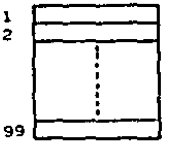


Tabla 3.5 Estructura de la tabla TOTSUC.

Cada elemento está asociado con un nodo. En cada elemento, el bit 15 está asociado con la sucursal 1 de ese nodo, el bit 14 con la sucursal 2, etc. El bit estará en 0 cuando no haya sucursal asociada; estará en 1 cuando sí exista sucursal.

La tabla tiene como finalidad detectar datagramas cuya dirección destino no existe y eliminarlos de la red.

ALGORITMO

La filosofía del algoritmo de EXL3 es parecida a la de EXEC (descrito con detalle en el siguiente capítulo) en los siguientes aspectos:

- EXL3 controla la ejecución de los módulos L2S y L2NN del nodo, tal como EXEC controla a los módulos de una sucursal;
- la política de atención a los módulos en ambos casos es la misma;
- EXEC utiliza "Pipes" (del sistema operativo Cromix) para comunicarse con los módulos, al igual que EXL3 los usa para comunicarse con L2S y L2NN;
- ambos manejan dos colas de prioridades para dar más atención a ciertos módulos que a otros;
- las acciones que debe efectuar EXL3 al inicio y fin de una sesión de trabajo son muy parecidas a las que realiza EXEC.

Por estas razones, en este algoritmo sólo se detalla aquello que no hace EXEC como: determinación de la ruta de un datagrama, control del tráfico de datagramas, eliminación de datagramas erróneos. El resto de las acciones de supervisión, ejecución y comunicación con los módulos L2S y L2NN se tomará como semejante a lo descrito en el siguiente capítulo para el módulo EXEC.

Rutina INICIO

1. Inicia la ejecución de los L2S y L2NN, según la cantidad de sucursales y nodos que estén ligados a éste. 2. Establece pipes de comunicación con ellos. 3. Coloca los índices de los L2S en la cola NORMAL y los de los L2NN en ALTAP. Se hace esto para dar mayor atención a los módulos que comunican con otros nodos ya que existe mayor tráfico entre éstos, que entre nodo y sucursales. La atención a ambos módulos se hace en base a una política de round-robin, como en el caso de EXEC. Por ejemplo, en base a la configuración de la figura 3.7, la política de atención es: L2S(suc. 1), L2NN(nodo 1), L2NN(nodo 4), L2S(suc. 2), L2NN(nodo 1), L2NN(nodo 4),...., L2S(suc. 10), L2NN(nodo 1), L2NN(nodo 4), L2S(suc. 1), etc.

O sea, por cada sucursal atendida se da atención a dos nodos.

Rutina FIN

1. Termina la ejecución de los L2S y L2NN. 2. Da de baja el sistema de transacciones (es decir, la parte correspondiente al nodo).

Rutina PRINCIPAL

Diagrama de la figura 3.8.

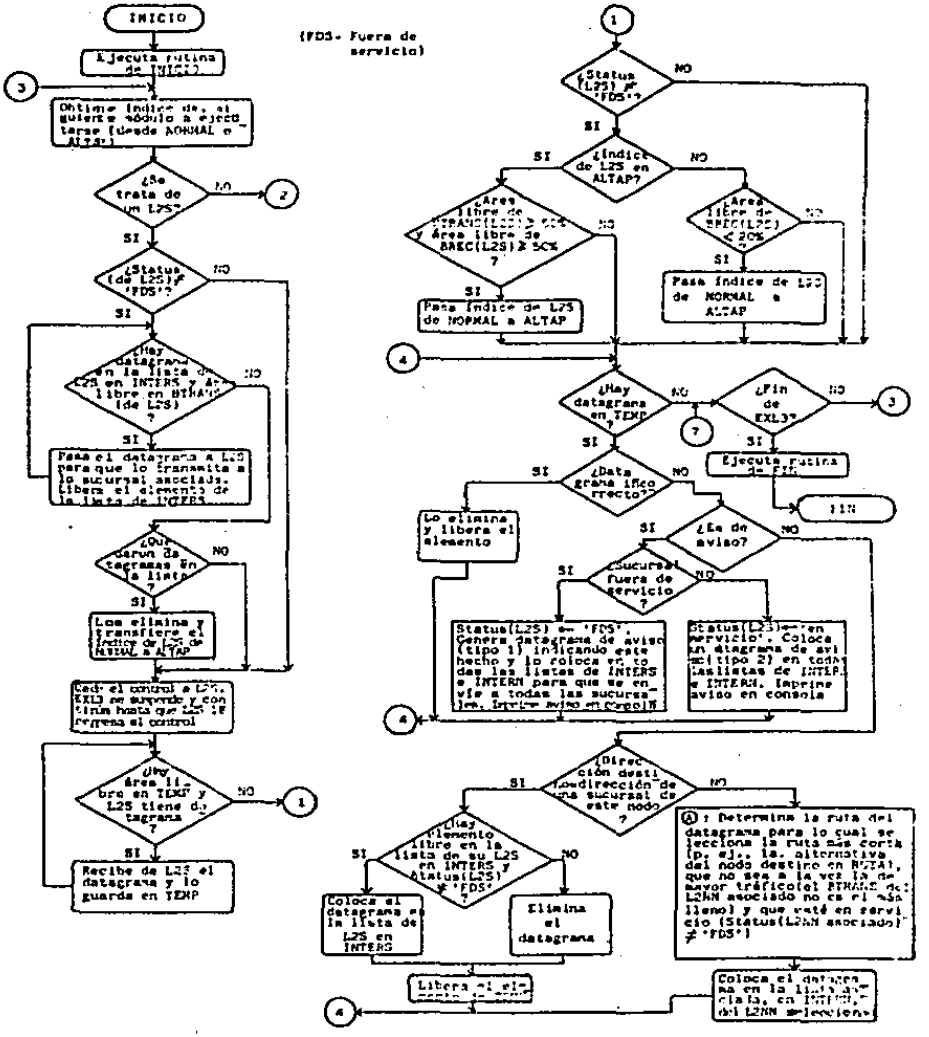


Fig. 3.8 Algoritmo del módulo EXL3.

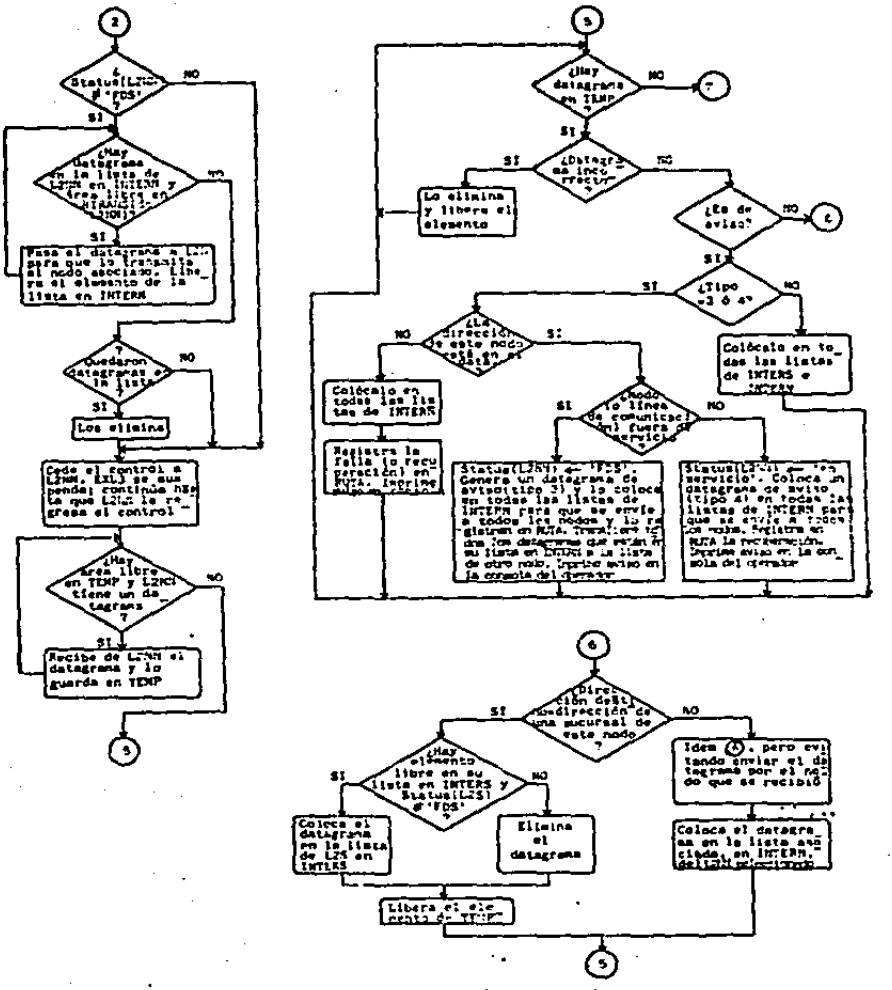


Fig. 3.8 Algoritmo del módulo EXL3 (Cont.).

3.4 PROCESAMIENTO GLOBAL DE UNA TRANSACCION

En esta sección se especifican las acciones concretas que deben efectuarse para llevar a cabo el procesamiento de una transacción, ya sea que ésta tenga que manejarse localmente o que tenga que procesarse globalmente.

3.4.1 Formato de las Cuentas Manejadas por el Sistema

Todo el conjunto de cuentas que pueden manejarse está distribuido entre todas las sucursales del sistema bancario. Por lo tanto es necesario establecer un formato para el manejo de las mismas que indique qué cuentas están registradas en qué sucursales, señalando además el tipo de cuenta de que se trata. El formato propuesto para las cuentas consiste de cuatro números según se muestra a continuación:

N1 - N2 - N3 - N4

Cada uno de estos números tiene la siguiente función:

a) N1 - indica el nodo al cual está conectada la sucursal donde está registrada la cuenta. El número está formado por dos dígitos para manejar nodos del 01 al 99, cantidad mas que suficiente para cualquier tamaño de la red propuesta.

b) N2 - indica de qué sucursal se trata dentro de todo el conjunto de sucursales conectadas al mismo nodo. Está formado también por dos dígitos para manejar sucursales del 01 al 10, que es el máximo que puede conectarse a un nodo.

c) N3 - representa un dígito que especifica el tipo de cuenta que se maneja. Los códigos son:

Código	Tipo de Cuenta
1	Cheques
2	Ahorro
3	Tarjeta de Crédito
4	Valores retirables en días pre-establecidos

d) N4 - representa un número asignado al cliente en la sucursal en la cual se registra. El número varía desde 1 hasta un cierto máximo. En principio no se da un valor concreto, pero debe ser lo suficientemente grande para poder manejar una gran cantidad de clientes en cada sucursal.

Del formato propuesto para las cuentas se observa que, para una misma sucursal, todas tienen el mismo número de nodo y sucursal. Por lo tanto para estas cuentas basta con guardar el tercer y cuarto números para su identificación dentro de la propia sucursal. Los números de nodo y sucursal para dichas cuentas se tienen en un archivo llamado IDENT.DAT, descrito con detalle en la sección 4.3.2. El número de cuenta sigue siendo único dentro de todo el sistema.

3.4.2 Información Enviada por una Sucursal

En esta sub-sección se describe el formato utilizado para transmitir información entre las sucursales, relacionada con el procesamiento de las transacciones en sí mismas, mas que con los protocolos usados para transmisión entre las diferentes micros de la red.

Como se mencionó en la sección 3.2, una sucursal puede generar cuatro clases de datagramas:

1 - SOLICITUD

2 - RESPUESTA

3 - CONFIRMACION

4 - AVISO

En cualquiera de los cuatro casos siempre se transmiten 10 bytes en el campo de DATOS -además de los requeridos por el datagrama y por el paquete HDLC-, sea que estos contengan información o no. Esto se hace para uniformizar la cantidad de bytes transmitidos entre punto y punto de la red y, sobre todo, para tener un mismo tamaño, para el manejo de los paquetes de información, en las diferentes estructuras que utilizan los diversos módulos del software de la red.

A continuación se describe la información que contiene cada uno de estos bytes; en el caso de un CONFIRMACION sólo almacenan información relevante los bytes 1, 3 y 4, guardando ahí el tipo y número de cuenta manejada. Los bytes 2 y 5 a 8 guardan ceros.
El caso de un datagrama de AVISO se trata en la sección 5.1.

a) Byte 1 - Contiene información la cual indica que a partir de este byte comienza el campo de datos de un datagrama. Almacena los siguientes valores:

11000XXX

Los primeros 5 bits contienen un patrón fijo para indicar el comienzo del campo. Los 3 últimos bits indican el tipo de cuenta

que se va a manejar de acuerdo con lo descrito en el inciso c) de la sub-sección anterior.

b) Byte 2 - Contiene información que indica el tipo de transacción que se va a efectuar sobre la cuenta, así como información sobre el resultado de la operación después que la transacción es realizada. Los primeros 3 bits son usados para indicar el tipo de transacción de acuerdo a lo siguiente:

Código	Tipo de Transacción
1	Saldo
2	Depósito
3	Retiro

Los últimos 5 bits se usan para indicar las siguientes acciones:

Código	Tipo de Acción
1	Solicitud para efectuar la transacción
2	Transacción terminada satisfactoriamente
3	Error: no existe este tipo de cuenta
4	Error: no existe este número de cuenta
5	Error: no se pudo accesar la cuenta
6	Error: saldo menor que la cantidad pedida (sólo en caso de que la transacción sea Retiro)

c) Bytes 3 y 4 - Contienen el número de cuenta que se está manejando. Como son dos bytes en principio se pueden manejar cuentas en el rango de 1 a 65535.

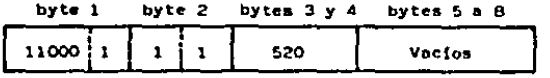
d) Bytes 5 a 8 - Contienen la cantidad de dinero que se maneja en la transacción. Cuando se solicita un saldo estos bytes están vacíos. Después que se hace la transacción, aquí se guarda la cantidad que tiene como saldo la cuenta. En caso de depósito o retiro, cuando se solicita la transacción, en estos bytes se coloca la cantidad que se va a depositar o retirar. Cuando se realiza la transacción estos bytes quedan vacíos.

e) Byte 9 - Contiene un dígito verificador del número de cuenta manejado, considerando nodo, sucursal y tipo de cuenta, también. Su objetivo es mantener la integridad del número de cuenta a manejar con la transacción, para evitar usar otra cuenta que se pudiera obtener como producto de un error en la transmisión. Se obtiene sumando el contenido de los bytes 1, 3, 4 anteriores y el de los bytes de la dirección de la micro destino. A la suma se le aplica módulo 256 y el resultado es el dígito verificador.

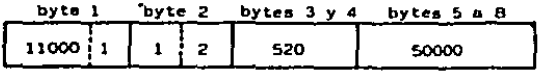
f) Byte 10 - Contiene un dígito verificador de la información de todo el datagrama. Su objetivo es mantener la integridad de todo el datagrama. Se obtiene sumando el contenido de los 18 bytes que forman el datagrama, exceptuando éste, y aplicando módulo 256. Cuando en una sucursal se recibe un datagrama, si no concuerdan

los valores de estos dos bytes con lo que deberían tener, L3S lo descarta.

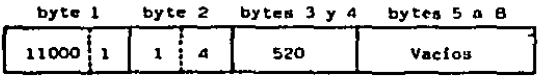
A continuación se muestra un ejemplo de la información que pueden tener estos bytes (exceptuando los dos últimos) cuando se hace una transacción con una cuenta de cheques:



En este ejemplo se está solicitando el saldo de la cuenta de cheques número 520. Después de efectuar la transacción la información que se envía como respuesta puede ser:



Se está contestando que la transacción sobre la cuenta 520 se realizó satisfactoriamente. El saldo de la cuenta es de 50,000 pesos. Si al efectuar la transacción ese número de cuenta no existe, se envía la siguiente información:



Finalmente, estos 10 bytes constituyen el campo de DATOS de los datagramas intercambiados entre las sucursales de la red y siempre existirán ya sea que contengan información o no.

3.4.3 Procesamiento de las Transacciones

En esta parte se describen en forma detallada todas las acciones que se deben realizar para llevar a cabo el procesamiento de una transacción de un cliente, ya sea que éste se encuentre registrado en la sucursal en la cual se origina la transacción o no. Se indica la secuencia de pasos a efectuar, así como los diferentes módulos y estructuras que se involucran para procesar una transacción.

En algunas partes se hace referencia a cuestiones del supervisor local que son tratadas con más detalle en el siguiente capítulo. En otras, se mencionan aspectos que ya fueron tratados en secciones anteriores, en cuyo caso sólo se hace referencia a la sección en cuestión.

En la figura 3.9 se muestran los módulos involucrados en el procesamiento de una transacción; también se marcan con números las partes en las cuales se efectúan los pasos descritos en el algoritmo.

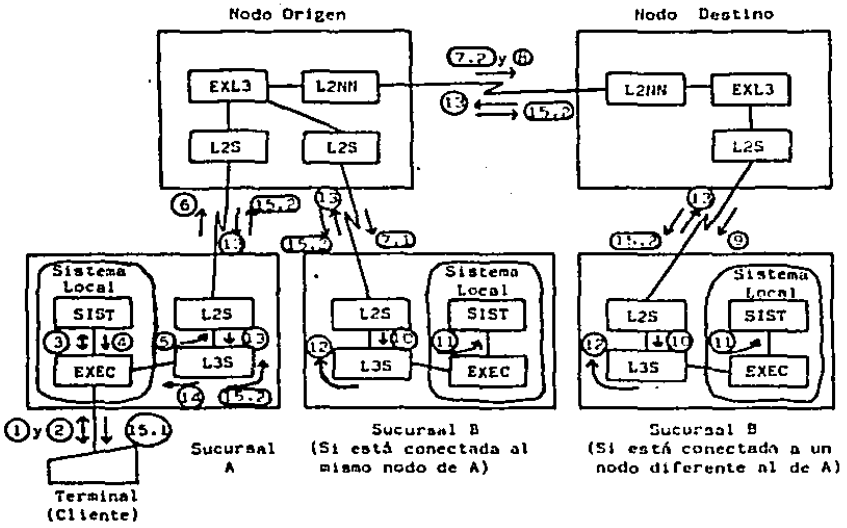


Fig. 3.9 Módulos involucrados en el procesamiento de una transacción.

El algoritmo a utilizar es el siguiente:

1. Un cliente llega a la sucursal A del sistema bancario a efectuar una transacción con una de sus cuentas (o la única, según el caso). El cliente solicita realizar la transacción y a través de una de las terminales se da inicio al procesamiento de la misma. El supervisor local atiende a todas las terminales de la sucursal empleando una política de round-robin, de tal manera que cuando le da atención a dicha terminal detecta que se va a iniciar una transacción haciendo todos los preparativos necesarios para procesarla.

2. Se solicitan al cliente los siguientes datos: transacción a efectuar, número de cuenta y cantidad a manejar en caso de que la transacción sea depósito o retiro.

3. Una vez que los datos son leídos, EXEC los pasa a SIST para que los procese. SIST compara los dos primeros números de la

cuenta, contra la información almacenada en el archivo IDENT.DAT. Si concuerdan el número de nodo y el de sucursal dentro del nodo, entonces es una cuenta que está registrada en esa sucursal. SIST efectúa todo lo descrito en el siguiente capítulo para procesar la transacción, genera los resultados correspondientes y los envía a EXEC para que éste a su vez los mande a la terminal en la cual se atiende al cliente, terminando con esto la transacción.

4. En caso de que no concuerde el número de nodo y/o el de sucursal, entonces es indicativo de que la cuenta está registrada en otra sucursal (B, por ejemplo). En base al tipo de cuenta y a la transacción, SIST forma el conjunto de los 8 primeros bytes, según lo descrito en la sub-sección anterior, y los envía a EXEC junto con el número de nodo y sucursal a donde se debe enviar dicha información.

5. EXEC pasa dicha información a L3S para que éste construya el datagrama de SOLICITUD correspondiente y le agregue los dos dígitos verificadores. Una vez construido, lo pasa a L2S.

6. L2S recibe el datagrama de L3S y realiza todo lo descrito en el algoritmo de la sección 3.1 para transmitirlo al nodo al que está conectada la sucursal. Con el datagrama, L2S forma el paquete HDLC y lo transmite a dicho nodo. Aquí cabe mencionar que aunque ésta y las acciones anteriores se han descrito en forma secuencial, en realidad se llevan a cabo cuando EXEC les da atención a los módulos involucrados y sólo hasta entonces se ejecutan. La política de atención dada por EXEC se describe en el siguiente capítulo.

7. El módulo L2S del nodo recibe el paquete HDLC enviado desde la sucursal y lo guarda en sus estructuras. Cuando EXL3 le da atención al módulo, recibe el datagrama correspondiente. En base a la información que contiene el campo de dirección destino del datagrama, EXL3 determina hacia dónde debe enviarlo.

   7.1 Si el número del nodo destino es su mismo número, entonces es un datagrama que se debe enviar a una sucursal que está conectada a este mismo nodo. EXL3 envía entonces el datagrama al módulo L2S que atiende a esa sucursal. Este módulo construye el paquete HDLC correspondiente y lo transmite a la sucursal B. Continúa en el punto 10.

   7.2 Si la sucursal B está conectada a otro nodo, EXL3 aplica su algoritmo para determinar cuál es la ruta que debe seguir el datagrama para alcanzar su destino, tomando como base que recorra la menor distancia posible. Una vez seleccionada la ruta envía el datagrama al módulo L2NN que transmite sobre la línea conectada al primer nodo de la ruta.

8. L2NN recibe el datagrama y lo transmite al nodo conectado en el otro extremo de la línea de transmisión. En ese nodo existe otro módulo L2NN que recibe el datagrama, lo envía al EXL3 de ese nodo y éste determina si es para una sucursal conectada a él o es para otro nodo. Si sucede lo último, se repite lo descrito en los puntos 7.2 y actual, hasta que el datagrama alcance el nodo destino.

9. Cuando el datagrama alcanza el nodo destino, el EXL3 que existe en esa micro selecciona el módulo L2S que atiende a la sucursal marcada en la dirección destino y se lo envía. L2S construye el paquete HDLC correspondiente y lo transmite a la sucursal B.

10. El módulo L2S de la sucursal B recibe el paquete HDLC, lo almacena en sus estructuras y después pasa el datagrama a L3S.

11. L3S recibe el datagrama, checa que esté correcto, almacena en sus estructuras los datos que correspondan, rescata los 8 bytes de información y los envía a EXEC con una indicación de que es una transacción solicitada desde otra sucursal.

12. EXEC pasa esta información a SIST; éste procesa la transacción y con los resultados que se produzcan, genera de nuevo 8 bytes de información los cuales envía a EXEC para que éste, a su vez, los pase a L3S a fin de que mande a la sucursal que hizo la solicitud, el datagrama de RESPUESTA que contiene el resultado de la operación.

13. Se repiten todas las acciones descritas en los puntos 5 a 10 para transmitir el datagrama que contiene la respuesta, desde la sucursal B hacia la A.

14. El módulo L3S de la sucursal A recibe el datagrama, rescata los bytes de información de la respuesta y los envía a EXEC. Este a su vez los transmite a SIST.

15. SIST compara la información recibida con los datos almacenados en sus estructuras, determina que es la respuesta a la transacción del cliente y efectúa lo siguiente:

15.1 Envía a la terminal en la cual se atiende al cliente, la información producto del procesamiento de la transacción, lo cual constituye el aviso de que ya fue realizada.

15.2 Envía una indicación a L3S para que éste genere un datagrama de CONFIRMACION para mandarlo a la sucursal B, a fin de indicarle que su respuesta fue bien recibida. Para que esta confirmación llegue a B, de nuevo deben repetirse las acciones descritas en los puntos 5 a 10. Cuando la confirmación llega a B, entonces B determina que su respuesta llegó bien a A y termina la comunicación con esa sucursal. Si en A no se recibe ningún datagrama adicional desde B, con respecto a la misma transacción, entonces A determina que la confirmación llegó bien a B y termina la comunicación con dicha sucursal.

Con ambas acciones termina el procesamiento de la transacción.

CAPITULO 4

SISTEMA LOCAL

4.1 GENERALIDADES

En este capitulo se describe en forma detallada la parte local del sistema que corresponde, principalmente, al supervisor local y a la parte que procesa y registra en si las transacciones.

La descripción del sistema local se hace enmarcando a éste dentro de las caracteristicas generales del sistema global, sobre todo en cuanto a modularización y capacidad de expansión se refiere. Sin embargo, dado que esta parte es la que fue implantada, en algunos casos se hace referencia concreta ya sea a caracteristicas físicas del equipo utilizado o a valores especificos empleados en la elaboración de los programas. De cualquier manera en todos los casos se cuidó el conservar los principios generales expuestos en la descripción del sistema global.

La implantación de este sistema local se hizo utilizando una microcomputadora Cromemco System 3, teniendo los siguientes elementos conectados: una unidad de diskettes de 8" con capacidad de 1.2 Mb, una unidad de disco fijo con capacidad de 20 Mb, 512 Kb de memoria central, 5 terminales y, como procesador central, el micro Z80 [CROM81]. La información manejada por el sistema de transacciones es almacenada en el diskette de 8" y accesada directamente a través de uno de los módulos del sistema local. El disco fijo se emplea para guardar los programas que conforman al sistema local y para efectuar respaldos de la información guardada en el diskette. En principio, el sistema está pensado para ser utilizado en una micro con dos unidades de diskette, una para la información de las transacciones y otra para programas y respaldos. En el apéndice B se muestra la configuración general de este equipo y se da una descripción detallada de sus componentes, sobre todo de la unidad de diskettes ya que ésta fue programada directamente, a nivel de ensamblador, para acelerar los tiempos de respuesta a las transacciones, y del IOP y QUADART que son los dispositivos mencionados en el capitulo 2 para controlar, a nivel de hardware, el intercambio de paquetes entre micros adyacentes.

El sistema local, tanto supervisor como transacciones, se desarrolló para ser ejecutado bajo Cromix -el sistema operativo de la micro, el cual es un subconjunto de UNIX-. Cuando se ejecuta el sistema local crea un ambiente de trabajo en el cual tiene el control tanto de las terminales como del diskette. Esto es, el sistema local toma el control de la mayor parte de la micro, dejando a Cromix solamente la gestión de la memoria asignada para la ejecución de los programas y el manejo del disco fijo. La atención de las terminales, el manejo del diskette, la atención a los programas, la supervisión de su ejecución y algunas otras funciones, son controladas totalmente por el sistema local. Las únicas operaciones que pueden efectuarse bajo este estado son las de las transacciones. Este tipo de sistema local es el que correría en cada micro de una sucursal de la red.

Las principales funciones que efectúa el sistema local son:

- controlar la ejecución del módulo que realiza el procesamiento de las transacciones bancarias en línea;
- controlar las operaciones de lectura/escritura que se llevan a cabo con el diskette;
- supervisar las comunicaciones realizadas con la terminales, a través de las cuales se da atención a las solicitudes para efectuar alguna transacción determinada;
- controlar las comunicaciones con micros de otras sucursales cuando se procesa una transacción global.

Todas las funciones anteriores se llevan a cabo en línea. Además de éstas, en el sistema local también se pueden efectuar las siguientes operaciones, entre otras:

- apertura de nuevas cuentas;
- cancelación de cuentas;
- generación de "estados de cuenta";
- respaldo y rescate de información.

Estas operaciones, sin embargo, se realizan fuera de línea ya que para las mismas no interesa una respuesta en tiempo real.

En las siguientes secciones se describe en detalle la organización general de este sistema local, así como la estructura de los archivos empleados y el algoritmo de los módulos que lo componen. En general, los módulos de este sistema están diseñados de acuerdo a las características descritas en la parte global del sistema.

4.2 ORGANIZACION GENERAL

El esquema general del sistema local es el que se muestra en la figura 4.1.

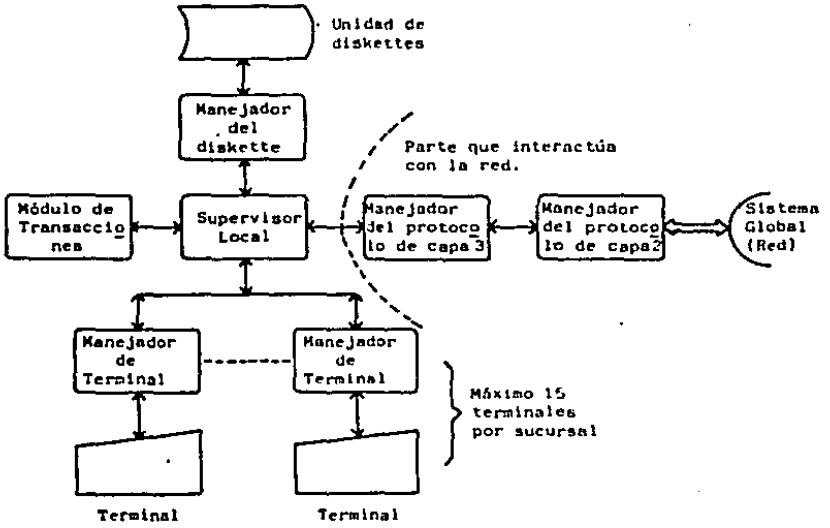


Fig. 4.1 Esquema general de la parte local del sistema.

En el esquema se muestran todos los módulos que integran al sistema local. Estos módulos son los que están ejecutándose en la micro que atiende a una sucursal. En todas las micros de sucursales se repiten estos módulos ya que en cada una se llevan a cabo las mismas funciones.

A continuación se describen a grandes rasgos las funciones principales que llevan a cabo cada uno de estos módulos y la forma en que se inter-relacionan para que el sistema local pueda funcionar.

a) Supervisor local.- Es el encargado de controlar la ejecución de los demás módulos en la micro local. Tiene como tareas principales las siguientes: iniciar la ejecución del resto de los módulos cuando comienza una sesión de trabajo; sincronizar la ejecución de cada módulo, o sea controlar cuando se le debe dar

atención para que se ejecute; controlar el intercambio de información entre los diferentes módulos; llevar a cabo las acciones necesarias para que el sistema sólo trabaje localmente cuando la sucursal queda desligada de la red por alguna falla; dar de baja ordenadamente al sistema cuando se termina una sesión de trabajo.

Sobre este módulo recae la responsabilidad de sincronizar y controlar la operación total de los demás módulos para que el sistema local pueda funcionar. Es a una micro de sucursal, lo que EXL3 es a una micro de nodo. Dentro del sistema este módulo recibe el nombre de EXEC.

b) Módulo de transacciones.- Se encarga de controlar el procesamiento de una transacción. Cada transacción ordenada desde una terminal es procesada a través de varias etapas las cuales controla este módulo. Cuando la transacción es local, efectúa todo lo necesario para que con la información que tiene el diskette local se satisfaga la misma y a la vez quede registrada. Cuando es global, genera la información necesaria para que se construya el datagrama a enviar a la sucursal que contiene la cuenta a manejar. Cuando se recibe una solicitud de otra sucursal, el módulo usa información del diskette local para construir el datagrama que satisface dicha solicitud. Este módulo recibe el nombre de SIST.

c) Manejador del diskette.- Lleva a cabo el manejo de la unidad de diskette local. Cada que se necesita utilizar el diskette para satisfacer una transacción, este módulo es activado. Se encarga de: iniciar su funcionamiento cuando se va a efectuar una operación de lectura/escritura con el diskette; controlar la realización de dicha operación; en caso de escritura, recibir la información correspondiente de EXEC y enviarla al diskette; en caso de lectura, efectuar la operación inversa; desactivar la unidad cuando la operación ha concluido.

El modulo recibe el nombre de DRVDSK dentro del sistema.

d) Manejador de terminal.- Realiza funciones similares al módulo anterior pero con una terminal. Cuando se va a escribir información en la terminal, controla la recepción de la misma desde EXEC y luego la escribe en la terminal. Cuando el operador de la terminal teclea información, controla la recepcion de cada carácter enviandolos a EXEC hasta que se oprime el fin de linea.

Existe un módulo de este tipo por cada terminal que esté en linea en la sucursal. Básicamente son los mismos y sólo tienen pequeñas diferencias en cuanto a la identificación de cada terminal. Se reconocen con los nombres de DRVTT1, DRVTT2,...,DRVTTn, donde n tiene un valor máximo de 15.

e) Manejador del protocolo de la capa 3.- Es el módulo que se encarga del manejo de los datagramas cuando estos se usan para procesar una transacción global. Sus funciones ya fueron descritas en la sección 3.2; es el módulo L3S.

f) Manejador del protocolo de la capa 2.- Es el que se encarga de la transmisión correcta de paquetes HDLC entre dos puntos adyacentes de la red. Al igual que el módulo anterior, sus funciones ya fueron descritas en la sección 3.1;es el módulo L2S.

Estas son a grandes rasgos las funciones que realizan los diferentes módulos que componen el sistema local que corre en una micro de sucursal. Las estructuras que emplean asi como su algoritmo se detallan en las siguientes secciones.

4.3 ARCHIVOS DEL SISTEMA LOCAL

En esta sección se describen las caracteristicas de los archivos usados en el sistema local. Como se mencionó en la sección 4.1, estos archivos se almacenan en un diskette de 1.2 Mb que es el que se utilizó con la micro en la cual se implantó este sistema local. Cada archivo ocupa una cantidad fija y especifica de pistas del diskette. Por razones del tamaño del diskette, cada archivo tiene un tamaño máximo fijo el cual permite manejar la información de una cierta cantidad de clientes. Esta cantidad puede ser aumentada cambiando ciertos parámetros que se tienen en los programas que usan estos archivos. En principio se considera que con la cantidad de clientes que actualmente se puede manejar es suficiente para efectos de implantación y ejecución experimental de este sistema.

En la figura 4.2 se muestra un esquema general de la distribución de estos archivos en el diskette, asi como una indicación de los sectores (o cilindros) ocupados por los mismos.

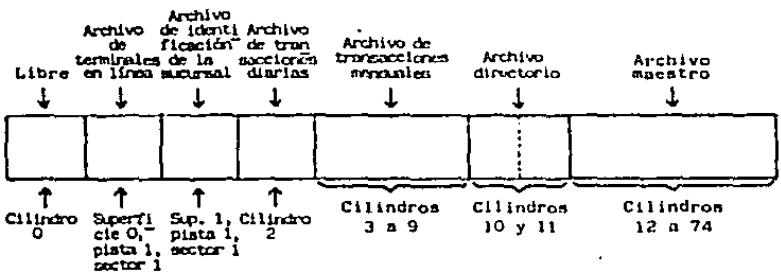


Fig. 4.2 Distribución de los archivos del sistema local en el diskette.

El diskette tiene dos superficies numeradas 0 y 1. Cada superficie tiene 77 pistas numeradas del 0 al 76; en total son 154 pistas. La conjunción de dos pistas con el mismo número pero en superficie diferente forma un cilindro. En total se tienen 77 cilindros. Cada pista contiene 16 sectores numerados del 1 al 16. Cada sector es de 512 bytes. En la figura 4.2 se indica que parte del diskette ocupa cada archivo. Por ejemplo, el archivo de transacciones diarias ocupa todo el cilindro 2.

Para accesar un archivo es necesario especificar la dirección física en la cual se localiza en el diskette. El acceso a los archivos lo hace el programa DRVDSK directamente a nivel de sectores, pistas y superficies, por medio de esta dirección física. En las siguientes sub-secciones se describe en detalle el contenido de estos archivos.

4.3.1 Archivo de Terminales en Línea

Es un archivo que se utiliza para almacenar la cantidad actual de terminales que se tienen registradas en el sistema local; o sea la cantidad actual de terminales que hay en la sucursal. También aquí se registra la cantidad y el número de las terminales que están desactivas (posiblemente por falla) en la sucursal.

Este archivo ocupa un bloque de 512 bytes del diskette y tiene un solo registro cuya estructura es la siguiente:

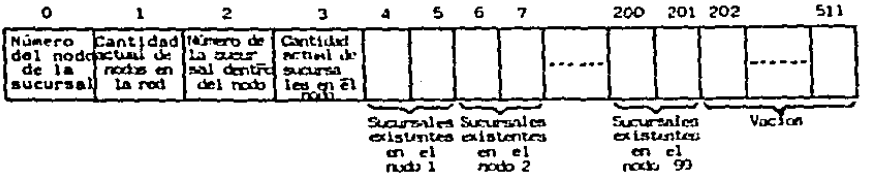
Byte: 0	1	2	3	------	k+1	511
Cantidad actual de terminales en la sucursal	Cantidad actual de terminales desactivas	Número de la 1a. terminal desactiva	Número de la 2a. terminal desactiva	------	Número de la k-ésima terminal desactiva	Resto de --los bytes-- vacios

El byte 0 almacena el total de terminales en la sucursal ya sea que estén activas o no. Cada terminal tiene asignado un número entero entre 1 y n (n<=15). Cuando una terminal queda fuera de línea (p. ej., por descompostura) se registra en este archivo como terminal desactiva. Cuando se inicia una sesión de trabajo, EXEC ve el contenido de este archivo para determinar a cuáles terminales les debe dar atención y a cuáles no. Cuando una terminal funciona de nuevo, simplemente se quita del archivo para que, al inicio de otra sesión, EXEC vuelva a considerarla.

El archivo ocupa la superficie 0, pista 1 y sector 1 del diskette.

4.3.2 Archivo de Identificación de la Sucursal

Contiene información que identifica en forma única a la sucursal dentro de toda la red e información que indica cuántas, y cuáles, sucursales están asociadas a cada nodo en la red. Ocupa un bloque del diskette y su estructura es la siguiente:



Los valores almacenados en los bytes 0 y 2 identifican en forma única a la sucursal dentro de todo el sistema. El número del nodo estará entre 1 y el valor almacenado en el byte 1; máximo habrá 99 nodos. El número de sucursal estará entre 1 y el valor guardado en el byte 3; máximo habrá 10 sucursales por nodo.

A partir del byte 4 se ocupan 2 bytes por nodo, para indicar cuántas y cuáles sucursales están asociadas a los mismos. La información guardada en los bytes 4 a 201 se carga en la tabla TOTSUC (descrita en el algoritmo de SIST) cuando el sistema local comienza a ejecutarse (es la misma tabla que maneja EXL3). Su finalidad es detectar si los números de nodo y sucursal dados para un número de cuenta, existen realmente en el sistema o no y así poder detectar error o no para dicho número.

El archivo ocupa la superficie 1, pista 1 y sector 1 del diskette.

4.3.3 Archivo de Transacciones Diarias

Contiene el registro diario de las transacciones de depósito o retiro efectuadas durante una sesión de trabajo con las cuentas registradas en la sucursal. Se utiliza un registro de 8 bytes por cada transacción y su esquema es el siguiente:

La información almacenada en cada campo es la siguiente:

Campo	Información almacenada
1	Monto de la transacción. Una cantidad positiva indica un depósito; una negativa, un retiro.
2	Número y tipo de cuenta. Los dos bits altos del 5o. byte se usan para indicar el tipo de cuenta de acuerdo con lo siguiente:

Bits	Tipo de cuenta
00	Cheques
01	Ahorro
10	Tarjeta de crédito
11	Valores retirables en días pre-establecidos

Los 14 bits restantes se emplean para el número de cuenta.

3	Número de la sucursal que hizo la transacción. El byte 7 se emplea para guardar el número de nodo y el 8 para el número de la sucursal dentro del nodo. Cuando la transacción es local, esta información se toma del archivo anterior; cuando es global, se toma de la dirección de la micro origen que viene en el datagrama que contiene la solicitud.

Cuando una sesión de trabajo termina, toda la información de este archivo se pasa al archivo de transacciones mensuales; después, los registros del mismo se dejan libres, listos para la siguiente sesión.

El archivo ocupa el cilindro 2 del diskette, o sea 32 sectores del mismo.

4.3.4 Archivo de Transacciones Mensuales

Contiene el registro de las transacciones de depósito o retiro efectuadas durante el mes actual con las cuentas registradas en la sucursal. Para cada transacción se emplea un registro de 8 bytes cuyo formato es semejante al de los registros del archivo anterior. La información almacenada en cada registro es del mismo tipo también.

Cada que termina una sesión de trabajo se pasa a este archivo toda la información que se guardó en el archivo anterior durante la sesión. Así en el presente archivo se van acumulando todas las transacciones de depósito o retiro efectuadas durante el mes que transcurre. Al final del mes con la información guardada en este archivo se generan los "estados de cuenta" de las cuentas registradas en la sucursal. Para separar la información de una sesión de otra, se emplea un registro especial con las siguientes características:

- en el campo de número y tipo de cuenta, se pone 11 en los dos bits altos del byte 5 y 0 en los 14 bits restantes. Con esto se indica que el registro separa información de dos sesiones de trabajo;
- en el campo de monto se guarda la fecha de la sesión. Todos los registros a continuación de éste contendrán transacciones efectuadas en esa fecha;
- si el campo de número y tipo de cuenta está en ceros, indicará que se trata de un registro libre.

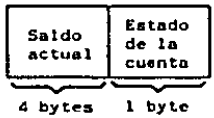
Al final del mes, después de generar los estados de cuenta, los registros se dejan libres para el siguiente mes.

El archivo ocupa los cilindros 3 a 9 del diskette. Cada sector almacena 64 registros de 8 bytes cada uno.

4.3.5 Archivo Directorio

Contiene para cada cuenta, el saldo actual, al inicio de una sesión de trabajo, y el "estado" de la misma. En principio este archivo almacena únicamente información de cuentas de cheques. Para manejar los demás tipos de cuentas habría que utilizar archivos semejantes por cada tipo. Como se mencionó en el capítulo 1, en este trabajo sólo se implantó lo referente a las cuentas de cheques.

Por cada cliente se usa un registro como el siguiente:



La información almacenada en cada registro es:

Campo	Información almacenada
1	Saldo actual de la cuenta al inicio de una sesión de trabajo.
2	Estado de la cuenta. Cada bit de este campo tiene la siguiente función:

Bit	Función
0	0-libre, 1-en uso(o sea, asignada a un cliente)
1	0-no suspendida, 1-suspendida
2	0-no cancelada, 1-cancelada(durante un mes, estos dos últimos bits pueden cambiar de 0(->)1 en cualquier momento)
3	0-activa, 1-dada de baja(si está dada de baja, al final del mes se desocupa el registro, después de generar el estado de cuenta)

4 0-tiene el mínimo, 1-abajo del mínimo(al final del mes se actualiza este bit, según se tenga el saldo mínimo mensual -en pesos- o no)
5 a 7 Sin uso

Cuando un registro está libre, contiene ceros. Los registros del archivo se consideran numerados del 1 al 6,000. El registro 1 estará asignado al cliente con cuenta 1 y así sucesivamente. Por el tamaño del archivo, en principio sólo se pueden manejar 6,000 clientes por sucursal. El archivo ocupa los cilindros 10 y 11 del diskette. Cada sector almacena 100 registros, ocupando 500 bytes del mismo.

4.3.6 Archivo Maestro

Contiene la información general de un cliente. Por cada cliente se emplea un registro con la siguiente estructura:

Nombre del cliente	Dirección	Ciudad y Estado	Teléfono	Benefi-ciario	Saldo al mes anterior	Saldo promedio al día
38 bytes	50 bytes	20 bytes	8 bytes	38 bytes	4 bytes	4 bytes

La información almacenada en cada campo es la siguiente:

Campo	Información almacenada
1	Nombre del cliente
2	Dirección, que incluye: calle, número, colonia y código postal.
3	Ciudad y Estado.
4	Teléfono.
5	Beneficiario.
6	Saldo al mes anterior.
7	Saldo promedio al día. Cada mes se inicia con 0; después se le suma cada día: saldo actual/30.

Al igual que con el archivo directorio, este archivo sólo contempla el manejo de cuenta de cheques. Para manejar los demás tipos de cuenta habría que agregar campos que contuvieran los números de cada cuenta, así como el saldo al mes anterior, todo para un mismo cliente. No se contempla más esta posibilidad en el presente trabajo.

Cuando un registro está libre, todo se encuentra a ceros. El archivo tiene capacidad para almacenar información de 6,000 clientes. Los registros se consideran numerados del 1 al 6,000. El registro 1 se asigna al cliente 1, etc. Existe una correspondencia de uno a uno entre los registros del directorio y los de este archivo.

El archivo ocupa los cilindros 12 a 74 del diskette. Cada sector almacena tres registros del archivo.

4.4 SUPERVISOR LOCAL: EXEC

El supervisor local es la parte central del sistema local, ya que es el encargado de supervisar y controlar la ejecución del resto de los módulos. En esta sección primeramente se describe la filosofía que emplea para llevar a cabo esta tarea y después se detallan las estructuras y algoritmo que usa en su ejecución.

Comunicación con y entre módulos.- Una de las funciones de EXEC es la de sincronizar la ejecución de los módulos que se ejecutan en la micro de una sucursal. Para lograr esto, cuando ésta se ordena, EXEC establece con cada uno de los módulos, canales de comunicación a través de los cuales lleva a cabo la sincronización de su ejecución. Estos canales no son otra cosa mas que los "pipes" manejados en UNIX [CROM81]. Cuando EXEC manda a ejecución a un módulo establece con él dos pipes de comunicación, según se muestra en la figura 4.3.

Fig. 4.3 Canales de comunicación entre módulos.

Uno de los pipes se usa para que EXEC reciba información desde el módulo -entrada-; el otro se emplea para que EXEC se la mande -salida-. Con cada uno de los módulos que EXEC manda a ejecutar establece dos pipes como los mostrados en la figura 4.3. Sólo hay un caso en el cual EXEC no establece relación directa con un módulo y es en el de la relación que existe entre L3S y L2S. En este caso L3S es el que ordena la ejecución de L2S y es el que establece los pipes entre los dos. Cuando le da atención a un módulo para que éste se ejecute, EXEC le envía un conjunto de bytes -a través del pipe de salida- para que inicie su ejecución. En ese momento EXEC se suspende y espera hasta que el módulo le responda con otra serie de bytes -a través del pipe de entrada-, la cual le indica que éste acaba de realizar su función y EXEC puede volver a tomar el control del procesador. Cuando le da atención a otro módulo repite este proceso. Al intercambiar esta serie de bytes también se envía información que utilizan los módulos para efectuar algún proceso. De esta forma se establece la comunicación entre EXEC y los demás módulos y además se sincroniza la ejecución de los mismos.

Politica de atención.- EXEC utiliza dos colas para dar atención a los módulos. Una, denominada ALTAP, es de prioridad alta y la otra, llamada NORMAL, es de prioridad normal. Cada módulo tiene un índice por medio del cual es manejado. Estos índices son colocados en una u otra cola dependiendo del grado de atención que se le de a cada módulo. Por cada dos módulos atendidos en ALTAP se atiende uno en NORMAL.

EXEC emplea una política de round-robin para dar atención a los módulos que se encuentran en estas colas; esto es, cede el control del procesador a éstos conforme los va encontrando secuencialmente en una u otra cola. En NORMAL siempre se encuentran los manejadores de terminales (DRVTTi). Cuando en una terminal se inicia una transacción, su módulo se pasa a ALTAP para que reciba más atención. Cuando ésta termina, el módulo se regresa a NORMAL para que reciba menos atención. L3S siempre se encuentra en ALTAP.

SIST y DRVDSK no se colocan en ninguna cola debido a que cuando se necesitan son ejecutados de inmediato. Esto es, cuando se debe procesar parte de una transacción, en ese momento se ejecuta SIST; lo mismo, cuando es necesario accesar el diskette, DRVDSK se ejecuta de inmediato. En esta situación también se encuentra L2S, aunque quién le cede el control es L3S en lugar de EXEC.

La figura 4.4 esquematiza esta política.

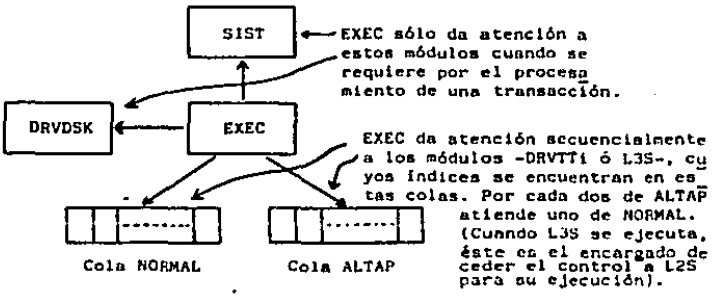


Fig. 4.4 Política de atención de EXEC.

Política de ejecución.- La idea central de este supervisor es que al iniciar la ejecución de un módulo, le deje a éste el control total del procesador. El módulo lo usará hasta que termine completamente su función o hasta que sea necesario ejecutar otro para que pueda continuar. En ambos casos el módulo mismo es el encargado de devolver el control del procesador al supervisor para que éste pueda dárselo a otro módulo diferente. Es decir, no hay

requisa (pre-emption) [HANS73] por parte de EXEC hacia estos módulos.

Lo anterior implica que cada módulo no tiene asignado un "pedazo de tiempo" para su ejecución, con lo cual, por un lado, se evita guardar registros, status, etc., al hacer el cambio de contexto al pasar a ejecutar otro módulo [HANS77]; pero por el otro, se pueden ocasionar problemas serios de eficiencia, en caso de que existiera una tarea que empleara mucho tiempo para ejecutarse, ya que no permitiría que otras tareas usaran el procesador por no existir un tiempo límite en su utilización. No es el caso de este sistema, ya que las funciones efectuadas por los distintos programas que lo componen son, en general, poco complejas, y por ende se ejecutan "rápidamente".

Cuando se está manejando una transacción siempre se trata de procesarla totalmente desde el principio hasta el final. Habrá casos en los que no sea posible hacer esto y entonces será necesario pasar a ejecutar otra solicitud, fundamentalmente para evitar retrasos en el tiempo de respuesta a una transacción. En general estos casos se presentan cuando es necesario teclear alguna información en terminal para continuar procesando una transacción local, o cuando se espera la respuesta desde otra sucursal en el caso de una transacción global. Tanto el supervisor como el módulo de transaccciones tienen los elementos necesarios para manejar estas situaciones, según se observará más adelante.

Los tres párrafos anteriores expresan la filosofía central de este supervisor. EXL3 observa esta misma filosofía en la supervisión que aplica a los módulos que se ejecutan en un nodo. Bajo este esquema, a continuación se describen las principales estructuras de información y el algoritmo empleados por EXEC.

ESTRUCTURAS DE INFORMACION

a) Tabla TAREA.- Es un arreglo de registros que se emplea para guardar información relacionada con cada módulo controlado por EXEC. Tiene el siguiente formato:

Nombre del módulo	Estado del módulo

Tabla 4.1 Estructura de la tabla TAREA.

El primer campo sirve para almacenar el nombre del módulo. El segundo es un byte que contiene información sobre el estado actual que guarda cada módulo. La función de cada bit es la siguiente:

Bit	Función
0	Tarea activa. Está en 1 si la tarea es susceptible de ser ejecutada. Está en 0, si durante la sesión actual no se va a ejecutar(sería el caso de un módulo DRVTTi que no se ejecuta porque la terminal que atiende queda fuera de servicio).
1	Tarea suspendida. El bit está en 1 cuando la tarea se suspende temporalmente. Se pone en 0 cuando puede ser ejecutada de nuevo.
2	Tarea en alta prioridad. Se pone en 1 cuando la tarea se inserta en la cola de alta prioridad. Está en 0 si se encuentra en la cola de prioridad normal.
3	Lectura de terminal. Este bit sólo se usa con módulos DRVTTi. Cuando está en 1 indica que el módulo espera leer de terminal. Cuando está en 0 es que va a escribir en ella.
4 a 7	Sin uso.

Cada módulo tiene asignado un número que sirve para identificarlo dentro del sistema. Este número actúa como índice para accesar su elemento correspondiente en este arreglo. Los índices asignados son:

Indice	Módulo
1	SIST
2	DRVDSK
3	L3S
4	L2S
5	DRVTT1
6	DRVTT2
.	.
.	.
n+4	DRVTTn (n<=15)

b) Cola NORMAL - Contiene los índices de las tareas listas a ejecutarse con prioridad normal. Se accesa secuencialmente de tal manera que las tareas se ejecutan una después de otra conforme se van encontrando. Cuando se inicia una sesión los índices de todos los módulos DRVTTi se colocan en esta cola. Su tamaño es de 16 bytes.

c) Cola ALTAP - Almacena los índices de las tareas que se ejecutan con prioridad alta. También se accesa secuencialmente. Por cada dos módulos ejecutados con prioridad alta, se ejecuta uno con prioridad normal.

d) Cola TSUSP - Aquí se insertan los índices de las tareas que se suspenden temporalmente. Las tareas susceptibles de ser

suspendidas son las que atienden a las terminales. Las posibles causas de suspensión son:

- tratar de accesar una cuenta bloqueada cuando se realiza una transacción local;
- esperar por una respuesta desde otra sucursal al efectuar una transacción global; y
- quedar fuera de servicio una terminal durante el transcurso de una sesión.

Cuando la causa que provoca la suspensión termina, el índice del módulo se quita de este arreglo. Tiene un tamaño de 16 bytes.

e) Arreglo BUF.- Es un arreglo de una dimensión de 517 bytes. Se utiliza para almacenar temporalmente la información que se transmite entre los diferentes módulos controlados por EXEC.

f) Arreglos ENT y SAL.- Son arreglos de una dimensión de 20 bytes cada uno. Almacenan los números asignados a los pipes que sirven para establecer comunicación entre EXEC y los demás módulos.

ALGORITMO

Rutina INICIO

(Al ejecutar esta rutina, EXEC crea un ambiente en el cual tiene el control total de los módulos que se ejecutan bajo el sistema local).

1. Guardar en TAREA los nombres de los módulos que van a ser directamente controlados por EXEC.
2. Iniciar la ejecución de DRVDSK para poder accesar la información almacenada en el diskette, estableciendo además los pipes de comunicación con él.
3. Leer la información del archivo de terminales en línea para determinar cuáles están activas y cuáles no.
4. Eliminar a SHELL del sistema a fin de que Cromix no reconozca a las terminales y EXEC pueda controlarlas.
5. Iniciar la ejecución de DRVTT1 a DRVTTn (n<=15) para controlar las terminales. Si la terminal correspondiente a un módulo está desactiva, éste no se ejecuta. Establece también los pipes con cada módulo.
6. Colocar los índices de DRVTT1 a DRVTTn en la cola NORMAL.
7. Iniciar la ejecución de SIST estableciendo los pipes con él. Lee el archivo directorio y transmite su contenido a SIST.
8. Iniciar la ejecución de L3S estableciendo los pipes con él. L3S inicia a su vez la ejecución de L2S y establece pipes entre los dos. Se coloca el índice de L3S en ALTAP.

Rutina FIN

(Al ejecutarse esta rutina, EXEC da de baja al sistema local y deja a la micro en el estado anterior en que la tenía Cromix).

1. Termina de ejecutar SIST. Al terminar, SIST envia información a EXEC para que actualice el archivo directorio.
2. Termina de ejecutar los módulos activos DRVTT1 a DRVTTn. Libera las terminales y permite que Cromix las reconozca de nuevo. Establece las caracteristicas originales de dichas terminales.
3. Termina de ejecutar DRVDSK y libera la unidad de diskettes.
4. Termina de ejecutar L3S y éste a su vez a L2S.
5. Termina su propia ejecución y libera a la terminal asociada.
6. Cede el control a Cromix.

Rutina PRINCIPAL

Diagrama de la figura 4.5.

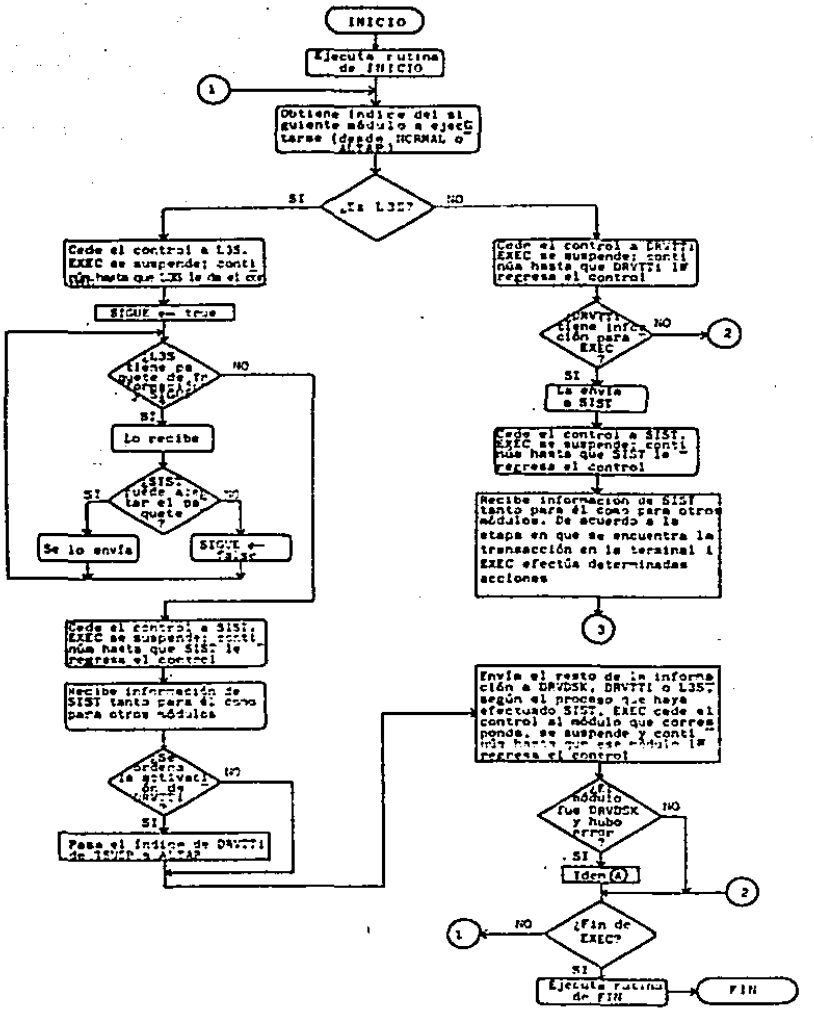


Fig. 4.5 Algoritmo del módulo EXEC.

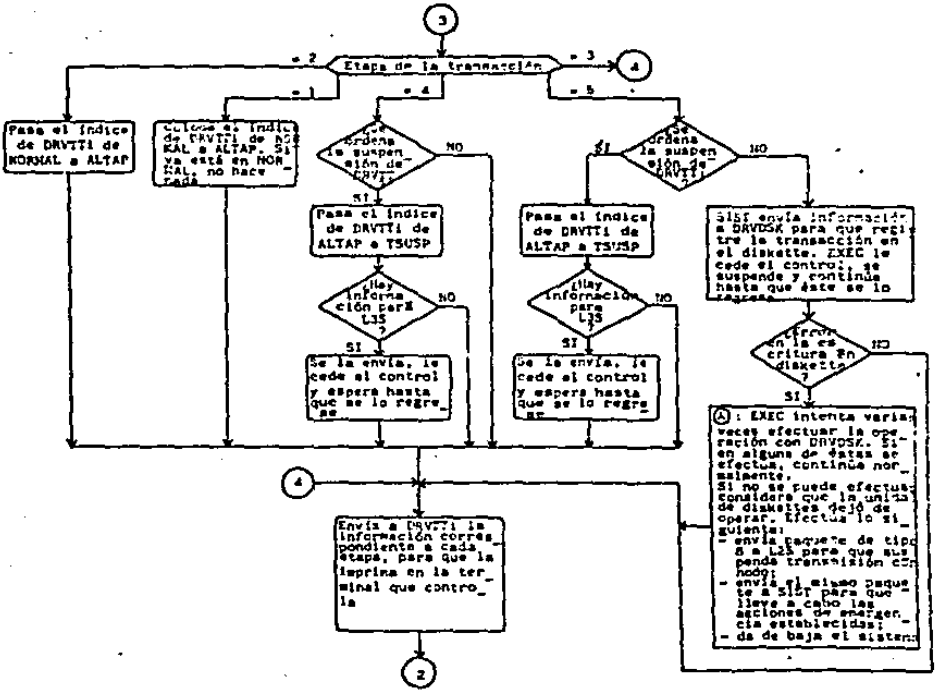


Fig. 4.5 Algoritmo del módulo EXEC (Cont.).

4.5 MODULO DE TRANSACCIONES: SIST

Este módulo es el encargado de procesar todas las transacciones que se efectúen con el sistema, tanto locales como globales. A continuación se describen sus principales estructuras, así como el algoritmo que lo compone, recalcando que sólo se trata lo relativo al manejo de cuenta de cheques.

ESTRUCTURAS DE INFORMACION

a) Tabla CLI.- Es un arreglo de 6,000 elementos el cual se utiliza para almacenar en memoria central el contenido del archivo directorio. Su estructura es semejante a la de los registros de este archivo y se muestra en la tabla 4.2.

Saldo actual	Estado de la cuenta

Tabla 4.2 Estructura de la tabla CLI.

Cada elemento de la tabla corresponde a una cuenta. El segundo campo de cada elemento es un byte cuyos bits tienen una función idéntica a los del byte de status del archivo directorio (sección 4.3.5). Sólo se usa un bit más, el número 5, para indicar si la cuenta está siendo ocupada en un momento dado o no. El bit estará en 0 si en ese momento la cuenta está desocupada; estará en 1, en caso contrario.

Al principio de una sesión en la tabla se almacenan el saldo y estado actuales de cada cuenta de cheques registrada en la sucursal. Durante el transcurso de una sesión, este saldo así como el estado de las cuentas pueden modificarse, dependiendo de las transacciones efectuadas sobre ellas. Cuando termina la sesión, el contenido de esta tabla se transfiere tal cual al archivo directorio para reflejar el último estado de las cuentas.

Todas las cuentas se tienen en memoria central con el fin de que el tiempo de respuesta al procesamiento de una transacción sea el más corto posible. Con las cuentas en memoria se minimiza el número de accesos a disco. Por ejemplo, cuando se desea consultar un saldo, basta con accesar el elemento de CLI que corresponda a la cuenta para obtenerlo, sin necesidad de accesar el disco. Cuando se trata de un depósito o retiro, CLI se usa en conjunción con el arreglo TRANSD para que con un solo acceso al diskette quede registrada la operación.

Cada elemento del arreglo ocupa 5 bytes, 4 para el saldo y uno para el status de la cuenta. El tamaño total del arreglo es de 30,000 bytes.

b) Tabla TRANSD.- Es un arreglo de 64 elementos el cual se emplea para ir almacenando temporalmente cada transacción de depósito o retiro efectuada durante una sesión de trabajo. Su estructura es semejante a la del archivo de transacciones diarias y se muestra en la tabla 4.3.

Monto de la transacción	Número y tipo de cuenta	Número de la sucursal que hizo la transacción

Tabla 4.3 Estructura de la tabla TRANSD.

Los campos de un elemento del arreglo tienen la misma función que los campos de un registro del archivo de transacciones diarias (ver sección 4.3.3). Cada elemento del arreglo corresponde a una transacción.

Cada que se realiza un depósito o retiro con una de las cuentas, la transacción se guarda secuencialmente (en el primer elemento libre) en el arreglo. Después de guardarla, el arreglo completo se escribe en el archivo de transacciones diarias, sin importar que esté lleno o no. Cuando los 64 elementos se ocupan, el arreglo se pone a ceros y las nuevas transacciones que se efectúen se guardan en el siguiente sector disponible de ese archivo. Esto se hace para que con un solo acceso al diskette quede registrada la transacción.

Si por algún motivo el sistema se suspende, el estado que tenían las cuentas antes de la suspensión puede ser reconstruido; para ello bastaría con tomar del archivo directorio el saldo de las cuentas al inicio de la sesión y con la información guardada en el de transacciones diarias, se repetirían todos los depósitos y retiros efectuados hasta antes de la falla, quedando las cuentas en CLI con el saldo que tenían en ese momento.

Cada elemento del arreglo ocupa 8 bytes, 4 para el monto, 2 para la cuenta y 2 para la sucursal que efectuó la transacción. El arreglo ocupa 512 bytes en total.

c) Tabla LSUSP.- Es un arreglo de 64 elementos donde cada uno ocupa 11 bytes. Esta tabla se emplea para guardar la información perteneciente a una transacción que se suspende, debido a que la cuenta que va a usar está siendo ocupada por otra transacción en ese momento. La transacción que se inserta en esta cola puede ser

local o global. Cuando es local la información que se guarda es la mostrada en la figura 4.6.

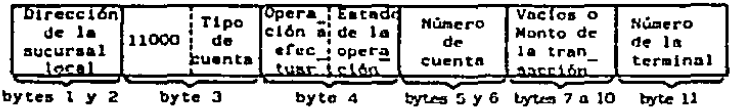


Fig. 4.6 Formato de la información guardada en LSUSP cuando la transacción es local.

La información guardada en los bytes 3 a 10 es la misma que la indicada en la sección 3.4.2. En el byte 11 se guarda el número de la terminal en la cual se está atendiendo la transacción. Cuando la transacción suspendida es global, el formato de la información es el de la figura 4.7.

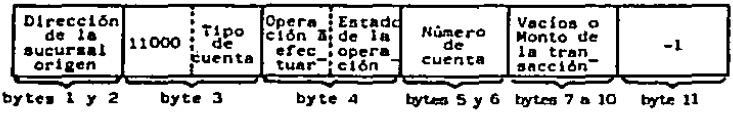


Fig. 4.7 Formato de la información guardada en LSUSP cuando la transacción es global.

La información de los bytes 3 a 10 es la indicada en la sección 3.4.2.

Una transacción permanece en esta cola hasta que se libera la cuenta que va a usar. En el momento que sale de esta cola, la transacción es procesada.

d) Tabla RGPORC - Es un arreglo de 64 elementos, cada uno de 10 bytes. Sirve para guardar la información enviada como respuesta a una sucursal (B) que envió un datagrama de solicitud a esta sucursal (A). La información que se guarda es la mostrada en la figura 4.8.

Fig. 4.8 Formato de la información guardada en RGPORC.

La información de una respuesta se elimina cuando:

- se recibe la confirmación, desde la sucursal B, de que llegó bien el datagrama de respuesta enviado por A; o
- se recibe un datagrama de aviso indicando que B quedó fuera de servicio; o
- esta sucursal (A) queda trabajando sólo en modo local.

e) Tabla GSUSP - Es un arreglo de 20 elementos donde cada uno ocupa 11 bytes. Se emplea para almacenar los bytes de información enviados en un datagrama de solicitud, cuando la cuenta que se va a manejar se localiza en otra sucursal de la red. El formato de este conjunto de 11 bytes es el de la figura 4.9.

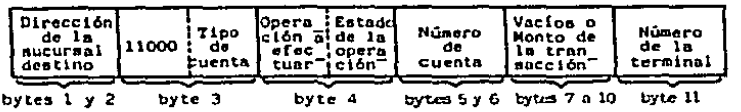


Fig. 4.9 Formato de la información guardada en GSUSP.

El último byte contiene el número de la terminal donde se atiende la transacción.

Cuando desde la sucursal en cuestión (B) se recibe el datagrama de respuesta, la transacción se saca de esta cola, la respuesta se envía a la terminal que atiende la transacción y a L3S se le pasan los datos necesarios para que envíe el datagrama de CONFIRMACION.

f) Cola TCONF - Es un arreglo de 32 elementos de 5 bytes cada uno. Se usa para almacenar temporalmente información a ser enviada a otras sucursales por medio de datagramas de CONFIRMACION. En la figura 4.10 se muestra el formato de esta información.

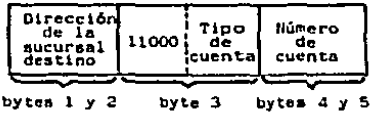


Fig. 4.10 Formato de la información guardada en TCONF.

Los elementos se liberan en cuanto la información se pasa a L3S.

g) Tabla SUCSF.- Se emplea para que SIST registre aquí las direcciones de las sucursales que quedan fuera de servicio durante una sesión. Esto se hace con la finalidad de que no se envíen datagramas adicionales a dichas sucursales. Su formato es:

Nodo de la sucursal	Número de la sucursal dentro del nodo

Tabla 4.4 Estructura de la tabla SUCSF.

La dirección de una sucursal se quita de aquí en cuanto se recibe el datagrama de aviso indicando que ésta entró de nuevo en servicio.

h) Tabla TOTSUC - Es la misma tabla que aparece en EXL3 (inciso h, sección 3.3). Se emplea para evitar enviar datagramas a sucursales que no existen en el sistema.

i) Arreglos ETAPA, OPER y CUENTA.- Son arreglos de una dimensión de 15 elementos cada uno. Se utilizan para guardar información relacionada con el procesamiento de una transacción. Cada elemento está asociado con una terminal de la sucursal.

Como se observará en el algoritmo, cada transacción se procesa por etapas desde el inicio hasta el final de la misma. Como se mencionó en EXEC, si una transacción se procesara totalmente sin dar atención a otra, el sistema podría quedar "amarrado" esperando alguna respuesta del operador en la terminal, o una respuesta desde otra sucursal, lo cual ocasionaría que se incrementara el tiempo de respuesta a otras transacciones. Para evitar esto es necesario que las transacciones se vayan desarrollando en paralelo, para lo cual se debe registrar en qué etapa se queda una determinada transacción cuando se pasa a atender a otra diferente. El arreglo ETAPA se emplea para tal fin. Como cada elemento está asociado a una terminal, en cada uno se va guardando un número entero que indica en qué etapa se encuentra la transacción que se procesa en cada terminal. Cuando se retoma una transacción suspendida, se sabe en cuál etapa continuar tan sólo con inspeccionar dicho arreglo. En el esquema de la figura 4.11 se muestra el aspecto que podría tener en un momento dado.

1	2	3	4	----------	15
1	5	2	4		

Fig. 4.11 Posible aspecto del arreglo ETAPA durante una sesión.

En el ejemplo, la transacción de la terminal 1 se encuentra en su etapa 1, la de la terminal 2 en su etapa 5, y así sucesivamente.

Los arreglos OPER y CUENTA almacenan el tipo de transacción que se está efectuando y la cuenta que se está manejando, respectivamente.

Los elementos que tienen el mismo índice en los tres arreglos están asociados con la misma terminal. Cada elemento de ETAPA y OPER consta de 2 bytes. Cada elemento de CUENTA es de 4 bytes.

En este caso sólo se considera cuenta de cheques. Para manejar las demás habría que agregar un arreglo TIPO, en el cual se guardaría el tipo de cuenta a manejar con la transacción.

j) Cola TEMP - Es un arreglo de 64 elementos de 11 bytes cada uno. Almacena temporalmente los paquetes de información recibidos desde L3S. Conforme SIST va procesándolos, se liberan los elementos de la cola.

k) Variable LOCAL - Es una variable lógica que se pone en "verdadero" cuando esta sucursal (A) queda trabajando sólo localmente (o sea, no puede enviar información a la red por algún motivo). La variable tomará un valor "falso" en el momento en que la sucursal queda ligada de nuevo a la red.

l) Arreglo BUFTT.- Es un arreglo de una dimensión de 110 bytes el cual se utiliza cuando SIST intercambia información con un módulo de terminal.

ALGORITMO

Rutina INICIO

1. Ordena la lectura del archivo directorio para cargar su información en el arreglo CLI; con esto, establece el estado inicial de CLI.
2. Establece el estado inicial de los demás arreglos que utiliza.
3. Para todas las terminales que están activas, pone un 1 en los elementos asociados del arreglo ETAPA. Con esto se indica que en todas las terminales las transacciones se encuentran en la etapa 1. Esta etapa de hecho significa que aún no empieza transacción alguna.
4. Envía mensaje de inicio a todas las terminales activas.
5. Hace LOCAL=false.

Rutina FIN

1. Envía todo el contenido de CLI a DRVDSK para que se escriba en el archivo directorio, a fin de conservar los saldos con los cuales quedaron las cuentas al final de la sesión.
2. Envía a todas las terminales el mensaje:

FIN DE SESION
EL SISTEMA EMPIEZA A DARSE DE BAJA

para indicar que la sesión ha concluido y el sistema comienza a darse de baja.

Rutina PRINCIPAL

Diagrama de la figura 4.12.

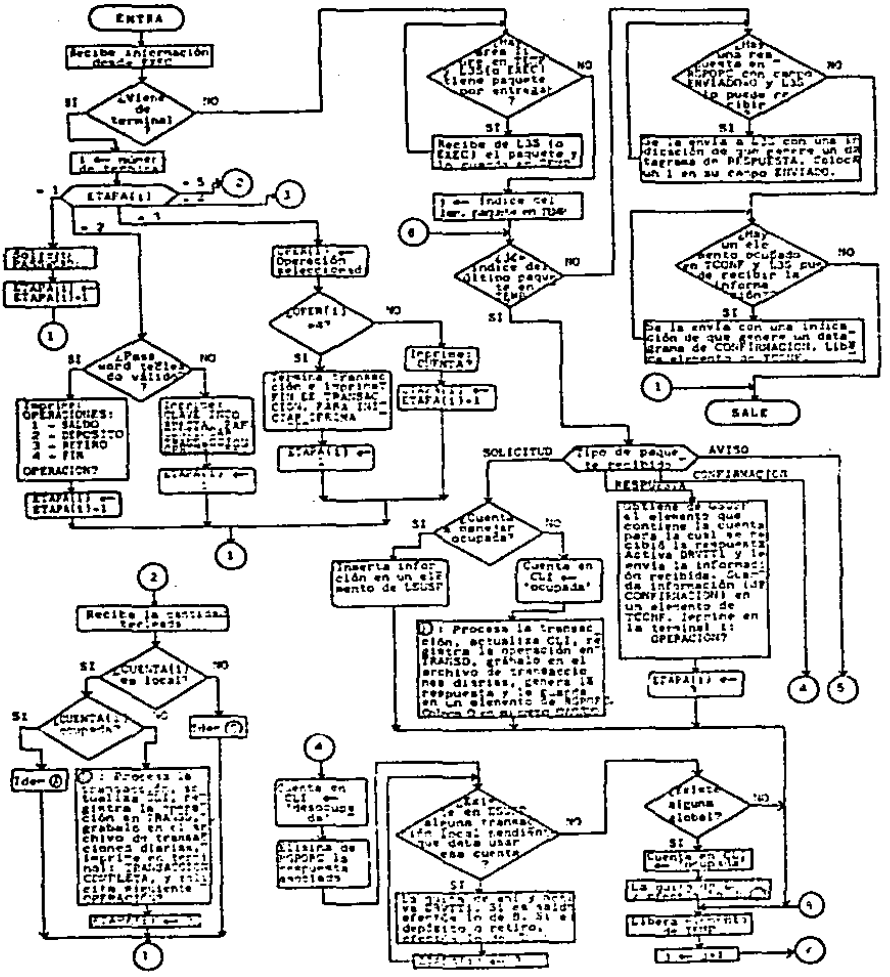


Fig. 4.12 Algoritmo del módulo SIST.

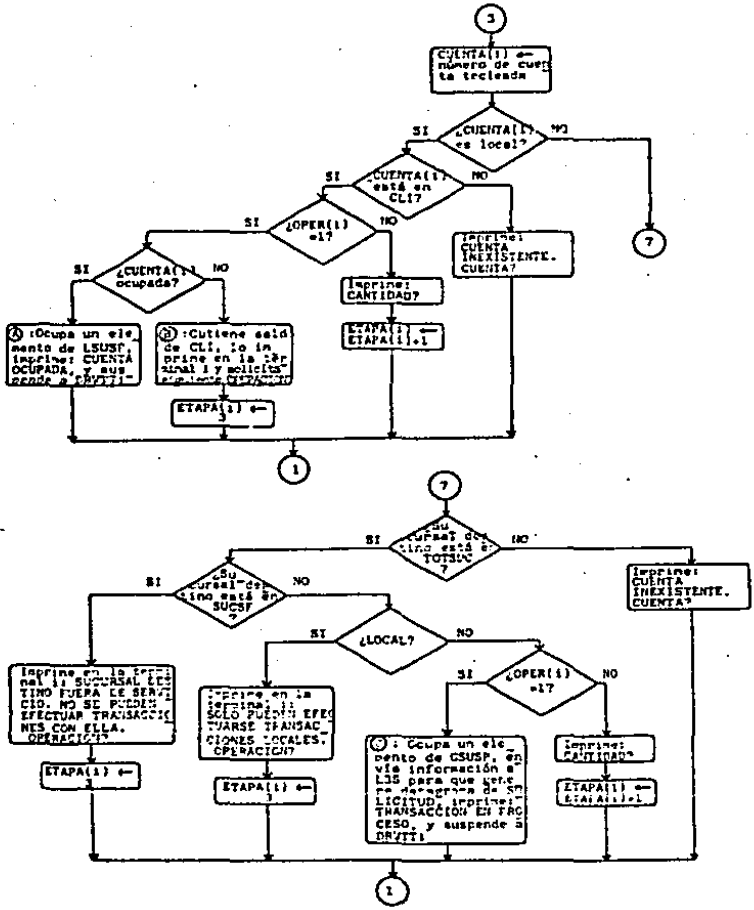


Fig. 4.12 Algoritmo del módulo SIST (Cont.).

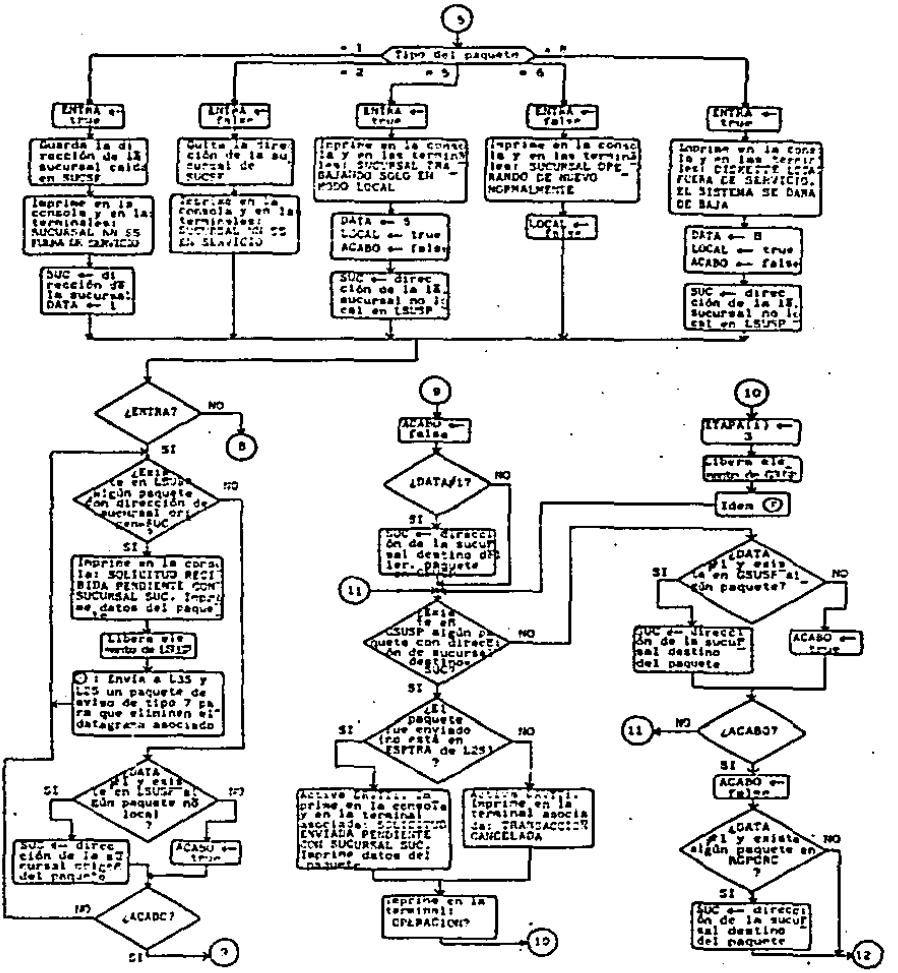


Fig. 4.13 Algoritmo del módulo SISF (Cont.).

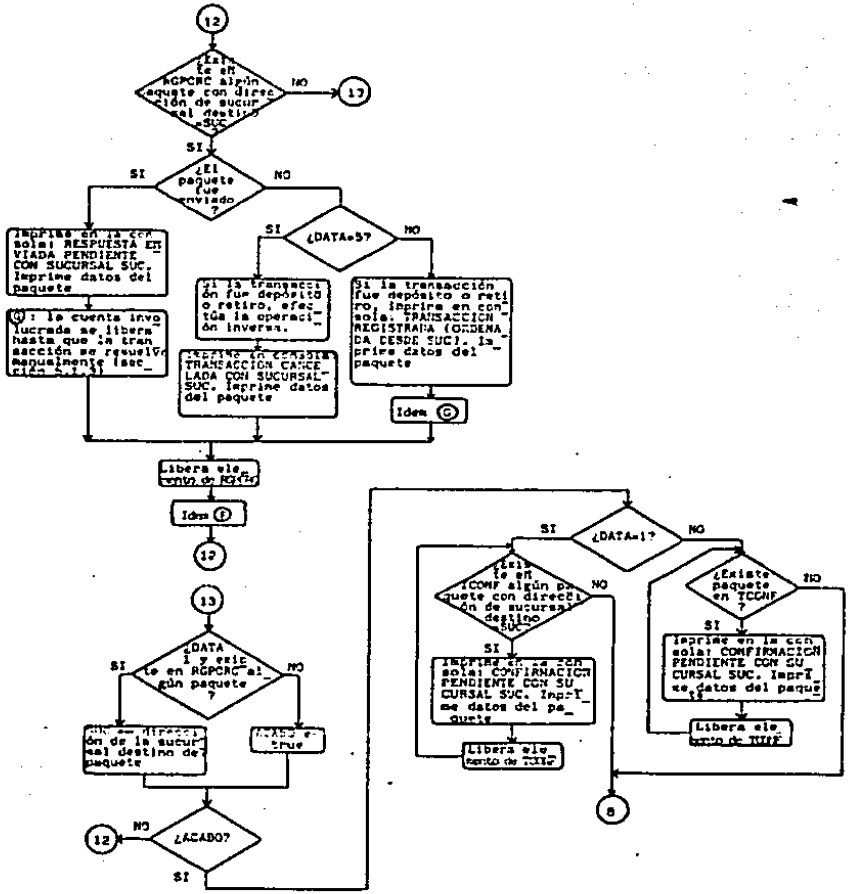


Fig. 4.12 Algoritmo del módulo SIST (Cont.).

4.6 MANEJADOR DEL DISKETTE: DRVDSK

Es el módulo que se encarga de controlar directamente la operación de la unidad de diskettes. Cuando se va a efectuar una operación de lectura/escritura con el diskette, DRVDSK se encarga de iniciarla, controlarla y supervisar su terminación. Si se trata de una lectura, lleva a cabo lo necesario para accesar la información en el diskette y enviarla a EXEC; si se trata de una escritura, recibe de EXEC la información correspondiente y, manipulando los registros de la unidad, efectúa lo necesario para que ésta quede grabada en el diskette. Cuando hay error en la operación avisa a EXEC para que éste realice lo necesario para recuperar la falla.

Este módulo realiza funciones similares a las de un "driver" o "handler" de un periférico. A través de lenguaje ensamblador programa directamente los registros de operación de la unidad (descritos en el apéndice B); esto es, no emplea las rutinas que Cromix proporciona para tal fin. El programar directamente estos registros le permite accesar la información del diskette directamente a nivel de superficie, pista y sector.

Lo anterior se hace para dar un tiempo de respuesta más corto cuando se efectúa un acceso al diskette al registrar una operación de depósito o retiro en una transacción. El tiempo es más corto ya que se evita pasar por todas las rutinas que necesita usar Cromix cuando hace un acceso al diskette. Aunque la programación de este módulo, en principio, fue difícil -por la complejidad natural que se tiene cuando se programan directamente los elementos de hardware de un periférico-, una vez ya dominado fue de bastante utilidad por el beneficio, en tiempo, que se tiene al accesar directamente la información del diskette, a nivel de sectores, empleando un solo programa.

A continuación se describen las principales estructuras usadas y el algoritmo de este módulo.

ESTRUCTURAS DE INFORMACION

Se utiliza un solo arreglo, de nombre BUF, de 517 bytes. Se emplea para intercambiar información entre EXEC y DRVDSK. Cuando EXEC envía información a DRVDSK, a través de los pipes, ésta se recibe en este arreglo. La información puede contener una orden de lectura de un sector o puede indicar la escritura de un bloque de 512 bytes. Cuando DRVDSK envía información a EXEC, en este arreglo se coloca la misma. Si se recibió orden de lectura, en él se envía el bloque leído; si fue de escritura, sólo se envía el resultado de la operación.

Cuando se recibe información desde EXEC, cada byte tiene el siguiente uso:

Byte(s)	Uso
1	1- mueve cabeza de lec./esc.; 0- no la mueve.
2	1- escribe un bloque de 512 bytes; 0- lee el bloque.
3	Número de la superficie del sector.
4	Pista del sector.
5	Número del sector.
6 a 517	Si es escritura, contiene la información a escribir; si es lectura, están vacíos.

Cuando se envía información a EXEC, los bytes se usan de la siguiente forma:

Byte(s)	Uso
1 a 4	No se emplean.
5	Resultado de la operación: 0- bien; 1- error.
6 a 517	Si se ordenó lectura, contiene la información leída; si fue escritura, vacíos.

ALGORITMO

1. Establece las condiciones iniciales de trabajo de la unidad de diskettes, que consisten, básicamente, en activar la unidad -o sea arrancar el motor de la unidad- y en colocar la cabeza de grabación en la pista 0 del diskette.

2. Suspende su ejecución hasta que recibe de EXEC una orden de lectura o escritura. Cuando la recibe efectúa lo siguiente:
   2.1 Si se trata de una lectura:
      2.1.1 Si en BUF se recibió indicación de mover la cabeza, DRVDSK coloca en los registros correspondientes de la unidad, los valores adecuados para que la cabeza se mueva a la pista y sector indicados, y selecciona la superficie que se va a manejar.
      2.1.2 Deshabilita interrupciones, ya que para poder controlar la lectura no debe efectuarse operación alguna en el resto del sistema, debido a que se tiene un intervalo aproximado de 14 microsegundos para recibir en BUF cada byte enviado por la unidad de diskettes.
      2.1.3 Coloca la unidad en modo AUTOWAIT. En este modo, DRVDSK suspende su ejecución, después de ordenar la lectura de un byte, hasta que la unidad le envía el byte que acaba de leer. En cuanto se termina de leer un sector, la unidad sale de este modo.
      2.1.4 En un ciclo lee los 512 bytes de un sector y los guarda en BUF.
      2.1.5 Habilita de nuevo las interrupciones para que el sistema continúe operando normalmente.
      2.1.6 Coloca en el byte correspondiente en BUF el status final de la operación.
      2.1.7 Envía el contenido de BUF a EXEC a través de los pipes establecidos entre los dos. Continúa en el punto 3.
   2.2 Si se trata de una escritura en el diskette lleva a cabo

acciones parecidas a las anteriores, con la diferencia de que ahora los valores a escribir se toman de BUF y se envían a la unidad para que los grabe en el sector indicado del diskette, moviendo la cabeza a donde éste se localiza, si fuese necesario. En este caso a EXEC sólo se envía el resultado de la operación.

3. Regresa al punto 2 a esperar otra orden de lectura/escritura. Cuando recibe la indicación de que el sistema va a darse de baja, DRVDSK apaga el motor de la unidad de diskettes y termina su trabajo.

Como comentario final de este módulo cabe mencionar que, aunque el algoritmo es más sencillo que en el caso de SIST y EXEC, constituyó una de las partes más difíciles de implantar debido a la complejidad que representó el programar directamente, a nivel de lenguaje ensamblador, los registros programables de la unidad de diskettes; sobre todo, por la poca información que proporcionan los manuales al respecto.

La mayor parte de este módulo se obtuvo después de una larga serie de pruebas de ensayo y error, hasta que se llegó a un dominio aceptable de la unidad. A pesar de todo, constituyó una buena experiencia y para efectos de los tiempos de respuesta a las transacciones, aportó una mejoría notable ya que con el acceso directo al diskette no hubo necesidad de pasar por las rutinas de Cromix que hubieran agregado bastante "overhead" a las transacciones de depósito y retiro.

4.7 MANEJADOR DE TERMINAL: DRVTT

Este módulo es el que se encarga de efectuar el manejo de una terminal. Cuando se va a escribir información en ella, este módulo la recibe desde EXEC y controla su escritura en la misma; cuando el operador teclea caracteres en la terminal, DRVTT controla la recepción de los mismos y al detectar el fin de línea, envía todo el conjunto a EXEC para que éste los reciba como un paquete.

Hasta cierto punto el módulo actúa como un "handler" de terminal, aunque no es tan completo como el manejador del diskette. Por un lado, aunque controla cuestiones tales como: quitar "eco" de la pantalla, detectar si un carácter se tecleó, controlar teclas especiales como <CTRL><U>, <CTRL><Z>, etc.; no lleva a cabo todas las actividades propias de un manejador de terminal como: control de paridad, velocidad de transmisión, etc., y esto se le deja a las rutinas de Cromix. Por otro lado, sus funciones las realiza empleando llamadas a Cromix, sin hacerlas directamente en ensamblador con lo cual se agrega un cierto "overhead" en la transmisión de caracteres hacia/desde la

terminal.

Para cada una de las terminales conectadas a la micro existe uno de estos módulos. Todos son semejantes y solo tienen pequeñas diferencias en cuanto a la identificación que usan para apoderarse de una terminal. Cada módulo se reconoce con los nombres DRVTT1, DRVTT2,..., DRVTTn, con n<=15. Todos son independientes entre sí, cada uno controlando la operación de una terminal.

ESTRUCTURAS DE INFORMACION

Al igual que en el caso de DRVDSK se utiliza un solo arreglo, también de nombre BUF, de 110 bytes. A través de él se efectúa el intercambio de información entre EXEC y DRVTTi(i=1,...,15(máximo)). La utilización que se hace de este arreglo es similar a la de ese programa.

Cuando se recibe información desde EXEC cada byte contiene lo siguiente:

Byte(s)	Uso
1	Sin uso.
2	0- lectura desde la terminal; 1- escritura en la misma.
3	En caso de lectura: 0- sin eco; 1- con eco. En caso de escritura no se usa.
4	Si se ordena lectura, contiene 0. Si se ordena escritura, contiene el número de caracteres a escribir.
5 a 110	Si es escritura, contienen la información a escribir; si es lectura, están vacios.

Cuando se envia información a EXEC los bytes contienen lo siguiente:

Byte(s)	Uso
1	Número de la terminal que envia la información.
2 y 3	Sin uso.
4	Si se ordenó lectura: se envia el número de caracteres leidos en caso de que se haya detectado el fin de linea; 0, en caso contrario. Si se ordenó escritura: se envia el valor 1.
5 a 110	Si se ordenó lectura: se envia la información leida si se detectó el fin de linea; están vacios, en caso contrario. Si se ordenó escritura, se envia el resultado de la operación: 0- bien, 1- mal.

ALGORITMO

1. Establece las siguientes condiciones iniciales:

terminal.

Para cada una de las terminales conectadas a la micro existe uno de estos módulos. Todos son semejantes y sólo tienen pequeñas diferencias en cuanto a la identificación que usan para apoderarse de una terminal. Cada módulo se reconoce con los nombres DRVTT1, DRVTT2,..., DRVTTn, con n<=15. Todos son independientes entre sí, cada uno controlando la operación de una terminal.

ESTRUCTURAS DE INFORMACION

Al igual que en el caso de DRVDSK se utiliza un solo arreglo, también de nombre BUF, de 110 bytes. A través de él se efectúa el intercambio de información entre EXEC y DRVTTi(i=1,...,15(máximo)). La utilización que se hace de este arreglo es similar a la de ese programa.

Cuando se recibe información desde EXEC cada byte contiene lo siguiente:

Byte(s)	Uso
1	Sin uso.
2	0- lectura desde la terminal; 1- escritura en la misma.
3	En caso de lectura: 0- sin eco; 1- con eco. En caso de escritura no se usa.
4	Si se ordena lectura, contiene 0. Si se ordena escritura, contiene el número de caracteres a escribir.
5 a 110	Si es escritura, contienen la información a escribir; si es lectura, están vacios.

Cuando se envía información a EXEC los bytes contienen lo siguiente:

Byte(s)	Uso
1	Número de la terminal que envía la información.
2 y 3	Sin uso.
4	Si se ordenó lectura: se envía el número de caracteres leídos en caso de que se haya detectado el fin de línea; 0, en caso contrario. Si se ordenó escritura: se envía el valor 1.
5 a 110	Si se ordenó lectura: se envía la información leída si se detectó el fin de línea; están vacios, en caso contrario. Si se ordenó escritura, se envía el resultado de la operación: 0- bien, 1- mal.

ALGORITMO

1. Establece las siguientes condiciones iniciales:

- se apodera de la terminal para controlar totalmente la lectura/escritura de caracteres con ella;
- quita el eco para que no aparezca un carácter impreso en la pantalla, en caso de que se teclee en forma anticipada a su lectura;
- da la indicación para que el control regrese al programa después de que haya leído un byte de la terminal. Esto se hace para que el programa no se quede "colgado" esperando a que se teclee el fin de línea.

2. Suspende su ejecución hasta que recibe atención por parte de EXEC. Cuando esto sucede efectúa lo siguiente:

2.1 Si recibe una orden de escritura, toma la información correspondiente de BUF y la escribe en la terminal. Cuando termina la operación, checa el estado final y lo envía a EXEC. Continúa en 3.

2.2 Si recibe orden de lectura:

2.2.1 Determina si se tecleó algún carácter. Si así fue, lo lee y lo guarda en BUF. Aquí cabe mencionar que para efectuar la lectura se emplea una función de Cromix, la cual permite regresar el control a este programa aunque un carácter no haya sido tecleado. Si no se hiciera así, la terminal, y por consiguiente el programa, quedarían "colgados" esperando que se oprimiera alguna tecla. Con esto se paralizaría totalmente el sistema y no habría forma de dar atención al resto de las terminales. Como esto no sucede así, es fácil determinar si se regresa el control a EXEC o no, dependiendo de que haya más caracteres tecleados o no.

2.2.2 Repite la lectura mientras haya más caracteres tecleados.

2.2.3 Si el último carácter leído es el de fin de línea, envía toda la información tecleada a EXEC. En caso contrario, envía un cero en el byte 4 de BUF para que ante EXEC aparezca como si ningún carácter hubiera sido tecleado. La finalidad de lo anterior es la siguiente: una línea tecleada sólo será procesada hasta que DRVITI detecte el fin de línea. Mientras esto no suceda, DRVITI hará aparecer ante EXEC que en la terminal no se tecleó información alguna. Con esto EXEC pasa a dar atención a otra terminal en la cual se esté efectuando otra transacción. Esto acelera los tiempos de respuesta a las transacciones, ya que impide que se le de demasiada atención a una terminal en la cual aún no se da el fin de línea y permite, a la vez, dar atención a otras terminales en las cuales también se están realizando transacciones en ese momento.

3. Regresa al punto 2 a esperar otra orden de lectura/escritura. Cuando se recibe la indicación de que el sistema va a darse de baja, DRVITI restablece las condiciones de operación originales de la terminal, la libera y concluye su trabajo.

CAPITULO 5

RECUPERACION DE FALLAS Y SEGURIDAD

En este capitulo se describen algunas politicas a seguir para recuperar fallas y errores que se presenten durante la operación del sistema. Asimismo, se tratan aspectos relacionados con la seguridad de éste.

5.1 RECUPERACION DE FALLAS Y ERRORES

En esta sección se describen las politicas que se deben seguir cuando se presenta una falla o error, tanto en el funcionamiento y operación de la red como en el procesamiento de las transacciones. Se comentan también las acciones de emergencia que efectúan los módulos de software para recuperar algunas de estas fallas. En los algoritmos presentados en los dos capitulos anteriores, estas acciones se muestran en forma más concreta por lo cual aqui sólo se mencionan brevemente.

5.1.1 Recuperación de Errores en la Transmisión entre dos Puntos Adyacentes

En la transmisión de información entre dos puntos adyacentes de la red -sean sucursales o nodos- se pueden presentar problemas relacionados con la operación de las capas 1 y 2; o sea, problemas que pueden ocurrir con la conexión física de las micros con el equipo de comunicación, así como en la transmisión de bits de un punto a otro de la red. Entre los problemas que se pueden presentar están los siguientes: falla de los modems, falla de la linea de comunicación, errores en la transmisión de los bits, fallas momentáneas de algún circuito o componente de un equipo, marcos perdidos y marcos inválidos de paquetes HDLC.

Cualquiera de estas fallas ocasiona que un paquete HDLC enviado entre dos micros adyacentes sea incorrecto y, por lo tanto, rechazado por la micro receptora. Cuando esto sucede, la micro receptora tiene que indicar de alguna manera a la micro

emisora, que no están recibiéndose correctamente sus paquetes. El protocolo HDLC prevé estos casos y ya tiene incluidos tanto los mecanismos para la detección de errores, como los procedimientos a seguir para efectuar la retransmisión de paquetes erróneos. En el Apéndice A se da una descripción detallada de estos mecanismos y procedimientos los cuales son contemplados por los módulos L2S y L2NN (sección 3.1) para controlar el intercambio correcto de información a estos niveles.

Por cada paquete HDLC enviado de una micro a otra, se necesita recibir en la micro emisora otro paquete HDLC (tipo I o tipo S) que contenga la confirmación de que aquel fue bien recibido por la micro receptora. Si por alguna causa dicha confirmación no es recibida en un cierto periodo de tiempo o lo que se recibe es una indicación de que el paquete no fue correcto, entonces el paquete original es retransmitido hacia la micro receptora. Si después de un cierto número de retransmisiones la aceptación no es recibida, ya sea porque nunca llega o porque siempre se recibe que el paquete transmitido es erróneo, entonces se considera que: algún componente del equipo de transmisión está fallando (p. ej., un modem); alguno o los dos puertos de transmisión de las micros están fallando, la línea de comunicación entre ambas micros está fuera de servicio o la micro receptora quedó fuera de servicio.

En cualquiera de estos casos la transmisión entre ambas micros es suspendida por los módulos L2S (o L2NN, según el caso) y avisan a EXEC (o EXL3) para que tomen las medidas pertinentes. Las acciones correctivas que se deben efectuar corresponden al nivel 3 por lo que son descritas en la siguiente sub-sección.

5.1.2 Recuperación de Errores en la Transmisión entre Dos Puntos Cualesquiera

La mayor parte de los problemas que se presentan en la transmisión de información entre dos sucursales cualesquiera están relacionados con la capa 3; o sea, con el Sistema de Transporte de paquetes de información de la red.

Para tratar algunos de estos problemas se pueden usar datagramas especiales, que llamaremos de aviso, que lleven información relacionada con el problema detectado. Estos datagramas también son de 18 bytes y tienen el formato mostrado en la figura 5.1.

Los primeros 6 bytes tienen la misma función de un datagrama normal. En el tercer byte, los valores 1 en el primer y tercer subcampos, indican que el datagrama es de aviso. Los bytes 7 y 8 sirven para identificar a quién(es) está dirigido el datagrama.

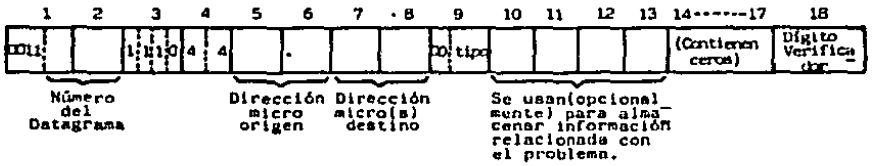


Fig. 5.1 Formato de un datagrama de aviso.

Los valores que pueden tomar son:

Byte 7	Byte 8	Destinatario
0	0	Todos los nodos.
0	-1	Todas las sucursales de la red.
m	0	Nodo m (1<=m<=99).
m	-1	Todas las sucursales del nodo m.
m	n	Nodo m, sucursal n (1<=n<=15). Este caso corresponde al de un datagrama normal.

Los últimos seis bits del noveno byte se usan para indicar el tipo de falla detectado, de acuerdo con la siguiente tabla:

Tipo	Función	Bytes adicionales usados
1	Sucursal fuera de servicio	10 y 11, para contener la dirección de la sucursal.
2	Sucursal de nuevo en servicio	Idem.
3	Linea de transmisión caida entre dos nodos o uno de los nodos fuera de servicio	10,11 y 12,13 para contener las direcciones de los nodos involucrados.
4	Linea en servicio de nuevo entre dos nodos o nodo en servicio	Idem.
5	Sucursal trabajando sólo en modo local	Ninguno.
6	Sucursal ligada de nuevo a la red	Ninguno.
7	Elimina datagramas (SIST lo envia a L3S y L2S)	Ninguno.
8	Unidad de diskettes fuera de servicio (EXEC lo envia a SIST y L2S)	Ninguno.

Dependiendo del problema, se pueden emplear los bytes 10, 11, 12 y 13 para almacenar direcciones de micros involucradas en el mismo. Si no se usan, almacenarán ceros al igual que los bytes 14 a 17.

Como se observa estos datagramas no sólo se usan para indicar una falla, sino también para señalar su recuperación. En general, cuando sea necesario intercambiar datagramas en la red para avisar alguna situación especial se empleará este tipo de datagramas. También se usan cuando algún módulo de sucursal quiere avisar alguna situación especial a otro módulo dentro de la misma sucursal (p. ej., el de tipo 8).

A continuación se describen brevemente los diferentes errores que se pueden presentar a este nivel, cómo recuperarlos y el código usado para los datagramas en cada caso.

a) Suspensión de la transmisión entre un nodo y una sucursal - En la sub-sección anterior se mencionó que por problemas en la capa 1 (o en la 2) puede suspenderse la transmisión entre dos puntos adyacentes de la red. Cuando esto ocurre entre un nodo y una sucursal se deben llevar a cabo varias acciones, según sea la causa de la suspensión.

En primer lugar, si la micro de la sucursal continúa operando, entonces la sucursal sólo trabajará en modo local satisfaciendo las transacciones de aquellas cuentas que estén registradas en la misma. Si se reciben solicitudes para cuentas que están en otra sucursal, no serán procesadas y sólo aparecerá en la terminal correspondiente un mensaje indicando que la sucursal está trabajando únicamente en modo local. El aviso y recuperación de esta falla se efectúa sólo localmente (dentro de la propia sucursal) por medio de datagramas de aviso tipos 5 y 6, respectivamente.

Si la micro de la sucursal no opera, entonces definitivamente no se llevará a cabo ninguna transacción y será necesario esperar hasta que el problema sea resuelto.

La política a seguir con los clientes para estos dos últimos casos puede ser: no llevar a cabo niguna operación o realizar únicamente operaciones de depósito y de retiro, hasta un cierto límite, llevando un registro manual de éstas. Cuando la micro opere de nuevo y quede enlazada a la red, entonces se registrarán estas operaciones en los archivos correspondientes para que las cuentas involucradas queden actualizadas como corresponde.

En segundo lugar, el módulo EXL3 del nodo al cual está conectada la sucursal, deberá llevar a cabo las siguientes acciones para indicar que ésta se encuentra fuera de servicio:

- indicar en la tabla ASIGS que esa sucursal quedó fuera de servicio;
- eliminar todos los datagramas, que tiene en ese momento, dirigidos a esa sucursal;
- generar un datagrama de aviso con código=1 y la dirección de dicha sucursal para enviarlo a los demás nodos, para que éstos, a su vez, lo transmitan a todas sus sucursales.
  En cada sucursal receptora se registra en la tabla SUCSF (inciso g, sección 4.5) esta dirección, para ya no enviar más

datagramas a la misma. Además de las acciones adicionales indicadas en los algoritmos de 4.4 y 4.5 para este tipo de falla, se envían mensajes de aviso a las terminales cada que en alguna de éstas se pretenden efectuar transacciones con cuentas de dicha sucursal.

Una vez hecho esto, el EXL3 del nodo verifica constantemente si la sucursal ya está de nuevo en servicio, por medio de paquetes HDLC de supervisión. Si ésta ya se encuentra trabajando, contestará con un paquete semejante al nodo, para que éste se entere de que pueden intercambiar información de nuevo. EXL3 cambia entonces el status de la sucursal en ASIGS y envía a las demás sucursales un datagrama de aviso con código=2 y la dirección de la sucursal, para que aquellas la quiten de su tabla SUCSF y puedan establecer de nuevo comunicación con ella. Con esto se da lugar a que puedan efectuarse de nuevo transacciones con cuentas registradas en dicha sucursal.

b) Suspensión de la transmisión entre dos nodos.- Cuando la transmisión entre dos nodos se suspende se debe verificar si las micros de los nodos continúan operando o no.

En cada nodo debe existir una micro de respaldo cuyo propósito es sustituir a la micro principal cuando ésta falle. Por cada nodo en la red debe existir por duplicado: el procesador central y el procesador frontal. Al fallar cualquiera de éstos, su cambio se haría manualmente (posiblemente por medio de un interruptor que desconecte unos y conecte a los otros). Lo ideal sería efectuar un proceso automático de conmutación el cual se efectuaría al ocurrir la falla. No se trata esa posibilidad en este trabajo. Dado que el objetivo principal del proceso de recuperación es que el nodo no deje de funcionar mucho tiempo, se propone la primera alternativa ya que es peor no hacer nada esperando a que la falla fuese arreglada.

Al fallar la micro primaria obviamente va a perderse la información manejada por la misma en ese momento. Existen dos tipos de información que toda micro de un nodo maneja: una puede denominarse de TRANSITO y otra PERMANENTE. La primera estará formada por todos los datagramas que se encuentran en tránsito por ese nodo, ya sea hacia sus sucursales o hacia otros nodos. Al perderse esta información no hay mayor problema, ya que el protocolo de capa 3 prevé la pérdida de mensajes y tiene los mecanismos apropiados para su retransmisión. La información permanente es la que almacenan las tablas ASIGS, ASIGN, TOTSUC y RUTA del módulo EXL3. Para poder recuperar la última información almacenada por éstas basta con grabar en un archivo su contenido cada que ocurre un cambio en ellas durante una sesión (las tablas son más o menos estáticas). Así cuando inicie su trabajo la micro de respaldo, cargará estas tablas con el contenido de ese archivo para que queden tal como estaban antes de la falla.

Los nodos vecinos del que falló comunican a los demás nodos de la red, con datagramas de aviso tipo 3, que éste quedó fuera de servicio, para que registren la falla en su tabla RUTA y así

eviten ese camino al enviar datagramas.

Si la falla no es en la micro, puede estar en el modem, en la línea de comunicación o en el puerto de transmisión. En cualquiera de los tres casos se considera que la línea entre un nodo y otro está caída. Cuando esto sucede ambos registran la falla en su tabla ASIGN (inciso b, sección 3.3) y la comunican también a los demás nodos con datagramas de aviso tipo 3.

Al igual que con las sucursales, los nodos tratarán de establecer comunicación entre sí constantemente, a través de paquetes HDLC de supervisión. Cuando la falla es recuperada, ambos recibirán este tipo de paquete lo cual les indicará que el problema fue resuelto. En ese momento cambian el byte de Status de su tabla ASIGN, reanudan el intercambio normal de información entre ellos y comunican la recuperación, con datagrama de aviso tipo 4, a los demás nodos de la red.

Es conveniente que en todas las micros, tanto de nodos como de sucursales, existan equipos de "no-break" para evitar que éstas dejen de operar por falta de energía eléctrica.

c) Errores varios.- A continuación se mencionan errores adicionales que pueden presentarse en esta capa de la red; la mayoría ya se comentó en secciones anteriores. En este inciso sólo se tratan brevemente:

- mensajes perdidos: cuando un mensaje se pierde ya sea porque se descompone brevemente una micro, porque la dirección destino se altera o por cualquier causa,entonces el datagrama enviado por una sucursal nunca llegará a su destino. El manejo de los datagramas en una sucursal ya prevé este caso y un datagrama enviado no será eliminado sino hasta que se reciba la confirmación de su recepción correcta desde la sucursal destino (secciones 3.2 y 3.4.3).
- mensajes duplicados: el mecanismo empleado para el manejo de datagramas en una sucursal permite detectar la recepción de mensajes duplicados. Al controlar esto se evita procesar doblemente una misma transacción.
- falla en equipo periférico de una micro: en todas las micros -sean nodos o sucursales- existen unidades de disco (fijo o diskettes) y una o más terminales (en sucursales) para procesar las transacciones. Cuando una unidad de disco falle en una sucursal, EXEC enviará un datagrama de tipo 8 a SIST y L2S para indicarles esta falla. SIST termina de procesar todas las transacciones que está efectuando en ese momento y L2S suspende sus transmisiones con el nodo asociado. Después de esto, EXEC da de baja el sistema y la sucursal se considera como fuera de servicio. Las acciones que efectúa el nodo son las indicadas en el inciso a) anterior. Cuando la falla se recupere, se llevarán a cabo los mismos procedimientos de recuperación indicados en ese inciso.
  En un nodo normalmente se tendrán dos unidades de diskette, de tal manera que si falla una, simplemente entrará en acción la de respaldo, cambiando el diskette de operación de una

unidad a la otra. En este caso, la falla del diskette no es tan crucial como en el caso de una sucursal, por lo que la falla prácticamente no afecta la operación del nodo.
Con respecto a las terminales, se pueden tener una o dos de emergencia por sucursal para usarlas cuando fuera requerido. Lo importante de esto es que si falla una terminal, no afecta en forma relevante al sistema.

5.1.3 Recuperación de Errores en el Procesamiento de una Transacción

En esta parte se describen las políticas y procedimientos a seguir cuando una transacción es interrumpida antes de que se procese totalmente.

Una transacción se interrumpe cuando alguna de las micros -de sucursales- involucradas en su procesamiento queda fuera de servicio, ya sea por falla en alguno de sus componentes o por cualquier causa. El efecto neto es que la transacción puede quedar incompleta y, dependiendo del punto en el cual se suspendió, habrá que tomar medidas de emergencia o no.

A continuación se detallan las medidas de recuperación que se deben adoptar. Se emplean de nuevo los términos sucursal A y B para denotar a las sucursales origen y destino involucradas en el procesamiento de la transacción.

a) La sucursal A queda trabajando sólo en modo local - Cuando la sucursal queda desligada de la red, por falla de la línea o del nodo asociado, pero continúa trabajando en modo local, se llevan a cabo las siguientes acciones (las cuales ya están contempladas en los algoritmos de los módulos del sistema local):

- en las terminales y consola se imprime un mensaje el cual indica que la sucursal sólo está operando localmente;
- todas las solicitudes recibidas en las terminales para efectuar transacciones globales se eliminan, si los datagramas respectivos no fueron transmitidos al nodo asociado. Si se transmitieron, en la consola se imprime el número de las sucursales involucradas;
- todas las respuestas, a solicitudes recibidas desde B, se eliminan si los datagramas correspondientes no se enviaron al nodo. SIST lleva a cabo la operación inversa indicada por la respuesta (p. ej., si se había efectuado un depósito, ahora se registra un retiro, por igual monto, sobre la misma cuenta). Se imprimen en la consola los números de las sucursales involucradas.
  Si los datagramas se enviaron, también se imprimen en la consola los números de las sucursales involucradas y además se bloquean las cuentas usadas en las transacciones;

- para todos los datagramas de confirmación que no hayan sido enviados, imprime los números de las sucursales a los cuales iban dirigidos;
- las transacciones locales se siguen procesando normalmente, a menos que la cuenta esté bloqueada por alguna transacción global;
- las solicitudes globales recibidas que no hayan sido procesadas, se eliminan y en la consola se imprime el número de las sucursales involucradas; y
- no se procesan más transacciones globales.

Todas las transacciones que hayan quedado pendientes, serán resueltas estableciendo contacto telefónico con las sucursales involucradas en las mismas. Por medio de un programa auxiliar (EMERG, sección 6.3) en la sucursal A se accesará directamente el diskette para modificar manualmente las transacciones que hayan quedado pendientes, en caso de que esto sea requerido. Esto sólo se hará para depósitos y retiros; los saldos se descartan ya que con ellos no hay problemas adicionales.

Las cuentas de la sucursal A involucradas en transacciones globales permanecerán bloqueadas hasta que la transacción inconclusa sea resuelta. Con esto se evita generar inconsistencias para dichas cuentas.

Cuando la falla se recupera, la micro de la sucursal A se liga a la red sólo hasta que todas las transacciones globales pendientes han sido resueltas. SIST y EXEC (de A) indican a L2S el momento en el cual debe empezar a transmitir al nodo asociado.

b) Falla en la sucursal B - Cuando la sucursal B queda fuera de servicio, el resto de las sucursales del sistema registra la dirección de la misma en SUCSF en cuanto reciben el datagrama de aviso de tipo 1.

Las acciones que efectúan con las transacciones pendientes con esa sucursal, son parecidas a las del inciso anterior, sólo que únicamente se afectan los datagramas intercambiados con esa sucursal. También se establece contacto telefónico con ella para resolver las transacciones pendientes que no pudieron ser resueltas automáticamente por EXEC y SIST.

Mientras no se reciba el datagrama de aviso tipo 2 que indica que la sucursal B está de nuevo en servicio, no se efectuarán transacciones adicionales con ella.

c) Micro de sucursal deja de operar - Cuando esta falla ocurre no se podrán efectuar acciones adicionales. Debido a esto, no hay manera de resolver las transacciones pendientes ni siquiera por vía telefónica.

En este caso las sucursales que tenían transacciones pendientes con la sucursal que deja de operar, tendrán que establecer una política que podría ser la siguiente:

- en caso de saldos, cancelar la transacción sin efectuar acción adicional alguna;
- en caso de depósitos, registrar la operación manualmente. Cuando la sucursal esté de nuevo en servicio, registrarán esta operación por medio de una transacción global;
- en caso de retiros, fijar un límite en el monto del retiro y registrar la operación manualmente. Cuando la sucursal opere de nuevo, se hará lo mismo que en el caso anterior.

5.2 SEGURIDAD

En esta sección se comentan algunas políticas a seguir para evitar que personas no autorizadas tengan acceso a la información manejada con este sistema, con el fin de prevenir que dichas personas puedan llevar a cabo, de algún modo, transacciones sobre las cuentas.

No se hace un análisis detallado de todos los mecanismos de protección que pudieran contemplarse en cada uno de los puntos de la red susceptibles de ser vulnerados. Simplemente se propone un procedimiento para transmitir la información entre sucursales en forma codificada, de tal manera que cualquier mensaje extraño que llegue a una sucursal, sea descartado si no cumple con esta codificación.

El término seguridad también comprende el proteger el equipo físico contra accidentes como fuego, daños intencionales, inundaciones, etc.; no se hacen mayores comentarios a este respecto, porque se considera que en toda instalación existen forzosamente medidas de protección adecuadas contra este tipo de percances.

También entran aquí las medidas que se deben tomar para que la información no sea destruida o dañada por fallas de hardware o de software. En la sección anterior ya se discutieron los procedimientos de recuperación contra los diferentes tipos de error que se pueden presentar en la operación de la red, por lo que no se ahonda más en ellos.

Existen ya varios mecanismos de protección implícitos en la red. Por ejemplo, cuando una estación, sea nodo o sucursal, recibe un paquete HDLC, éste debe cumplir con el formato establecido, o sea marcos inicial y final y CRC correctos. Si llega un paquete que no cumple con estas características inmediatamente es descartado. Además si el paquete HDLC no es de la longitud exacta (24 bytes) también es descartado. Existen además dos dígitos verificadores en el datagrama, para el número de cuenta a manejar y para todo el datagrama, que ocasionan que éste sea descartado si no concuerdan con la información que viene en el mismo.

Se incluyen además dos mecanismos adicionales de protección. Uno es simple y consiste en preguntar por una palabra clave cada que un operador se va a conectar a la red por medio de una terminal en una sucursal. Es un mecanismo sencillo pero permite evitar que cualquier persona pueda conectarse a la red desde una de estas terminales. El otro es más complejo ya que se emplea Criptografía para codificar la información que contiene un datagrama.

La Criptografía [MART81] es un mecanismo con el cual se puede tener seguridad de que los datos no van a ser leidos, copiados o alterados durante su transmisión por las líneas de comunicación. En el proceso de codificación los bits de los datos son mezclados de tal manera que resulta difícil descifrar el contenido de los mismos, si no se conoce dicho proceso. Este método se aplica con los datagramas antes y después de su transmisión. En el esquema de la figura 5.2 se muestra el lugar en el cual esto se hace.

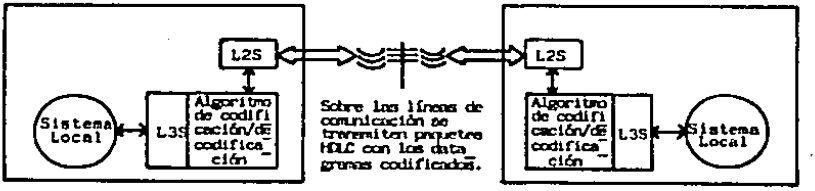


Fig. 5.2 Esquema de codificación/decodificación de un datagrama.

Cuando el módulo L3S construye un datagrama, antes de pasarlo a L2S le aplica el algoritmo de codificación. Una vez codificado es recibido así por L2S y transmitido de igual forma dentro de un paquete HDLC, hacia la sucursal destino. Cuando en ésta, L2S lo recibe, lo pasa tal cual a L3S. En el momento de recibirlo L3S aplica el algoritmo de decodificación para obtener el datagrama original, guardándolo así en sus estructuras. Los módulos EXL3 de los nodos también tienen este algoritmo y lo aplican para determinar la dirección destino del datagrama y así seleccionar la ruta por la cual debe ser enviado.

A continuación se describe el algoritmo para codificar/decodificar un datagrama [MART81][KERN76]. Este algoritmo utiliza una clave de 64 bits divididos en 8 bytes de 8 bits cada uno, para realizar la criptografía. La clave se utiliza de la siguiente manera: el primer byte se usa con el primer byte del datagrama, el segundo con el segundo, y así sucesivamente. Cuando la clave se acaba es reusada desde el principio tantas veces como sea necesario.

ALGORITMO DE CODIFICACION

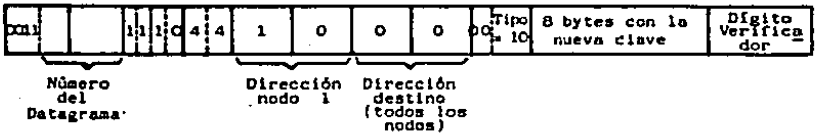
1. Tomar la clave y dividirla en 8 bytes. Identificarlos como K1,...,K8.
2. Identificar a cada byte del datagrama como D1,...,D18.
3. Hacer i=1 y j=1.
4. Generar byte Ni = Di xor Kj.
5. Hacer j=j+1; si j>8, hacer j=1.
6. Hacer i=i+1; si i<=18, regresar a 4. En caso contrario, el algoritmo termina. El conjunto de bytes N1,...,N18 constituyen el datagrama codificado.

El algoritmo de decodificación para obtener el datagrama original es el mismo y se aplica sobre el datagrama codificado utilizando la misma clave. Esto es cierto ya que se aprovecha el principio de simetria del o-exclusivo:

$$a \oplus b \oplus b = a$$

donde a es el datagrama original y b la clave. El orden de evaluación es irrelevante.

Como se observa el algoritmo es muy sencillo; la clave es bastante larga con lo cual es poco probable que pueda ser descifrada. Para dar un poco más de seguridad con respecto al manejo de la clave, ésta puede ser cambiada diario antes de iniciar una sesión de trabajo. La clave puede ser generada en forma aleatoria por un nodo y transmitirla al resto de la red para que durante esa sesión trabajen con dicha clave. El procedimiento a seguir es: un nodo central, el 1 por ejemplo, genera una clave aleatoria minutos antes de iniciar la sesión del día. Con la clave nueva genera el siguiente datagrama de aviso (éste, y los dos siguientes, tienen el mismo formato que los descritos en la sección 5.1.2, aunque con TIPO diferente):



una vez construido lo envia a los demás nodos codificándolo aún con la clave anterior. Cuando los demás nodos lo reciben, lo transmiten tal cual a todas sus sucursales para que registren la nueva clave que será usada durante esa sesión. Todas las sucursales le envian a su nodo datagramas del tipo:

codificados aún con la clave vieja, para indicarle que recibieron la nueva clave. El nodo les confirma la recepción de éste con un datagrama:

también codificado con la clave vieja. A partir de este momento, cualquier datagrama que envíe/reciba una sucursal se codificará/decodificará utilizando la clave nueva.

Una vez que los nodos han recibido la respuesta de todas sus sucursales, envían datagramas de tipo 11 al nodo central, codificados con la clave vieja, para notificar que tanto ellos como las sucursales ya tienen la clave nueva. El nodo central confirma con un datagrama de tipo 12 que envía a todos los nodos. A partir de este momento los nodos usarán la clave nueva cada que necesiten codificar/decodificar un datagrama.

CAPITULO 6

IMPLANTACION DEL SISTEMA LOCAL

6.1 GENERALIDADES

En esta sección se dan detalles concernientes a la implantacion del sistema local. Se ahonda en cuestiones relacionadas con el equipo utilizado, software empleado, limitaciones en cuanto a capacidades del sistema, caracteristicas de operación, etc. O sea, cuestiones que no están relacionadas directamente con el diseño del sistema.

a) Equipo utilizado.- Para efectuar la implantación se empleó la microcomputadora Cromemco Sistema 3 [CROM81]. Los periféricos conectados a esta máquina, asi como la forma en que fueron utilizados ya fue descrita en la sección 4.1. En el apéndice B se da una descripción más detallada de este equipo. En principio éste se usó para la implantación debido a que presentaba mayores facilidades para su utilización, sin interferir demasiado con el aspecto productivo de esta máquina en el lugar en el cual elaboré este trabajo.

b) Software empleado.- El sistema local se desarrolló para ser ejecutado bajo Cromix. Este sistema operativo es una versión simplificada de UNIX y es el utilizado con las computadoras Cromemco. En el apéndice B tambien se enuncian las diversas "directivas del sistema" empleadas en la elaboración de los módulos del sistema local.

En cuanto a programación todos los módulos, excepto DRVDSK, fueron elaborados utilizando el lenguaje C. El compilador proporciona funciones para efectuar llamadas al sistema operativo, con lo cual todos los módulos se encuentran totalmente programados en este lenguaje. Lo anterior da una cierta portabilidad al sistema debido a que la mayor parte del mismo está escrito en lenguaje de alto nivel. El módulo DRVDSK se programó en lenguaje ensamblador Z80 para accesar en forma más directa al diskette, principalmente, y asi acelerar los tiempos de respuesta para las transacciones de depósito y retiro. En la sección 4.6 se dieron más detalles con respecto a este módulo.

En todos los módulos los principios de modularidad y segmentación fueron observados, para facilitar tanto la programación de los mismos como su análisis posterior.

c) Capacidades del sistema.- Como se mencionó en la sección 4.1, en la implantación de este sistema tuvieron que darse valores concretos tanto para las capacidades de los archivos como para cantidades específicas a manejar dentro de los programas; esto debido a las restricciones impuestas por el equipo utilizado, principalmente. En la tabla 6.1 se detallan las capacidades específicas que tienen actualmente algunos de los archivos del sistema.

Archivo	Capacidad máxima actual
Transacciones diarias	2040 transacciones de depósito o retiro.
Transacciones mensuales	14336 transacciones de depósito o retiro.
Directorio	6000 cuentas (saldo y estado).
Maestro	6000 clientes (información general).

Tabla 6.1 Capacidades de almacenamiento de algunos archivos del sistema.

Para los archivos de Terminales en línea y de Identificación de la Sucursal no existe problema en cuanto a su capacidad.

En cuanto a los programas también existen algunas restricciones similares. Por ejemplo en el caso de EXEC, tenemos que sólo maneja actualmente 4 terminales (que es el total con que cuenta el equipo utilizado). Tanto éste como los valores indicados para los archivos están manejados en forma paramètrica, de tal forma que resulta fácil modificarlos en cualquier momento, si esto fuese requerido.

d) Ejecución y operación del sistema local.- Cuando el sistema local se ejecuta crea su propio ambiente de trabajo y se apodera del control de la mayor parte de la micro. Sólo deja a Cromix la parte de gestión de la memoria central y el manejo del disco fijo.

En este ambiente de trabajo única y exclusivamente pueden efectuarse acciones relacionadas con el manejo de las transacciones bancarias en línea. Cualquier operación de depósito y retiro queda registrada por el sistema. Cualquier operación de saldo es satisfecha con los datos de las cuentas sin dejar registro alguno.

En la implantación actual cuando se ejecuta el sistema se apodera de todas las terminales de la micro, una de las cuales actúa como consola. En ésta aparecen diagnósticos producidos cuando ocurre algún error o falla. También aquí se puede ejecutar algún programa que esté monitoreando el funcionamiento del sistema. En las demás sólo pueden efectuarse transacciones.

Cuando se termina una sesión de trabajo, el sistema se da de baja a través de la consola. Después de esto, Cromix toma de nuevo el control total de la micro, con lo cual puede operar en su modo normal de trabajo. En este momento pueden ejecutarse cualquiera de los programas descritos en la sección 6.3 para llevar a cabo algún respaldo, generar listados, etc.

Este proceso se repite con cada sesión de trabajo.

6.2 TIEMPOS DE RESPUESTA

Con el sistema local ya implantado se efectuaron mediciones experimentales de los tiempos promedio de respuesta del mismo a las diferentes transacciones de saldo, depósito y retiro. Estas mediciones se efectuaron en forma indirecta a través del comando de Cromix: PS -AL, el cual despliega en terminal el status de los procesos que se encuentran corriendo en ese momento. Entre la información listada por ese comando se encuentra el tiempo empleado en ejecución por un proceso, el cual se muestra con una precisión de milésimas de segundo.

Por medio de este comando se llevaron a cabo las mediciones que se muestran a continuación. En la figura 6.1 se muestra la gráfica del tiempo promedio de respuesta dado por el sistema local para realizar una transacción de saldo. En la gráfica aparecen los tiempos medidos cuando el sistema atiende a 1, 2 y 3 terminales simultáneamente. En cada caso se efectuaron 15 mediciones de tiempos para obtener este tiempo promedio de respuesta.

En la figura 6.2 se muestra la gráfica obtenida cuando se realiza una transacción de depósito. Esta gráfica también vale para las operaciones de retiro ya que en ambos casos se realiza el mismo tipo de procesamiento; la única diferencia consiste en que en los depósitos se registra una cantidad positiva y en los retiros, una negativa. Los tiempos mostrados corresponden a mediciones tomadas cuando el sistema atiende a 1, 2 y 3 terminales simultáneamente. En total se efectuaron 45 mediciones en cada caso para obtener el tiempo promedio de respuesta.

En ambos casos solo se tomaron tiempos para 3 terminales como máximo debido a que la otra disponible se empleaba, precisamente, para medirlos.

Los tiempos mostrados en las gráficas son la suma global promedio de los tiempos individuales empleados por DRVTT, SIST, EXEC y DRVDSK para el procesamiento de una transacción local.

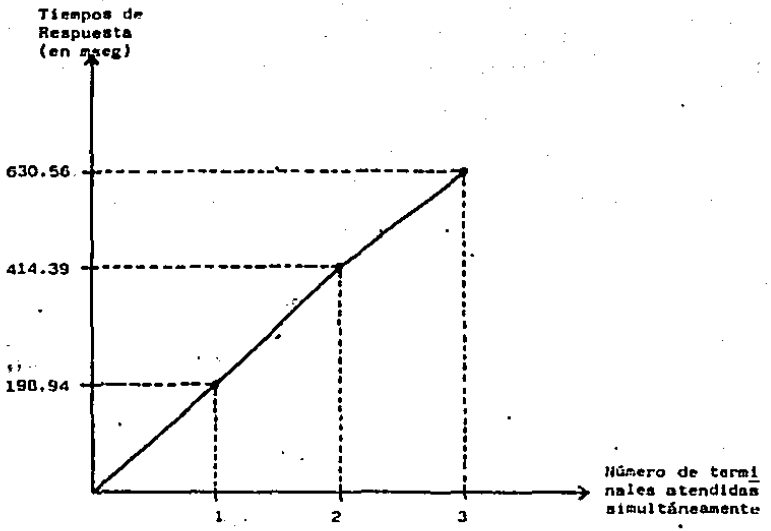


Fig. 6.1 Tiempos de respuesta en el procesamiento de un saldo.

De las gráficas se pueden sacar las siguientes conclusiones:

- el tiempo de respuesta del sistema cuando se procesa un saldo es de 198.94 mseg, en promedio;
- el tiempo de respuesta a una transacción de depósito (o retiro) es de 301.86 mseg, en promedio;
- con respecto a los saldos, se observa que conforme se atiende a más terminales (transacciones) simultáneamente, se va agregando un tiempo de 17 mseg, en promedio, por cada terminal adicional. Esto trae como consecuencia, extrapolando, que al atender a 15 terminales simultáneamente, el tiempo aproximado de respuesta es de 3.222 segundos;
- con respecto a los depósitos (o retiros) al atender a más terminales se aumenta un tiempo promedio de 19 mseg por terminal adicional. Si se atienden simultáneamente a 15 terminales, el tiempo aproximado de respuesta es de 4.793 segundos.

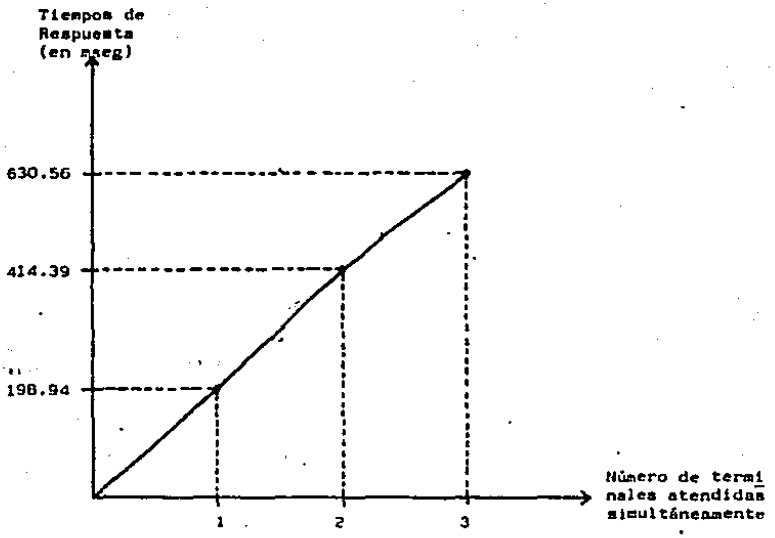


Fig. 6.1 Tiempos de respuesta en el procesamiento de un saldo.

De las gráficas se pueden sacar las siguientes conclusiones:

- el tiempo de respuesta del sistema cuando se procesa un saldo es de 198.94 mseg, en promedio;
- el tiempo de respuesta a una transacción de depósito (o retiro) es de 301.86 mseg, en promedio;
- con respecto a los saldos, se observa que conforme se atiende a más terminales (transacciones) simultáneamente, se va agregando un tiempo de 17 mseg, en promedio, por cada terminal adicional. Esto trae como consecuencia, extrapolando, que al atender a 15 terminales simultáneamente, el tiempo aproximado de respuesta es de 3.222 segundos;
- con respecto a los depósitos (o retiros) al atender a más terminales se aumenta un tiempo promedio de 19 mseg por terminal adicional. Si se atienden simultáneamente a 15 terminales, el tiempo aproximado de respuesta es de 4.793 segundos.

Lo anterior da una idea de qué tan rápido responde el sistema cuando se procesa una transacción local ya sea de saldo, depósito o retiro.

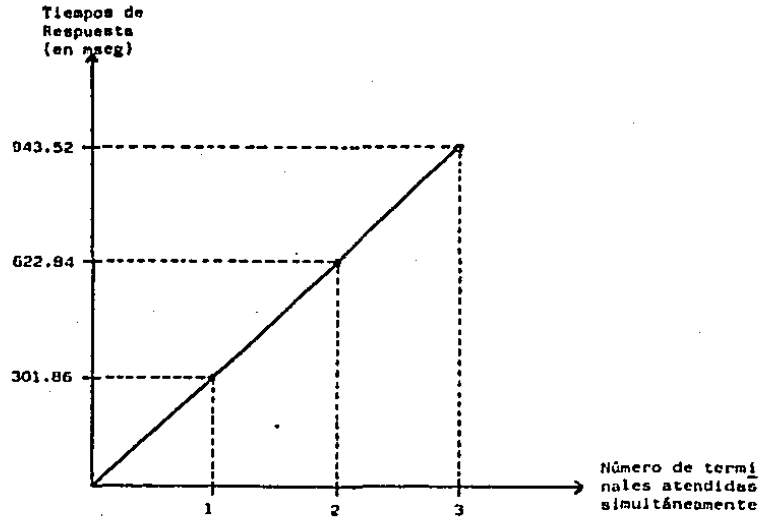


Fig. 6.2 Tiempos de respuesta en el procesamiento de un depósito (o retiro).

A continuación se comentan algunas mejoras que se pueden hacer al sistema para dar tiempos de respuesta aún más cortos:

a) Mediciones directas desde los programas.- Esto, mas que mejora, representa una alternativa para medir más precisamente los tiempos de respuesta del sistema local. Como se comentó anteriormente las mediciones de los tiempos se hicieron en forma indirecta y no desde los propios módulos. El uso de este comando introdujo un margen de tiempo adicional, el cual se cuantificó hasta cierto punto, aunque quedó un margen de incertumbre acerca de la exactitud de esta cuantificación. Las mediciones no se hicieron directamente desde los módulos debido a que con las llamadas a Cromix, para obtención de tiempo, sólo se logran precisiones máximas de segundos lo que no sirve para determinar los tiempos de respuesta. También se trató de programar varios de los timers de la micro para medir estos tiempos sin tener éxito, ya que al pasar

de un módulo a otro se perdía el control de estos timers.

Es casi seguro que los tiempos mostrados en las gráficas son mayores a los tiempos de respuesta reales del sistema, aunque no se pudo cuantificar exactamente en qué proporción son mayores.

b) Utilización del MC68000.- En la sección anterior se mencionó que el micro usado por el sistema es el Z80. La microcomputadora Sistema 3 posee también otro micro que es el MC68000 (descrito en el apéndice B). Si se hubiera empleado este micro para ejecutar el sistema, se habría ganado en cuanto a tiempo de ejecución ya que el MC68000 es, por lo menos, dos a tres veces más rápido que el Z80, según se observa en la tabla 6.2 (obtenida de un manual de Cromemco [CROM81]).

1. Averaged Benchmarks

	Z8002/68000	8086-1/68000	LSI11/68000
TIME	1.71	2.16	5.08
BYTES	1.20	1.57	1.75
LINES	1.23	2.17	1.68

Number shown is ratio of other processor's performance to MC68000 performance

2. Digital Filtering Application
V.P Nelson & B.T. Nagel, Jr.
Auburn University

Time Lag from Input to Output, in Microseconds:

MC68000	82.25
Z8000	156.25
8086	212.8
TMS9900	253.9

3. Blacksburg Group Inc. Benchmark,
from 16-Bit Microprocessor Handbook

Processor	Initialization (mS)	Sort (Sec.)	String Search (uS)	Square Root (uS)
8086 (5 MHz)	4.683	6.854	375	312.236
Z8002 (4 MHz)*	3.450	4.727	225	134/187
LSI/11	0	10.500	979	457/628
TMS9980 (2 MHz)	0	33.000	2250	680/860
MC68000 (8 MHz)	1.581	2.016	424	88/121

*All times are calculated

0 Initialization time part of Sort time

Tabla 6.2 Tiempos de respuesta del microprocesador MC68000 vs. tiempos de respuesta de otros micros.

Aunque en la tabla aparece el micro Z8000 y el utilizado para implantar el sistema fue el Z80, se puede extrapolar fácilmente que el sistema respondería más rápido si se utilizara el MC68000 en lugar del Z80 para su ejecución.

No se usó este micro debido a que el compilador C del cual se disponía sólo genera código para el Z80.

c) Programación del IOP y Quadart.- La entrada/salida con terminales se manejó utilizando llamadas que proporciona Cromix para la gestión de entrada/salida con cualquier periférico. Esto agrega un "overhead" al procesamiento de las transacciones, el cual no fue posible cuantificar. Si se hubieran programado directamente, a nivel de ensamblador, tanto el IOP como el Quadart (descritos en el apéndice B) para el manejo de las comunicaciones con las terminales, se habría ganado también en el tiempo de respuesta como sucedió con el manejador del diskette. Esta programación no se hizo debido a que se consideró que el "overhead" agregado no era tan grande como en el caso del diskette, ya que aquí no se maneja información estructurada, y la inversión, en tiempo y esfuerzo, que debía hacerse en programación no redituaba en un beneficio considerable en los tiempos de respuesta.

d) Utilización de disco fijo.- El equipo utilizado cuenta con un disco fijo tipo Winchester. Toda la información manejada por el sistema local actualmente se maneja con la unidad de diskettes del equipo. Empleando la unidad de disco fijo para este propósito, hubiera traído como consecuencia tiempos de respuesta más cortos para las transacciones de depósito y retiro, ya que el acceso a estos discos es más rápido. No se usó debido a que contenía información importante para la Institución propietaria del equipo y para no interferir con la misma se optó por no utilizarlo.

6.3 PROGRAMAS AUXILIARES

Como parte del sistema local se desarrollaron una serie de programas auxiliares usados para efectuar operaciones que, por su naturaleza, no requieren realizarse en línea.

Todos estos programas se escribieron en lenguaje C, utilizando también algunas de las llamadas a Cromix para realizar sus funciones. La mayoría utiliza al módulo DRVDSK ya que, en general, deben llevar a cabo operaciones que manejan la información almacenada en el diskette. En general, la primera acción que efectúan estos programas es, precisamente, ordenar la ejecución de DRVDSK para que puedan accesar la información del diskette.

Todos los programas se ejecutan en forma independiente y corren bajo Cromix cuando el sistema local no está activo (excepto EMERG). Sus funciones no requieren respuestas en tiempo real por lo cual no es necesario que se ejecuten bajo el control de EXEC.

A continuación se describen las principales funciones efectuadas por estos programas.

1) Programa CREAR
LLeva a cabo la creación de los archivos usados por el sistema local, definiendo su estructura así como el contenido inicial de los mismos. Por su naturaleza, este programa sólo se ejecuta una vez para definir la estructura inicial de los archivos.

2) Programa TERMS
Este programa se utiliza para listar y modificar el contenido del archivo de terminales en línea. Entre las funciones que realiza se encuentran las siguientes:

- imprime el número de terminales registradas en la sucursal;
- permite modificar este número;
- permite activar terminales (o sea, ponerlas en línea);
- permite desactivarlas;
- lista los números de las terminales desactivas.

3) Programa IDSUC
Permite listar y modificar el contenido del archivo de identificación de la sucursal. Con este programa se puede modificar cualquiera de los campos de este archivo. Se usaría cuando hubiera algún cambio en la configuración de la red que afectara a la sucursal.

4) Programa ACT
Se utiliza para modificar el archivo maestro. Las principales funciones que lleva a cabo son:

- dar de alta a clientes, solicitando nombre, dirección, beneficiario, etc., y guardando esta información en el archivo maestro. Al dar la alta, asigna un número de cuenta al cliente y afecta la cuenta correspondiente en el archivo directorio para indicar que está en uso;
- dar de baja a clientes, con lo cual se borra la información de éstos y se liberan los registros ocupados tanto en el archivo maestro como en el directorio;
- modificar información de clientes, permitiendo alterar cualquier campo del registro de un cliente;
- cancelar o suspender temporalmente la cuenta de un cliente, con lo cual no pueden efectuarse temporalmente transacciones con la misma;
- rehabilitar una cuenta cancelada o suspendida temporalmente.

5) Programa LISTC
Genera listados con los datos de los clientes registrados en el archivo maestro. Presenta las siguientes opciones para la generación de los listados:

- todos los clientes de la sucursal;
- clientes en un rango de cuentas;
- clientes con saldo promedio (al mes anterior) menor que un mínimo permitido;
- clientes con cuenta cancelada (temporalmente);

- clientes con cuenta suspendida (temporalmente);
- clientes dados de baja;
- consulta de datos de clientes (por pantalla).

6) Programa TDM
Transfiere la información almacenada en el archivo de transacciones diarias al de transacciones mensuales. El programa se ejecuta al final de una sesión de trabajo. Al hacer la transferencia, coloca primeramente en el de mensuales un registro con la fecha de la sesión, para indicar que la información que viene a continuación se registró en dicha fecha. Con esto, en el archivo de mensuales se van acumulando las transacciones efectuadas en el mes, para que, al final del mismo, se generen los estados de cuenta de los clientes. Una vez efectuada la transferencia, el programa pone a ceros los registros del archivo de diarias con el fin de que queden listos para recibir la información generada en la siguiente sesión de trabajo.

7) Programa ESTC
Genera los estados de cuenta mensuales corrspondientes a cada cliente registrado en el archivo maestro. Con la información acumulada en el archivo de transacciones mensuales, genera para cada cliente un estado de cuenta que contiene: todas las transacciones de depósito y retiro efectuadas por el mismo durante el mes, el saldo al mes anterior y el actual, el estado de su cuenta y los datos generales del cliente. Una vez generados los estados de todos los clientes de la sucursal, el archivo de mensuales se limpia para que quede listo para acumular las transacciones del siguiente mes.

8) Programa EMERG
Es un programa auxiliar que se emplea para modificar o desplegar en pantalla la información de un sector específico del diskette. Recibe como datos la superficie, pista y número del sector a accesar, así como la operación a efectuar. Se emplea cuando una sucursal deja de operar o queda trabajando sólo localmente. Con el se accesan directamente registros del diskette para modificarlos a fin de resolver alguna transacción global, de depósito o retiro, que haya quedado inconclusa al ocurrir la falla.

9) Programa RESPL
Efectúa un respaldo en disco fijo de toda la información almacenada en el diskette. El respaldo se efectuaría al finalizar una sesión de trabajo para guardar el estado de los archivos hasta ese momento.

10) Programa RESCT
Realiza la operación inversa de la anterior para rescatar la información almacenada en el disco fijo.

CONCLUSIONES

En las siguientes lineas se anotan algunos comentarios sobre aspectos importantes tratados en cada capitulo de este trabajo, para al final dar algunas conclusiones generales sobre todo el sistema.

Red de Comunicación

El diseño modular de la red permite que ésta pueda crecer facilmente, sin importar la cantidad de nodos y sucursales que existan en el sistema. Obviamente hay un limite (99 nodos y 990 sucursales, máximo) ya que no se puede crecer hasta el infinito. Sin embargo, dentro de estos limites, el crecimiento es más o menos directo y los cambios que se deben hacer, tanto en hardware como en software, al hacer una expansión, son llevados al mínimo posible. Existe gran flexibilidad en el crecimiento del sistema.

Software de Control de la Red

Los algoritmos de L2S y L3S cumplen con las funciones requeridas para los protocolos de las capas 2 y 3 de la red. En el caso de L2S es importante señalar que la transmisión/recepción de paquetes con una micro adyacente se deja completamente al procesador frontal. Este es el encargado de vigilar que no se pierda ningún byte en la transmisión/recepción de los mismos. L2S sólo debe controlar la cantidad de paquetes enviados y recibidos. O sea, la parte del hardware juega un papel muy importante en el control del envio y recepción de los paquetes de información.

El algoritmo de EXL3 permite deducir que no hay un nodo en particular que tenga el control de toda la red. Puede ser que en un cierto nodo se de inicio y fin a una sesión de trabajo, pero una vez hecho esto, no se centraliza el control en ningún nodo de la red. Al no existir centralización, se asegura una disponibilidad casi completa del sistema, ya que ante la falla de cualquier nodo, o sucursal, el resto puede seguir operando.

Sistema Local

Con respecto al sistema local se puede anotar lo siguiente: las funciones que realiza EXEC son las de un Supervisor (o monitor) el cual controla la ejecución de los demás programas de la micro, sincroniza las comunicaciones con ellos, les cede el control del procesador para que se ejecuten, etc. Utiliza las facilidades que proporciona Cromix para elaborar programas de esta naturaleza: creación de procesos, empleo de pipes para comunicación inter-procesos, intercambio de información con simples ordenes de lectura/escritura, etc.; todo esto a través de llamadas al sistema operativo. No es propiamente un ejecutivo ya que, por ejemplo, no administra memoria; ni un kernel ya que no proporciona primitivas de sincronización y comunicación, por ejemplo; sin embargo las funciones que lleva a cabo se enmarcan dentro de las que éstos realizan. EXL3 se inscribe también dentro de esta idea, aunque además define rutas y controla el tráfico de la información que pasa por un nodo.

De SIST es importante señalar las labores de paralelismo que realiza para poder procesar varias transacciones, tanto locales como globales, al mismo tiempo.

Recuperación de Fallas y Seguridad

En lo referente a recuperación de fallas y errores no se trataron todas las posibles fallas que pueden ocurrir en la operación del sistema, ya que son muchas. Sólo se tocaron las más importantes, las cuales de una u otra forma afectan en forma relevante el funcionamiento del sistema cuando se presentan. En la mayoría de los casos los procedimientos de recuperación son automatizados y las propias micros, de nodo o de sucursal, son las encargadas de llevarlos a cabo. En algunos casos se proponen métodos manuales de recuperación ya que hacerlo de otra manera resultaría prácticamente imposible o tendería a producir un sistema demasiado complejo.

Con respecto a la seguridad se describen mecanismos de protección para garantizar tanto la integridad de la información como el no acceso a personas ajenas al sistema.

Implantación del Sistema Local

El tiempo de respuesta dado por el sistema local se inscribe dentro de las características de tiempo real, ya que procesar una transacción de saldo en 198.94 mseg y una de depósito o retiro en 301.86 mseg, da indicios de un sistema relativamente rápido.

Comentarios Generales sobre el Sistema

Los dos aspectos de mayor relevancia de este trabajo son: el uso de microcomputadoras para soportar la operación del sistema y la distribución de la información de los clientes entre todas estas micros con el consiguiente procesamiento distribuido de la misma.

El uso de micros multiusuario para atender a las diferentes sucursales y nodos de la red, produce un sistema que no es excesivamente caro ni complejo, que cumple con el objetivo de procesar las diferentes transacciones que realizan los clientes en todas las sucursales del sistema. Con el empleo de una micro más poderosa que la usada en este trabajo (p. ej., una MicroVax), con mayor capacidad de almacenamiento en discos, se haría más factible y atractiva la implantación de un sistema de este tipo, a la vez que se obtendrían tiempos de respuesta menores en el procesamiento de las transacciones.

El manejar en forma distribuida la información concerniente a los clientes, trae consigo las ventajas y desventajas de este tipo de procesamiento.

Por un lado se pueden citar las siguientes ventajas:

- al no tener centralizada la información se evita emplear grandes máquinas, con gran capacidad de almacenamiento, necesarias para dar satisfacción a todas las transacciones efectuadas por los clientes. Asimismo, se evita tener por duplicado a estas máquinas lo cual sería necesario para prevenir el caso de que la máquina central dejara de funcionar. Aún así es frecuente observar en los sistemas bancarios que en los días de mayor carga de trabajo -días de quincena, viernes, etc.-, no se hacen transacciones, o se hacen con cantidades restringidas a un cierto límite, por falla del sistema central. Esto prácticamente paraliza las actividades del sistema automatizado, obligando a los empleados a efectuar las transacciones en forma manual;
- con el sistema descentralizado se asegura una disponibilidad casi total del mismo. Si falla un nodo, afecta temporalmente a las sucursales asociadas a él, pero el resto de las sucursales sigue operando. Si una sucursal queda desligada de la red, por falla en su línea de transmisión, puede seguir operando localmente satisfaciendo transacciones con las cuentas registradas en la misma. Esto es, la falla de un nodo, o sucursal, no ocasiona que el resto del sistema automatizado quede inactivo como en el caso anterior.

Como posibles desventajas se pueden mencionar las siguientes:

- al tener la información descentralizada es necesario introducir más mecanismos y procedimientos de control, tanto administrativos como de supervisión, que en un sistema centralizado. Esto se hace necesario para garantizar tanto el funcionamiento adecuado del sistema como la integridad de

la información que éste maneja;
- relacionado con lo anterior, se deben agregar más mecanismos de seguridad y protección para evitar que se puedan efectuar transacciones fraudulentas a través de la red. Bajo este esquema existen más puntos vulnerables en el sistema a través de los cuales se puede accesar al mismo sin autorización. Algunos aspectos de esta naturaleza se comentaron en la sección 5.2, aunque faltó ahondar un poco más en el tema. Por ejemplo, se podrían haber establecido diferentes grupos de usuarios, con diferentes niveles de privilegios, de tal manera que cada grupo pudiera accesar sólo cierta parte del sistema.

Este trabajo representa, simplemente, una alternativa a procesos que ya se realizan actualmente aunque en forma centralizada.

BIBLIOGRAFIA

[BECK85] Beck, L. E. "SYSTEM SOFTWARE an introduction to systems programming". Addison-Wesley. 1985.

[CHOR80] Chorafas, D. "Data Communications for Distributed Information Systems". Petrocelli Books, Inc. 1980.

[CROM81] Cromemco. "Manuales varios de Cromemco". Cromemco, Inc. 1981, 1982 y 1983.

[HABE76] Habermann, A. N. "Introduction to Operating System Design". Science Research Associates, Inc. 1976.

[HANS73] Hansen, P. B. "Operating System Principles". Prentice-Hall. 1973.

[HANS77] Hansen, P. B. "The Architecture of Concurrent Programs". Prentice-Hall. 1977.

[HOLT78] Holt, R. C.; Graham, G. S.; Lazowska, E. D. y Scott, M. A. "Structured Concurrent Programming with Operating Systems Applications". Addison-Wesley. 1978.

[KERN76] Kernighan, W. B. y Plauger, P. J. "Software Tools". Addison-Wesley. 1976.

[KERN85] Kernighan, W. B. y Ritchie, D. M. "El Lenguaje de Programación C". Prentice-Hall Hispanoamericana. 1985.

[KING83] King, J. L. "Centralized versus Decentralized Computing: Organizational Considerations and Management Options". Computing Surveys. December 1983.

[MART81] Martin, J. "Computer Networks and Distributed Processing". Prentice-Hall. 1981.

[NICH82] Nichols, E.; Nichols, J. y Musson, K. "Data Communications for Microcomputers". McGraw Hill. 1982.

[QUAR86] Quarterman, J. S. y Hoskins, J. C. "Notable Computer Networks". Communications of the ACM. October 1986.

[STAL85] Stallings, W. "Data and Computer Communications". Macmillan Publishing Company. 1985.

[WEIT80] Weitzman, C. "Distributed Micro/Minicomputer Systems". Prentice-Hall. 1980.

[WICK82] Wicklund, T. L. "MINI-EXEC: a Portable Executive for 8-bit Microcomputers". Communications of the ACM. November 1982.

APENDICE A

INTERCAMBIO DE INFORMACION CON EL PROTOCOLO HDLC

En esta parte se describe a grandes rasgos el proceso que se efectúa con el protocolo HDLC para intercambiar información entre dos maquinas conectadas a la misma linea de comunicación [MART81].

Bajo este protocolo la maquina que transmite se conoce como "estación primaria" y la que recibe como "estacion secundaria". Una estación secundaria normalmente opera en uno de dos modos: normal o asincrono. En el primer modo, la secundaria puede transmitir solamente en respuesta a un mensaje (que denominaremos, comando) enviado por la primaria. El mensaje (que llamaremos, respuesta) que transmite la secundaria puede consistir de uno o más paquetes y debe indicar cuál es el último de ellos. No puede transmitir otra vez, sino hasta que reciba otro comando. La estación primaria es responsable por time-outs, retransmisión y todas las formas de recuperación de errores.

En el modo de respuesta asincrono el secundario puede iniciar en cualquier momento la transmisión de uno o más paquetes. Estos pueden contener información o ser enviados sólo para propósitos de control. El secundario es responsable por time-outs y retransmisión si uno de los paquetes no es recibido correctamente.

En una red como la propuesta es deseable tener altos volúmenes de tráfico viajando en ambas direcciones entre las micros. Para alcanzar mayor eficiencia, este protocolo define lo que se conoce como "Estación Balanceada". Aqui cada estación asume ambos papeles: primaria y secundaria. Todas las estaciones tienen el mismo conjunto de protocolos. Cualquier estación puede recibir tanto comandos como respuestas. Cualquier estación puede iniciar la transmisión en cualquier momento. Las estaciones tienen igual responsabilidad para la recuperación de errores. Este tipo de operación se conoce como "Modo Balanceado" y es el modo normal de operación entre las micros de la red.

Campo de Control.- Cada paquete HDLC contiene un campo de control (un byte) que sirve para indicar el tipo de paquete que se está transmitiendo, así como para indicar los números de secuencia de los paquetes transmitidos. Los intercambios entre nodos son controlados a traves de estos números. Usualmente son de tres bits (y opcionalmente de 7 bits, por medio de extensiones que no se discuten aquí) permitiendo contar de 0 a 7 y de nuevo desde 0.

Como se mencionó en la sección 2.3.2 existen 3 tipos de paquetes HDLC, de los cuales se usan dos en este diseño: Tipo I (denominado, paquete I) y Tipo S (denominado, paquete S). Cada paquete I enviado es identificado por el número Ns (ver fig. 2.7). La recepción de varios paquetes I es confirmada enviando una aceptación que contiene el número de secuencia (Nr) del siguiente paquete I que la estación receptora espera recibir. Esto implica que todos los paquetes previos a Nr han sido recibidos correctamente. Los paquetes S, que pueden llevar aceptaciones, también tienen el Nr en sus últimos tres bits. Las aceptaciones también pueden ser llevadas por un paquete I de respuesta a un comando. El paquete I lleva entonces dos números de secuencia: su propio número, Ns, y el número que indica la aceptación de los paquetes recibidos en esa estación, Nr.

Control del Intercambio de Información.- Cuando una estación -primaria o secundaria- envía paquetes I, puede enviar de uno hasta siete paquetes; el último en el grupo debe tener el bit P/F -del byte de control- en 1. Cuando el paquete es un comando enviado por un primario, este bit es llamado "Bit de Poll" y solicita una respuesta del secundario direccionado. Cuando el paquete es una respuesta enviada por un secundario el bit es llamado "Bit de Final" y solicita una aceptación del primario (la cual puede venir, a su vez, en un paquete I). El bit P/F indica entonces que una estación espera una respuesta antes de transmitir otro paquete. La figura A.1 muestra situaciones en las cuales una estación envía datos y la estación receptora responde con más datos.

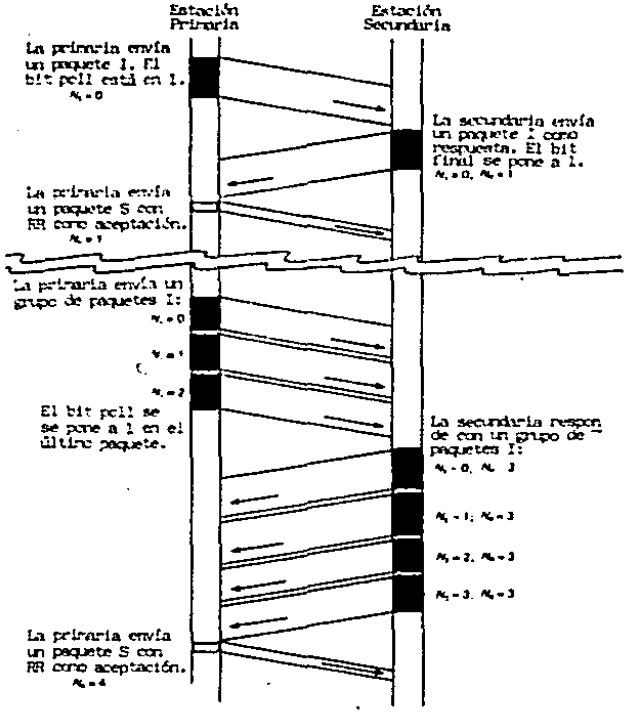


Fig. A.1 Intercambio de Información entre dos Estaciones.

Operación en Full-Duplex.- En la red propuesta es deseable tener flujo de datos en ambas direcciones sobre la línea de transmisión. La figura A.2 muestra mensajes de longitud variable viajando en ambas direcciones, tal como sucede en la red aquí considerada.

El flujo en un sentido es independiente del flujo en el otro. Cualquier estación puede ser primaria. En la figura se observa que un error ocurre en el paquete Nr=6. Cuando el error ocurre, la estación que recibe este paquete detecta el error y no actualiza su contador de Nr. En los paquetes que envía sigue indicando que el siguiente paquete esperado es el 6. Después de un tiempo pre-establecido, la estación que originó el paquete dañado lo envía de nuevo, junto con todos los paquetes que le siguen.

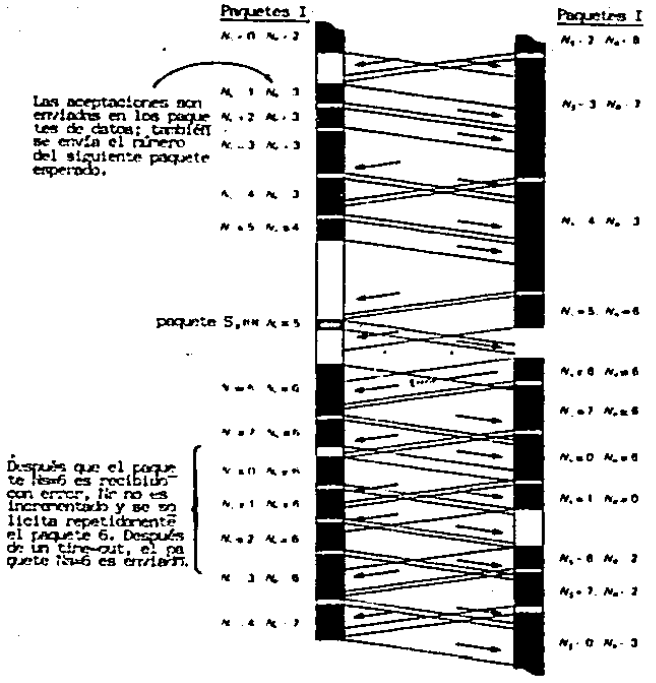


Fig. A.2 Intercambio de Información en Full-Duplex.

Existen dispositivos que permiten manejar este tipo de protocolos a nivel de hardware. El usuario únicamente se encarga de controlar el tipo de paquete que se va a enviar (I o S) y de la información que se va a transmitir en cada paquete. El dispositivo controla el MARCO, CRC, transmisión síncrona y todo aquello que es controlable por hardware.

El proceso descrito en la operación en full-duplex, es el usado en este diseño para controlar el intercambio de información entre dos micros adyacentes en la red.

APENDICE B

CARACTERISTICAS DE LA MICROCOMPUTADORA SISTEMA 3

En el presente apéndice se muestra un panorama general de la organización de la microcomputadora SISTEMA TRES (de Cromemco) [CROMA1]. Esta micro es la que se utilizó para desarrollar el sistema local presentado en este trabajo. En el apéndice se muestran a grandes rasgos los principales elementos de hardware y software que componen a la micro. La descripción de estos elementos se hace enmarcándolos dentro del trabajo aquí presentado.

B.1 COMPONENTES DEL HARDWARE

En la figura B.1 se presenta un esquema general de la arquitectura de esta micro en el cual se muestran los principales componentes de la misma.

A continuación se da una breve descripción de cada uno de estos elementos.

a) Procesador central CPU.- El procesador central de esta computadora está constituido por dos microprocesadores: el Zilog Z80A de 8 bits y el Motorola MC68000 de 16 bits. El MC68000 es el que se usa principalmente para soportar todas las operaciones de la micro. Por ejemplo, el sistema operativo Cromix que usa esta computadora se ejecuta en su mayor parte en el MC68000.

El Z80A se proporciona por cuestiones de compatibilidad para poder ejecutar en esta micro todo el software disponible para el mismo. La micro proporciona todo el control interno -tanto en hardware como en software- para poder ejecutar programas en uno u otro microprocesador en forma totalmente transparente al usuario.

El sistema local presentado en esta tesis está implantado usando como soporte el Z80A, ya que el compilador (de lenguaje C) empleado para la traducción de los programas del sistema, genera únicamente código para el Z80.

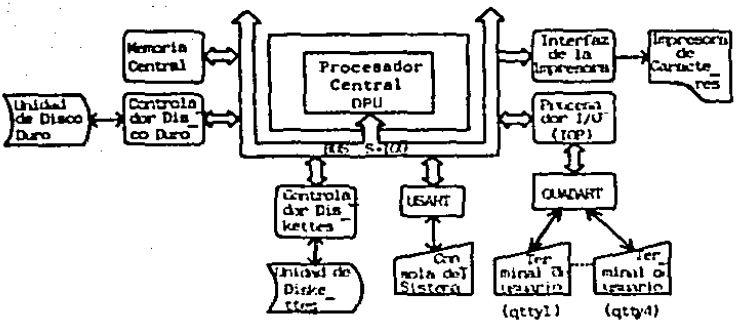


Fig. B.1 Arquitectura de la microcomputadora Sistema 3.

El usar uno u otro microprocesador no influyó en el diseño del presente trabajo ya que el control de ambos lo lleva la propia computadora. Afectó la implantación del sistema local en lo concerniente al tiempo de respuesta en el procesamiento de una transacción, ya que el Z80A es más lento que el MC68000.

b) Memoria central.- Tiene un tamaño actual de 512 Kbytes pudiendo ser extendida hasta 16 Mbytes (teóricamente). La memoria se divide en bloques de 64 Kb cada uno. Cada bloque se dice que es un banco de memoria y se numeran a partir de 0. Los bancos 0 y 1 (128Kb) están reservados para el sistema operativo Cromix. El banco 0 almacena la parte de Cromix ejecutada bajo el MC68000; el banco 1, la parte controlada por el Z80A. Lo anterior deja un total de 6 bancos libres para ejecutar programas.

Cada programa que está siendo ejecutado ocupa un banco de memoria. Esto daría un máximo de 6 programas que estuvieran ejecutándose al mismo tiempo. Sin embargo, existe una opción, al ligar un programa, que permite que un mismo banco sea ocupado por tantos programas como quepan en el mismo. Esta opción permite que el sistema local aquí descrito se pueda expandir fácilmente para manejar un máximo de 15 terminales simultáneamente.

c) Controlador y unidad de disco fijo.- Manejan un disco fijo de tecnología Winchester con capacidad de 20 Mbytes. El disco se utiliza para almacenar el sistema operativo Cromix, los programas producto de este trabajo y programas y archivos de otros usuarios.

No influye en el sistema presentado por lo que no se dan mayores detalles del mismo.

d) Impresora de caracteres.- Es una impresora ATI de baja velocidad. No afecta el diseño del sistema.

e) Controlador 64FDC y unidad de diskettes.- La unidad maneja diskettes de 8", doble lado, doble densidad, con una capacidad de 1.2 Mbytes. Se utiliza para almacenar en un diskette los archivos manejados por el sistema. Juega un papel importante en este trabajo por lo que en la sección B.1.1 se da una explicación más detallada del funcionamiento del controlador. En la configuración actual sólo existe un controlador y una unidad.

f) Consola del sistema.- Es una terminal de video a través de la cual se da de alta y baja a Cromix. Por medio de esta terminal se inicia la ejecución del sistema de transacciones. Cuando éste se encuentra corriendo, esta terminal se emplea como consola del mismo. Se reconoce con el nombre de TTY1.

g) Procesador de Entrada/Salida y Quadart.- Se utilizan para controlar las demás terminales de la microcomputadora. El IOP es un procesador frontal pequeño que controla el intercambio de información entre el CPU y las terminales. El Quadart es un multiplexor para las terminales. Actualmente la micro tiene conectado un IOP y un Quadart. En la sección B.1.2 se describen con más detalle ya que también son importantes en este trabajo.

h) Terminales de usuario.- Son terminales de video. Se reconocen con los nombres de QTTY1 a QTTY4. En el sistema local actúan como terminales de usuario a través de las cuales pueden efectuarse transacciones bancarias. Actualmente se tienen cuatro terminales.

B.1.1 Controlador de la Unidad de Diskettes

Todos los archivos manejados en el sistema local de transacciones se encuentran almacenados en el diskette de 1.2 Mb. Para reducir el tiempo de respuesta del sistema al manejar una transacción es accesado directamente sin pasar por el sistema de archivos de Cromix. Así, el controlador de la unidad, llamado 64FDC, está programado directamente a nivel de ensamblador Z80. En la sección 4.6 se describió en detalle el módulo que programa estos componentes.

El 64FDC posee un conjunto de registros que pueden ser programados para accesar directamente el diskette. Posee además una serie de registros que permiten efectuar transmisión asíncrona de caracteres, que le permiten actuar como interfaz entre el DPU y una terminal. Es por medio de estos registros que se controla a la consola del sistema. También posee registros para programar timers que producen señales de interrupción en periodos de hasta 16.32 mseg. El conjunto total de registros se muestra en la tabla

B.1 junto con las direcciones de los puertos asignados a estos, dentro de todo el sistema de la microcomputadora.

PUERTO	ENTRADA	SALIDA
00	Registro de Estado UART	Baudaje UART
01	Registro Receptor de Datos UART	Datos UART
02	No Asignado	Comandos UART
03	Dirección Rutina de Servicio	Máscara de Interrupción
04	Registro de Estado Auxiliar	Comandos Auxiliar
05	No Asignado	Timer 1
06	No Asignado	Timer 2
07	No Asignado	Timer 3
08	No Asignado	Timer 4
09	No Asignado	Timer 5
30	Registro de Estado Diskette	Comandos Diskette
31	Pista Diskette	Pista Diskette
32	Sector Diskette	Sector Diskette
33	Datos Diskette	Datos Diskette
34	Banderas Diskette	Control Diskette
40	No Asignado	Selector Banco de Memoria

Tabla B.1 Registros del controlador 64FDC.

A continuación se describen las principales características de los registros directamente relacionados con el manejo de la unidad. Cada registro es de 8 bits; estos bits se numeran de la forma D0 a D7, siendo D0 el bit menos significativo.

a) Registro de comandos auxiliares (Salida).- Se utiliza sólo el bit D1 para indicar el lado que va a ser accesado en diskette. Un 1 selecciona el lado 0; un 0 selecciona el lado 1.

b) Registro de comandos (Salida).- En el se guarda el comando que va a ejecutar como siguiente operación el 64FDC. Los principales Comandos que puede ejecutar son:

RESTORE - coloca la cabeza de lectura/escritura en la pista 00 del diskette.
SEEK - coloca la cabeza de grabación en la pista indicada por medio del registro de datos.
STEP - avanza a la siguiente pista en la dirección en la cual se haya efectuado el último movimiento de la cabeza de grabación.
READ RECORD - lee un registro (512 bytes) del diskette. En doble densidad existe un lapso de 14 microsegundos para leer cada dato del registro. Si en este tiempo no se lee el dato, se pierde y toda la operación es marcada como errónea.
WRITE RECORD - escribe un registro (512 bytes) en el diskette. Las condiciones para la escritura de un dato son las mismas que en

el caso de una lectura.

Todos estos comandos se dan por medio de combinaciones pre-definidas de 0's y 1's que deben ser guardadas en este registro.

c) Registro de estado (Entrada).- Indica el "status" de una operación después que ésta se ha ejecutado. El estado que representan los bits varía de acuerdo al último comando ejecutado, según se muestra en la tabla B.2.

Ultimo Comando	D7	D6	D5	D4	D3	D2	D1	D0
SEEK, STEP, o RESTORE	No Listo	Protección Contra Escritura	Cabeza Abajo	No Encontrado	Error CRC	Pista 00	Indice	Ocupado
READ RECORD	No Listo	0	Tipo de Registro	No Encontrado	Error CRC	Dato Perdido	Dato Solicitado	Ocupado
WRITE RECORD	No Listo	Protección Contra Escritura	0	No Encontrado	Error CRC	Dato Perdido	Dato Solicitado	Ocupado

Tabla B.2 Afectación del registro de estado según el último comando ejecutado.

En todos los casos el bit en 1 indica la ocurrencia del evento.

d) Registro de la Pista (Entrada/Salida).- Contiene el número de la pista en la cual se encuentra actualmente la cabeza de grabación. Es incrementado en 1 cada que la cabeza es movida hacia el centro del diskette y decrementado en 1 cuando se mueve en sentido contrario.

e) Registro del sector (Entrada/Salida).- Contiene la dirección (número) del sector que se desea accesar. Cada pista contiene 16 sectores numerados secuencialmente a partir de 1.

f) Registro de datos (Entrada/Salida).- Es usado para almacenar bytes de datos durante las operaciones de lectura/escritura con el diskette. En operaciones de lectura, el byte de datos leído es transferido del diskette a este registro para que de ahí sea accesado por el procesador central. En operaciones de escritura, el procesador central carga este registro para que de ahí sea grabado en el diskette.

g) Registro de control (Salida).- Se emplea para definir la densidad y tamaño, así como el número de la unidad usada para manejarlo. También con este registro se controla la transferencia

de los bytes de datos por medio del bit D7. Cuando se pone un 1 en este bit, el 64FDC entra en modo auto-wait lo cual significa que la siguiente lectura que se haga del registro de banderas, pondrá al CPU en estado de WAIT (el CPU suspende su actividad) hasta que uno de cuatro eventos suceda:
1. El 64FDC emite un DRQ (Data Request). Este es el uso normal de auto-wait.
2. El 64FDC emite un EOJ. Esto termina el auto-wait. Es el método normal de terminación.
3. Se da un reset por hardware.
4. El timer del auto-wait llega a cero (terminación anormal).

Las operaciones de lectura/escritura efectuadas por el módulo DRVDSK se controlan por medio de las situaciones 1 y 2, ya que con esto se sincroniza la transferencia de bytes de datos entre el CPU y el 64FDC.

h) Registro de banderas (Entrada).- Se utiliza sólo el bit D0 (EOJ) el cual se pone en 1 cuando el 64FDC termina de ejecutar un comando. El bit es puesto a 0 cuando se lee el registro de estado.

B.1.2 Procesador de Entrada/Salida y Quadart

El Procesador de Entrada/Salida (IOP) y el Quadart son dispositivos de hardware que se encargan de controlar las comunicaciones entre el procesador central (DPU) y las terminales y/o modems.

El IOP es un pequeño procesador frontal de comunicaciones que se encarga de almacenar y formatear la información recibida desde las terminales/modems para pasarla ya pre-procesada al DPU y, a la vez, recibir de éste información que debe distribuir entre los mismos. El IOP recibe comandos y datos desde el procesador central y le regresa status y datos pre-procesados, descargándolo de la responsabilidad de monitorear y controlar las comunicaciones con la terminales/modems.

El Quadart es una tarjeta que contiene cuatro USART's los cuales permiten conectar hasta cuatro terminales/modems y un IOP. El IOP configura estos cuatro canales seriales, full-duplex, del quadart y ejecuta transferencias de información en paralelo con el mismo. El quadart formatea y almacena datos de entrada/salida, proporciona canales de comunicación, efectúa conversiones de datos paralelo serie y serie-paralelo y genera señales para control de modems. Además el quadart descarga al IOP de ciertas tareas proporcionando por hardware algunas funciones como: generación verificación de CRC al transmitir/recibir paquetes de información, generación/detección de bytes MARCO en paquetes HDLC (sección 2.3.2) y recepción/transmisión controlada al usar modems.

Entre ambos efectúan: labores de monitoreo y conmutación de las líneas usadas para conectar las terminales/modems, concentración de datos, manejo de protocolos y conversión de datos.

En un sistema como el descrito en este apéndice se pueden conectar hasta cuatro quadart's a un solo IOP. La figura B.2 muestra un esquema de esta configuración.

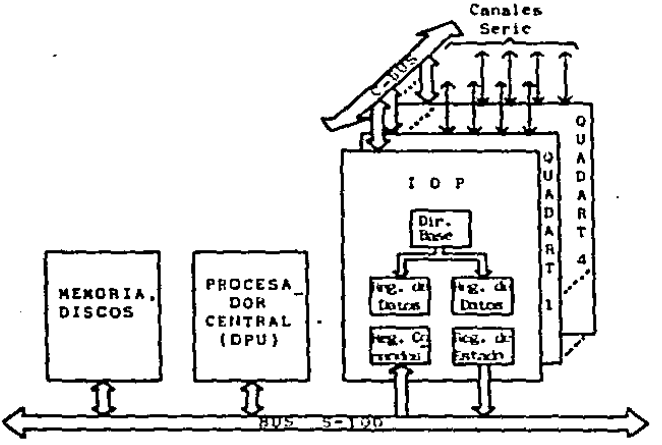


Fig. B.2 Configuración del SISTEMA-3 con cuatro Quadart's y un IOP.

Cada quadart es conectado al IOP por medio de un cable, llamado C-BUS. A través de éste, el IOP intercambia datos y señales en paralelo con los quadart's, sin interferir en nada con la información y señales intercambiadas entre el DPU, memoria, disco y el propio IOP a través del bus S-100 que es el que comunica a todo el sistema.

El DPU se comunica con el IOP a través de dos puertos de entrada/salida que tienen una cierta Dirección Base la cual es reconocida desde el procesador central. El DPU envía bytes al IOP a través del Puerto de Datos y del Puerto de Comandos, y recibe bytes desde el Puerto de Estado y el otro Puerto de Datos. La dirección de los dos puertos de datos es la misma; igual sucede con el puerto de comandos y el de estado.

Bajo esta configuración el IOP y los quadart's actúan como un sub-sistema de entrada/salida serial con respecto al procesador central, descargándolo completamente de la responsabilidad de

controlar las comunicaciones con las terminales/modems. En este esquema se podrian conectar hasta 16 terminales/modems, con lo cual se podrian implantar las configuraciones descritas en las secciones 2.2.1 y 2.2.2 para un nodo y una sucursal del sistema de transacciones.

El esquema detallado de este sub-sistema (con un quadart) se muestra en la figura B.3.

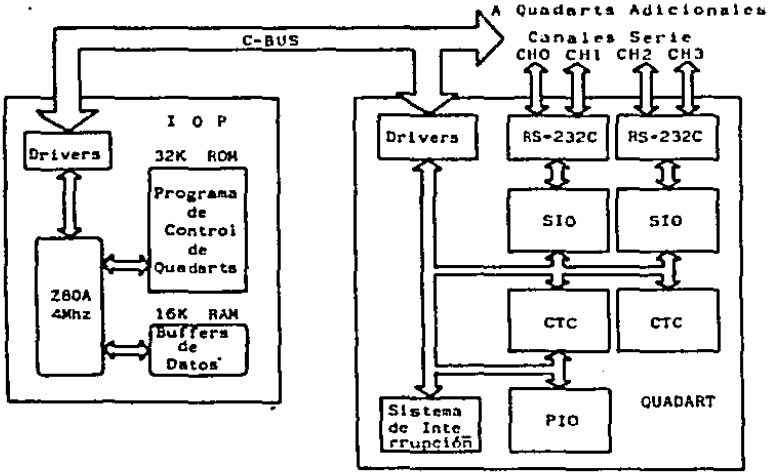


Fig. B.3 Sub-sistema de entrada/salida con un quadart.

Como se observa el IOP contiene un Z-80 que controla las operaciones de este procesador y el intercambio de información con el quadart. Un programa corriendo en la memoria ROM del IOP controla un quadart por medio de señales enviadas y recibidas a través del C-BUS. En una operación tipica de este sub-sistema, el quadart inicia la transferencia de datos/comandos por medio de interrupciones que hace al IOP; éste responde ya sea enviando datos/comandos o recibiendo datos/status a través de 21 puertos que maneja el quadart.

La transferencia de información se hace por medio de interrupciones para que el IOP atienda completamente a los cuatro canales y así evitar que pierda información que en un momento dado se estuviera enviando/recibiendo por los mismos. El programa de

control del quadart está formado básicamente por un segmento de inicialización del quadart, un segmento que corre continuamente supervisando la atención al quadart y varias rutinas de servicio las cuales responden a condiciones tales como: buffer de transmisión vacío, carácter recibido disponible, transiciones en la línea del modem y varias más, para cada canal usado. Los 16K bytes de RAM se usan como: área de almacenamiento de datos, variables RAM y stack. Este programa es proporcionado por el fabricante del equipo.

Con la labor que realizan conjuntamente el IOP y el quadart, la transmisión/recepción de bytes entre nodo y sucursal y entre nodos se puede confiar totalmente a ambos dispositivos ya que contienen todos los elementos necesarios, tanto de hardware como de software, para efectuar esta función sin perder bytes en el intercambio. El módulo L2S (sección 3.1) cuando tiene que enviar/recibir paquetes HDLC sólo tiene que preocuparse por dar la orden de escritura/lectura y el IOP, junto con el quadart, se encargan de controlar la transmisión de la información.

El quadart, por hardware, puede manejar tres protocolos para la transmisión/recepción de información. El programa en el IOP cuando inicializa el quadart especifica qué tipo de protocolo se va a manejar en cada canal. Entre esos protocolos se encuentra el HDLC que es el propuesto para la capa 2 (sección 2.3.2) del sistema descrito en este trabajo. El quadart por hardware puede efectuar las siguientes funciones con respecto a este tipo de paquetes:

- detectar, en la recepción, los marcos de inicio y fin de un paquete;
- insertar, en la transmisión, estos marcos;
- reconocer como suya la dirección que viene en un paquete;
- generar o checar el CRC para control de la integridad de la información.

Cuando se transmite un paquete HDLC, el usuario sólo debe encargarse de proporcionar al quadart los bytes de Dirección, Control e Información; éste se encarga de agregar los demás bytes que necesita el paquete y controlar la transmisión del mismo (los canales del quadart se pueden programar para que transmitan a una cierta velocidad, p. ej. a 4,800 ó 9,600 bps). En la recepción, el quadart detecta el inicio y fin de un paquete, checa la dirección y el CRC y entrega sólo los bytes de Control y de Información para aquellos paquetes que no son erróneos.

Este dispositivo puede emplearse para implantar el intercambio de paquetes HDLC entre dos micros adyacentes del sistema.

Finalmente, el quadart proporciona un conjunto completo de líneas compatibles con el RS-232C de tal manera que puede proporcionar servicio full-duplex en la transmisión/recepción de información. Cada canal contiene una interfaz RS-232C de tal

manera que puede conectársele una terminal o un modem. En nuestro sistema, cuando hablamos de nodos, en cada canal se podrían conectar modems para establecer las comunicaciones con otros nodos o con sucursales; en el caso de sucursales, uno de los canales tendría conectado un modem para comunicarse con el nodo asociado y los demas, terminales para dar atención a los clientes en la sucursal.

B.2 LLAMADAS DE CROMIX

En esta sección se listan todas las llamadas (primitivas) de Cromix que se utilizaron para la implantación del sistema local. Solamente son enunciadas, fundamentalmente con el propósito de dar una idea del tipo de funciones que llevan a cabo. En muchos casos hubo necesidad de probarlas varias veces hasta lograr dominarlas totalmente para poder efectuar la implantación del sistema local.

Llamada	Función
CHDUP	Duplica un canal de comunicación.
CLOSE	Cierra un archivo.
ERROR	Imprime un mensaje de error por el dispositivo estándar de errores.
EXIT	Termina un proceso.
FEXEC	Crea un nuevo proceso y ejecuta un programa en él.
GETDATE	Obtiene la fecha actual del sistema.
GETMODE	Regresa las características de operación de un dispositivo tipo carácter.
GETTIME	Obtiene la hora actual del sistema.
KILL	Envía señales a un proceso.
PIPE	Crea un canal de entrada/salida para comunicación inter-procesos.
PRINTF	Imprime resultados formateados en un archivo.
RDSEQ	Lee de un canal de entrada una determinada cantidad de bytes.
SETMODE	Cambia las características de operación de un dispositivo tipo carácter.
WAIT	Espera por la terminación de un proceso hijo.
WRSEQ	Escribe en un canal de salida una cantidad determinada de bytes.