UNIVERSIDAD NACIONAL DE INGENIERIA

Facultad de Ingeniería Eléctrica y Electrónica

"Sistema Automatico de Intercambio de Información en un Ambiente Heterogéneo de Plataformas de Computo "

TITULACION POR EXAMEN PROFESIONAL

Para Optar el Título Profesional de:

INGENIERO ELECTRONICO

Philip Meerovici Reinstein

Promoción 1989 - 2

LIMA - PERU - 1995

SUMARIO

El Banco Cafetero de Colombia ingresó a un proceso de Reingeniería del Negocio durante el año 1995. Durante este proceso de Reingeniería, se resaltó la necesidad de mejorar los servicios en las agencias del Banco, siendo primordial el mejorar el sistema de comunicación de las agencias con la Oficina Central. Luego de la conclusión del diseño e implementación de redes de comunicaciones, se solicitó la elaboración de un sistema que en forma automática permitiera el intercambio de archivos de información entre las agencias y la Oficina Central, con lo cual, ambos tendrían la información necesaria para la operación día a día.

Una vez estudiados los requerimientos del Banco se procedió a analizar su red de comunicaciones y los equipos de cómputo existentes. El resultado fue el diseño de un sistema de intercambio de archivos de información que trabajara en base a protocolos de comunicación y sistemas de cómputo dentro del marco de lo que se conoce como "Arquitectura Abierta" o "Sistemas Abiertos" y que fuera independiente del nivel físico del enlace de comunicaciones adoptado.

El sistema automático de intercambio de archivos de información se diseñó e implementó durante los meses de Julio a Setiembre de 1995 y se encuentra actualmente en funcionamiento.

SISTEMA AUTOMÁTICO DE INTERCAMBIO DE INFORMACIÓN EN UN AMBIENTE HETEROGÉNEO DE PLATAFORMAS DE CÓMPUTO

Título:	Sistema automático de intercambio de información en un ambiente heterogéneo de plataformas de cómputo
Autor:	Philip Meerovici Reinstein
Grado al que opta:	Ingeniero Electrónico
Facultad:	Ingeniería Electrónica
Ciudad y año:	Lima, Enero de 1996

Extracto

El presente documento es el resultado del trabajo realizado en el Banco Cafetero de Colombia, llevado a cabo entre los meses de Julio a Setiembre de 1995, y comprende la implementación de un sistema de transferencia automática de información entre la oficina central (computadores centrales (*Hosts*) IBM y UNISYS) y las oficinas regionales (centros de proceso) del Banco (PCs), permitiendo la colección y/o distribución de información entre todas las oficinas.

Para la elección del protocolo de comunicaciones se tomó en cuenta la visión estratégica del Banco, que desea mantener una "Arquitectura Abierta de Sistemas", con el fin de obtener soluciones con arquitecturas soportadas por múltiples proveedores. Por este motivo se decidió el uso de APPC (*advanced program to program communication*), el cual forma parte del esquema de sistemas abiertos de IBM (*IBM Open*

Blueprint) y está ampliamente soportado por múltiples plataformas, incluyendo las plataformas a usar en el proyecto (UNISYS e IBM, tanto al nivel Host como al nivel PC).

La elección del APPC nos permitió lo siguiente:

- Las sesiones de comunicación son establecidas de aplicación a aplicación, siendo éstas manejadas totalmente por la arquitectura.
- La aplicación en la PC decide paramétricamente con que aplicación desea comunicarse. Con ello se logra utilizar la potencia de proceso de la PC que tiene menor costo que la de los Hosts.
- No es necesario escribir aplicaciones en los procesadores principales que ejecuten la función de enrutamiento ("Switch").

Los sistemas operativos usados fueron:

- UNIX en el UNISYS A-18
- VM en el IBM ES/9000 (S390)
- OS/2 en las PCs

y como lenguaje de programación se uso REXX (*REstructured eXtended eXecutor language*), el cual es nativo del OS/2 y del VM, siendo soportado por todas las plataformas IBM.

El sistema se encuentra actualmente en producción.

TABLA DE CONTENIDO

INTRODUCCIÓN

Dentro del actual régimen de desregulación del sistema financiero nacional, privatización de la banca estatal e incremento de los costos laborales en Colombia, las Entidades Bancarias en dicho país empiezan a enfrentar un mercado que ofrece márgenes decrecientes de utilidad, menores volúmenes de operación, mayor riesgo en la colocación de créditos y todo ésto lo lleva a la necesidad de maximizar la eficiencia y productividad de la entidad bancaria.

Para poder sobrevivir en este medio, los bancos deberán basar su competitividad en la prestación de un excelente servicio, en el desarrollo de nuevos productos o combinación de productos competitivos y ofrecerlos al mercado antes y mejor que cualquier otro en el sistema financiero, y en el logro de una optimización de rentabilidad por producto, cliente y segmento de cliente. Entonces el Banco líder será el que ofrezca oportunamente una relación total Banco-Cliente y la mejor combinación de servicios interconectados e integrados.

Es dentro de este marco de necesaria competitividad, que el Banco Cafetero de Colombia, también conocido como Bancafé, empujado por la necesidad de mantenerse como uno de los bancos líderes del mercado colombiano, ingresa a un proceso de "Reingeniería de la Empresa" durante el año 1995.

Las principales características del Banco son las siguientes:

299 oficinas divididas en 10 regiones, que cubren todo el país (con autonomía contable, financiera y administrativa), atendiendo aproximadamente a 700,000 clientes.

- 87 oficinas (de las 299) se encuentran actualmente interconectadas (junto con 24 cajeros automáticos propios más los servicios de cajeros automáticos de Redeban y Servibanca), ofreciendo servicios de ventanilla en línea.

  Para atender las necesidades de integración de las oficinas no interconectadas (212), el Banco cuenta con 31 centros de proceso donde se realiza la entrada de datos para su transmisión al sistema principal y la impresión de reportes.

  El Banco espera ampliar la red de oficinas interconectadas a 187, contando las 112 restantes con un computador personal y un modem para la comunicación. También espera incrementar el número de cajeros automáticos a 100.

El proceso de "Reingeniería" permitió al Banco Cafetero detectar muchos procesos por mejorar y optimizar, o eliminar.

Uno de los procesos revisados, relacionado con la forma de operar de las diversas oficinas del Banco, trajo a la luz los siguientes problemas:

- En muchas ocasiones las oficinas del Banco tienen que iniciar la atención al público sın listados actualizados, con el consiguiente malestar de los clientes y la pérdida de negocios.

- El Banco no llega a conocer con exactitud la posición del día anterior, lo que genera riesgos en su relación con las entidades fiscalizadoras del gobierno.
- El proceso de cambios y nuevas versiones de programas en cada una de las oficinas es sumamente lento.
- Los procesos de transferencia de información son hechos manualmente por operadores en ambos extremos.
- Las herramientas de transmisión utilizadas no proveen facilidades de compresión y reinicio de puntos de sincronismo por pérdidas de línea.
- El protocolo de comunicaciones es diferente al utilizado en las oficinas y por ello costosa su integración.

Todas estas situaciones causan malestar en los empleados del Banco, tanto los que atienden al público, como los que manejan el sistema.

La propuesta realizada al Banco, consistente en el desarrollo de herramientas automáticas de intercambio de información entre las agencias del Banco, permitiría:

- La transferencia eficiente de archivos desde y hacia los centros de proceso.
- La construcción y restauración eficiente de Servidores y Clientes en caso de problemas.
- Una recuperación simple en caso de desastres.

Las herramientas de distribución de software pueden utilizarse para distribuir archivos, documentos o manuales de procedimientos y productos necesarios para la operación bancaria.

Los requerimientos funcionales del Banco fueron los siguientes:

- Todo el proceso de transferencia deberá ser completamente automático.
- Se deberá operar sobre facilidades de comunicaciones dedicadas o conmutadas.
- Se ofrecerá facilidades de compresión de datos para disminuir los tiempos de transmisión.
- Se podrá transferir archivos entre los sistemas operativos VM (IBM), UNIX (A-18, Unisys) y OS/2.

CAPITULO I
CONCEPTOS SOBRE LOS PROTOCOLOS Y TIPOS DE COMUNICACIÓN

1.1 X.25

X.25 es un conjunto de recomendaciones emitidas por la International Telegraph and Telephone Consultative Committee (CCITT).

X.25 define los requerimientos de equipos usados para conectar computadoras a redes públicas de conmutación de paquetes.

En una red pública de conmutación de paquetes, múltiples usuarios comparten los mismos canales y rutas en el mismo tiempo. Este tipo de red es una alternativa a redes constituidas por líneas dedicadas a pares de usuarios y usuarios múltiples.

Una red pública de conmutación de paquetes transporta unidades de información que son divididas en segmentos denominados paquetes.

Uno no necesita saber en que forma la información es llevada a su destino o que procedimientos son usados dentro de la red para la transmisión de los paquetes.

Una red que usa el estandar X.25 provee la facilidad para conectar varios sistemas a la red y además el transporte de la información. Ésto significa que el estandar provee un camino para conectar físicamente equipos, que transferirá la

información desde un Equipo Terminal de Información (DTE: Data Terminal Equipment, usualmente la computadora del usuario) hacia un Equipo Terminante del Circuito de Información (DCE: Data Circuit - Terminating Equipment, usualmente el modem). Se necesitará un nivel más alto de protocolo para transferir la información a través de la red. Para el usuario de red X.25, otros niveles del sistema de comunicación manejarán el lenguaje aplicación a aplicación (APPC por ejemplo).

X.25 define 3 niveles de conexión entre el DTE (Equipo terminal de Información) y el DCE (Equipo Terminante del Circuito de Información):

- Conexión física
- Protocolo de enlace
- Procedimientos de empaquetado

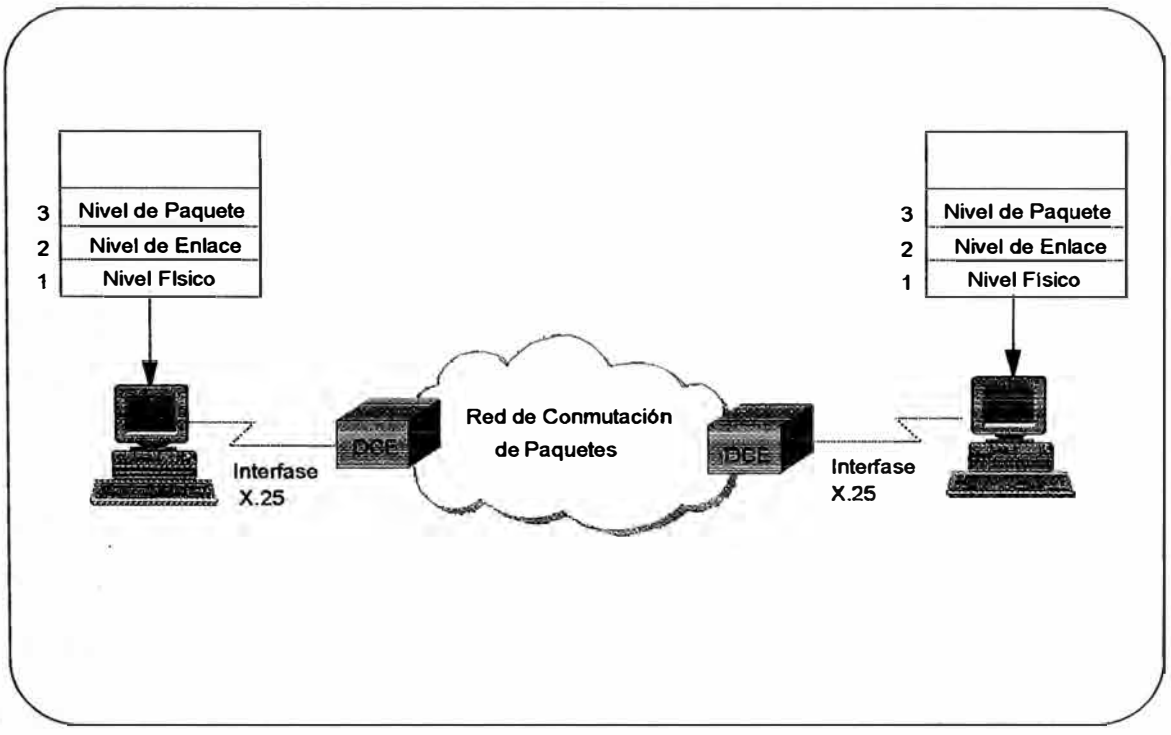

Figura 1

1.1.1 Conexión física

La conexión física (1) define el control de la conexión eléctrica entre el Equipo Terminal de Información (DTE) y el Equipo Terminante del Circuito de Información (DCE). Las funciones de control incluyen activación, mantenimiento y desactivación del circuito entre el dispositivo en comunicación y el punto de entrada a la red.

1.1.2 Protocolo de enlace

El protocolo de enlace (2) también conocido como el nivel de estructura (*frame level*) define los procedimientos usados para transferir información a través del enlace entre el DTE y la red en forma precisa. Estos procedimientos corresponden a los del HDLC (High-level Data Link Control) y son similares a los del SDLC (Synchronus Data Link Control) usados en las comunicaciones SNA.

El protocolo de enlace le da un formato a la información y consiste de procedimientos para establecer llamadas, transferir estructuras (*frames*) y desconectar llamadas, y además es el primer nivel de los procedimientos de recuperación.

1.1.3 Procedimientos de empaquetado

Los procedimientos de empaquetado (3) definen como la información y la información de control son estructurados en paquetes, y como son establecidas las llamadas, mantenidas y liberadas.

Este nivel divide un enlace físico único en múltiples enlaces lógicos. Cada canal lógico puede establecer una conexión lógica, llamada circuito virtual, hacia otro DCE. La

información fluye a través del nivel de paquetes en unidades manejadas en forma independiente llamadas paquetes. X.25 define los formatos de los varios tipos de paquetes.

Dado que la información es transmitida en paquetes, un enlace físico puede transmitir información desde muchos circuitos virtuales distintos en forma simultánea. La facilidad de compartir un enlace físico ahorra muchos recursos de red. Un DTE puede tener muchos circuitos virtuales establecidos al mismo tiempo en enlace físico hacia una red X.25.

1.2 SDLC

SDLC (Synchronus Data Link Control) es la versión ampliamente usada por IBM del conjunto de estándares correspondientes a HDLC (High Level Data Link Control).

SDLC se implementa en muchas formas, tales como enlaces punto a punto, enlaces multi-punto, y en líneas conmutadas o dedicadas.

Una de las implementaciones más usadas de SDLC es el uso del modo de respuesta normal no balanceada (Unbalanced Normal REsponse Mode) del HDLC, lo cual significa que el enlace es manejado por una estación principal.

SDLC maneja las conexiones de enlace entre las estaciones con el uso de estados de transmisión. La conexión se describe por uno de los siguientes 4 posibles estados:

Activo: El tráfico está fluyendo entre las estaciones.

- Desconectado: La estación está físicamente desconectada del enlace.

- En espera: No se está envíando tráfico; en lugar de tráfico se transmiten 1s continuos.
- Transitorio: Es el tiempo entre que la estación secundaria recibe un *frame* y envía una respuesta a la estación primaria.

1.3 APPC

APPC (*Advanced Program-to-Program Communication*, también conocido como LU 6.2) es un protocolo de comunicaciones que permite la conversación entre programas en diferentes computadores. APPC provee la interfase entre los programas y el *hardware* y *software* de la red y define las reglas que usan los programas para intercambiar información.

APPC es una forma de comunicar con alta velocidad programas en distintas computadoras, desde computadoras portables y estaciones de trabajo hasta computadoras medianas y grandes (hosts).

El software que compone al APPC está disponible para muchos sistemas operativos, IBMs y no-IBMs, en algunos casos formando parte de éstos y en otros casos, como paquetes de software adicionales.

APPC sirve de interfase entre los programas de aplicaciones y la red.

Cuando la aplicación de comunicaciones en la estación de trabajo envía información al software del APPC, el APPC toma la información y la envía a una interfase de red, tal como una tarjeta adaptadora de token-ring. La información viaja a través de la red hacia otra computadora, donde el software del APPC recibe la información de la interfase de red.

APPC regresa la información a su formato original y la envía a la aplicación de comunicaciones correspondiente.

1.3.1 Conceptos de APPC

La parte de la aplicación de comunicaciones que inicia o responde a las comunicaciones APPC es llamada programa de transacción.

Un programa de transacción es parte del programa que maneja transacciones (intercambio de información) con otro programa.

La comunicación entre dos programas de transacción es llamada conversación, y es semejante a la conversación entre dos personas, en la que se siguen reglas de como comenzar y terminar una conversación, como se toman turnos para hablar y como se intercambia información.

En forma análoga, el APPC es llamado un protocolo pues provee de reglas que gobiernan como se inician y terminan las conversaciones entre programas de transacción, que programa "habla" primero, y como la información es intercambiada.

APPC consiste de un set de reglas muy bien definidas que buscan cubrir todas las situaciones posibles de comunicación.

Un programa APPC puede mantener varias conversaciones activas a un mismo tiempo, con el mismo o distintos programas de transacción.

Los programas de transacción requieren de una pareja para poder llevar a cabo una conversación. Esto significa que los programas de transacción son desarrollados en pares (*partner transaction programs*).

El lenguaje de comunicación entre un programa de transacción y otro consiste en verbos tales como:

ALLOCATE	Inicio de una conversación con otro programa de transacción.
SEND DATA	Envío de información al programa de transacción *partner*.
RECEIVE	Recepción de información desde el programa de transacción *partner*.
DEALLOCATE	Finaliza la conversación con otro programa de transacción.

El conjunto de verbos APPC constituyen los APIs (*application programming interface* - interfase programable de aplicación) del APPC, que representan la interfase entre los programas de transacción y el software APPC.

1.3.2 APPC y computación Cliente/Servidor

Cliente/Servidor es actualmente conocido como un sinónimo de computación, como la arquitectura de sistemas de la siguiente generación o como la plataforma de computación abierta de los 90s (entre muchos otros apelativos).

Debido a que la transición a los Sistemas Abiertos y arquitecturas Cliente/Servidor no puede ser ejecutada en forma instantánea, y las compañias (sobre todo las grandes) poseen una gran inversión de capital en aplicaciones en sistemas hosts, muchas mantendrán por un buen tiempo las aplicaciones existentes e información en los *mainframes* (además de que hoy en día no se ha igualado la gran capacidad de manejo de canales y puertos de entrada/salida que poseen los *mainframes*).

Los intentos iniciales de implementación de tecnología Cliente/Servidor por lo general consisten en el desarrollo de nuevas aplicaciones para estaciones de trabajo en un ambiente LAN. Sin embargo se desea que estas aplicaciones puedan formar parte de la red y que permitan a las estaciónes de trabajo accesar la información contenida en los *mainframes.* Luego, la computación Cliente/Servidor requiere de una tecnología sofisticada de comunicaciones que esté diseñada para funcionar en distintas máquinas, tanto en ambientes LAN como WAN.

Aquí es donde interviene APPC jugando un rol muy importante en la computación Cliente/Servidor, permitiendo a programas ser ejecutados sin modificación alguna en muchos tipos de enlaces de comunicaciones, desde LANs token-ring y Ethernet hasta conexiones asíncronas vía modems.

1.3.3 CPI-C

APPC provee un set de funciones de comunicación programa-a-programa consistente en distintas plataformas. Sin embargo, la arquitectura no especifica una Interfase de Programación de Aplicaciones (API) común para la implementación de estas funciones. Por lo tanto, cada sistema operativo que soporta APPC desarrolló sus propios APIs, muy ligados al propio sistema operativo.

Ésto implica que de trabajar en sistemas operativos distintos, se deberá aprender el uso de dos o más conjuntos de APIs.

Para solucionar este problema aparece el CPI-C (*Common Programming Interface for Communications*), el cual trae un

conjunto de APIs estandar, conocidos como llamadas CPI-C, para todos los sistemas que soportan CPI-C.

Al usar CPI-C se podrá hacer uso de un mismo conjunto de APIs para aplicaciones Cliente/Servidor en distintos sistemas operativos.

1.3.4 Conectividades APPC probadas en el Banco

Los siguientes tipos de conectividades APPC fueron probados en forma exitosa en la red del Banco:

- Conectividad APPC del computador IBM al Unisys A18, en ambos sentidos:

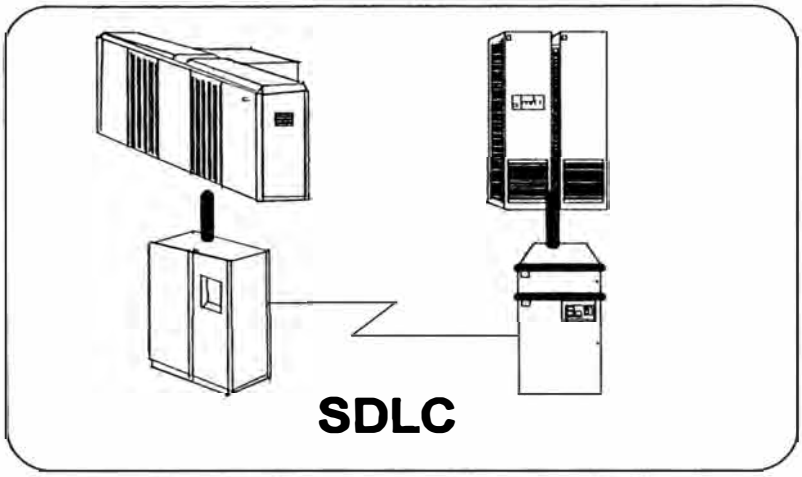

Figura 2

- Conectividad APPC del computador IBM al Servidor de Transferencias, en ambos sentidos:

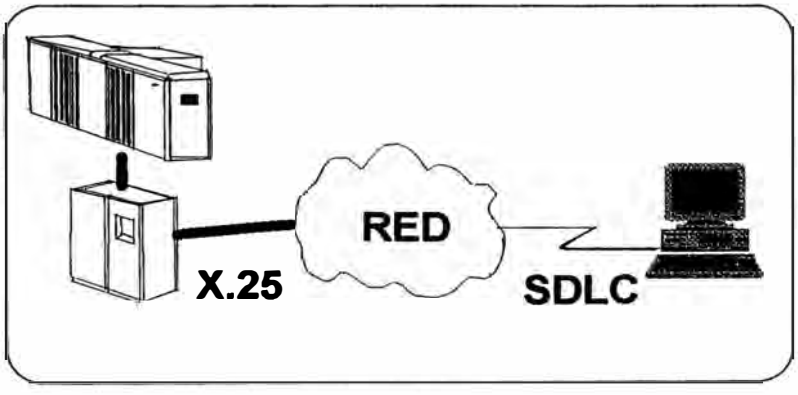

Figura 3

- Conectividad APPC de Servidor de Transferencias a Centro de Proceso, en ambos sentidos:

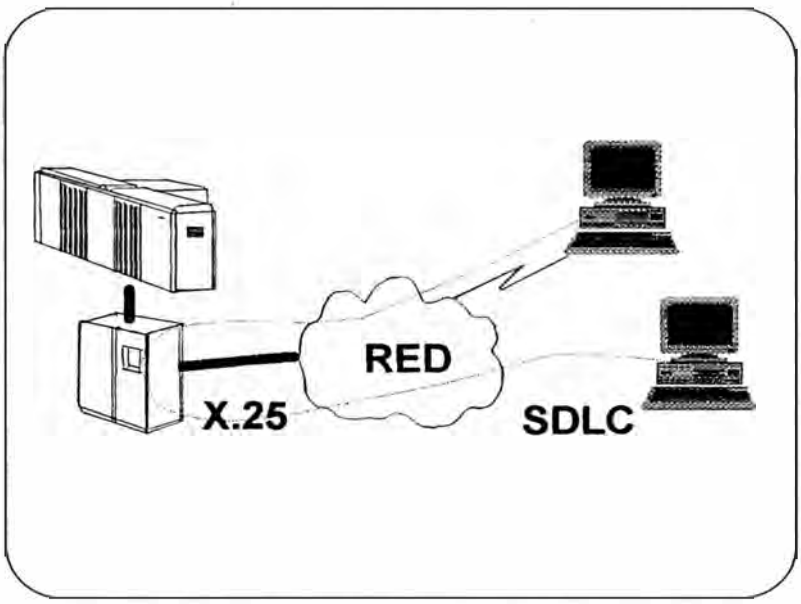

Figura 4

- Conectividad "Cross-domain" del Servidor de Transferencias al Unisys A18 a través del computador IBM, en ambos sentidos:

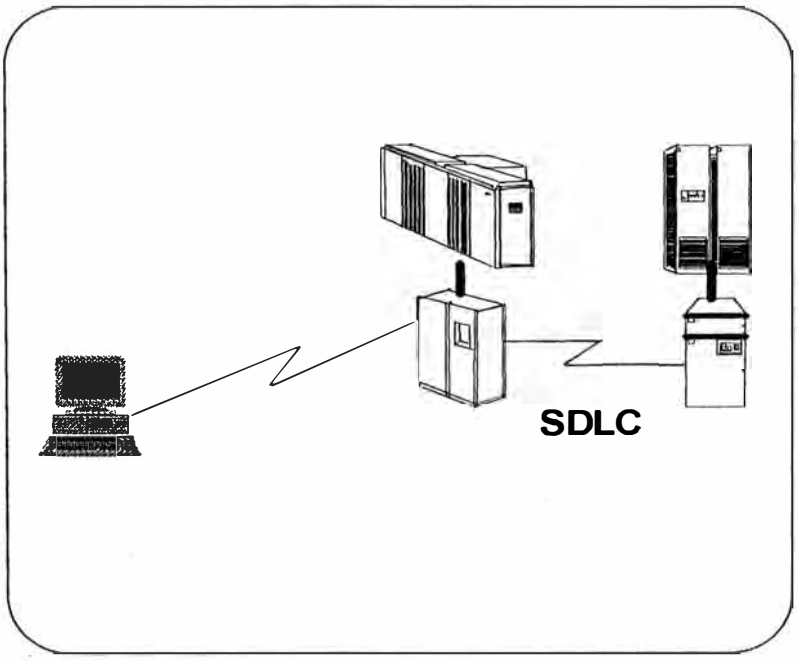

Figura 5

- Conectividad APPC del computador Unisys A18 al Servidor de Transferencias, en ambos sentidos:

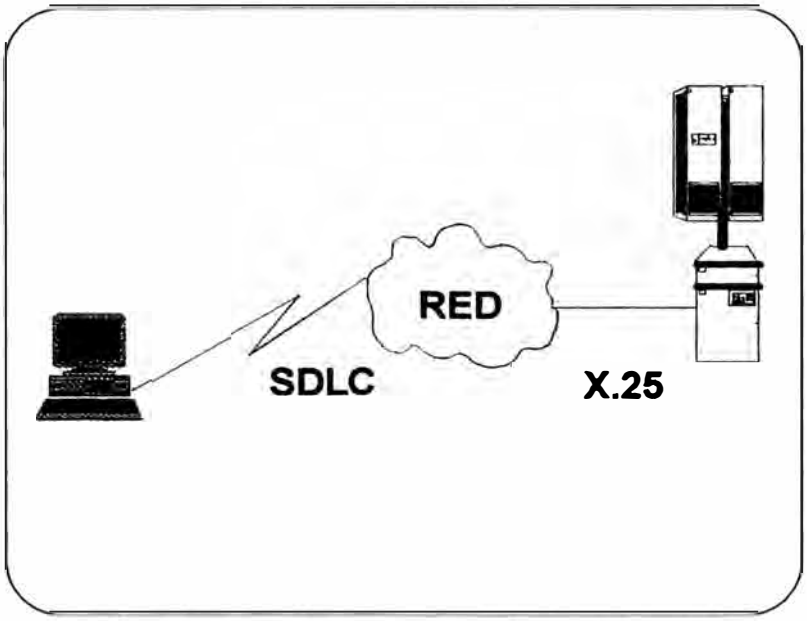

Figura 6

- Conectividad APPC del Servidor de Transferencias al Unisys A18, a través del computador IBM, en ambos sentidos:

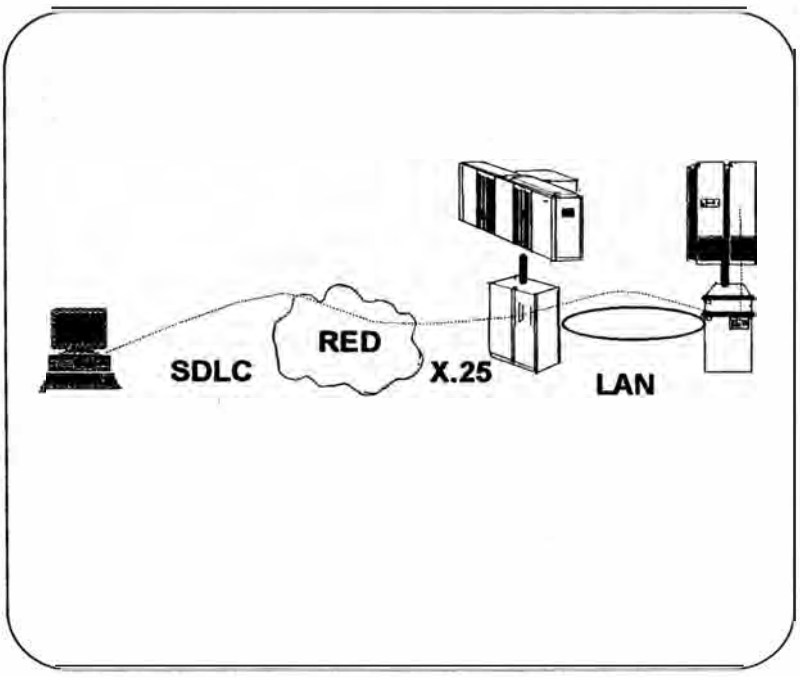

Figura 7

CAPITULO II
ARQUITECTURA DE COMPONENTES DE LA RED DEL BANCO

2.1 Descripción de la arquitectura

En la figura 8 se puede notar que se ha configurado un "Backbone" conformado por los MAB (Manejadores de Ancho de Banda), los NP (Procesadores Nodales) y los R (Enrutadores).

Esta configuración es la usada en la Dirección General del Banco en Bogotá y en cada una de sus nueve Regionales.

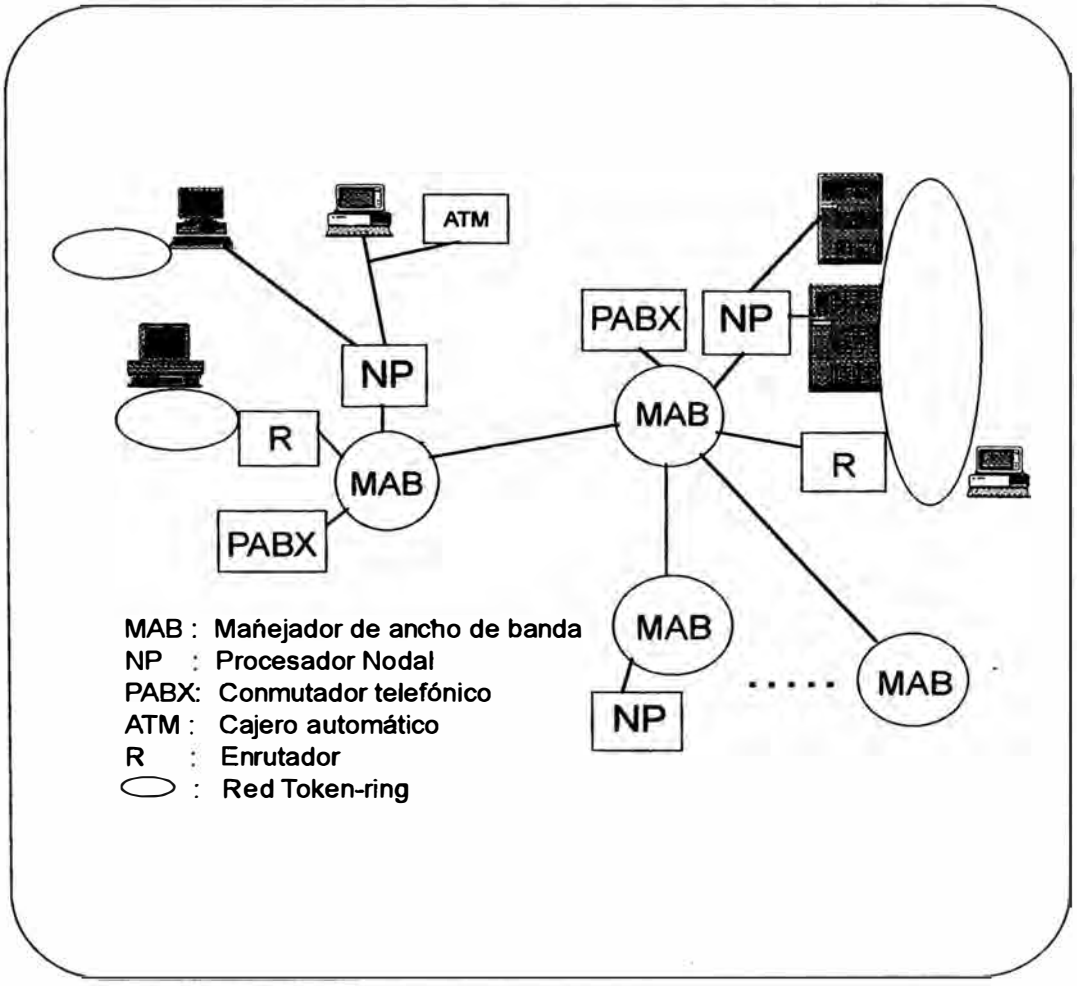

Figura 8

Al procesador nodal en la Dirección General de Bogotá se conectan:

- Un Sistema/88 con software ON/2 el cual maneja a través de la red todos los cajeros automáticos del país.
- Un controlador de comunicaciones 3745 con ACF/NCP y NPSI (para soporte X.25) el cual maneja a través de la red todas las oficinas del país.
- Todas las oficinas y cajeros de la ciudad de Bogotá.

Los medios de transmisión que conectan cada una de estas Regionales con la Dirección General están provistos por la red digital de Telecom (el Banco no ha implantado una red propia de transmisión y está contratando los servicios de Telecomunicaciones).

2.2 Medios de comunicaciones de la red

El tipo de medio de comunicación a usar está determinado por la disponibilidad, el tráfico a cursar y el nivel de importancia de la oficina.

En la figura 9 se muestran cuatro tipos de conexión:

- TIPO A: Conexión terrestre dedicada, con velocidades menores a 19.2Kbps, entre una oficina y un Procesador Nodal en la Regional
- TIPO B: Conexión digital, terrestre o satelital con velocidades iguales o mayores a 64Kbps canal dedicado, entre cada una de las Regionales y la Dirección General del Banco en Bogotá.
- TIPO C: Conexión satelital, con velocidades menores o iguales a 19.2Kbps, entre una oficina y la Dirección General del Banco en Bogotá.

- TIPO D: Conexión a través de la red telefónica conmutada a velocidades menores a los 19.2Kbps, tanto entre oficinas y la Dirección General del Banco en Bogotá o para las oficinas con sus Regionales.

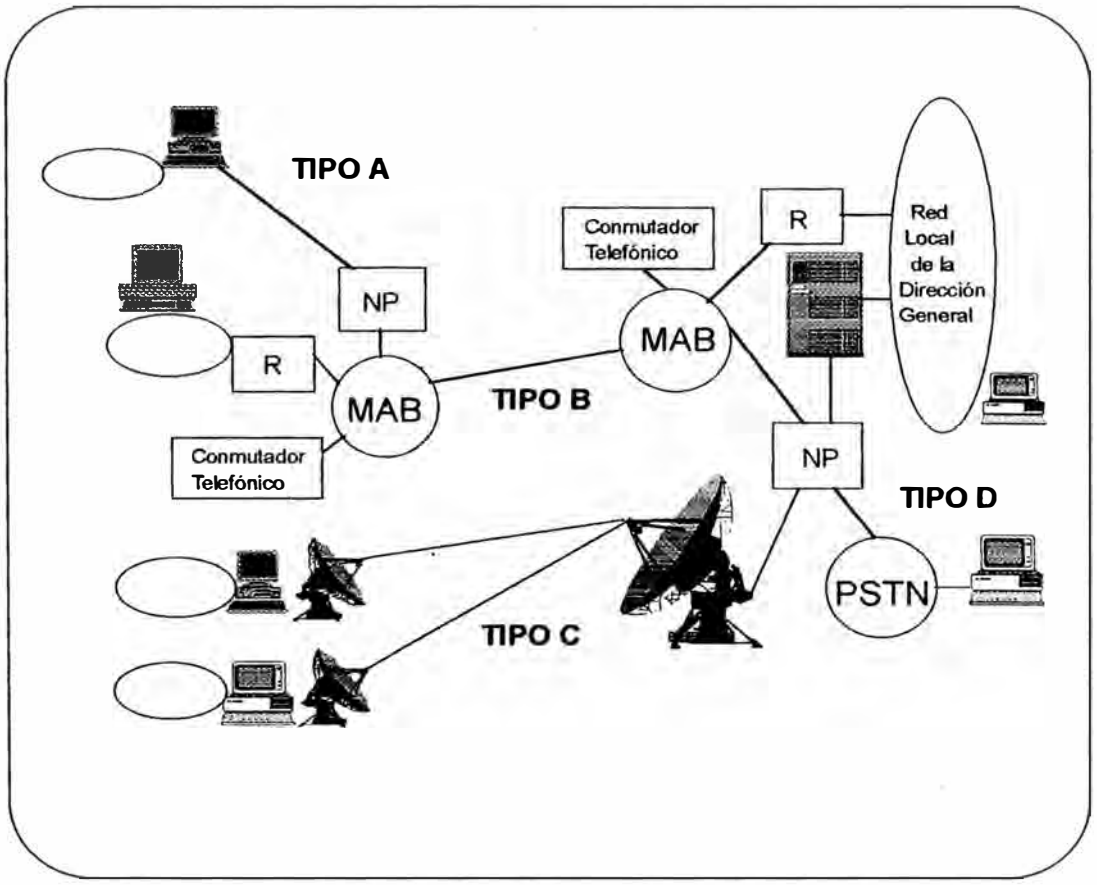

Figura 9

Los manejadores de ancho de banda (figura 10) se encargan de racionalizar el uso del medio de comunicaciones. Ellos forman el backbone de la red de transmisión y son los encargados de enrutar a través de los medios de comunicación cada uno de los canales. Estos dispositivos asignan el ancho de banda en forma dinámica de acuerdo a la demanda de canales de voz y datos.

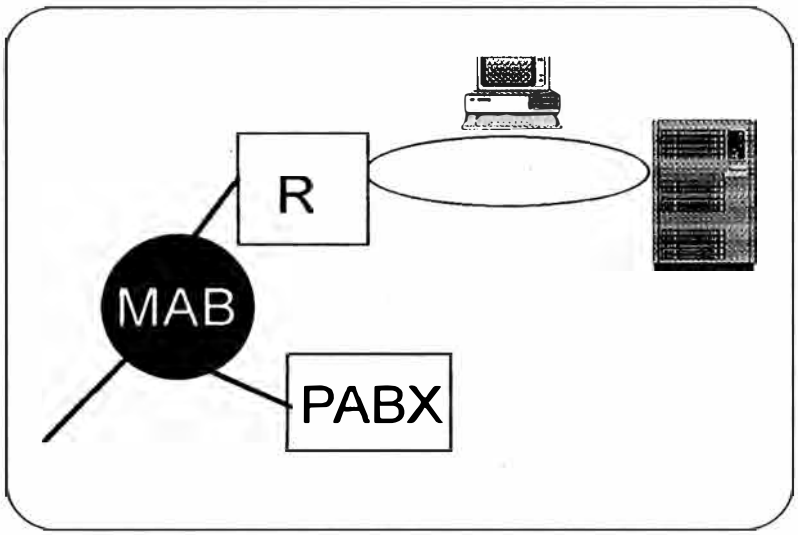

Figura 10

Estos dispositivos también digitalizan y comprimen las señales de voz analógicas, para poder compartir el medio con los datos (figura 11).

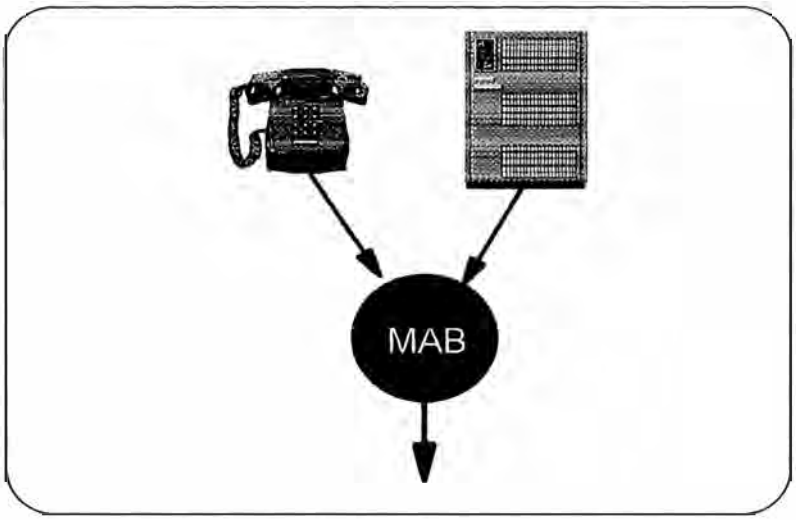

Figura 11

El uso de procesadores nodales (NP: figura 12) permite que el Banco pueda soportar eficientemente diversos protocolos diseñados para medios analógicos. Los procesadores nodales trabajan con tecnología de conmutación de paquetes brindando los servicios de acceso y conmutación a las oficinas y cajeros.

Las oficinas y cajeros trabajan bajo protocolo SDLC y están conectadas a través de medios analógicos al NP.

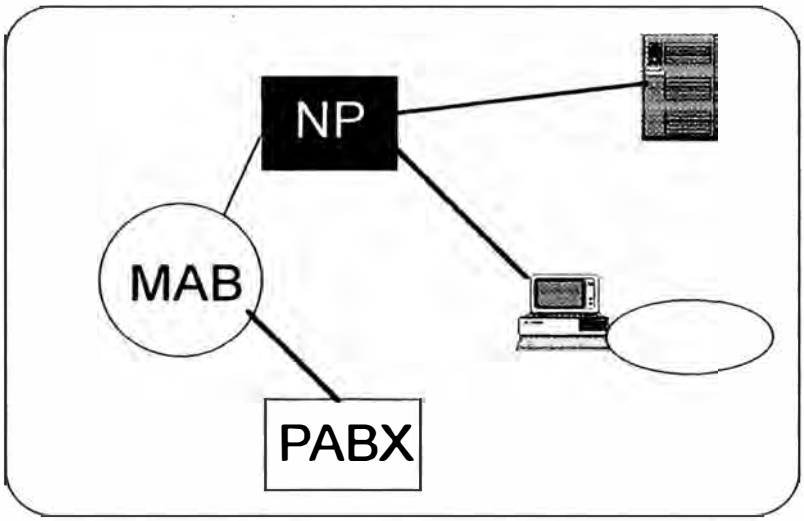

Figura 12

El uso de enrutadores (R: figura 13) permite que el Banco pueda transmitir por el medio diversos protocolos.

Los enrutadores permiten construir una red multiprotocolo en la que se puede transmitir tráfico SNA, TCP/IP, Netbios e IPX. También ofrecen un mejor tiempo de respuesta en el filtrado de tráfico.

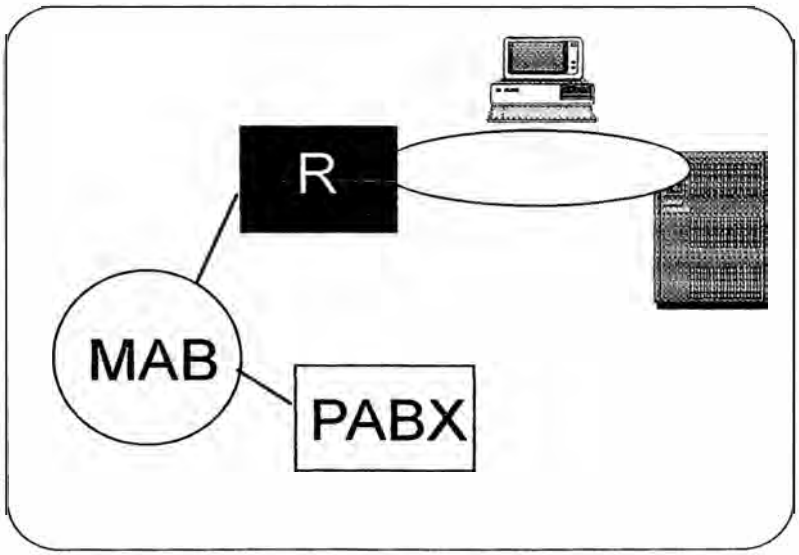

Figura 13

Todo lo anterior permite que cada Región pueda ser considerada como una sub-red independiente, facilitando de esta forma el direccionamiento total de la red y así su manejo.

CAPITULO III
IMPLEMENTACIÓN DEL SISTEMA

3.1 Descripción general

El sistema cuenta con un servidor de transferencias PC que usa el sistema operativo OS/2 Warp. Este sistema operativo se caracteriza por ser Multitarea, es decir, tener la capacidad de procesar varias tareas al mismo tiempo, en áreas de memoria totalmente independientes, lo que en este caso permite realizar transferencias de información desde y hacia distintos puntos, en forma simultánea.

La transferencia de información se realiza por medio de utilitarios de uso público basados en el protocolo APPC en todos los casos, salvo en el caso del sistema operativo VM, que en esta instalación no contaba con este protocolo, y por lo tanto, se usó EHLLAPI (Enhanced High Level Language Application Program Interface), el cual permite interactuar directamente desde un programa con las sesiones de emulación terminal a este sistema.

Los computadores que son interconectados entre sí (figura 14) consisten en UNISYS B28s (BTOS), computadores personales basadas en Intel (IBM), un sistema IBM ES/9000 (VM) y un UNISYS A18 (UNIX).

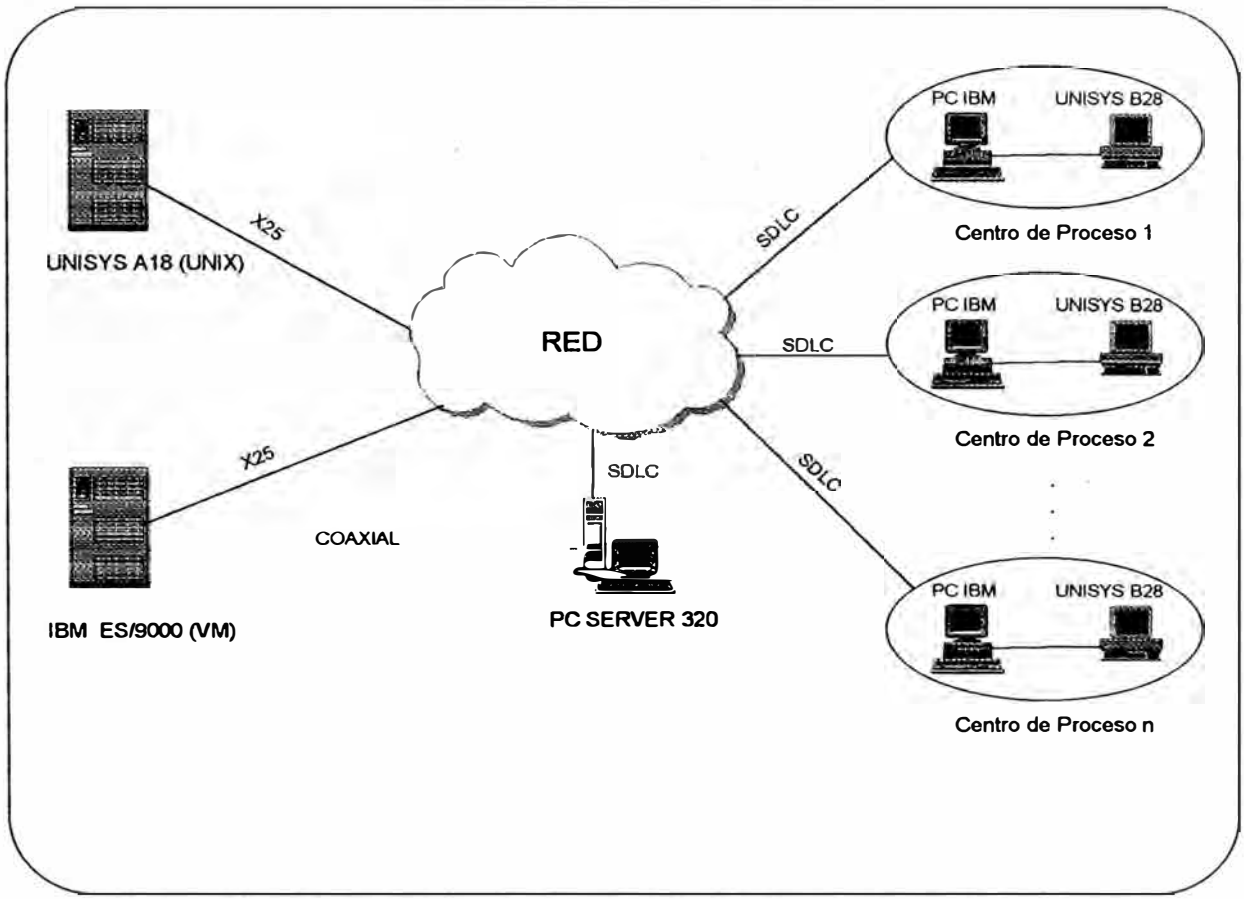

Figura 14

3.2 Arquitectura del sistema

El sistema cuenta con una serie de directorios de trabajo que usan la nomenclatura WKS001, WKS002,...WKS0nn, donde nn representa el numero de procesos simultáneos de transferencias con los Centros de Proceso, y es un parámetro del sistema. También cuenta con los directorios de trabajo WKSRVM, WKSTVM (recepción y transferencia desde y hacia el VM, respectivamente) y WKSRUN, WKSTUN (recepción y transferencia desde y hacia el UNIX, respectivamente).

Para el control de cual directorio de trabajo puede ser usado por estar disponible, y cual está siendo usado en un determinado momento, el sistema hace uso de un directorio de

control WKSMON, el cual contiene archivos de control por cada directorio de trabajo que use el sistema (figura 15).

El contenido de los archivos de control es nulo. El control se lleva a cabo por el FN (*filename*) y EXT (*extension*) de cada uno de ellos. El FN determina el nombre del directorio de trabajo (WKS001, WKS002, WKSTVM, etc) y el EXT determina su estado en ese momento.

Para un archivo de control WKS00n.xxx, los diferentes estados posibles xxx son:

1. **DIR** El directorio WKS00n se encuentra disponible para ser usado por cualquier proceso.
2. **TXR** El directorio WKS00n está siendo usado para transferir archivos desde un Centro de Proceso (C.P.) hacia el servidor de transferencias, para su posterior envío al VM o UNIX. El proceso de transferencia se encuentra activo en ese momento.
3. **TXW** El proceso de transferencia desde el Centro de Proceso X y el servidor de transferencia ya culminó y el directorio WKS00n contiene los archivos resultados de la transferencia. Está a la espera que se arranque un proceso de distribución de archivos hacia los directorios de transferencia al VM o UNIX (proceso WKSTCPY), para finalmente regresar al estado DIR (disponible) nuevamente.
4. **TXC** El directorio WKS00n está en medio del proceso de distribución de los archivos que le fueron transferidos desde el Centro de Proceso, hacia los directorios de distribución al VM o UNIX, según sea el destino final de los archivos (determinado por el EXT de los mismos).

5. **TXH** El directorio WKSTVM o WKSTUN está transmitiendo a Host.
6. **RXC** El directorio WKS00n está en medio del proceso de selección de los archivos que fueron recibidos desde el VM y el UNIX en los directorios WKSRVM y WKSRUN respectivamente (proceso WKSRCPY) y que serán copiados a este directorio para su posterior distribución al Centro de Proceso correspondiente.
7. **RXW** El directorio WKS00n ya contiene los archivos que serán distribuídos al Centro de Proceso identificado por la extensión de los mismos, y está a la espera del arranque del proceso correspondiente al envío de estos archivos.
8. **RXR** Los archivos que contiene el directorio WKS00n están siendo transmitidos en ese momento al Centro de Proceso respectivo (los archivos se encuentran empaquetados en uno solo).
9. **RXH** El directorio WKSRVM o WKSRUN está recibiendo de Host.
10. **XXX** El directorio WKS00n se encuentra marcado como inutilizable por algún error. Se deberá revisar el Log de Errores del sistema para la determinación del mismo.

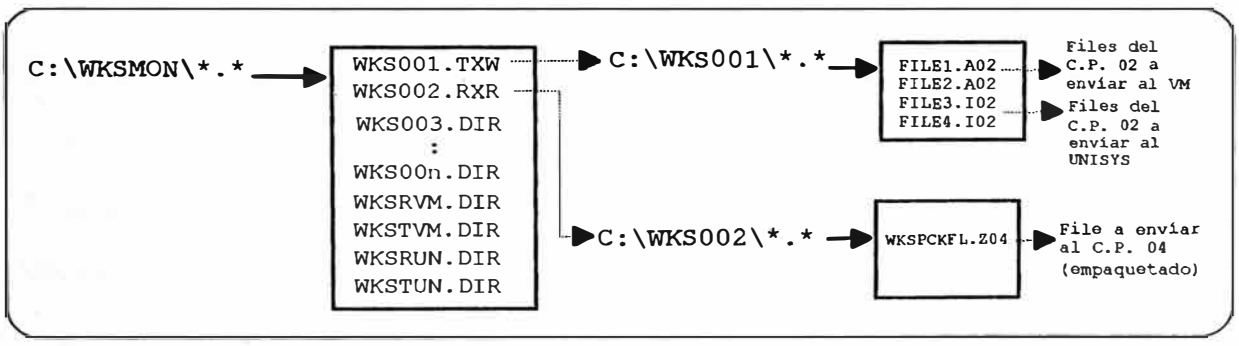

Figura 15

El Sistema además cuenta con un archivo de control WKSTN.LST el cual contiene la información necesaria para contactar a los Centros de Proceso.

Esta información está conformada por:

1. Identificación del Centro de Proceso
2. Nodo APPC
3. Dirección APPC
4. Usuario de conexión al C.P.
5. Contraseña de conexión al C.P.
6. Directorio en el C.P. desde el cual transmitir información al Servidor de Transferencias
7. Directorio en el C.P. hacia el cual transmitir información desde el Servidor de Transferencias

El archivo de control, junto con todos los programas del Sistema de Transferencias, se localizan en el directorio C:\WKSDIR.

3.3 Descripción del funcionamiento

El proceso de transferencia es controlado por el Servidor de Transferencias, quien inicia las tareas de transmisión y recepción de archivos desde y hacia los Centros de Proceso (C.P.) según haya disponibilidad de directorios de trabajo.

También se encarga de iniciar procesos de transmisión y recepción de archivos desde y hacia el VM y UNIX. A diferencia de la transferencia con Centros de Proceso, éstas se caracterizan por realizarse con directorios de trabajo fijos, 2 para el VM (WKSTVM y WKSRVM) y 2 para el UNIX (WKSTUN y WKSRUN), por lo cual no estarán supeditados a la disponibilidad de directorios de trabajo.

3.3.1 Proceso de transmisión de archivos

El monitor de transferencias (WKSMON) hará un barrido por todos los Centros de Proceso y en caso que alguno de ellos tuviera archivos para transmisión y hubiera directorios de trabajo disponibles, se procederá a transferir los archivos correspondientes.

Para llevar a cabo este proceso de transmisión desde el Centro de Proceso, primero se cambiará el estado del directorio de trabajo desde DIR a TXR. Luego se empaquetan en el C.P. los archivos a transmitir en uno solo, además de compactarlos. Dentro del archivo empaquetado se incorpora uno creado por el programa empaquetador, que contiene la lista de todos los archivos a ser transmitidos (WKSPCKFL.LST). Con ésto el proceso de transmisión se simplifica al tener que transmitir un solo archivo en lugar de varios.

Una vez recibido el archivo en el Servidor de Transferencias, se procede a desempaquetarlos y descompactarlos, recuperando los archivos originales. Se verifica que todos los archivos que aparecen en la lista creada por el programa empaquetador hayan sido recibidos, y se confirma al C.P. su correcta recepción, además de registrarlo en el Log de Archivos correspondiente a ese C.P. Al terminar la operación de transferencia de archivos, el estado del directorio cambia de TXR a TXW, que corresponde a un directorio de trabajo con archivos en espera a ser transmitidos al VM y/o al UNIX.

Si hubiera algún problema durante el proceso de transmisión (tal como caída de línea de comunicación), se retorna el estado del directorio de trabajo a DIR y al

inicio del siguiente barrido de C.P., se intentará un nuevo proceso de transmisión.

Cuando el programa monitor (WKSMON) detecta la existencia de un directorio de trabajo en estado TXW, inicia el proceso WKSTCPY, el cual se encarga de distribuir los archivos que fueron transmitidos al directorio de trabajo, a los directorios de transmisión al VM o al UNIX. Durante este proceso el estado del directorio de trabajo es cambiado a TXC, para evitar que se pueda arrancar algún otro proceso WKSTCPY contra este directorio. Se determina cual archivo va al VM y cual al UNIX por la extensión de los mismos. Una extensión que se inicie con la letra A dirigirá los archivos al directorio de transmisión al VM, y una extensión que se inicie con la letra I, al UNIX.

La distribución consiste en copiar el archivo al directorio de transmisión correspondiente y luego borrarlo del directorio de trabajo.

Al finalizar esta tarea de distribuición, se cambia el estado del directorio de trabajo de TXC a DIR, quedando totalmente liberado para participar en algún otro proceso de transmisión o recepción.

Si hubiera algún problema durante el proceso de distribución, el programa regresará el estado del directorio de trabajo a TXW para que posteriormente se arranque un nuevo proceso de distribución y se intente autocorregir el error del sistema.

En forma periódica el programa monitor arranca procesos de transmisión y recepción de archivos hacia el VM y UNIX.

Para el caso de transmisión hacia ambos sistemas, el programa monitor arranca las tareas WKSHLSNV (VM) y WKSHLSNU (UNIX). Estas tareas revisan los correspondientes directorios de transmisión y de encontrar archivos en ellos, comienzan a transmitirlos.

En el caso del VM, la transmisión se realizará por medio de APIs de EHLLAPI a la sesión de emulación correspondiente a transferencia hacia el VM (sesión A).

En el caso del UNIX, la transmisión se realizará por medio de AFILE (APPC) hacia dicho sistema.

Durante el proceso de transmisión, el estado del directorio correspondiente al sistema operativo hacia el cual se están enviando archivos, será cambiado de DIR a TXR, para evitar que pueda activarse algún otro proceso de transmisión hacia el mismo sistema.

Al finalizar la transmisión de cada archivo, se graba en el Log de Archivos del VM o UNIX la finalización de la transferencia. Al finalizar la transmisión de todos los archivos, se regresa el estado del directorio de transmisión correspondiente al estado DIR.

Con ésto se da por finalizado el proceso de transmisión de archivos desde un C.P. hacia el VM y/o UNIX.

Proceso de envío de archivos a los Hosts
VM RX
VM TX
File1.A01
File3.A01
(WKSPACK + WKSGET)
C.P. 01
C.P. 02
C.P. 03
C.P. n
WKS001
WKS001
WKS001
WKS001
WKS002
WKS003
WKSnnn
(WKSLPCK)
St DIR -> TXR
St TXR->TXW
(WKSTCPY)
St TXW->TXC->DIR
WKSTSNW
WKSTCPY
WKSTVM
WKSTVM
WKSTVM
WKSRVM
File1.A01
File3.A01
ST TXR->DIR
ST DIR->TXR
WKSTUN
WKSTUN
WKSTUN
WKSRUN
File2.I01
Filen.I01
ST TXR->DIR
ST DIR->TXR
(WKSHLSNU)

Figura 16

3.3.2 Proceso de recepción de archivos

El monitor de transferencias (WKSMON) en forma periódica arranca los procesos de recepción de archivos del VM (WKSHLRVM) y UNIX (WKSHLRUN) siempre que no exista ya uno de estos procesos ejecutándose (ésto se determina al encontrar al directorio WKSRVM o WKSRUN en el estado RXH). Estos procesos transferirán los archivos desde el VM y UNIX hacia los directorios WKSRVM y WKSRUN respectivamente.

Cuando el monitor de transferencias encuentre un directorio de trabajo disponible (estado DIR) después de haber arrancado un proceso de transferencia (WKSGET), arrancará un proceso WKSRCPY (siempre que no exista otro activo) el cual se encargará de seleccionar archivos a ser transmitidos a los C.P. correspondientes desde los directorios WKSRVM y WKSRUN. Para poder realizar ésto, el WKSRCPY obtiene la lista de archivos en el directorio WKSRVM y le añade la lista de archivos en el directorio WKSRUN. Luego selecciona el primer archivo de la lista total y determina a que C.P. lo debe transmitir (lo obtiene de los 2 últimos dígitos de la extensión del archivo), y por último selecciona todos los archivos de la lista total que tengan el mismo destino que el primer archivo y los mueve al directorio de trabajo.

Durante este proceso el directorio de trabajo cambió del estado DIR al estado RXC (mientras se realizaba la selección y movida de archivos) y por último quedó con el estado RXW (en espera de transmisión al C.P.).

Cuando el monitor de transferencia detecta la existencia de un directorio en estado RXW, inmediatamente arranca el proceso WKSPUT que se encarga de envíar todos los archivos de ese directorio al C.P. correspondiente. Inmediatamente se cambia el estado del directorio de trabajo de RXW al estado RXR. Este proceso primero ejecuta el proceso WKSPACK para empaquetar todos los archivos del directorio de trabajo en uno solo, el cual será transmitido al C.P. El WKSPACK además de empaquetar los archivos, incluye dentro del empaquetado un archivo conteniendo la lista de archivos a ser envíados (WKSPCKFL.LST). Luego se ejecuta el utilitario AFILE para transmitir el archivo empaquetado al C.P. y finalmente se ejecuta en remoto en el C.P. el programa WKSUPCK, que se encargará de desempaquetar los archivos envíados.

Una vez finalizada la transmisión, se procede a una verificación del envío. La verificación consiste en efectuar la transmisión del archivo WKSPCKFL.LST desde el C.P. al servidor de transferencias, leer su contenido y notificar al C.P. la lista de los archivos envíados. También se grabará en el LOG de transferencias el correcto o fallido intento de transmisión al C.P.

Al finalizar el proceso, se cambiará el estado del directorio de trabajo del estado RXR al estado DIR, y este directorio quedará disponible para ser asignado a cualquier otro proceso.

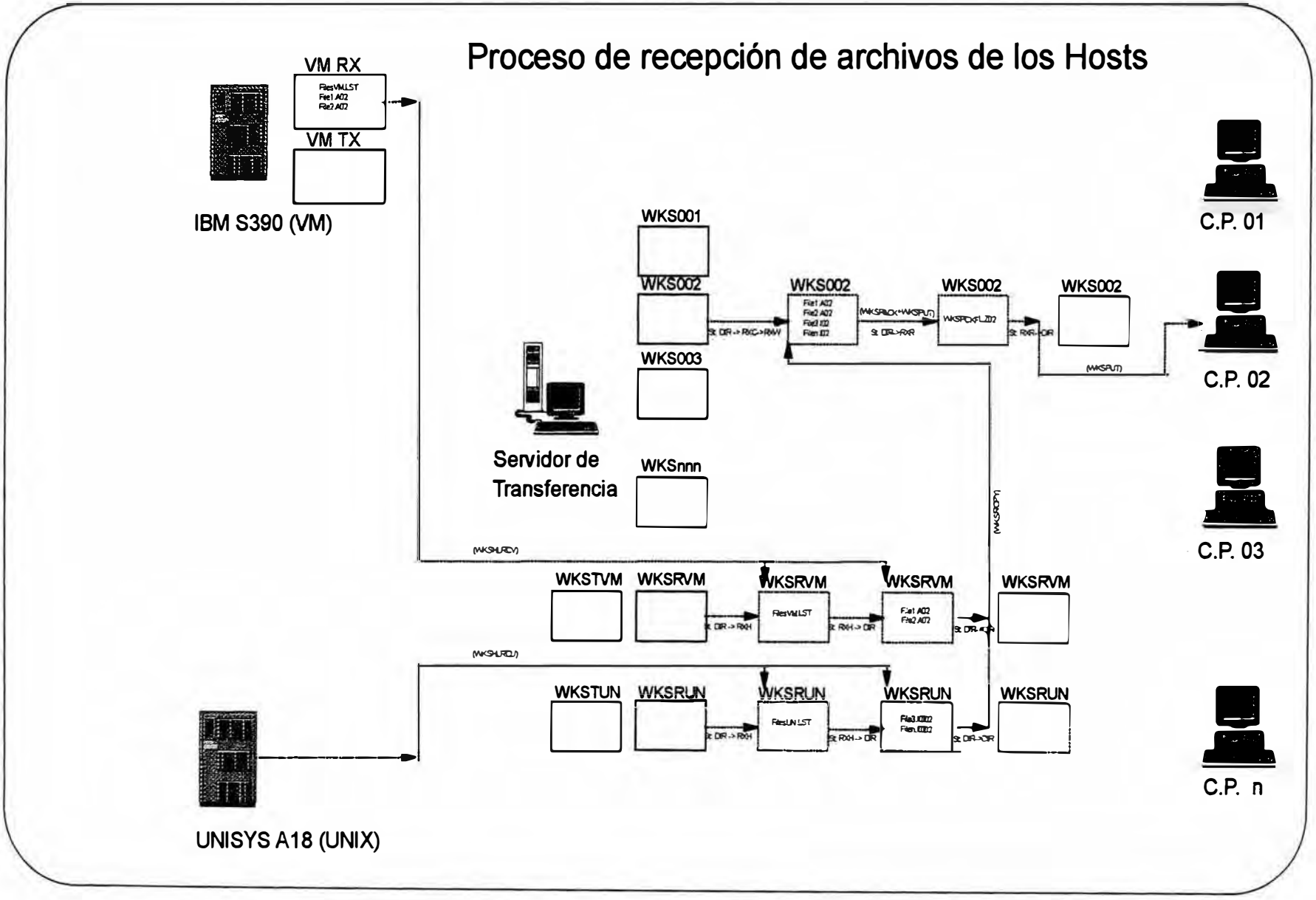
Proceso de recepción de archivos de los Hosts
VM RX
VM TX
IBM S390 (VM)
UNISYS A18 (UNIX)
Servidor de Transferencia
WKS001
WKS002
WKS003
WKSnnn
WKS002
WKS002
WKS002
C.P. 01
C.P. 02
C.P. 03
C.P. n
WKSTVM
WKSRVM
WKSRVM
WKSRVM
WKSRVM
WKSTUN
WKSRUN
WKSRUN
WKSRUN
WKSRUN

Figura 17

Si se diera el caso que fallara la transmisión al C.P., se terminará la ejecución del proceso WKSPUT, y se cambiará el estado del directorio de trabajo del estado RXR al estado RXW, para que se reintente posteriormente la transferencia.

3.4 Software usado

El sistema de control de transferencia fue programado en su totalidad en el lenguaje REXX, que es nativo al sistema operativo OS/2. También hace uso de utilitarios de uso general que hacen uso del protocolo APPC para establecer sesiones de comunicación entre el servidor de transferencias y los Centros de Proceso.

Para la transferencia con el sistema operativo VM, hace uso de APIs de EHLLAPI.

Utilitarios Usados:

- **AFILE** Establece una sesión APPC entre dos estaciones y permite a la estación que ejecuta el comando, enviar un archivo hacia la estación contactada, o transferirse un archivo desde aquella.
- **AREXEC** Establece una sesión APPC entre dos estaciones y permite a la estación que ejecuta el comando, arrancar un proceso en la estación contactada.

  PKZIP Reune un conjunto de archivos en un solo archivo, empaquetándolos y compactándolos.
- **PKUNZIP** Ejecuta la función inversa al PKZIP.
- **TERSE** Comprime un archivo en PC y descomprime en VM
- **RXCOM** Utilitarios de comunicaciones para REXX

Relación de Programas:

- **WKSRES** Restauración del Sistema.
- **WKSMON** Programa monitor del sistema de transferencia de archivos.
- **WKSPACK** Empaquetador de archivos a transmitir.
- **WKSGET** Recepción de archivos desde el Centro de Proceso hacia el servidor de transferencias.
- **WKSTCPY** Distribución de archivos hacia directorios de transferencia al VM y UNIX.
- **WKSUPCK** Desempaquetador de los archivos transmitidos.
- **WKSRCPY** Selección de archivos a transmitir a un Centro de Proceso desde los directorios de recepción del VM y UNIX.
- **WKSPUT** Envío de archivos desde servidor de transferencias hacia Centro de Proceso.
- **WKSNTFY** Notifica al operador del Servidor de Transferencias o a los operadores de los C.P. la conclusión de algún evento de transferencia de archivos o el acontecimiento de algún error en el sistema.
- **WKSFLLOG** Grabación de log de transferencias de archivos.
- **WKSERLOG** Grabación de log de errores del sistema.
- **WKSHLLOG** Conexión de las sesiones de emulación del sistema operativo VM.
- **WKSHLSNV** Envío de archivos al VM desde el directorio WKSTVM.
- **WKSHLRCV** Recepción de archivos del VM al directorio WKSRVM.
- **WKSHLSNU** Envío de archivos al UNIX desde el directorio WKSTUN.

- **WKSHLRCU** Recepción de archivos del UNIX al directorio WKSRUN.
- **WKSHLCOM** Ejecución de comandos en las sesiones de emulación de VM.
- **WKSRB28** Recibe en el C.P. el archivo envíado desde el B28 (UNISYS).

3.4.1 Descripción de los utilitarios

AFILE

AFILE provee de operaciones de transferencia básica de archivos.

En un simple comando, AFILE transfiere el archivo original desde la computadora origen hacia el archivo de destino en la computadora de destino.

El programa AFILE consiste de dos extremos: el lado de la computadora cliente y el lado de la computadora servidora. En el lado de la computadora cliente, el usuario inicia el programa AFILE y los parámetros correspondientes. Como resultado, el programa AFILED es iniciado en el lado de la computadora servidora (a menos que se especifique otro nombre de transacción con el parámetro -t). Luego los programas AFILE y AFILED se comunican usando APPC y completan la transacción.

Para que la computadora cliente y la computadora servidora puedan "hablar" una con otra, ambas computadoras deben ser configuradas. Esta configuración involucra la definición de cierta información APPC al programa AFILE y en la plataforma APPC de las computadoras. AFILE, en ambiente OS/2, se

configura en el Communications Manager/2.

La información específica que se debe saber de AFILE para esta configuración es:

TP_NAME : AFILED

Nombre Ejecutable : AFILED.EXE

Nombre del modo de ejecución : #INTER

Formato:

AFILE *lu_origen:archivo_origen lu_destino:archivo_destino*

[-a/-b -m *nombre_modo* -t *nombre_tp* -n -u *usuario* -p *contraseña*]

Parámetros:

AFILE usa indicadores para especificar las características de transferencia y opciones de seguridad. Los indicadores pueden ser especificados solo una vez.

lu_origen La dirección en la red en la que el archivo a ser transferido se localiza.

La dirección de red puede ser un nombre de LU (Unidad Lógica) totalmente especificado o un nombre simbólico CPI-C (Common Program Interface for Comunications).

Si no se ha especificado una *lu_origen*, el archivo de origen se asume como un archivo local.

archivo_origen El nombre del archivo que será copiado.

lu_destino La dirección de red en la que el archivo será copiado. La especificación de esta dirección es semejante a la de la *lu_origen*.

Si no se ha especificado una *lu_destino*, el archivo de destino se asume como un archivo local.

archivo destino El nombre del archivo destino.

-a Transmisión ASCII. Es el modo de transmisión por omisión.

-b Transmisión Binaria

-m Nombre de la modalidad de conversación APPC. El valor por omisión es #BATCH.

-t Nombre de la transacción a ejecutar en el servidor. El valor por omisión es AFILED.

-n Sin seguridad. Fuerza al AFILE a no hacer uso de seguridad en la conversación (seguridad CPI-C=NONE).

-u Usuario; el usuario en la máquina contactada. Si se especifica un usuario, también se deberá especificar una contraseña.

-p Contraseña; el código de seguridad asociado al usuario en la máquina contactada.

AREXEC

AREXEC debe su nombre a APPC Remote EXECution (Ejecución Remota en APPC).

AREXEC permite ejecutar cualquier comando en una estación de trabajo remota.

Toda salida a stdout o stderr será dirigida a la pantalla de la estación ejecutora.

Este programa no está diseñado para trabajar con comandos que requieren del ingreso de información por parte del usuario o que trabajan en pantalla completa.

AREXEC se compone de dos programas de transacción: AREXEC, el cual corre en el lado cliente, y AREXECD, el cual corre en el lado servidor.

Para configurar AREXEC en ambiente OS/2 se deberá accesar

a la parte correspondiente de configuración APPC en el Communications Manager/2.

La información específica que se debe saber de AREXEC para esta configuración es:

TP_NAME : AREXECD

Nombre Ejecutable : AREXECD.EXE

Nombre del modo de ejecución : #BATCH

Formato:

AREXEC [parámetros opcionales] *destino comando*

El *destino* y el *comando* son los únicos parámetros requeridos. Los demás parámetros son opcionales.

Parámetros:

destino Identifica la computadora en la cual se ejecuta el programa AREXECD. Puede ser una dirección APPC o una dirección simbólica CPI-C.

comando El comando (cadena de caracteres) que será ejecutado en la computadora destino.

-m *mode_name* Modalidad de comunicación. (Por omisión: #INTER)

-t *tp_name* El TP: programa transaccional a ejecutar en la computadora contactada. (Por omisión: AREXECD)

-u *usuario* El usuario con el que se conectará en la computadora contactada.

-p *contraseña* Contraseña de conexión a la computadora contactada.

-n Este parámetro fuerza a AREXEC a no usar seguridad en la conversación.

PKZIP

Este utilitario permite reunir un conjunto de archivos en uno solo, con el contenido de cada uno de ellos compactado con el algoritmo que mejor se aplique a su estructura.

Formato usado:

PKZIP -m *archivo_o* *archivos_i*

Parámetros:

-m borra los archivos después de incluirlos en el archivo empaquetado

archivo_o Archivo empaquetado

archivos_i Archivos a empaquetar

PKUNZIP

Este utilitario permite recuperar los archivos que fueron empaquetados y compactados por el utilitario PKZIP.

Formato usado:

PKZIP -o *archivo_i* *directorio*

Parámetros:

-o regrabar sobre los archivos existentes con el mismo nombre

archivo_i Archivo empaquetado

directorio Directorio en el cual se recuperarán los archivos empaquetados

TERSE

Este utilitario provee de una compactación muy efectiva y puede ser usado en adición al PKZIP, aunque no dará un beneficio significativo el uso en conjunto de ambas herramientas.

Su uso será muy efectivo en el momento de la transmisión de archivos al VM, dado que en VM se cuenta con la misma herramienta capaz de descompactar la información del TERSE de la PC. Ésto también es válido para la compactación en el VM y su posterior descompactación en la PC.

El uso de esta herramienta es opcional para el sistema y la ventaja principal se dará en el tiempo de transferencia desde y hacia el VM.

Formato usado:

TERSE *archivoi* *archivoo* -c -q

Parámetros:

- Para la opción de compactación:

  archivoi: Archivo a compactar

  archivoo: Nuevo archivo compactado. Si no se especifica *archivoo*, *archivoi* será compactado con el mismo nombre.

  -c: Opción de compactación

  -q: Opción de eliminación de mensajes informativos a la pantalla

- Para la opción de descompactación:

  archivoi: Archivo a descompactar

  archivoo: Archivo descompactado. Si no se especifica *archivoo*, *archivoi* será descompactado con el mismo nombre.

-d: Opción de descompactación

-q: Opción de eliminación de mensajes informativos a la pantalla.

RXCOM

RXCOM es un utilitario que provee de una librería de funciones de soporte de comunicación con pórticos seriales al REXX.

Este utilitario es usado por el programa WKSRB28 para recibir en los C.P. los archivos envíados desde el computador B28 (UNISYS) a través del "null modem" que los une.

Las funciones que añade al REXX son las siguientes:

RxComOpenPort: Abrir un puerto serial (p.e. COM1) y obtener un identificador de éste.

Formato: rc = RxComOpenPort(Nombre_de_Puerto, Identificador)

RxComClosePort: Cerrar un puerto serial.

Formato: rc = RxComClosePort(Identificador)

RxComQueryLineControl: Obtención de información de control: baudios, tamaño de armazón (frame size), paridad y bits de parada.

Formato: control = RxComQueryLineControl(Identificador)

RxComSetLineControl: Especificación de la información de control: baudios, tamaño de armazón (frame size), paridad y bits de parada.

Formato: rc = RxComSetLineControl(Identificador, Baudios, Frame, Paridad, StopBits)

RxComSetReadMode: Especificar la modalidad de lectura en fija o variable para las subsiguientes lecturas.

Formato: rc = RxComSetReadMode(Identificador, modo, som, eom, scm, trail, transp)

Descripción de parámetros:

Identificador: Identificador obtenido de un RxComOpenPort previo.

- *modo*: modalidad de lectura: 'FIXED' o 'VARIABLE'

Para la modalidad 'VARIABLE' se usan los siguientes parámetros:

- *som*: Inicio de la secuencia de mensaje. Consiste en una cadena de máximo 4 caracteres que identifican el inicio del mensaje. Puede no ser especificado.
- *eom*: Secuencia de fin de mensaje. Es una cadena de máximo 4 caracteres que indentifican el final del mensaje. Es obligatorio para la modalidad de lectura de archivos variables.
- *scm*: Mensajes de caracter único: Consiste en una cadena de caracteres en la cual cada uno conforma un mensaje completo. Estos mensajes son normalmente 'ACK' y 'NAK' o nueva línea (linefeed).

trail: Número de caracteres finales que siguen a la secuencia de fin de mensaje. Por lo general son caracteres de control.

- *trans*: Modo transparente: Se especifica por 'TRANSP' o 'NOTRANSP'. Para protocolos que pueden transferir información binaria dentro del armazón del mensaje, es importante que ninguna porción de la información sea confundida con el mensaje de *eom*. Por lo tanto, siempre que el primer caracter de la secuencia de *eom* aparezca en la porción de información, se insertará un caracter adicional por el transmisor. El receptor deberá luego eliminar el caracter adicional durante o después de recibir el mensaje.

RxComWrite: Envío de una cadena de caracteres al puerto serial.

Formato: rc =RxComWrite(Identificador, Cadena)

RxComRead: Leer una cadena de caracteres del puerto serial.

Formato: rc = RxComRead(Identificador, Dest, Long, Timeout)

Descripción de parámetros:

Identificador: Identificador de un RxComOpenPort previo.

- *Dest*: Nombre de la variable donde será colocada la información recibida.

Long: Longitud a leer en bytes.

Timeout: Tiempo en milisegundos para completar la lectura.

3.4.2 Descripción de los programas

WKSRES

Este programa inicializa el directorio de monitoreo (WKSMON) y los directorios de trabajo. Se ejecuta durante la instalación del sistema para la creación de los directorios de trabajo y del directorio de monitoreo (wksmon) con el parámetro COLD y se ejecuta cada vez que se inicia el proceso principal del sistema (WKSMON) para hacer una reinicialización de los estados de cada directorio de trabajo con el parámetro WARM.

Formato usado:

WKSRES *t* *n*

Parámetros:

t: Tipo de reinicio del sistema:

- COLD: Se inicializan los directorios de trabajo existentes, borrando su contenido y cambiando su estado a DIR en el directorio de monitoreo (wksmon). Se crean los directorios de trabajo faltantes hasta completar *n* directorios de trabajo.

  También se inicializa el directorio de los LOGs del sistema, borrando su contenido, por lo cual se deberá considerar sacarles una copia de respaldo antes de ejecutar este programa.
- WARM: Se reinicializan los directorios de trabajo cambiando su estado en el monitor del sistema al estado que le permita continuar el proceso que fue interrumpido en la última detención del sistema.

  Los estados TXH, TXR y RXH son cambiados al estado DIR,

para permitir que el sistema reinicie los procesos de transmisión a Host, recepción del Host y transmisión desde los C.P.

El estado RXR es cambiado al estado RXW para que se reinicie el proceso de envío al C.P. correspondiente.

El estado TXC es cambiado al estado TXW para que se reinicie el proceso de distribución de archivos hacia los directorios de transmisión al VM y UNIX.

El estado RXC es cambiado al estado RXW para que se inicie la transferencia hacia el C.P. correspondiente. Si quedaron archivos correspondientes a ese C.P. sin ser seleccionados por el WKSRCPY, para su envío, por algún problema o la detención del sistema, éstos serán seleccionados en la siguiente ejecución del WKSRCPY y finalmente llegarán a su destino.

n: Indica el número de directorios de trabajo con que operará el sistema. Este número de directorios se refleja directamente en el número de procesos que pueden ser arrancados en forma simultánea en el sistema de transferencia.

WKSMON

Este programa controla el arranque de los procesos de transferencia en el sistema desde y hacia los C.P. y Hosts.

El programa hace un barrido de todos los C.P. (Centros de Proceso) inscritos en el archivo de control WKSTN.LST en un lazo sin fin, arrancando en forma consecutiva procesos de recepción de información desde el C.P. correspondiente, procesos de distribución de archivos hacia los directorios de

transmisión a Host, siempre que exista algún directorio de trabajo pendiente de transmisión al Host, procesos de selección de archivos desde los directorios de recepción de los Hosts para su distribución a los C.P., envío de los archivos a los C.P. correspondientes ante la existencia de un directorio de trabajo pendiente de envío a C.P. y procesos de recepción y envío de archivos desde y hacia los Hosts.

Este programa ejecuta el programa WKSRES con el parámetro WARM para ejecutar una reinicialización normal del sistema.

Formato usado:

WKSRES

Parámetros:

No hay parámetros de ejecución.

WKSPACK

El programa WKSPACK tiene como función el empaquetamiento de los archivos a ser transmitidos, en uno solo, además de su previa compactación.

Admite dos tipos de compactación: la proporcionada por el PKZIP y la proporcionada por el TERSE, según los parámetros que se le especifiquen.

Este programa es ejecutado en remoto en el C.P. antes de transferir sus archivos al servidor de archivos (desde el proceso WKSGET) y también es ejecutado en el servidor de archivos antes de transferir sus archivos al C.P. correspondiente (desde el proceso WKSPUT).

Este programa no intentará empaquetar más de una vez los archivos contenidos en el directorio de trabajo en caso que detectara la existencia de un archivo previamente

empaquetado. Esto permite la ejecución de un reproceso en caso de error.

Formato usado:

WKSPACK *dir packfile ext flag*

Parámetros:

dir: Directorio cuyos archivos serán empaquetados

packfile: Nombre asignado al archivo empaquetado

ext: Extensión a darle al archivo empaquetado

flag: Opción de empaquetamiento:

- P: Se compactarán y empaquetarán los archivos con PKZIP
- T: Se compactarán los archivos con TERSE
- B: Se compactarán (TERSE) y se empaquetarán (PKZIP) los archivos.

WKSGET

El WKSGET ejecuta la transferencia de archivos desde un C.P. hacia el servidor de transferencia de archivos. Durante su ejecución, se cambia el estado del directorio de trabajo desde DIR hacia TXR. Al finalizar la transferencia, si ésta terminó bien, se cambia el estado del directorio de trabajo a TXW. En caso que hubiera terminado mal, se regresaría el estado de éste a DIR.

Este programa ejecuta los utilitarios AFILE y AREXEC, y ejecuta también los programas WKSPACK y WKSUPCK.

Este programa es arrancado como un proceso independiente desde el programa WKSMON.

Formato de uso:

WKSGET *wksid wrkdir ex tsk*

Parámetros:

wksid: Identificación del C.P. remoto

wrkdir: Directorio de trabajo hacia el cual enviar los archivos

ex_tsk: Indicador de cierre de sesión

- *ex_tsk* <> "": Se cerrará la sesión OS/2 al finalizar la ejecución del proceso
- *ex_tsk* = "": La sesión de OS/2 no será finalizada (útil durante la ejecución de pruebas)

WKSTCPY

La función del programa WKSTCPY es la de distribuir los archivos de un directorio de trabajo (después de la ejecución de un WKSGET) hacia los directorios de transmisión al VM y UNIX (WKSTVM y WKSTUN respectivamente). La distinción del destino de cada archivo se logra debido a que el primer caracter de la EXT de cada uno de ellos determina el Host al cual está dirigido (A para VM e I para el UNIX).

Durante la ejecución de WKSTCPY el estado del directorio cambiará a TXC. Al finalizar la ejecución de este programa, si su ejecución fue satisfactoria, el estado del directorio de trabajo será cambiado a DIR. En caso contrario, se le regresará al estado TXW para posibilitar el arranque de un posterior WKSTCPY que termine la tarea iniciada por el anterior.

Este programa es ejecutado desde el proceso WKSMON tan pronto éste detecta la existencia de un directorio de trabajo con el estado TXW.

Formato de uso:

WKSTCPY *wrkdir* *ex_tsk*

Parámetros:

wrkdir: Directorio desde donde copiar los archivos

ex_tsk: Indicador de cierre de sesión

- *ex_tsk* <> "": Se cerrará la sesión OS/2 al finalizar la ejecución del proceso
- *ex_tsk* = "": La sesión de OS/2 no será finalizada (útil durante la ejecución de pruebas)

WKSUPCK

Tanto en las transmisiones hacia los C.P. desde el servidor de transferencias, como en las transmisiones que vienen desde los C.P. al servidor de transferencias, se transmiten todos los archivos en un solo archivo compactado y empaquetado.

El programa WKSUPCK tiene como función el desempaquetamiento de los archivos al ser recibidos en su destino.

Este programa es ejecutado en remoto en el C.P. después de haber concluido la transferencia del archivo empaquetado (desde el proceso WKSPUT) y es ejecutado en el servidor de transferencia, después de haber recibido éste el archivo empaquetado transmitido desde el C.P correspondiente (proceso WKSGET).

Formato usado:

WKSUPCK *dir* *packfile* *ext* *flag*

Parámetros:

dir: Directorio cuyos archivos serán desempaquetados

packfile: Nombre del archivo empaquetado

ext: Extensión del archivo empaquetado

flag: Opción de desempaquetamiento:

- P: Se desempaquetarán los archivos (PKUNZIP)
- T: Se descompactarán con TERSE los archivos
- B: Se descompactarán (TERSE) y se desempaquetarán (PKUNZIP) los archivos.

WKSRCPY

La función de este programa es la de mover archivos desde los directorios de recepción de los Hosts (WKSRVM y WKSRUN) hacia algún directorio de trabajo para su posterior envío al C.P. correspondiente. Cada archivo que es copiado al directorio de trabajo es borrado del directorio de recepción origen.

La forma de selección es la siguiente: Se obtiene un listado del contenido de ambos directorios de recepción de Host y se toma el primer archivo de la lista. Se obtiene el C.P. de destino (últimos dos dígitos de la extensión del archivo) y se selecciona de la lista todos aquellos archivos que tengan el mismo destino. Éstos son los archivos que serán movidos del directorio de recepción de Host al directorio de trabajo.

Formato de uso:

WKSRCPY *wrkdir ex tsk*

Parámetros:

wrkdir: Directorio de trabajo a donde mover los archivos

ex_tsk: Indicador de cierre de sesión

- *ex_tsk* <> "": Se cerrará la sesión OS/2 al finalizar la ejecución del proceso
- *ex_tsk* = "": La sesión de OS/2 no será finalizada (útil durante la ejecución de pruebas)

WKSPUT

El WKSPUT ejecuta la transferencia de archivos desde el servidor de transferencia de archivos hacia el C.P. correspondiente.

Durante su ejecución, se cambia el estado del directorio de trabajo desde RXW hacia RXR.

Al finalizar la transferencia, si ésta terminó bien, se cambia el estado del directorio de trabajo a DIR. En caso que hubiera terminado mal, se regresaría el estado de éste a RXW para reintentar la transmisión. Este programa ejecuta los utilitarios AFILE y AREXEC, y ejecuta también los programas WKSPACK y WKSUPCK.

Este programa es arrancado como un proceso independiente desde el programa WKSMON.

Formato de uso:

WKSPUT *wrkdir* *ex_tsk*

Parámetros:

wrkdir: Directorio de trabajo desde el cual se enviarán los archivos

ex_tsk: Indicador de cierre de sesión

- *ex_tsk* <> "": Se cerrará la sesión OS/2 al finalizar la ejecución del proceso
- *ex_tsk* = "": La sesión de OS/2 no será finalizada (útil durante la ejecución de pruebas)

WKSNTFY

Este programa tiene como función la notificación de eventos a los C.P. y al mismo servidor de transferencia. Los eventos notificados pueden ser la correcta recepción de archivos transmitidos, o la ocurrencia de cualquier error, que deba ser notificado. La notificación se manifiesta en la forma de una ventana en la estación a notificar.

Esta función es ejecutada desde todos los procesos que deban realizar alguna notificación.

Formato de uso:

WKSNTFY *tipo* *msg*

Parámetros:

tipo: Tipo de ventana a mostrar:

- INF: Informativa
- WNG: Advertencia
- ERR: Error

msg: Mensaje a ser mostrado

WKSFLLOG

Genera un Log de cada transmisión realizada o fallida desde y hacia el servidor de transferencia.

Formato de uso:

WKSFLLOG *pgnam* *flog* *msg*

Parámetros:

pgnam: Nombre del programa que origina el Log

flog: Nombre del archivo del Log

msg: Mensaje a grabar en el Log

WKSERLOG

Genera un Log de errores del sistema. En caso de querer notificarlos al operador del servidor de transferencias, hará uso de la función WKSNTFY, además de grabar el error en el Log de errores.

Formato de uso:

WKSERLOG *pgnam sever msg*

Parámetros:

pgnam: Nombre del programa que reporta el error

sever: Severidad del error

- *sever* = 0: No hay notificación del error.
- *sever* > 0 y *sever* < 4: Se genera notificación informativa.
- *sever* >= 4 y *sever* < 8: Se genera notificación de advertencia.
- *sever* >=8: Se genera notificación de error.

msg: Texto explicativo del error.

WKSHLLOG

Este programa es ejecutado durante la carga final del sistema operativo del servidor de archivos y su función es la de conectar las dos sesiones del sistema operativo VM. Se comunica con el VM por medio del EHLLAPI.

Formato de uso:

WKSHLLOG *userid password session*

Parámetros:

userid: Usuario a conectar en la sesión

password: Contraseña del usuario a conectar

session: Sesión del VM a usar (A o B)

WKSHLSNV

El programa WKSHLSNV se encarga de la transferencia al sistema operativo VM (sesión A) de todos los archivos que se encuentren en el directorio WKSTVM. La transferencia se realiza haciendo uso de funciones del EHLLAPI.

Formato de uso:

WKSHLSNV *ex_tsk*

Parámetros:

ex_tsk: Indicador de cierre de sesión

- *ex_tsk* <> "": Se cerrará la sesión OS/2 al finalizar la ejecución del proceso
- *ex_tsk* = "": La sesión de OS/2 no será finalizada (útil durante la ejecución de pruebas)

WKSHLRCV

El programa WKSHLRCV transferirá desde la sesión B de VM todos aquellos archivos que se encuentren en la READER de la misma, hacia el directorio de trabajo WKSRVM.

El proceso ejecuta en la sesión B de VM el programa WKSRDR, el cual hace una lectura de todos los archivos pendientes de ser recibidos que se encuentran en la READER. Se genera en VM un archivo de nombre FILESVM.LST el cual contiene la lista de archivos a ser recibidos de la READER de VM y transmitidos al servidor de transferencias, junto con su

correspondiente número de SPOOL (para poder recibirlos en VM).

Una vez transmitido el archivo FILESVM.LST al servidor de transferencias, se procederá a recibir y transmitir cada uno de los archivos presentes en la READER de la sesión B del VM.

La transferencia se realiza haciendo uso de funciones del EHLLAPI.

Formato de uso:

WKSHLRCV *ex_tsk*

Parámetros:

ex_tsk: Indicador de cierre de sesión

- *ex_tsk* <> "": Se cerrará la sesión OS/2 al finalizar la ejecución del proceso
- *ex_tsk* = "": La sesión de OS/2 no será finalizada (útil durante la ejecución de pruebas)

WKSHLSNU

El programa WKSHLSNU se encarga de la transferencia al sistema operativo UNIX de todos los archivos que se encuentren en el directorio WKSTUN. Para ejecutar la transferencia de archivos, se hace uso del utilitario AFILE.

Formato de uso:

WKSHLSNV *ex_tsk*

Parámetros:

ex_tsk: Indicador de cierre de sesión

- *ex_tsk* <> "": Se cerrará la sesión OS/2 al finalizar la ejecución del proceso

- *ex_tsk* = "": La sesión de OS/2 no será finalizada (útil durante la ejecución de pruebas)

WKSHLRCU

El programa WKSHLRCU transferirá desde el UNIX todos aquellos archivos que se encuentren en la lista FILESUN.LST, hacia el directorio de trabajo WKSRUN. El archivo FILESUN.LST se genera en el mismo UNIX y es transferido inicialmente al servidor de transferencias para poder conocer los nombres de los archivos a transmitir.

Este archivo se genera y transmite al servidor de archivos al ejecutar el utilitario AFILE solicitando la transferencia desde el UNIX al servidor del archivo 'DIR'.

Formato de uso:

WKSHLRCU *ex_tsk*

Parámetros:

ex_tsk: Indicador de cierre de sesión

- *ex_tsk* <> "": Se cerrará la sesión OS/2 al finalizar la ejecución del proceso
- *ex_tsk* = "": La sesión de OS/2 no será finalizada (útil durante la ejecución de pruebas)

WKSHLCOM

Este programa ejecuta un comando en la sesión VM especificada.

La sesión deberá encontrarse en el estado VM READ o RUNNING, pues de lo contrario, la ejecución del comando será fallida (y se notificará de ésto al operador del sistema de transferencias).

Formato de uso:

WKSHLCOM *commd session*

Parámetros:

commd: Comando a ejecutar

session: Sesión de VM a usar

WKSRB28

Este programa se ejecuta en cada C.P. cuando se desea transferir un archivo desde el B28 conectado a éste. El B28 ejecuta el sistema operativo BTOS y en él corre el programa ATE (Asynchronus Transfer Emulation), el cual se comunica por medio de un null modem al C.P. y conversa con el programa WKSRB28, el cual recibirá en el C.P. cualquier archivo envíado desde el B28.

Formato de uso:

WKSRB28

Parámetros:

No hay parámetros.

3.5 Interelación de los componentes del sistema

3.5.1 Arquitectura de Sistemas Abiertos de IBM (*IBM Open Blueprint)*

El Open Blueprint es una arquitectura basada en estándares que define los servicios requeridos por las aplicaciones en un ambiente distribuído y heterogéneo (conformado por equipos de múltiples proveedores). Se representa gráficamente por un conjunto de bloques, cada uno representando funciones que proveen los servicios requeridos por las aplicaciones en los ambientes distribuídos y heterogéneos (figura 18).

Los bloques en el diagrama del Open Blueprint no corresponden a productos específicos. El Open Blueprint está implementado por productos distintos en distintas plataformas. El Open Blueprint describe técnicas para construir un sistema abierto, heterogéneo y distribuído que puede ser expandido o modificado por componentes alternos y/o por nuevas funciones y tecnologías.

Como se ve en la figura 18, existen varios conjuntos de servicios en el Open Blueprint:

- **Servicios de red**
  - **Semántica común de transporte:** soporta comunicaciones independientes del protocolo en redes distribuídas.
  - **Servicios de transporte:** provee los protocolos para el transporte de información de un sistema a otro, tales como SNA/APPN, TCP/IP, OSI, NETBIOS e IPX.
  - **Sub-redes:** proveen funciones relacionadas con facilidades de transmisión, tales como varios tipos de LANs, WANs, canales, ATM y tecnologías emergentes, tales como inalámbrica.
- **Servicios de sistemas distribuidos**
  - **Servicios de comunicación:** provee los mecanismos para que partes de aplicaciones distribuídas o de manejo de recursos del sistema, puedan conversar unas con otras.
  - **Servicios de administración de objetos:** provee de servicios para objetos comunes, tales como acceso transparente a objetos locales o remotos.

- **Servicios distribuídos:** asisten en la comunicación entre partes de una aplicación distribuída y los manejadores de recursos del sistema, al proveer de funciones comunes tales como directorio y seguridad.

- **Habilitación de aplicaciones**
  - **Servicios de presentación:** definen la interacción entre las aplicaciones y el usuario.
  - **Servicios de aplicación:** son funciones comunes tales como correo electrónico, que están disponibles para ser usadas por todas las aplicaciones.
  - **Servicios de acceso de datos:** permiten a las aplicaciones y manejadores de recursos el interactuar con distintos tipos de almacenamiento de información.
- **Administración del sistema:** proveen las facilidades a un administrador de sistemas o procedimientos automatizados para el manejo del sistema operativo de la red.

Servicios locales de sistemas operativos: operan dentro de los confines de un único sistema en la red. Ejemplos son el manejo de memoria y el despacho de trabajo.

Las herramientas de trabajo y los servicios del Open Blueprint ayudan al desarrollador de aplicaciones el implementar aplicaciones distribuídas que usen interfases estándares.

Aplicaciones
y
Servicios de
Habilitación de
Aplicaciones

Servicios de
Sistemas
Distribuidos

Servicios
de Red

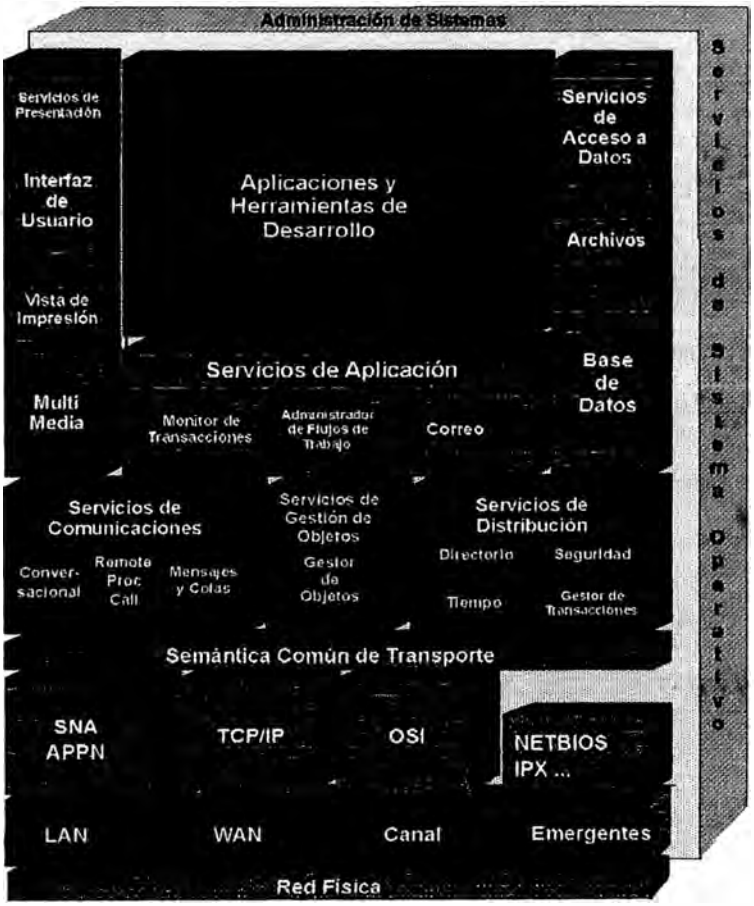

Figura 18

Para el Open Blueprint un sistema es parte de una red distribuída y la red es en sí un único sistema. Lo último conlleva a que el Open Blueprint permita el funcionamiento de una red de sistemas operativos como si se tratara de una unidad, de un "sistema operativo de la red". Un sistema operativo de red está compuesto por muchos sistemas separados uno del otro y conectados por una red de comunicación. Los servicios en cada sistema manejan los recursos en forma cooperativa a través de la red en la misma forma como un sistema operativo aislado y único maneja sus propios recursos en una sola plataforma. Cada nodo en la red puede pensarse como una estructura según el Open Blueprint La figura 19 representa una red en la cual cada uno de sus componentes puede ser considerado como una sola estructura, de acuerdo con el Open Blueprint. Esta figura representa la verdadera naturaleza distribuída del Open Blueprint.

El Open Blueprint provee las guías de integración de sistemas heterogéneos y la simplificación de los aspectos más complicados de la computación distribuída, tales como múltiples conexiones, múltiles contraseñas y directorios de aplicaciones únicos para la ubicación de recursos. Los productos que siguen las reglas del Open Blueprint proveen interfases y protocolos adecuados. Los productos que usan los manejadores de recursos especificados en el Open Blueprint, en lugar de los suyos propios, se encuentran perfectamente integrados en el Open Blueprint. Si cada aplicación usara su propio sistema de seguridad, un usuario debiera presentar una contraseña única para cada aplicación.

Si en cambio las aplicaciones y manejadores de recursos usan los servicios de seguridad del Open Blueprint, una sola contraseña sería suficiente para las aplicaciones interactuantes. Esta integración mejora la imagen de "un único sistema" del sistema distribuído tal como la percibe el usuario final y el desarrollador de aplicaciones.

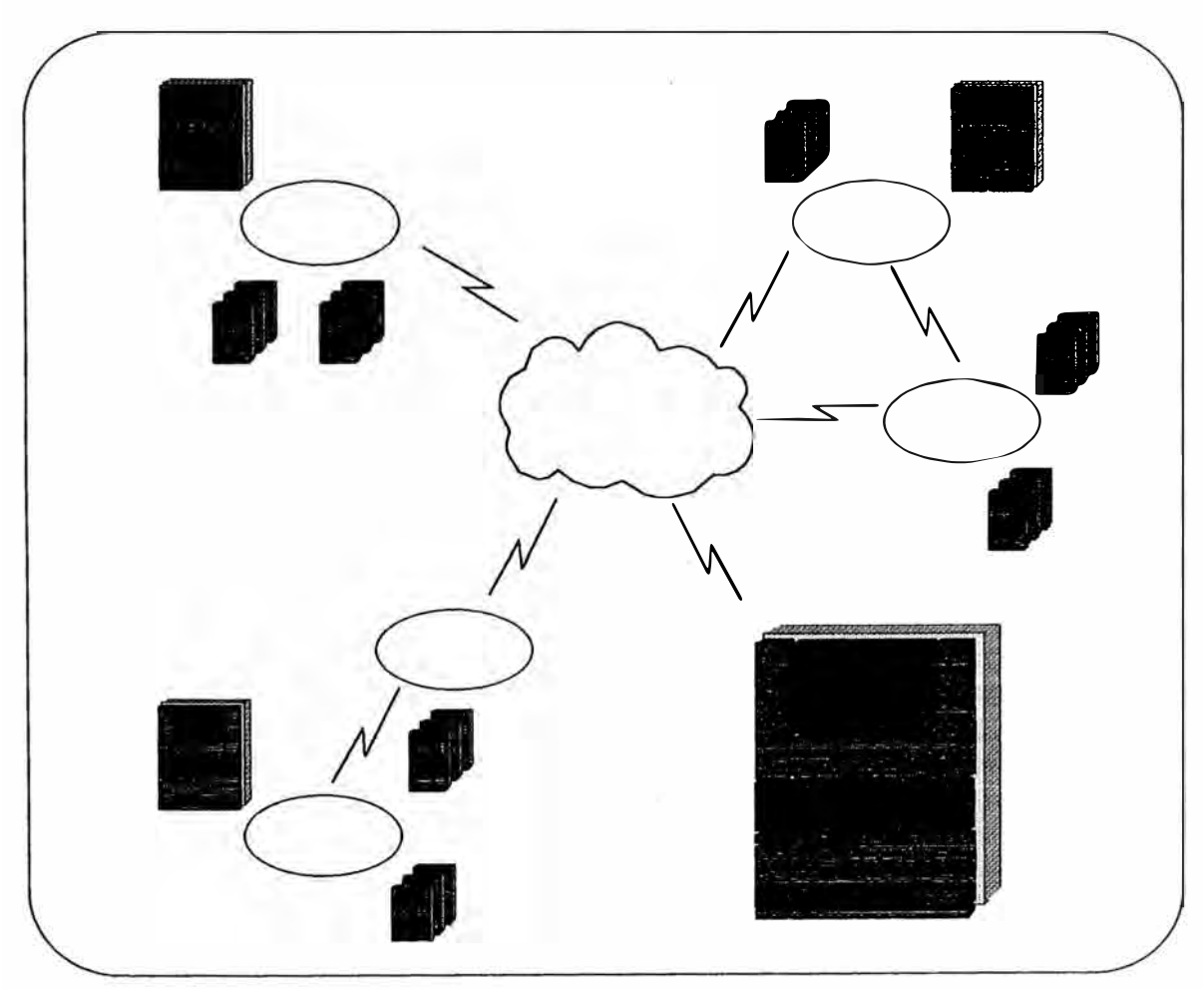

Figura 19

Todos los servicios deben estar disponibles por las aplicaciones en todas las plataformas, sin embargo, esto no significa que todos los servicios deban existir en cada una de ellas. Los servicios equivalentes en cada plataforma deben trabajar en conjunto para proveer de un soporte homogéneo y a todo lo largo y ancho de la red para las

aplicaciones distribuídas y Cliente/Servidor.

En conclusión, el Open Blueprint permite:

- a los usuarios finales, esconder las complejidades de la red y hacerles parecer ésta como un único sistema.
- a los desarrolladores de aplicaciones, el uso de interfases estándar les permitirá tener una visión de un único sistema en toda la red y además, el desarrollo de aplicaciones transportables a múltiples plataformas.
- a los administradores de sistemas, una forma consistente de administrar la red y esconder las complejidades de los desarrolladores de aplicaciones y usuarios.

3.5.2 Conectividad del sistema (caso 1: VM)

Tomando como modelo el IBM Open Blueprint y el conjunto de funciones que lo conforman, usamos el siguiente diagrama para representar la conectividad de los componentes propios del Sistema de Intercambio de Información en el caso de interconexión de los Centros de Procesos y el Host con sistema operativo VM (figura 20).

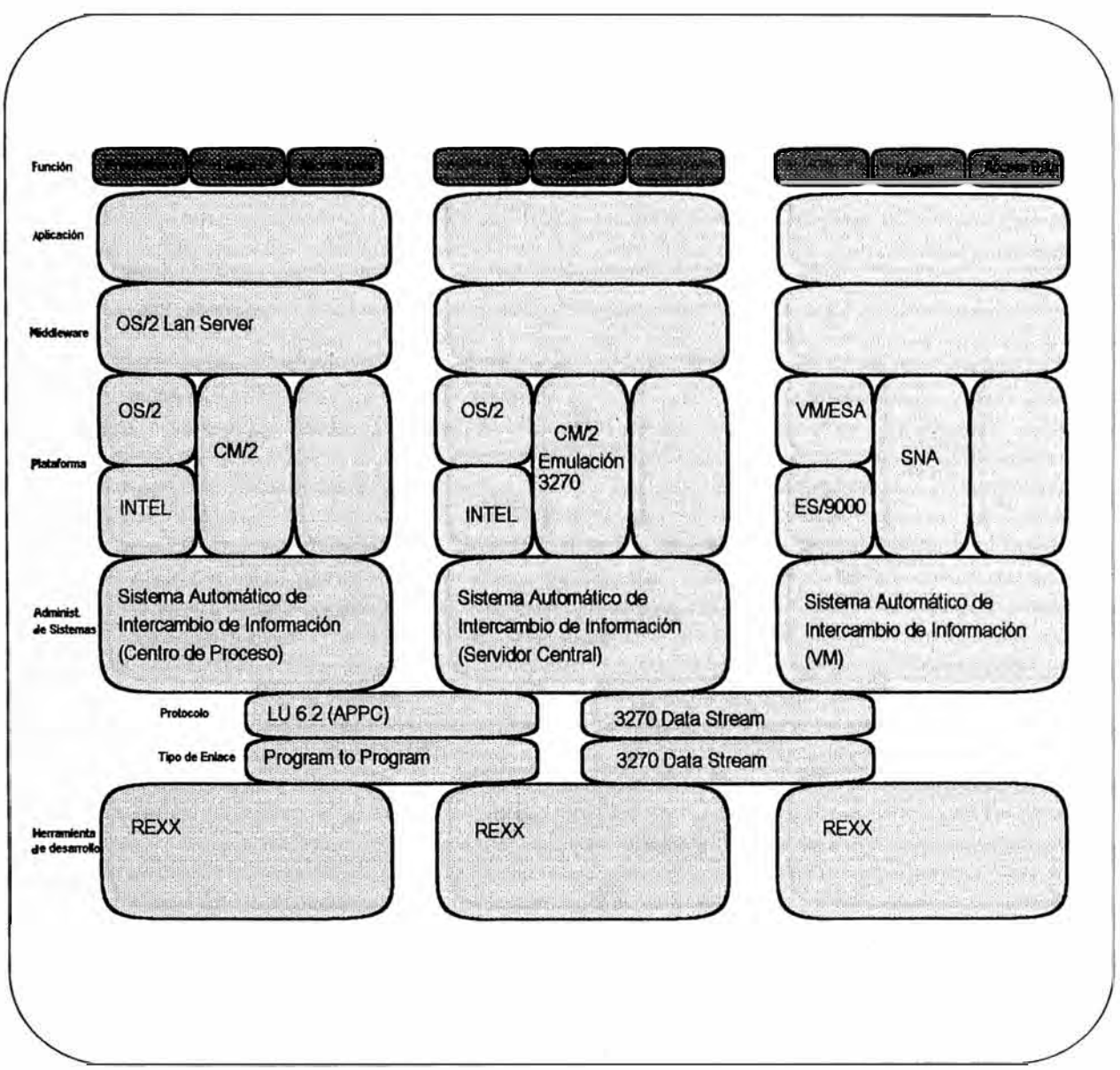

Figura 20

3.5.3 Conectividad del sistema (caso 2: UNIX)

El siguiente diagrama (figura 21) representa la conectividad de los componentes propios del Sistema de Intercambio de Información en el caso de interconexión de los Centros de Procesos y el Host con sistema operativo UNIX.

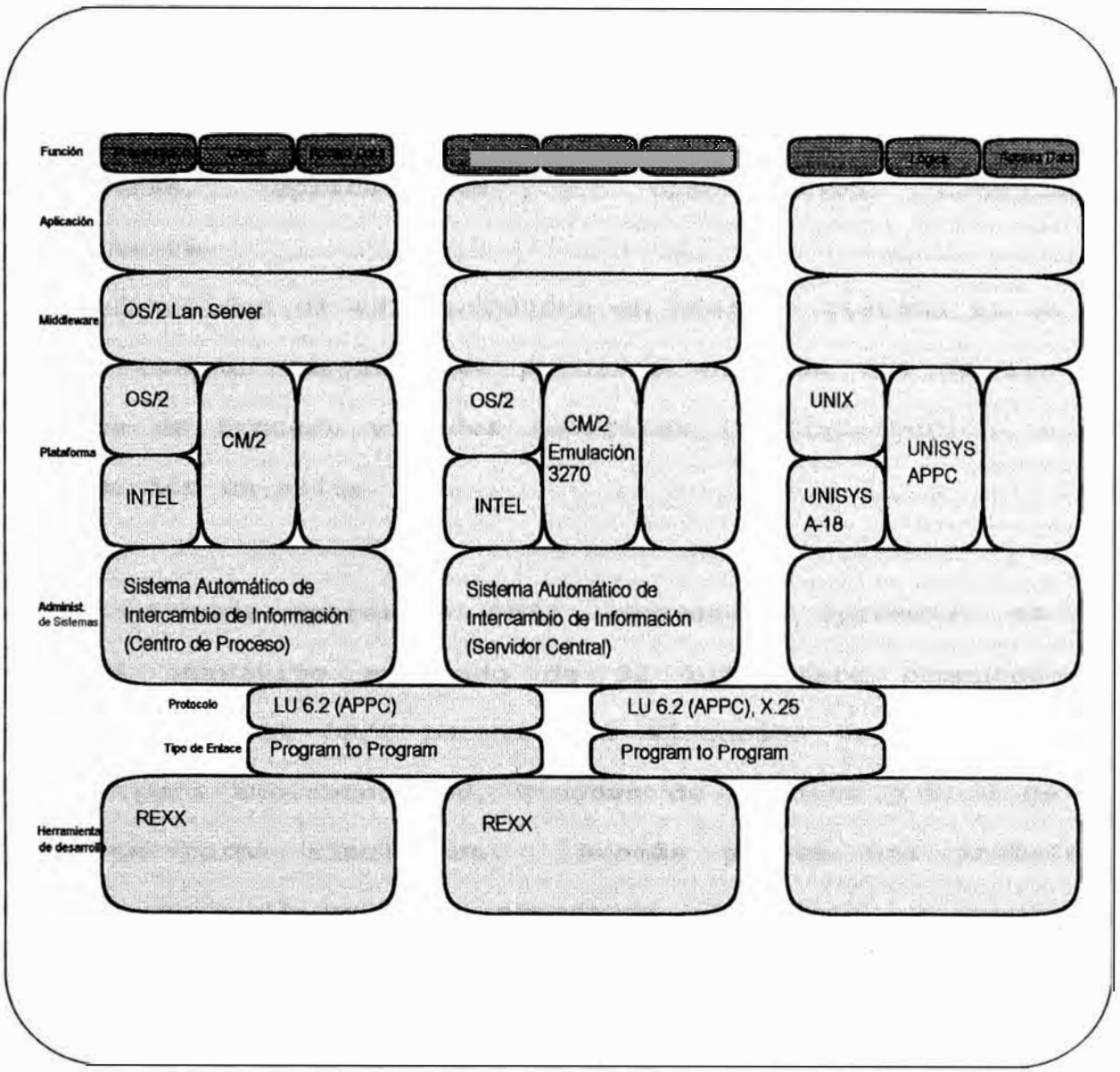

Figura 21

3.5.4 Descripción de los componentes

OS/2 LAN Server

El OS/2 LAN Server provee un ambiente de sistema operativo de red muy completo con la capacidad de compartir recursos con clientes del tipo OS/2, DOS, Windows y Macintosh. El acceso remoto a archivos y servicios de impresión en un servidor es totalmente transparente a las aplicaciones de los clientes.

Este producto opera en redes Token-Ring y Ethernet. Provee facilidades para definir, administrar y controlar el acceso a los recursos compartidos de la red tales como discos, impresoras, aplicaciones y dispositivos conectados serialmente.

La finalidad de este producto en nuestro sistema es la de contar con un servidor de archivos en cada uno de los 31 Centros de Proceso y poder facilitar la distribución de la información en ellos.

OS/2

El sistema operativo OS/2 (*Operating System/2*) es un sistema operativo avanzado de 32 bits para computadoras personales. El OS/2 permite la ejecución de aplicaciones hechas para ambientes DOS, Windows de 16 bits y OS/2 de 32 bits en forma simultánea. Además provee una interfase gráfica para el usuario, orientada al objeto, y seguridad integral del sistema operativo, la cual proteje el núcleo del sistema, aislándolo de las aplicaciones en ejecución en ese momento, al igual que aislando las mismas aplicaciones, unas de otras.

El Sistema de Intercambio de Información aprovecha la capacidad del OS/2 de ejecutar tareas en forma concurrente (multi-proceso), para poder efectuar varias tareas de transferencias de información desde y hacia distintos Centros de Proceso, al mismo tiempo.

INTEL

Se refiere a una PC IBM o compatible basada en la arquitectura de microprocesadores 80x86 (80386 en adelante)

CM/2

El Communications Manager/2 provee al sistema operativo OS/2 de servicios de comunicación avanzados, tales como emuladores de terminal 3270 y 5250, servicios de conectividad ISDN, soporte de APIs tales como EHLLAPI y CPI-C, soporte APPN de nodo final o nodo de red, APPC, etc.

Es gracias al CM/2 que podemos establecer la comunicación Program To Program entre los Centros de Proceso y el Servidor Central, y entre el Servidor Central y el UNIX. En el caso del VM, la comunicación entre el Servidor Central y este sistema operativo ser realiza por medio de emulación 3270 y EHLLAPI.

REXX

Este lenguaje de programación permite que los programas sean escritos en una forma limpia y estructurada. Provee muchas facilidades para manejo de cadenas de caracteres, asignación automática de tipo de variable y aritmética de alta precisión.

El REXX se encuentra disponible en muchos sistemas operativos, tales como VM/ESA, MVS/ESA (como parte de TSO/E), AIX/6000, OS/400, OS/2 y DOS.

VM/ESA

El VM/ESA es un sistema operativo interactivo para computadores grandes IBM.

Los usuarios tienen sus propias máquinas virtuales, las cuales aparentan estar totalmente a su disposición, pero cuyas tareas son realizadas compartiendo los recursos físicos del computador. El Programa de Control (*CP*) controla los

recursos físicos. Cada máquina virtual ejecuta su propio sistema operativo, generalmente CMS. También se pueden ejecutar otros sistemas operativos tales como VM y VSE en un ambiente VM.

ES/9000

Los computadores IBM ES/9000 representan la serie de computadores de mayor capacidad de proceso de IBM. Pueden correr sistemas operativos tales como MVS, VM y VSE.

SNA

SNA maneja los controles de transmisión, direccionamientos, control de flujo de información y servicios de presentación. Provee interfases con dispositivos del tipo LU2 y LU3 y facilidades administrativas. Permite la emulación de un terminal 3270 desde un terminal remoto en una conexión asíncrona y también permite la comunicación vía X.25 a través de una red conmutada de paquetes.

UNIX

El sistema operativo UNIX es un sistema operativo multi-proceso, paginable y de memoria virtual el cual puede operar como un sistema operativo de usuario único o como multi-usuario. Como protocolo de comunicación nativo usa el TCP/IP pero igualmente puede soportar otros protocolos tales como APPC.

UNISYS A-18

Esta serie de computador Unisys pertenece a una familia de computadores desde medianos hasta muy grandes, de propósito general. Estos procesadores compiten con los procesadores del tipo IBM ES/9000.

CONCLUSIONES

Hoy en día es muy común el encontrar deficiencias en el intercambio de información, envío de información actualizada, manejo de cambios y resolución de problemas en compañías que poseen redes de computadoras (medianas o grandes), semejantes a los encontrados en el Banco Cafetero. Problemas como éstos causan evidentes trastornos en el funcionamiento de las compañías y en los resultados que éstas esperan obtener. Se debe recordar en todo momento que las exigencias del mercado actual obligan a las empresas a responder muy rápidamente a los cambios de éste y a regirse por la política de "Calidad Total" en la prestación de sus servicios o en sus productos finales.

Esto nos trae como conclusión principal lo siguiente:

Cuando se opera una red grande, con inteligencia distribuida, la distribución eficiente y simple de la información y programas es una tarea crítica.

Lo anterior implica:

- Transferencia de archivos de datos
- Distribución de cambios:
  - Nuevas aplicaciones o versiones
  - Nuevos Sistemas Operativos o versiones

  Instalación de nuevas máquinas con sus respectivos códigos y programas

El uso de herramientas de transferencia de archivos, distribución automática de software y de un servidor de código asegurará:

- La transferencia eficiente de archivos desde los centros de proceso.
- La construcción y restauración eficiente de Servidores y Clientes en caso de problemas.
- El mantener imágenes idénticas en clientes y de ser posible, en servidores, lo cual facilitaría la administración de las redes.
- Una recuperación simple en caso de desastres.

BIBLIOGRAFIA

Client/Server Programming with OS/2 2.0

- Autor: Robert Orfali, Dan Harkey
- Fecha: 1993, 2da edición

- *Data Communications Concepts*
  - Autor: IBM Corp.
  - Código de Manual: GC21-5169-4

Data Link Protocols

- Autor: Uyless Black
- Fecha: 1993, 3era edición
- Editorial: Bell Atlantic Education Services

IBM Communications Manager/2 EHLLAPI Programming Reference

- Autor: Kevin Daugherty, IBM
- Fecha: Diciembre 1992
- Código de Manual: SC31-6163-00

- *OS/2 REXX: From Bark to Byte*
  - Autor: IBM International Technical Support Organization, Boca Raton Center
  - Fecha: Diciembre 1993
  - Código de Manual: GG24-4199-00

Los siguientes manuales fueron consultados en el disco compacto "*The Best of APPC, APPN & CPI-C*" de IBM Corp, compendio con fecha: Abril/95 (Número de Parte: SK2T-2013-04):

- *CM/2 APPC Programming Reference*
  - Fecha: 8/09/94
  - Código de Libro: CO6P6101
  - Código de Manual: SC31-6160-02
- *Communicating with APPC & CPI-C*
  - Fecha: 14/04/94
  - Código de Libro: D50D0701
  - Código de Manual: G325-0202-00
- *IBM Communications Manager/2 X.25 Programming Reference*
  - Fecha: 8/12/94
  - Código de Libro: CO6Q3001
  - Código de Manual: SC31-6167-01