ESCUELA SUPERIOR POLITECNICA DEL LITORAL

FACULTAD DE INGENIERIA ELECTRICA Y COMPUTACION

DISEÑO E IMPLEMENTACION DEL SISTEMA PARA REGISTRO DETALLADO DE LLAMADAS DEL PBX ERICSSON MD110 DE LA ESPOL

TESIS DE GRADO

Previa a la obtención del Título de:

INGENIERO EN COMPUTACION

Presentada por:

JUAN CARLOS PERALTA ESTEVES

Guayaquil - Ecuador

1.996

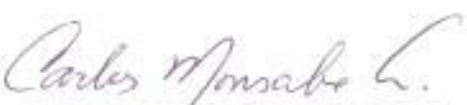
D E D I C A T O R I A

A mis padres

ING. ARMANDO ALTAMIRANO
SUB-DECANO DE LA FACULTAD

ING. GUIDO CAICEDO
DIRECTOR DE TESIS

ING. SIXTO GARCIA
MIEMBRO PRINCIPAL



ING. CARLOS MONSALVE
MIEMBRO PRINCIPAL

DECLARACION EXPRESA

"La responsabilidad por los hechos, ideas y doctrinas expuestos en esta tesis, me corresponden exclusivamente; y, el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL".

(Reglamento de Exámenes y Títulos Profesionales de la ESPOL).

JUAN CARLOS PERALTA ESTEVES

R E S U M E N

El presente trabajo describe los múltiples aspectos del uso de un PBX para la transmisión de voz de datos y presenta una solución para un problema en particular, cual es la tarifación de las llamadas al exterior desde el PBX ERICSSON MD110 adquirido por la ESPOL para el Campus Politécnico.

En los capítulos I y II se discuten los problemas y posibles soluciones a las situaciones que se presentan para administrar las facilidades que brinda un PBX digital, tomando en cuenta además, que la ESPOL se constituye como el primer abonado digital con un enlace de fibra óptica entre su PBX y una Central telefónica pública de EMETEL.

En el capítulo III se describen las bondades que brinda el PBX como medio de transmisión de voz y datos. Se describe también el uso de equipos adicionales para la transmisión de datos de los que éste hace uso.

En el capítulo IV se describen las diferentes opciones de interconección entre un computador personal (PC) con el PBX.

El PC constituye la herramienta principal en la implementación de la solución del presente proyecto, por lo que se hace necasario tener una visión de las diferentes alternativas con que se cuenta.

En el capítulo V se hace una descripción de la solución al problema del presente proyecto.

Finalmente en el capítulo VI se decribe los aspectos más importantes del sistema operativo OS/2 como la plataforma de desarrollo de la aplicación del sistema implementado en el presente trabajo. Este capítulo abarca específicamente los componentes de OS/2 relacionados directa o indirectamente con el proyecto, un estudio completo y detallado de todas los componentes y herramientas de este sistema operativo no es objeto de esta tesis.

CONTENIDO

INDICE DE FIGURAS

INTRODUCCION

Con el traslado de la ESPOL al nuevo Campus Politécnico en la Prosperina, surgió la necesidad de contar con un medio de comunicación moderno entre las diferentes dependencias de la Institución.

Como respuesta a ello, se instaló el PBX Ericsson MD110, el cual, a más de resolver el problema anterior, serviría también para la transmisión de datos por medio del tendido telefónico. Esta solución genera la necesidad de administrar adecuadamente este nuevo recurso.

En primer lugar, cómo saber que la distribución de las extensiones telefónicas es la adecuada? Es decir, existen áreas con exceso de extensiones y otras con déficit? Hay abuso en el uso del teléfono?. Para responder a estas y otras interrogantes, las estadísticas y consultas que se puedan obtener del registro detallado de llamadas, serían las herramientas con que contaría la administración del Sistema Integrado de Voz y Datos.

En segundo lugar, las llamadas internas del Campus no tienen un costo directo para la ESPOL, no ocurre así con las llamadas externas, en que al hacer uso de la infraestructura de EMETEL, dichas llamadas sí tienen costo para la ESPOL.

Ligado al punto anterior, surge la posibilidad de distribuir entre los diferentes centros de costo de la Institución la facturación de EMETEL por las llamadas desde el Campus Prosperina al exterior.

Puesto que la ESPOL es el primer abonado digital en el país que cuenta con enlace de fibra óptica entre su PBX y la Central Telefónica de EMETEL (Central Mapasingue), EMETEL ve al PBX como una Central más y por ello las llamadas externas son "transparentes" ya que EMETEL no dispone del equipo necesario para registrarlas y por tanto no puede planillarlas. La ley vigente no está actualizada a este tipo de enlaces.

Una solución es que la ESPOL entregue a EMETEL los datos para su respectiva facturación, labor que ejecutará el presente proyecto.

El PBX MD110 se comunica con el mundo exterior a través de diferentes puertos, uno de ellos es serial, al cual puede conectarse una impresora o un terminal asincrónico a fin de obtener el registro detallado de llamadas. Este proyecto tiene por objeto conectar un PC a dicho puerto para que lea el registro de llamadas y lo vaya guardando en un archivo de bitácora (logging).

Este PC debe, al mismo tiempo, permitir trabajar en otras aplicaciones y utilitarios (procesador de palabras, hoja de cálculo, etc.) sin que esto signifique detener la "conversación" entre el PC y el PBX ni la actualización del archivo bitácora. Es más, se podría obtener estadísticas de las llamadas mientras todo lo anterior está en ejecución.

CAPITULO I

1.GENERALIDADES

1.1. Funciones de un PBX.

Un PBX es un sistema de control y conmutación que da servicios de comunicación interna y que además está conectado a las redes de telecomunicaciones públicas. En algunos casos la abreviación PABX es usada para designar a un PBX automático, donde los usarios digitan sus propias llamadas; en otros, se usa PMBX para referirse a los PBX antiguos y manuales. En el presente trabajo nos referiremos simplemente al PBX como un sistema automático de conmutación controlado por un programa almacenado.

Hasta hace poco el PBX estaba considerado solamente como un sistema de conmutación telefónico, sirviendo las necesidades domésticas de comunicación de voz de una organización en un lugar, sea una oficina, fábrica,

hospital, hotel o universidad. En el presente trabajo nos estamos referiendo a un *PBX Integrado* como un sistema capaz de procesar comunicación de voz y de datos.

Es probable que un pequeño porcentaje de todos los sistemas de PBX instalados actualmente soporten tráfico de comunicación de voz y datos, pero es de esperar que este porcentaje aumente en buena medida con el transcurso de los años.

1.2. Facilidades del PBX de la ESPOL.

Debido a la extensión que ocupa el Campus Politécnico y a la amplia distribución de la ubicación de sus edificios, hay un problema de comunicación que debe ser resuelto. Como en todas las organizaciones, las necesidades de comunicación internas y externas, son satisfechas por centrales telefónicas.

La ESPOL no es la excepción, existe la necesidad de comunicarse internamente y para ello, fue necesario cablear todas las dependencias de la Institución

dejando en cada oficina por lo menos un punto telefónico (a partir de ahora se lo conocerá como **salida universal**) que en realidad consiste en un aplique de pared que tiene 2 conectores RJ-11.

Para distribuir el cableado telefónico por todo el Campus se ha hecho uso de cajas de distribución intermedia (CDI) y de cajas de distribución final (CDF). Una CDF puede alimentar (dar tono) a varias salidas universales mientras que una CDI puede alimentar a varias CDF.

Las CDI a su vez son alimentadas por los cables multipar que salen del Panel de Distribución Principal (MDF, Main Distribution Frame) ubicado en el sótano del Edificio del Rectorado, donde también se encuentra la Central Telefónica.

Además de la infraestructura física descrita, hay que considerar las herramientas de programación que brinda el PBX. Desde un computador personal (PC), es posible accesar al ambiente de programación del PBX, el cual permite la configuración de todas las líneas (analógicas y digitales) y componentes físicos del MD.

Esta estructura da mucha flexibilidad y capacidad de operación ya que permite:

- Habilitar una salida universal para voz y/o datos cuando ésta sea necesaria.

- Cambiar la ubicación física de una extensión.

- Habilitar paralelos de una extensión.

- Cambiar los atributos de una misma extensión ya sea de voz (por ej: que pueda o no hacer llamadas externas al Campus) o de datos (por ej: cambiar la velocidad/protocolo).

- Rapidez en la localización y corrección de fallas en las líneas.

- Al igual que en la red telefónica pública, las extensiones analógicas pueden transmitir datos con el uso de modems.

- Usando multiplexación por división de tiempo (TDM) es posible que por una extensión digital (físicamente un sólo par de hilos trenzados) se pueda mantener una conversación telefónica y al mismo tiempo acceder a los recursos de un computador.

- Acceso directo desde el exterior a una extensión interna de la ESPOL anteponiendo la serie numérica '269' a los 3 dígitos de la extensión con la cual se desea establecer comunicación.

- Capacidad de acceso remoto (por ej: hogar) a los computadores integrados en el Sistema de Voz y Datos.

- Acceso remoto al sistema de manera tal que, las facilidades que tiene una determinada extensión, las adquiera el teléfono externo que llama.

- Establecer claves de acceso para obtener los beneficios del uso de líneas de voz con determinadas bondades (por ej: DDI).

- Integración del sistema electrónico de seguridad, instalado en diversas dependencias de la ESPOL, a una estación central de control y monitoreo por medio de la red telefónica.

- Integración del sistema de control de acceso del personal administrativo de la ESPOL por medio de la red telefónica.

Algunas de estas facilidades no serían posibles si no fuera por el enlace de fibra óptica entre la Central Telefónica de EMETEL y el MD 110 de la ESPOL, implementación que no se había dado antes. Debido a este enlace el PBX de la ESPOL puede operar como si fuera una central pública remota a la central de Mapasingue.

Como esta implementación ha sido realizado antes, EMETEL no tiene los equipos necesarios para tarifar las llamadas al exterior generadas desde el PBX de la ESPOL.

Debido a esto, es necesario encontrar algún mecanismo tal que la imposibilidad de tarifar las llamadas por parte de EMETEL, no sea un impedimento para que los usuarios del sistema pueda aprovechar el enlace de fibra óptica, con lo cual las llamadas al exterior harían uso de las troncales digitales que dispone el PBX. Esto permite obtener tono de marcar en un menor tiempo, una comunicación más nítida, entre otros beneficios.

1.2.1.Objetivos.

El principal problema a ser resuelto por este proyecto es la tarifación de las llamadas al exterior realizadas desde el PBX de la ESPOL. Para ello se hará uso de una facilidad de la Central Telefónica, esta es el "registro detallado de llamadas" (CIL, Call Information Logging) que se hacen. Dado el esquema de trabajo de la MD110, el concepto de llamada no está limitado solamente a conversiones telefónicas internas o externas, sino también a las "llamadas" que hacen:

- Los equipos que integran el sistema electrónico de seguridad.
- Los equipos que integran el control de acceso del personal administrativo.
- Los equipos que permiten la transmisión de datos entre computadores y terminales. Sobre ellos se hablará en detalle más adelante.

Para capturar los datos que genera el CIL en un puerto serial del PBX, se hará uso de un PC, el cual se espera que no esté dedicado exclusivamente a esta tarea, sino que también sirva de apoyo a las actividades de la Unidad Sistema Integrado de Voz y Datos de la ESPOL. Esta última condición nos lleva a considerar un sistema operativo multitarea, tal como se indicó anteriormente.

Por lo expuesto, podemos indicar que los objetivos que se persiguen en el presente trabajo son los siguientes:

1. Obtener un registro detallado de las llamadas de voz y datos que se realizan en el MD110.

2. Procesar los datos del registro detallado de llamadas para obtener la información que permita a EMETEL facturar las llamadas al exterior realizadas a través de las troncales digitales del PBX.

3. Estudio de los principales componentes del Sistema Operativo OS/2.

CAPITULO II

2. ESTUDIO DE DIFERENTES OPCIONES

2.1. Generalidades.

En este capítulo se dará una justificación al porqué de la solución adoptada para el problema de tarifación.

En el capítulo anterior se mencionó el uso del CIL como fuente de información para conocer las llamadas que se realizan en el PBX. Este criterio no puede cambiar ya que el CIL es la única herramienta que provee el MD110 para dar a conocer esta vital información al proyecto.

Una de las alternativas que sugiere el fabricante para leer el CIL es conectar una impresora serial al puerto del PBX para que ésta imprima el registro de llamadas. Pero obviamente, por la cantidad de llamadas y por el formato del reporte (el cual se analizará con detalle

más adelante en otro capítulo) esta solución no es práctica.

Otra alternativa es conectar un PC al puerto del CIL para que éste registre las llamadas. Esta solución tiene la ventaja de que al tener el PC registradas las llamadas, éste está en capacidad de procesarlas y obtener información y reportes de acuerdo a las necesidades del usuario.

Por obvias razones, esta alternativa es la elegida como la más conveniente y práctica para obtener la tarifación de las llamadas salientes del Campus. Sin embargo, esta solución puede ser implementada en más de una manera en el PC y de este tema tratará la siguiente sección.

2.2. Opciones bajo el PC.

Antes de exponer las posibles alternativas para la captura del registro de llamadas, se debe recordar que al hablar de PC nos referimos a un computador basado en un procesador Intel x86. Además, el PC no sólo deberá realizar la captura, sino también la tarifación y

además deberá ser utilizado como herramienta de soporte a las actividades administrativas del departamento Voz y Datos, es decir el computador deberá hacerse cargo de más de una tarea a la vez.

La solución a este problema está íntimamente ligada al sistema operativo que el PC vaya a tener instalado. Entre las alternativas más comunes para la arquitectura de un PC son: DOS, Windows, OS/2 y Unix.

2.2.1.DOS

Bajo el ambiente monotarea del sistema operativo DOS existen en el mercado algunos productos que permiten conectar PCs a puertos seriales de un PBX para efectos de tarifación. Pero este tipo de solución (con la cual el presente proyecto no tendría sentido alguno) no puede ser aplicado a la ESPOL por dos motivos:

- El PC sólo puede realizar una tarea adicional además de la recepción y almacenamiento del CIL, esto limita el requerimiento de que el computador realice varias tareas a la vez.

- El PC además de receptar y almacenar los datos mediante un TSR, genera información de tarifación no compatible con los requerimientos de EMETEL.

Esto último, principalmente, hace que muchos de los productos del mercado sean descartados como solución. Esto sin contar el aspecto económico.

Por otro lado, pensar en el desarrollo de una solución bajo esta plataforma nos llevaría al problema de limitar la capacidad del PC de atender a varias tareas a la vez.

2.2.2.Windows

La plataforma Windows permite trabajar con varias tareas (DOS y Windows) a la vez mediante la conmutación de las mismas al procesador. Este sistema operativo basa su kernel en DOS, por lo tanto, muchas de las limitaciones del DOS se aplican también a Windows.

Si bien es cierto el manejo de las interrupciones es transparente para el desarrollador, además de que existen herramientas de desarrollo en el mercado, hay un aspecto de investigación que motiva a no seleccionar esta plataforma como se explica en la siguiente sección.

En este caso habría que desarrollar una aplicación que procese los datos que de alguna manera son capturados del PBX. Esta aplicación podría ser implementada con una DBMS popular como Fox ya que así sería fácil contar con personal técnico de experiencia y bajo costo que pueda dar mantenimiento a la aplicación para modificaciones o ampliaciones futuras.

2.2.3.OS/2

Este sistema operativo no basa su kernel en DOS. Está orientado a multitarea y ofrece una arquitectura (se estudiará en capítulo aparte) que brinda un ambiente mucho más robusto para que varias tareas (DOS, Windows y OS/2) coexistan en el sistema y se ejecuten a la vez.

La ESPOL cuenta con un paquete original de este producto que no ha sido ni siquiera instalado y dado que es un producto no muy popular en el medio, hay preferencia en utilizarlo porque además de tener el potencial de solucionar el problema de tarifación obligaría a estudiarlo para conocer sobre sus potencialidades y uso como plataforma para la implementación de soluciones futuras de la ESPOL o de terceros en los que esta última podría participar.

El esquema de solución sería el mismo que el expuesto en la sección anterior, es decir, captura de datos via RS-232 y el procesamiento de los mismos con un DBMS.

2.2.4.UNIX

Este sistema operativo, caracterizado por ser multitarea y multiusuario, se presenta ante este proyecto como una alternativa sobredimensionada puesto que el aspecto multiusuario no es un factor importante.

Por otro lado, habría que considerar los costos que involucraría el propio sistema operativo y el DBMS para el proyecto en sí. A esto hay que agregar los utilitarios (procesador de palabras, hoja de cálculo, etc) que bajo este sistema operativo se debería adquirir.

2.3. Conclusiones.

En base al breve análisis expuesto podemos conclur que necesitamos de un PC para que lea y procese los datos que emite el puerto del CIL del PBX.

En cuanto al sistema operativo la elección es OS/2 por ser multitarea y por efectos de investigación. Esto nos lleva a los siguientes requerimientos mínimos de hardware: procesador 386DX y 8 Mb RAM. Con respecto a la capacidad en disco, en capítulo posterior se analiza este aspecto.

CAPITULO III

3.SISTEMA INTEGRADO DE VOZ Y DATOS

3.1. Generalidades.

El PBX ERICSSON MD110 con el que cuenta la ESPOL, está conformado por bloques modulares (LIMs) cada uno de los cuales está controlado por un microprocesador digital. Esto le da al PBX una completa modularidad y una gran capacidad de crecimiento, permitiendo que cada LIM pueda funcionar autonomamente.

Los LIMS se interconectan a través de enlaces PCM de 32 canales cada uno (incluídos los de sincronización y señalización). Estos enlaces se concentran en un Group Switch (GS) el cuál es controlado por los LIMs. Toda la commutación, tanto para voz como para datos se realiza en el LIM, para lo cuál este contiene a un módulo de commutación en el tiempo con una capacidad de

512 puertos de entrada y salida. Cada puerto tiene una capacidad de transmisión de datos full-duplex a 64 Kbits/s.

Cada LIM puede equiparse con un cierto número de unidades para líneas de extensiones digitales (ELU-D), a cada una de las cuales pueden conectarse o la consola de la operadora, o teléfonos digitales o unidades adaptadoras de terminales (TAU). En todo caso, cualquiera de estas conecciones se realizan a través de un simple par trenzado.

En el caso de la ESPOL, se tienen 4 LIMs. Tres de ellos prestan servicio al área de Ingenierías y fisicamente se encuentran el el edificio de Gobierno Central (Rectorado). El cuarto LIM sirve al área de Tecnologías y se encuentra en el edificio de Gobierno de Tecnologías (edificio 36). Se dispone de un GS (Group Switch) que se encuentra en el área de Ingenierías (sótano del Rectorado). El enlace PCM entre las áreas de Tecnologías e Ingenierías se realiza a través de cables de fibra óptica, con una distancia aproximada de 1300 metros. El siguiente gráfico muestra la configuración de los principales componenetes del MD110.

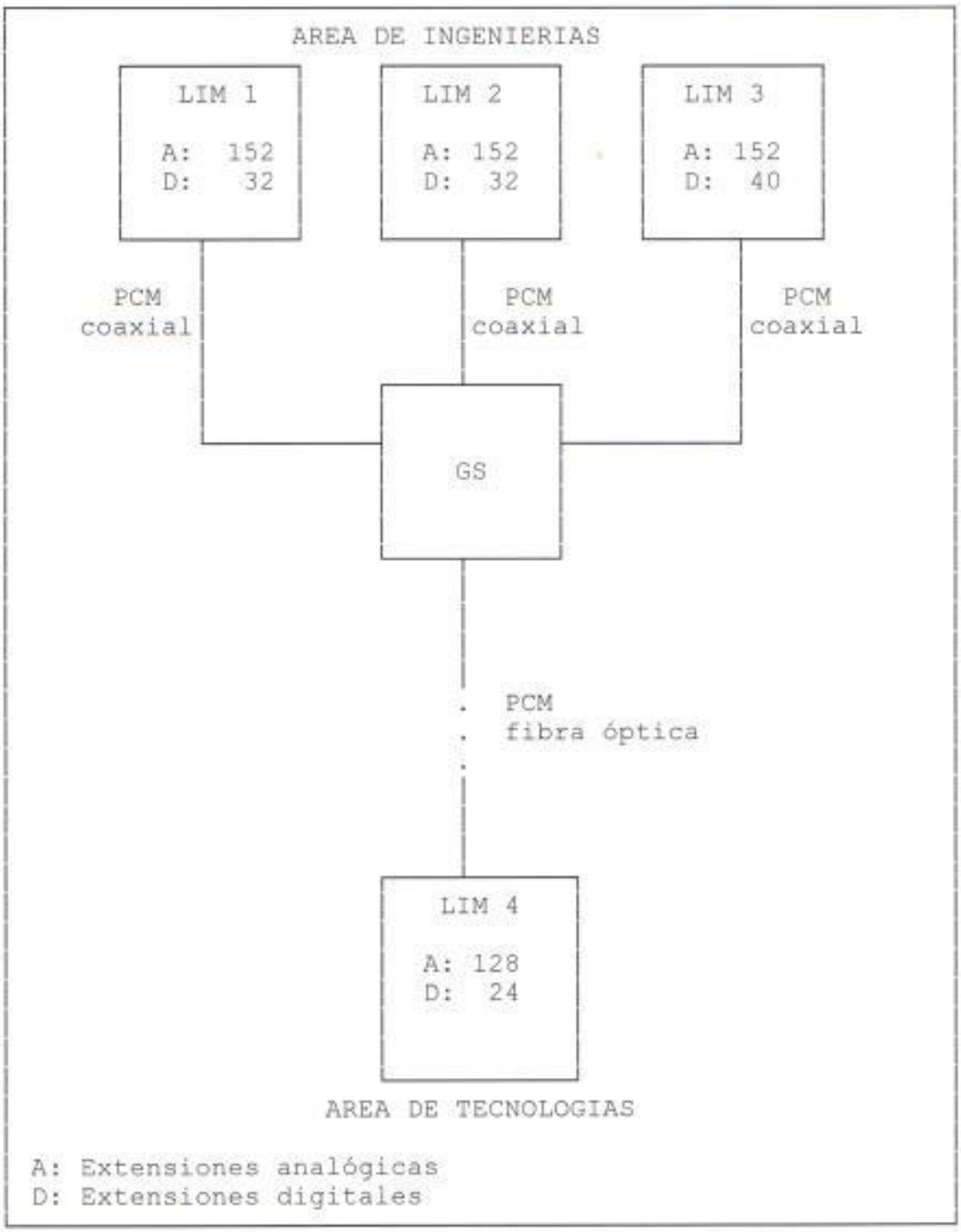


Fig. 3.1 Configuración del PBX de la ESPOL.

3.2. Facilidades para transmisión de datos.

Cada ELU-D puede ser usada para mantener una extensión sólo dedicada a voz o sólo dedicada a datos, o que integra voz y datos. La señalización para una extensión digital se da sobre un simple par trenzado y utiliza una transmisión multiplexada en el tiempo (TDM). Con esto se logra una transmisión integrada de voz y datos sobre un mismo par trenzado.

Al trabajar con una línea de extensión digital, puede usarse un canal de 64 Kbit/s para la transmisión de datos en modo "standalone" a través de una unidad adaptadora de terminal (TAU). Pero también se puede usar un canal de 16 Kbit/s cuando se desea la transmisión de voz y datos integrada. En este último caso se usa otro modelo de TAU que permite la conexión simultánea de un terminal y de un aparato telefónico.

Los canales de datos provistos por el PBX en una extensión digital permiten tanto la transmisión asíncrona como síncrona de datos.

3.2.1.Conexión de Modems.

Los modems pueden ser usados junto con el PBX para realizar cualquiera de las dos siguientes funciones:

1.- Uso de extensiones analógicas para transmisión de datos: Hasta el momento, se ha descrito el uso de líneas digitales para la transmisión de datos. A pesar de que las extensiones analógicas del PBX no brindan algún tipo de servicio especial en lo que a transmisión de datos se refiere, pueden ser usadas en unión de modems para este propósito.

El siguiente gráfico explica esta situación:

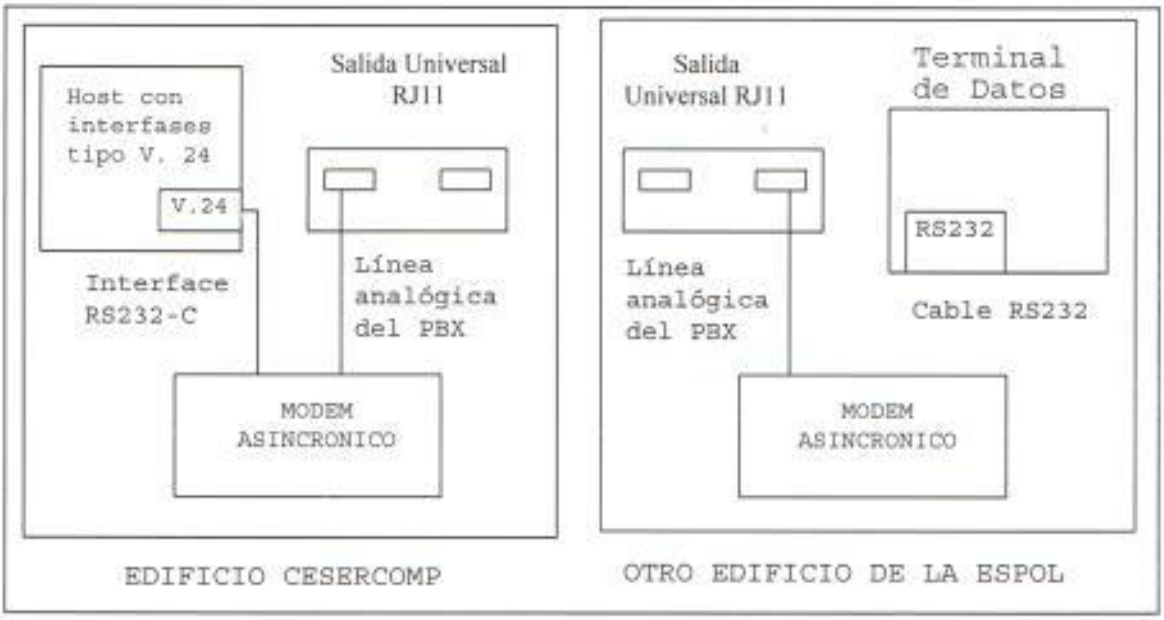


Fig.3.2 Uso de extensiones analógicas para comunicación de datos.

Lo mismo puede suceder para comunicación sincrónica (por ejemplo SDLC), en cuyo caso lo único que cambian son los modems y el terminal de datos que deben ser todos de tipo sincrónico.

En el caso de la ESPOL se han realizado pruebas de comunicación asincrónica entre dos equipos ubicados en diferentes edificios del Campus Prosperina y se ha podido trabajar satisfactoriamente. Las características de comunicación están dadas en este caso por todos

los equipos usados en la comunicación (modems, computador, terminales, etc.).

2.- Soporte de comunicación con usuarios y redes externas: Para soportar conecciones de datos externas vía la red telefónica pública (EMETEL), pueden ser conectados modems convencionales a un LIM vía un módulo de línea especial llamado MAU (Modem Adapter Unit). Si conectamos varios modems de esta manera, y los designamos como un grupo de modems, de tal forma que se permita a un usuario tener acceso por cualquier modem del grupo que se encuentre libre, entonces tenemos un *modem-pool* que permite dos tipos de trabajo:

- Que un usuario pueda comunicarse con uno de los computadores de la ESPOL a pesar de que se encuentre fuera de sus instalaciones (ej: su domicilio).

- Que los usuarios de la ESPOL puedan comunicarse con redes de datos que se encuentren fuera del perímetro físico del Campus Prosperina.

De esta manera se gana tanto en la utilizacion efectiva de los modems, así como en la facilidad de acceso a través de ellos.

El MAU provee una interfase V.24/V.28 (RS232-C) y una entrada para una línea telefónica analógica que permite la coneccion de modems normalizados. La línea pública puede ser conmutada o dedicada.

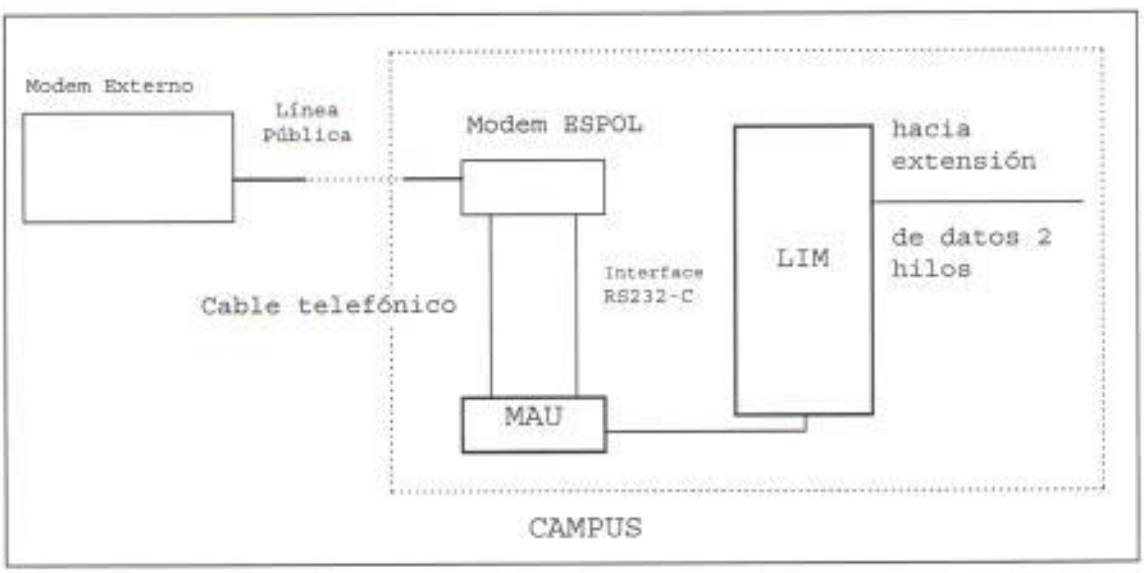


Fig. 3.3 Uso de MAUs para conectar modems externos.

Se pueden conectar varios MAUs al PBX, en el caso de la ESPOL se dispone de 8 MAUs que permiten la

conección de una batería de ocho modems para servicios externos.

3.2.2.Eliminación de cable coaxial.

Practicamente se ha eliminado el uso de cable coaxial, en lo que se refiere a la red SNA, en todo el Campus Politécnico por considerarlo costoso y voluminoso.

Para lograr la eliminación se han usado "baluns" como interfaz entre el conector coaxial, de los terminales sincrónicos o de los equipos con salidas sincrónicas (IBM 3174), y el par trenzado distribuido por todo el Campus.

En el lado del host, estos "baluns" están incorporados en un panel de conexiones coaxial-par trenzado, como se aprecia en la figura 3.4.

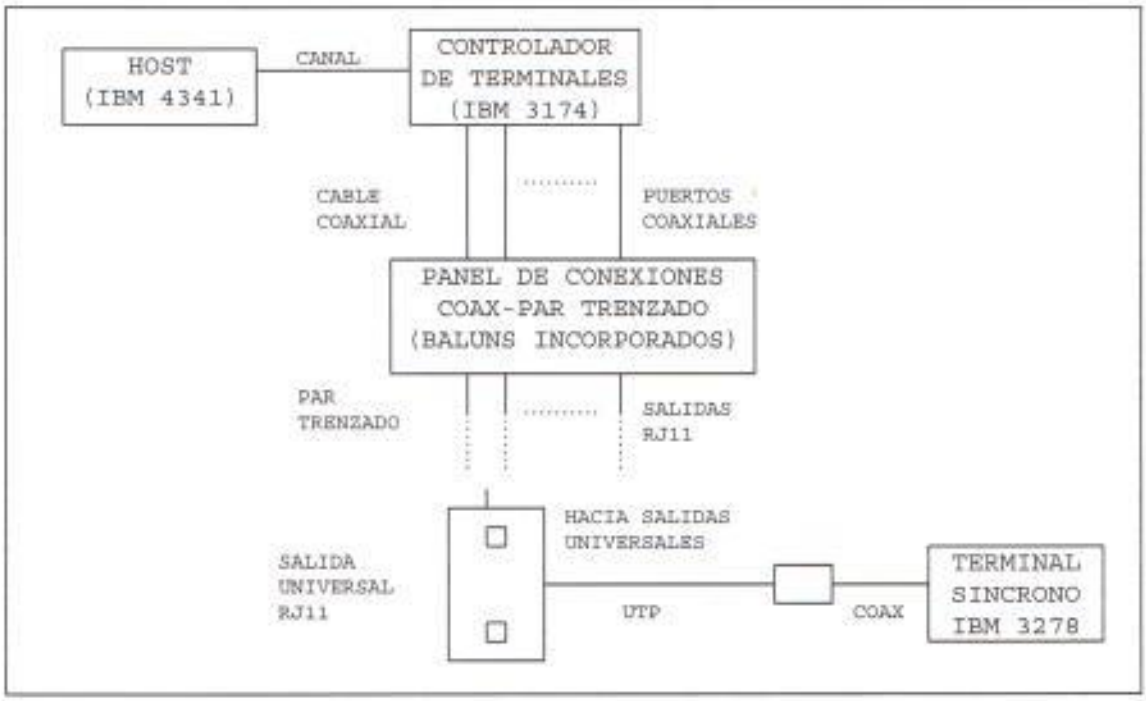


Fig. 3.4 Eliminación de cable coaxial
BAL

Los "baluns" permiten una distancia, entre controlador y terminal, sobre los 670 metros, lo que asegura poder dar servicio a cualquier terminal sincrónico dentro del Campus. Hay que recalcar que los enlaces entre los terminales sincrónicos y las controladoras de terminales no están conmutados a través del PBX.

3.2.3.Interconexión de computadores personales.

El sistema MD110 de Ericsson, provee de un servicio llamado MULTINET, el que brinda una funcionalidad muy parecida a la obtenida con una red de área local (LAN), lo que incluye funciones como compartición de archivos e impresoras así como diferentes tipos de gateways.

La red formada por MULTINET, usa extensivamente el cableado de la red telefónica, tan sólo necesita de un par para cualquier conexión. Por lo tanto se trata de una red barata y fácil de instalar y configurar.

La principal desventaja de MULTINET es la velocidad a la que trabaja: 64 Kbps, mientras que su principal ventaja es el permitir la existencia de una red dispersa que cubra toda una gran área geográfica (WAN: Wide Area Network).

Para la implementación de MULTINET se requieren de los siguientes recursos:

- PBX MD110 con suficiente número de extensiones digitales (una para cada computador a conectarse).

- El software de MULTINET que corre en el PBX.

- Una tarjeta TAU-M para cada estación de trabajo que se desee interconectar (ocupa un "slot" del PC).

- Un computador que actúe como servidor de la red. Este computador también debe estar provisto de una tarjeta TAU-M.

- Sistema operativo DOS versión 3.1 o superior en todos los PCs a interconectarse.

- El software usado en la actualidad para el manejo de comunicaciones en los PCs es LAN MANAGER.

El siguiente es un diagrama de bloques de MULTINET:

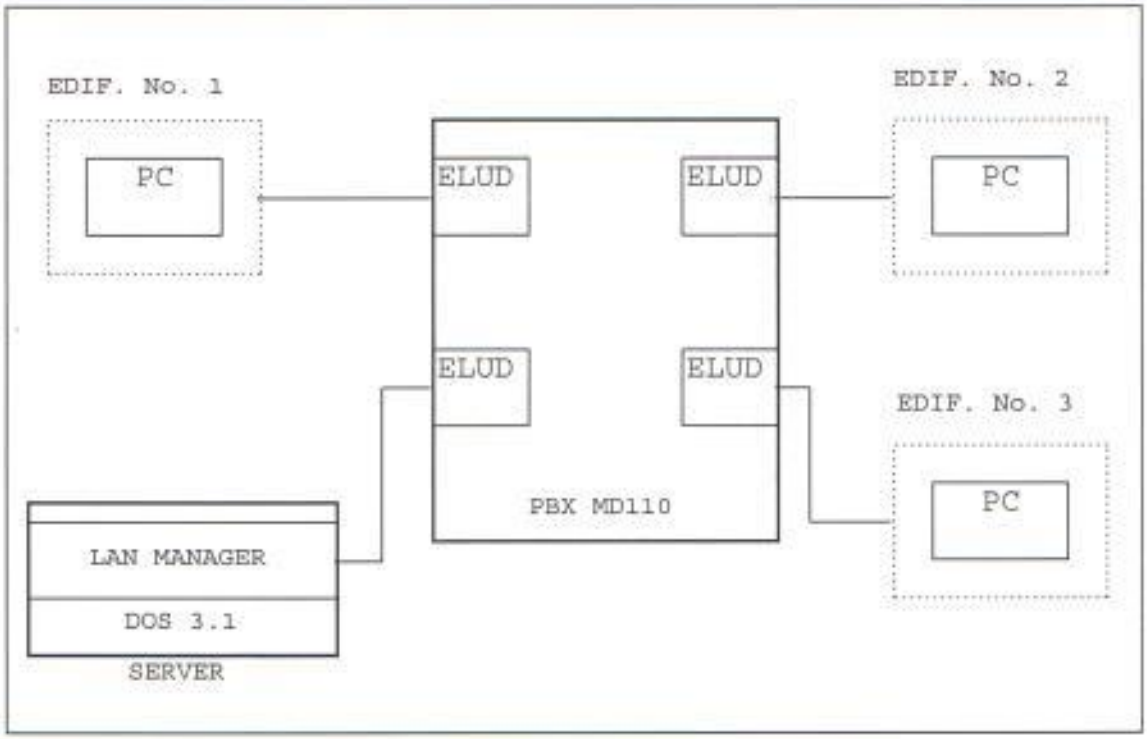


Fig.3.5 Esquema de MULTINET.

Quien se encarga del manejo de las líneas, control de tráfico y manejo de los comandos es el software que reside en la memoria del PBX. Las tarjetas TAU-M que deben instalarse en todos los computadores a interconectarse (incluyendo el server), se conectan al PBX a través de par trenzado. La distancia soportada es la misma que para la comunicación telefónica. Cabe indicar que el TAU-M no emula estándares de comunicacion serial.

La red formada por MULTINET es una red tipo anillo semi-permanente que trabaja a 64 Kb/s, el protocolo de acceso al medio está basado en el estándar IEEE 802.5. La red puede estar distribuida en todo el PBX, incluyendo a LIMs remotos. El PBX se encarga de la supervisión de la red, y garantiza siempre mantener el anillo cerrado, si una estación se daña o apaga inesperadamente, ésta será separada de la red automáticamente.

En cuanto a las aplicaciones, se garantiza que cualquier aplicación escrita para MS-DOS puede correr en MULTINET, con los beneficios de la compartición de archivos y tener un "spool" de impresoras.

3.2.4. Interconexión de redes usando MULTINET.

El software de MULTINET soporta hasta 63 redes MULTINET en el mismo PBX y un máximo de 254 estaciones de trabajo por LIM. Todas estas redes

MULTINET, pueden estar totalmente enlazadas entre ellas, extendiendo de esta manera el compartimiento de recursos y las posibilidades de comunicación de cada red.

Esta interconexión permite a los usuarios compartir los recursos de una manera fácil y transparente. La comunicación de las redes se realiza a través del conmutador de datos del MD110, para ello el servidor de cada red debe estar equipado con un TAU que establezca el enlace de comunicación con el PBX a través de una simple extensión digital (la velocidad puede llegar a 9600 bps).

Esta interconexión también es posible entre redes MULTINET y redes que no son MULTINET. Recordar que una red MULTINET no es una red física, sino que es lógicamente establecida a través del PBX. Las redes no MULTINET son redes físicas como Ethernet o Token-Ring, los servidores de estas redes están conectados al PBX de la forma descrita anteriormente para los servidores de las redes MULTINET. Las redes no MULTINET, deben usar el mismo software del MULTINET (LAN MANAGER). La

misma interconección es posible a través de modems simples, como se aprecia en la siguiente figura.

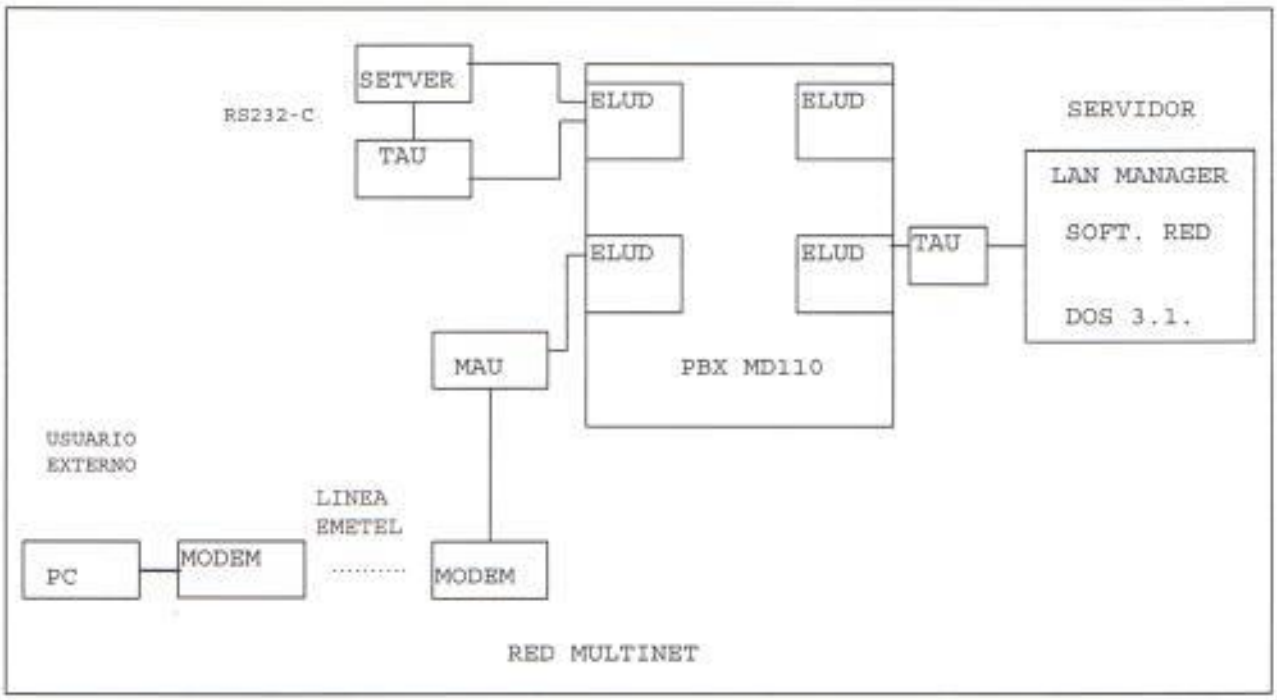


Fig. 3.6 Interconexión de redes con MULTINET.

3.3. Convertidor de protocolo IBM 7171.

La unidad de control para conectar dispositivos ASCII (Convertidos de Protoclo IBM 7171) provee la facilidad de conectar una variedad de dispositivos ASCII full duplex directamente a mainframes IBM 43xx, 308x o 3090.

Soporta conectar hasta 64 dispositivos ASCII via la interface eléctrica RS-232-C a velocidades de 50 a 19200 bits por segundo. Esta tasa de transmisión está limitada solamente por el enlace de comunicación y el dispositivo ASCII. Cuenta con reconocimiento automático de la tasa de transmisión (autobaud) por cada línea, para velocidaddes de 300, 600, 1200, 1800, 2400, 3600, 4800, 9600 y 19200 baudios.

Estos dispositivos pueden ser conectados directamente al IBM 7171 sin modem, o por líneas dedicadas o conmutadas usando line driver, modem o acopladores acústivos.

El IBM 7171 también provee conversión de protocolo ASCII a IBM 3270. Desde el punto de vista del host, el IBM 7171 aparece como una o dos unidades de control IBM 3274 modelo 1D. Los terminales ASCII conectados y las impresores del lado del host aparecen como terminales OBM 3278 o 3277 e impresoras IBM 3286.

La emulación 3270 permite a los dispositivos conectados al 7171 utilizar programas existentes para 3270 sin requerir modificaciones en el host.

La emulación 3270 extiende las capacidades de los dispositivos ASCII brindándoles funciones tipo 3270. Estas funciones incluyen: teclas funcionales de los programas, simulación de lápiz óptico, campos de entrada sólo numéricos, resaltado, saltar campos protegidos y funciones de edición tales como: inserción y borrado de caracteres, tabulación de campos hacia adelante y atrás, borrar hasta fin de campo, limpiar pantalla y movimientos de cursor.

Además de emulación 3270, también se han incluido funciones extendidas. Estas incluyen: manejo mejorado de nulos/blancos, características especiales de indentación y tabulación automática hacia adelante y atrás de columnas.

El IBM 7171 está diseñado para una fácil instalación, configuración y mantenimiento. Puede ser integrado en una red de comunicación ya establecida o en el punto de partida de una red recientemente diseñada.

3.4. Módulo de comunicación asincrónica TAU-2520.

Este tipo de unidad adaptadora de terminal, permite conectar directamente un teléfono digital y un terminal a la misma línea digital a través de un único par trenzado.

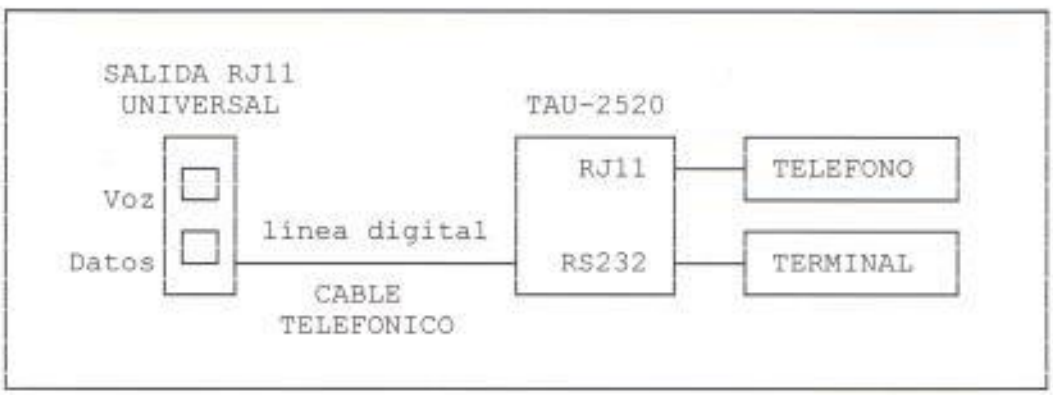


Fig. 3.7 Conexión del TAU-2520.

Cada salida universal posee dos RJ11 que dan servicio al usuario: uno para comunicación de voz y otro para datos.

El TAU-2520 cumple con la función de demultiplexar las señales de voz y de datos para el usuario final. Soporta la transmisión de datos asíncrona y

sincronicamente. A pesar de su capacidad integradora de voz y datos (dual mode), también se lo puede usar exclusivamente para la transmisión de datos "stand-alone mode". Puede actuar en modo full-duplex o en half-duplex.

Existen dos modos de operación:

1. *Hot line*: al activar el TAU-2520 automaticamente realizará la llamada a una extensión de datos previamente definida.

2. *Dial-up*: el usuario tendrá que "marcar" el número de la extensión de datos con la que desea comunicarse.

Las tasas de transmisión son:

STAND-ALONE:

Asincrónica (bps):

110,150,200,300,600,1200,1800,2400,4800,9600,19200

Sincrónica (bps):

600,1200,2400,4800,9600,12000,14400,19200

MODO DUAL :

Asincrónica (bps):

110,150,200,300,600,1200,1800,2400,4800,9600

Sincrónica (bps):

600,1200,2400,4800,9600,12000

El TAU-2520 puede ser usado con un computador personal, un terminal ANSI compatible, un terminal tipo VT100 o una impresora. En realidad, cualquier equipo terminal de datos que posea una interfase V.24/V.28 (RS232-C) podria hacer uso de las facilidades del MD110 a través del TAU-2520. El manejo de las llamadas de datos así como la programación de los parámetros para la interfase del terminal son desarrollados con la ayuda del firmware del TAU, el cual soporta los comandos AT de Hayes. Este firmware también provee un menú de interface para el almacenamiento de las extensiones que se utilizan con más frecuencia.

3.5. Módulo de comunicación asincrónica TAU-S.

Para aquellas aplicaciones donde no es necesaria la integración (por ejemplo: no se requiere teléfono), los terminales pueden conectarse a través de una unidad adaptadora llamada TAU-S. Funcionalmente es muy parecido al TAU-2520. Cualquier terminal de datos que utilice una interface V.24/V.28 (RS232-C) puede ser conectado al MD110 a través del TAU-S.

Las tasas de transmisión son:

Asincrónica (bps):

110,150,200,300,600,1200,1800,2400,4800,9600,19200

Sincrónica (bps):

600,1200,2400,4800,9600,12000,14400,19200

Las extensiones digitales que estén usando el TAU-S o el TAU-2520, pueden tener acceso a un grupo de TAUs. Un grupo de TAUs consiste en le agrupación de los TAUs conectados al equipo de computación con el que se trata

de establecer la comunicación (ej: un host), a los que se les ha asignado un número de grupo. De esta forma, un usuario puede hacer una llamada para transmisión de datos a través del número asignado al grupo, entonces el PBX lo conectará a un TAU libre que forme parte de este grupo.

CAPITULO IV

4.TRANSMISION DE DATOS ENTRE EL PC Y EL PBX

4.1. Interconexión.

El PC que almacena los datos que reporta el CIL puede estar conectado de manera local o remota al PBX. En ambos casos la conexión es RS-232.

En modo local, basta con un cable serial. Uno de sus extremos está conectado al puerto del CIL en el PBX y el otro extremo al puerto serial del PC. Esto obliga a que el PC se encuentre en el mismo lugar del PBX, debido a limitaciones de distancia propias de RS-232.

En el modo remoto se hace uso de dos TAU, de esta manera se aprovecha la red telefónica y hay flexibilidad en cuanto al lugar donde va a estar ubicado el PC. Para la implementación de este proyecto se hace uso de dos TAU-S, uno de ellos está conectado al puerto del CIL en el PBX y el otro TAU-S al puerto

serial del PC. A nivel de Central ambos TAU-S están programados para operar a 19200 bps, siendo el TAU-S conectado al PC el *maestro* y el TAU-S conectado al PBX el *esclavo*.

El *maestro* llama al *esclavo* apenas tenga la señal DTR en el puerto serial, cuando el esclavo contesta y se establece el enlace, el CIL puede enviar el detalle de las llamadas al PC. El CIL se encuentra activo en todo momento, por lo que si el *maestro* no llama al *esclavo* el PBX reportará la respectiva alarma.

La programación necesaria a nivel del PBX para activar el CIL de la manera descrita es la siguiente:

```
/*================================================*/
/* ACTIVATING FILE                                */
CLOHI:FILE=CIL,FORM=2,DMPSIZ=1;
CLODI:FILE=CIL;
CLTGI;
/*================================================*/
CLTGE;
CLODE:FILE=CIL;
CLOHE:FILE=CIL;
/*================================================*/
```

4.2. Protocolos de comunicación entre el PC y el PBX.

Los dos protocolos que pueden ser usados son, el uno ACK/NACK y el otro se lo conoce como no formateado. El protocolo a usar se selecciona con el comando EDICC del MD110.

Independiente del protocolo, el tablero SIU emite 1024 bytes dentro de los siguientes tiempos:

Baudios	Segundos
300	45
600	43
1200	22
2400	12
4800	7
9600	4
19200	3

4.2.1.Protocolo no formateado

Este protocolo significa que los datos son emitidos sin formato. No existe *checksum*. El equipo periférico no tiene la posibilidad de pedir retransmisión si los datos recibidos tienen fallas.

El periférico puede detener el envío seteando DTR como pasivo. Otra forma es que el equipo periférico envíe el carácter XOFF. En este caso el envío continuará si envía el carácter XON.

4.2.2.Protocolo ACK/NACK

Este protocolo es de tipo "poll" (llamado también *escrutinio*), donde el equipo periférico es el maestro y el MD110 es el esclavo. "Polling" significa que el maestro pregunta al esclavo si este último tiene algo que enviar. Al esclavo no se le permite transmitir hasta que haya sido "poleado" (carácter ASCII ENQ) por el equipo periférico. El maestro puede trasnmitir solamente cuando el SIU está listo para aceptar.

El protocolo usa palabras de 8 bits (7 de datos y uno de paridad par) y un bit de parada. Para este protocolo estas características son inalterables.

Los datos son transmitidos en mensajes en los cuales cada mensaje puede consistir de uno o más bloques (máximo 8). Cada bloque termina con un checksum. Si el checksum recibido o paridad son incorrectos, el equipo periférico pedirá retransmitir el bloque.

La figura 4.1 da una descripción gráfica del protocolo de comunicación entre el PC y el PBX:

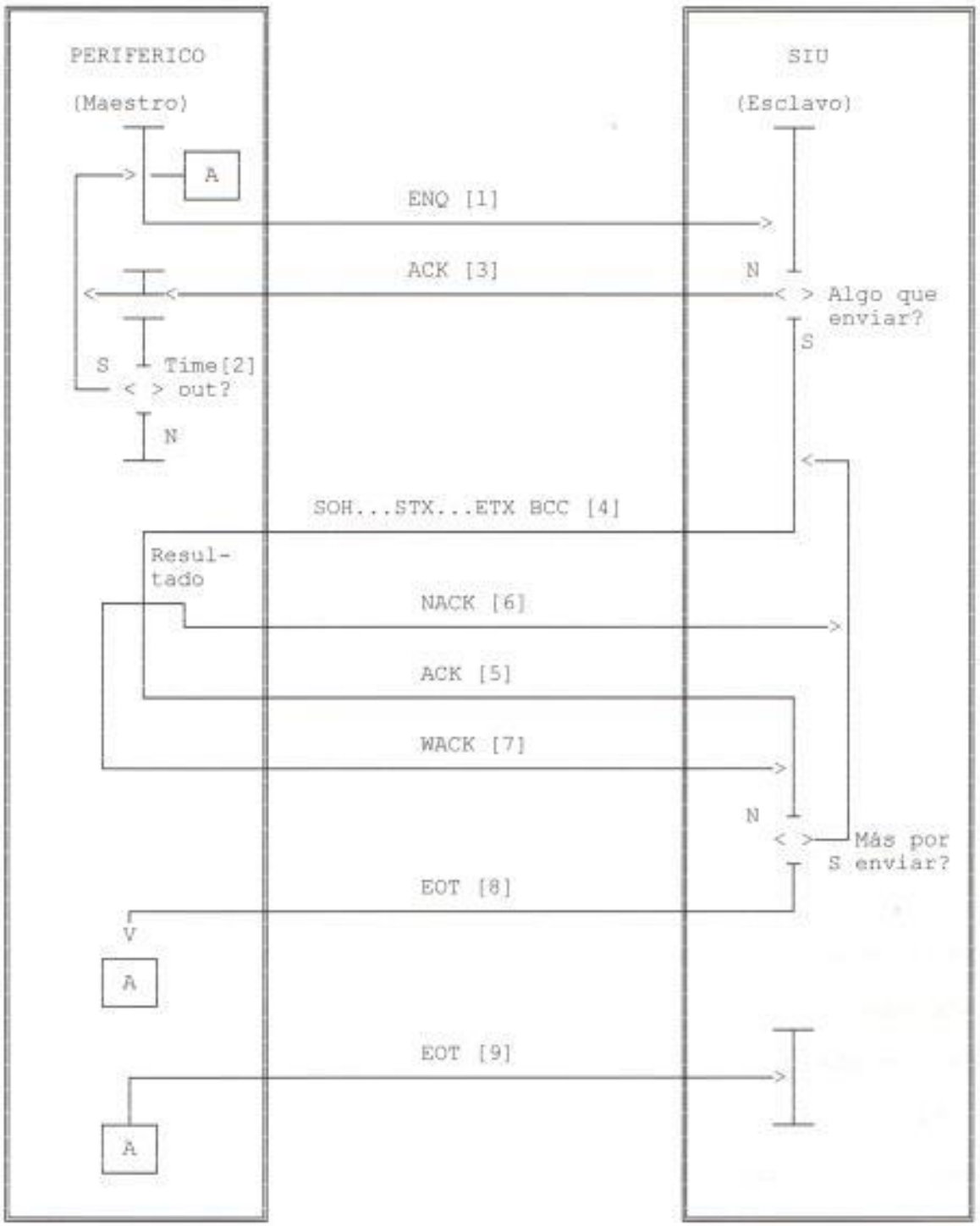


Fig. 4.1 Protocolo ACK NACK entre el PC y el PBX.

Los dígitos encerrados entre corchetes en el gráfico son una referencia al texto explicatorio que está a continuación:

[1] El maestro comienza el "polling".

[2] Se obtiene time out (el tiempo es establecido en el maestro al inicio del sistema). El maestro continúa el "polling".

[3] ACK: El esclavo ha aceptado el poll pero no tiene algo que enviar.

[4] El esclavo envía un mensaje.

[5] El mensaje ha sido aceptado.

[6] NACK: No al ACK. El maestro no acepta el mensaje. El esclavo deberá enviar el mensaje una vez más. Si un NACK es registrado por el esclavo cuando el bloque está siendo enviado, esto también debe resultar en una retransmisión.

Un "NACK dañado" deberá también ser considerado como un NACK, si es posible. Esto implica que, cuando el esclavo espera por ACK/NACK, una señal que no sea ENQ, ACK, NACK, WACK o SOH deberá considerarse como NACK dando por resultado una retransmisión en vez de time out.

[7] WACK: Espera por ACK. El maestro acepta el mensaje pero no está listo para aceptar el siguiente. Cuando el maestro esté listo enviará ENQ.

[8] EOT: Fin de texto. El esclavo concluye la transmisión porque no hay algo más que enviar.

[9] El maestro termina la transmisión. El mensaje ha sido repetido diez veces o no se ha obtenido respuesta cuando ACK/NACK/WACK fue enviado al esclavo.

4.3. Formatos.

Un bloque de datos que envía el PBX tiene el siguiente formato:

SOH	PRIO	BLOCK	SEQ	STX	TEXTO	ETX	BCC

La descripción de los campos es la siguiente:

FUNCION	DESCRIPCION	VALOR (Hex)
SOH	Inicio de cabecera	81
PRIO	Prioridad (no usado)	00
BLOCK	Contador de bloque	00 = sólo un bloque en el mensaje 01 = primer bloque . . . 07 = séptimo bloque 0F = último bloque en el mensaje
SEQ	Número de secuencia	0F = primer bloque después de que el SIU ha sido reinicializado y para el primer bloque en que cada mensaje que contenga varios bloques 00 = segundo bloque después del reinicio

		01 = tercer bloque, de aquí en adelante de manera alterna 001
STX	Inicio de texto (ASCII)	02
TEXTO	Máximo 128 caracteres ASCII con paridad par	
ETX	Fin de texto (ASCII)	03
BCC	Checksum[1] para bloques[2]	

El campo TEXTO, almacena los datos que reporta el CIL. Para el presente proyecto se ha utilizado el siguiente formato:

FORMATO STANDARD

DESCRIPCION	**BYTES**
Espacios	0 - 2
Fecha término de llamada	3 - 6
Hora término de llamada (hmms)	7 - 10
Espacio	11

---

[1] El checksum de los bloques es calculado de la siguiente manera: "OR exclusivo" de todos los caracteres desde PRIO inclusive hasta ETX incluido.

[2]. BB es el único byte en el bloque que carece de chequeo de paridad.

Tiempo duración de llamada	12 - 15
Espacio	16
Pulso intercentral	17 - 20
Espacio	21
Código de estado (status)	22
Espacio	23
Código de acceso 1	24 - 28
Código de acceso 2	29 - 33
Espacio	34
Número al que llama	35 - 54
Espacio	55
Número que llama	56 - 61
Espacio	62
Código de cuenta	63 - 77
Espacio	78
Código de autorización	79 - 84
Espacio	85
Tiempo encolado en troncal	86 - 87
Espacio	88
Código de la troncal	89 - 94
Carriage Return	95
Line Feed	96
Espacio	97 - 99

CAPITULO V

5.SOLUCION AL PLANILLAJE DE EMETEL

5.1. Requerimientos.

Dado que el enlace de fibra óptica entre el MD 110 y la Central Mapasingue de EMETEL R-2 hacen que el PBX de la ESPOL sea visto como una central de EMETEL[3], la facturación de las llamadas locales no es posible por 2 razones principales:

- Técnicamente, EMETEL no cuenta en sus centrales públicas con los equipos necesarios de control que permitan tarifar las llamadas.

- No existe, al momento de realizar el presente trabajo, la reglamentación adecuada que permita

[3] El PBX de la ESPOL es una central remota de la central Ericcson AXE instalada en Mapasingue, pero en la práctica el MD110 es visto como una Central Pública más.

establecer las políticas de tarifación en el caso de los clientes corporativos de EMETEL, que cuentan con PBX digitales conectados a centrales públicas.

Dado el tipo de enlace entre el PBX MD110 y la Central Mapasingue, se aprovechará la capacidad de envío de "ecos" intercentral de los equipos instalados, desde la Central Mapasingue a la ESPOL, de cada llamada local hecha por una extensión del Campus a la PSTN de EMETEL.

5.2. Descripción de la aplicación.

La aplicación instalada en el PC se encarga de registrar todas las llamadas que procesa el PBX. En el capítulo anterior se describió la estructura del registro que por cada llamada genera el PBX.

También se describió el hardware utilizado para poder efectuar las lecturas de los registros y el almacenamiento de los mismos en el PC, en un archivo ASCII. De las pruebas realizadas, aproximadamente se

almacenarán entre 100.000 a 110.000 registros[4] mensuales.

Para el procesamiento de los datos en el archivo, hacemos uso del DBMS FoxPro. Hemos hecho esta elección por varias razones :

- Buen tiempo de respuesta comparado a otros productos de similares características.

- Producto de uso muy extendido, lo que permite contar con mayor facilidad de personal entrenado en el uso del mismo, abaratando así los costos de mano de obra.

- Incluye herramientas que facilitan el trabajo y aumentan la productividad en la creación de sistemas, como son: generador de reportes, uso de ventanas, manejo de menús, documentación técnica, entre otras.

Este archivo ASCII es transformado y cargado en una tabla para ser procesado por un programa Fox con el fin de generar un nuevo archivo ASCII con este formato:

---

[4] El CIL no sólo registra las llamadas salientes locales sino todas las llamadas de voz y datos, lo que incluye la comunicación entre la red de seguridad del Campus Prosperina.

```
aammdd - aammdd     ←  Período de las llamadas
999999 99999999 aamm
999999 99999999 aamm
  .       .       .
  .       .       . ← Año y mes de la facturación
  .       . ←-------- Pulsos generados por el número
  . ←---------------- Número telefónico
  .       .       .
999    99999999     ← No. registros/Total de registros
```

Para la definición de este formato se trabajó en cojunto con personal del Departamento de Informática de EMETEL R2.

De igual modo, está establecida una política que establece que tanto la ESPOL, representada por el Sistema Integrado de Voz y Datos, como EMETEL cumplirán a fin de que mensualmente se procese la información generada por la ESPOL para la respectiva tarifación.

El nombre del archivo será *EPLaamm.TXT*, donde `aamm' hace referencia al año y mes de facturación.

La primera línea del archivo especifica el período comprendido entre la primera y última lectura del CIL que lo originó.

El campo número telefónico corresponde a la extensión interna que hizo la llamada local hacial el exterior del Campus, es decir, 269 + *número de extensión*.

El campo pulsos generados por el número indica el acumulado de pulsos, generados por llamadas locales, que durante el período de lectura, generó la extensión. De acuerdo a la ley vigente, por cada 3 minutos de conversación se genera un pulso si es que la llamada se hace de Lunes a Viernes entre las 7h00 y 19h00. Se genera un pulso por cada 6 minutos si la llamada se hace sábados, domingos, feriados y de lunes a viernes fuera del horario anterior.

El campo año y mes de la facturación es una simple referencia del mes al que corresponde la lectura del registro. Está incluído también para que tenga similitud con los registros para facturación generados por el COMAG.

La última línea del archivo contiene los *campos total de números* y *acumulado de pulsos*. El primero es un contador del número de registros del archivo y el segundo es un acumulador de los pulsos generados por los diferentes números. Estos campos tienen por objeto facilitar a EMETEL los procesos de validación y consistencia de datos.

Dado que todas las llamadas de voz y datos se encuentran en una tabla FoxPro, formato DBF, éstas pueden ser utilizadas no sólo para efectos de tarifación, principal objetivo de este proyecto, sino que también existe la posibilidad de realizar consultas que ayuden a la administración del sistema.

Una de las ayudas de administración implementadas en este proyecto, es el control de las llamadas al exterior. La idea es obtener un detalle de las llamadas que una extensión en particular ha realizado. Luego, se sumariza este detalle para obtener el tiempo total empleado en llamadas externas, generalmente, en el período de un mes.

Este proceso es muy útil para la ESPOL ya que, por ejemplo, permitiría cargar los gastos de llamadas telefónicas a los arrendatarios de la Institución que disponen de extensiones con salida al exterior mediante el dígito "9".

Internamente, se puede controlar al personal de la ESPOL que abusa de la facilidad de salida al exterior con el "9", habilitadas a sus extensiones, por motivos de trabajo.

Tanto el registro detallado de llamadas, como el proceso de generación del diskette para tarifación o el control de llamadas al exterior se las realiza al mismo tiempo, dadas las facilidades de multitarea que brinda el sistema operativos OS/2, plataforma de trabajo de este proyecto.

CAPITULO VI

6.SISTEMA OPERATIVO OS/2

6.1. Características.

OS/2 2.x es la primera versión del sistema operativo OS/2 que explota las características de los sistemas de 32 bits Intel 80386. Está diseñado para correr a partir de los procesadores 80386 DX/SX y superiores.

El sistema usa las características de paginamiento del 80386 para proveer un ambiente de memoria virtual de demanda de páginas que soporta un nuevo modelo de programación de 32 bits. Este modelo permite que las aplicaciones, los subsistemas y el sistema en sí utilice el conjunto de registros de 32 bits, las instrucciones de 32 bits y modelos de memoria, así como objetos de memoria mayores de 64 Kb.

OS/2 2.X provee compatibilidad para las aplicacciones y subsistemas OS/2 1.X. El ambiente de compatibilidad DOS ha sido mejorado aprovechando las ventajas del modo virtual 8086 del 80386, permitiendo múltiples sesiones DOS con ejecución en *background*. El sistema provee además un emulador del coprocesador numérico 80387 y soporte para los sistemas de organización de archivos FAT y HPFS.

6.1.1.Modelo de memoria plano.

El objetivo principal del ambiente de programación de 32 bits es proveer una arquitectura que permita a las aplicaciones, subsistemas y al sistema en sí ser portable a plataformas de procesamiento diferentes a las máquinas basadas en el procesador Intel 80X86.

Este requirimiento llevó al desarrollo de un nuevo modelo de memoria llamado *memoria plana*, el que permite a los procesos ver la memoria como un gran espacio de direcciones direccionables mediante desplazamientos de 32 bits.

La arquitectura de memoria plana es fácilmente portable a la mayoría de plataformas de procesamiento ya que el hardware debe proveer solamente un registro base capaz de direccionar un gran espacio de direcciones paginadas y un registro de desplazamiento usado como índice en el espacio de direcciones.

El modelo de memoria plano, como se lo implementa en el 80386, es también conocido como "modelo 0:32" porque solamente el desplazamiento de 32 bits en el espacio de direcciones de proceso es usado para obtener la dirección de un único byte de memoria. El modelo plano oculta de manera efectiva toda segmentación al programador de 32 bits, dando como resultado un modelo de programación portable con un mayor desempeño que el que un sistema segmentado pueda ofrecer.

El espacio de direcciones de proceso de OS/2 2.x consiste de un único gran segmento que los procesos direccionan usando desplazamientos de 32 bits. En el modelo plano, la unidad básica de localización y compartición es una página de 4

Kb. En vez de estar divida en segmentos como en OS/2 1.X, la memoria está dividida en objetos de memoria que consisten de una o más páginas de 4 Kb. A diferencia de los segmentos, los objetos de memoria no son relocalizables, son localizados en unidades de 4 Kb y están alineados en fronteras de páginas en el espacio de direcciones de proceso.

El tamaño del espacio de direcciones de proceso en OS/2 2.x es de 4 Gb, que es lo que soporta el 80386. Esta restricción, discutida más adelante, es debida a la compatibilidad con OS/2 1.X. La figura 7.1 ilustra el espacio de direcciones de proceso en el modelo plano.

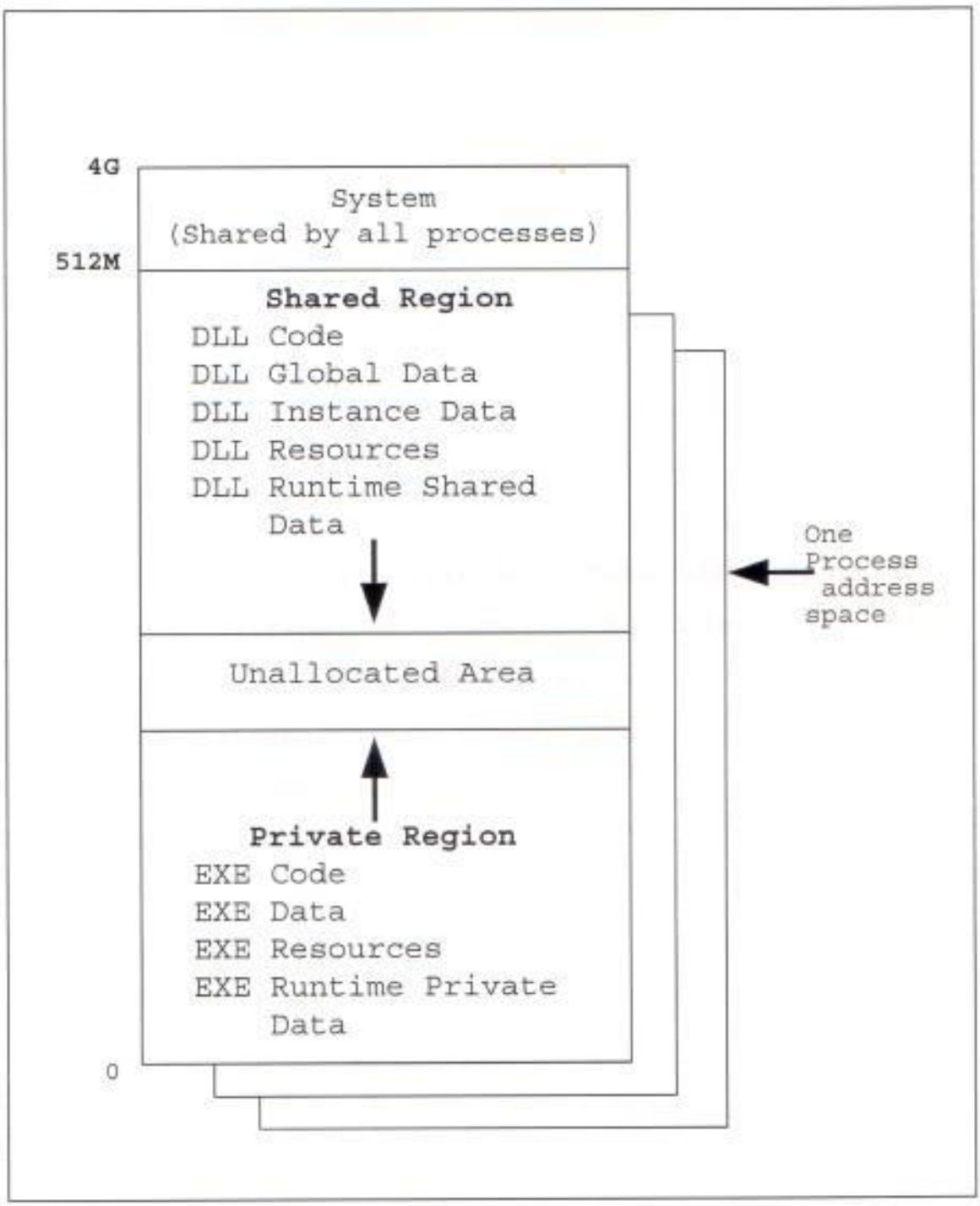


Fig. 6.1 Espacio de direcciones de proceso en OS/2 2.x.

En la figura 7.1 se aprecia que el sistema está en el tope de cada espacio de direcciones de proceso. En cualquier momento que una aplicación intente usar una dirección mayor de 512 Mb o una dirección que no ha sido colocada, se genera una falla de acceso. El espacio de direcciones de proceso en sí está dividido en regiones privadas y compartidas. Las localidades de memoria compartidas crecen desde el tope hacia abajo mientras que las privadas crecen desde el fondo hacia arriba del espacio de direcciones de proceso. El sistema garantiza que 64 Mb de memoria cada uno de memoria compartida y privada están disponibles al momento de carga del proceso.

La protección de memoria es una de las más significativas diferencias entre el modelo plano y el modelo segmentado de 16 bits. En este último, existe la protección basada en "por segmento"; los chequeos de acceso y límite existen en la carga de cada selector de segmento. Dentro del modelo plano, sin embargo, ya que

todos los objetos de memoria de la aplicación existen en el mismo segmento, las semánticas de protección INTEL son ignoradas y se usa la protección a nivel de página para administrar la memoria en el espacio de direcciones de proceso.

En el modelo segmentado o de 16 bits, los registros de segmento tienen que ser vueltos a cargar con selectores cada vez que un segmento diferente de 64 Kb tiene que ser accesado. Estas operaciones de carga de sslectores son muy costosas en el modo protegido de los procesadores 80X86 debido al cheque que debe existir para asegurar la protección del segmento. En el modelo plano, un desplazamiento de 32 bits relativo a la base del espacio lineal de direcciones se usa para direccionar cualquier byte de memoria sin recargar ningún selector. De hecho, los programas y subsistemas de 32 bits no usan o saben acerca de los registros de segmento.

Otro beneficio del modelo plano es que solamente existe un modelo de memoria para las aplicaciones (*small* sin restricciones de 64 Kb) en vez de

varios modelos (*small, medium, large, huge*) usado en OS/2 1.X y otros sistemas basados en procesadores 8088 y 80286. Los registros de 32 bits y la aritmética provista para la arquitectura Intel de 32 bits también incrementan el desempeño de las aplicaciones.

6.1.2.Paginamiento

La característica de paginamiento del 80386 es usada para soportar el modelo de memoria plano y múltiples espacios de direcciones DOS, también le permite a OS/2 proveer un diferente tipo de manejo de falta de memoria que el provisto por los sistemas 1.X. En OS/2 1.X, se usaba intercambio de segmentos para mantener al sistema corriendo en situaciones de falta de memoria. Debido al desempeño de entrada/salida de la mayoría de los discos duros, sin embargo, el intercambio de segmentos no se desempeña lo suficientemente bien para dar un almacenamiento genérico virtual.

El 80386, por otro lado, provee paginamiento. Desde el punto de vista del sistema, una gran cantidad de almacenamiento puede ser virtualizado en un disco duro a un menor costo de entrada/salida debido a que el tamaño de una página es menor que el de un segmento. El sistema también puede hacer un mejor trabajo de seguimiento del uso de memoria ya que los algoritmos de control de "vejez" de la memoria operan a nivel de página en vez de a nivel de segmento. Por estas razones, a partir de OS/2 2.x, un sistema de almacenamiento virtual por demanda de páginas, está diseñado de forma tal que el sistema corra aceptablemente en situaciones de falta de memoria.

6.1.3. Encadenamiento dinámico

OS/2 2.x provee soporte a librerías compartidas en la forma de Librerías de Encadenamiento Dinámicas (DLLs). La eliminación de la segmentación en el modelo plano es también reflejado en el modelo de encadenamiento dinámico

del sistema. En vez de que todas las llamadas sean "far", todos los objetos de código son "near"; no es necesario volver a cargar los segmentos cuando un API o una rutina dinamicamente encadenada es llamada.

Las DLLs de 32 bits son mapeadas en la región compartida en el espacio de memoria del proceso que las requiere al momento de carga y ejecutadas sin nivel de privilegio de entrada/salida. La figura 7.2 muestra cómo son implementadas las DLLs de 32 bits en el ambiente de modelo plano.

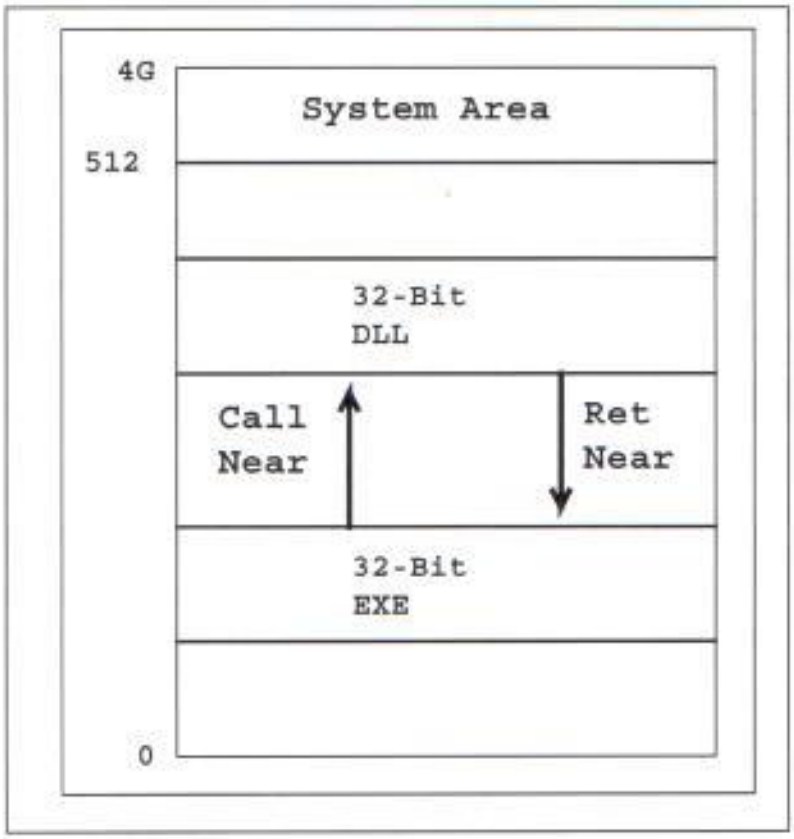


Fig. 6.2 DLL modelo plano.

6.1.4.API de 32 bits

OS/2 provee APIs de 32 bits que permiten a las aplicaciones bajo el modelo plano usar los servicios del sistema. El API de 32 bits ha sido diseñado a fin de que las aplicaciones y subsistemas que usan y proveen APIs de 32 bits

sean portables en código fuente a cualquier futura plataforma OS/2.

La arquitectura del API en OS/2 1.x y OS/2 2.x difieren significativamente. En OS/2 2.x ya no hay restricción de 64 Kb y los punteros de los API son del formato 0:32. Muchos de los nombres de los APIs han sido cambiados para dar mayor consistencia y algunas áreas de los API han sido mejoradas para brindar mejor funcionalidad.

6.1.5.Compatibilidad con OS/2 1.X

Como se indicó anteriormente, OS/2 2.x corre las aplicaciones bajo OS/2 1.x sin necesidad de cambios. Para ello, los diseñadores de OS/2 tuvieron que desarrollar una arquitectura en la cual los módulos de 16 bits y 32 bits puedan coexistir simultaneamente. Esta tarea es particularmente difícil debido a que el modelo plano y el segmentado son extremadamente diferentes. El principal requerimiento para que ambos sistemas puedan coexistir es un mecanismo

para convertir las direcciones de 16 bits en direcciones de 32 bits y viceversa. Una vez resuelto este problema, la tarea de servir una llamada a un API de 16 bits con una rutina de 32 bits, o lo contrario, se vuelve factible. La técnica usada para la conversión de direcciones entre el modelo segmentado y el modelo plano se llama *tejado de tablas de datos locales* (LDT tiling).

Un tejado de LDTs contiene a lo mucho 8192 descriptores de segmento, cada uno de los cuales mapea una región de 64 Kb del espacio de direcciones de proceso; las regiones mapeadas por descriptores contiguos están contiguos en el espacio de direcciones de proceso. El resultado es que un solo LDT puede mapear, a lo mucho, 512 Mb de espacio contiguo de direcciones en el modelo plano. Es por esto que OS/2 2.X restringe el tamaño del espacio de direcciones de proceso a 512 Mb.

Este tejado de LDTs crea un mapeo de direcciones entre las direcciones 16:16 y las 0:32 para

cualquier byte de memoria en el espacio de direcciones de proceso. Las siguientes fórmulas describen las conversiones de direcciones entre los modelos de memoria de 16 bits y 32 bits:

```
0:32 = ((SELECTOR(16:16) >> 3) << 16) + OFFSET(16:16)
```

El puntero 0:32 es construído de un puntero 16:16 tomando el selector, eliminándole el indicador de tabla y el nivel de privilegio, poniéndolo en los 16 bits de mayor orden del punturo 0:32 y añadiéndole el desplazamiento del puntero 16:16.

```
16:16 = MAKEP((HIWORD(0:32) << 3) + 7, LOWORD(0:32))
```

El puntero 16:16 se construye de un puntero 0:32 de manera inversa, tomando los 16 bits de mayor orden del puntero 0:32, se le suma el nivel de privilegio e información del indicador de tabla y ése es el selector del puntero 16:16. La parte de desplazamiento del puntero 16:16 son los 16 bits de menor orden del puntero 0:32.

La LDT es manejada separadamente; los selectores privados están localizados en un extremo de la LDT mientras los selectores compartidos están localizados en el otro. Esto no afecta la compatibilidad de los módulos de 16 bits ya que

su naturaleza segmentada hace que sean relocalizables por omisión.

6.2. Communication Manager

Este componente del OS/2 provee comunicación de datos entre una estación de trabajo OS/2 a otras estaciones y mainframes remotos sobre una red de área local, teléfono o línea dedicada. Esto permite que las aplicaciones que se ejecutan en sistemas remotos interactúen entre sí, intercambiando mensajes y archivos y compartiendo datos e impresoras.

El Communication Manager posee su propio API, un conjunto de comandos que los programas pueden usar para comunicarse con el mundo exterior. Esto incluye *comunicación avanzada de programa a programa* (APPC), un protocolo de comunicación punto a punto entre dos o más sistemas inteligentes.

6.2.1. Interfase para comunicación de dispositivos asincrónicos (ACDI)

Es la interface de programación que da soporte para quienes deseen usar los puertos seriales para comunicación asincrónica en un ambiente de multitarea. ACDI interactúa con los puertos seriales a través del sistema operativo y asegura que las interrupciones son manejadas y los datos son colocados en buffers para las aplicaciones según se requiera.

ACDI contiene verbos que permiten al programador fijar parámetros de comunicación asincrónica tales como, longitud del carácter, tasa de bits y modo control de flujo además de verbos para enviar y recibir datos a través del puerto serial.

ACDI es una interface de alto nivel, que no depende del hardware. Los desarrolladores de aplicaciones que usan ACDI no necesitan preocuparse del hardware instalado y las aplicaciones no necesitan ser cambiadas cuando el usuario cambia de hardware. De esta manera, un

programador puede desarrollar y usar un protocolo de comunicación asincrónica sin tener un conocimiento detallado del hardware de comunicación asincrónica o de los manejadores de dispositivos. Por ejemplo, el programador puede ejecutar el verbo *ComConnect* para establecer una conexión de comunicación asincrónica. ACDI dirige al modem conectado de manera apropiada (basado en cómo el usuario ha configurado el Communicaction Manager) para establecer la conexión.

ACDI puede ser accesado por múltiples aplicaciones en forma concurrente. Una sola aplicación puede ejecutar comunicación asincrónica sobre varios puertos seriales concurrentemente a través del ACDI (máximo 3 puertos seriales en las estaciones PS/2). Los dispositivos seriales configurados para usar el ACDI no pueden ser usados también por el programa LAN Requester del OS/2.

Las funciones del ACDI hacen lo siguiente:

- Independencia de los dispositivos y soporte a una variedad de equipos de comunicación asincrónica y tipos de conecciones tales como:
  - Lineas conmutadas y no conmutadas.
  - Modems (autodial/answer, manual dial/answer)
  - Conecciones directas

- Soporta conjuntos de comandos de dispositivos de comunicación asincrónica:
  - Comandos ATtention
  - CCITT V.25bis
  - No comando (conjunto de comandos dados por el usuario que soportan los circuitos que cumplen la norma CCITT V.24)

- Soporta protocolos tradicionales de comunicación asincrónica tales como:
  - Control de flujo
  - Detener proceso

- Provee puertos para la administración de recursos:
  - Tres puertos asincrónicos para computadores PS/2 (COM1, COM2 y COM3).

- Dos puertos asincrónicos para los computadores IBM AT.

- Provee interfases flexibles de entrada y salida para la transferencia de datos.

- Administra los buffers de las aplicaciones para datos recibidos y transmitidos.

- Comparte un puerto asincrónico en donde auto-dial y auto-answer son puestos en el mismo puerto.

La figura 7.3 da una vista general de cómo están estructuras las capas arquitecturales de ACDI.

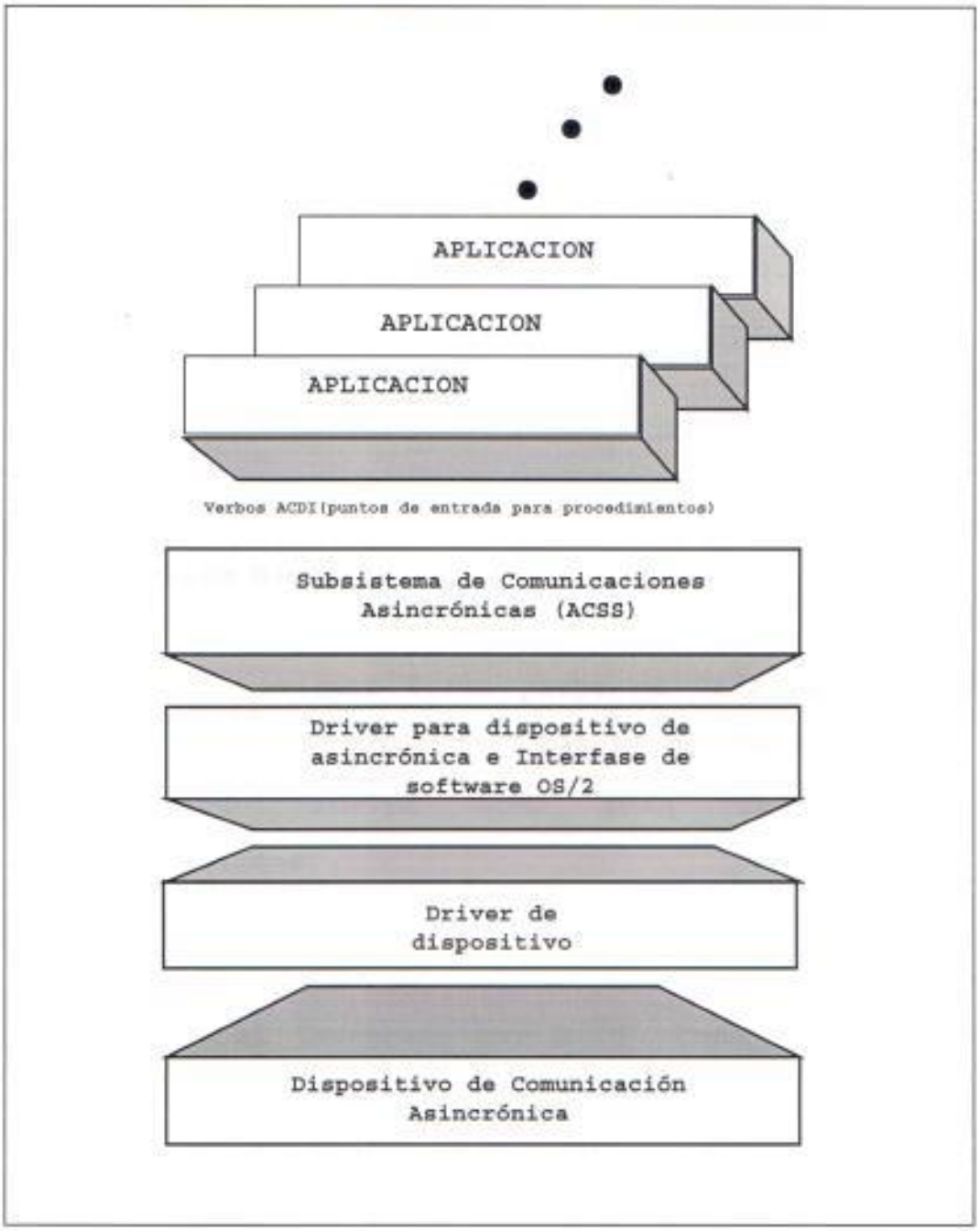


Fig. 6.3 Capas arquitecturales del ACDI

Las aplicaciones llaman al módulo de enlace dinámico ACDI el cual hace llamadas al *device driver* ACDI. ACDI programa el adaptador asincrónico y el modem conectado. Los manejadores de disposivos, incluídos en el Communication Manager, manejan el adaptador de comunicaciones asincrónico.

6.3. Database Manager

Esta herramienta forma parte de OS/2 para satisfacer los requerimientos de bases de datos tanto para usuarios finales como para programadores de aplicaciones.

Es un manejador de base de datos basado en el modelo relacional inventado por E. F. Codd en el Centro de Investigaciones de IBM en San José. Soporta el lenguaje de preguntas SQL (Structured Query Language). El modelo relacional ha sido ampliamente aceptado por la industria y los usuarios finales. La principal ventaja de este modelo es la clara separación entre la

percepción del usuario y la implementación interna de los datos.

SQL, originalmente desarrollado para el Proyecto R de IBM se ha convertido en una norma de la industria. Es considerado sencillo de aprender y lo suficientemente poderoso para expresar preguntas sofisticadas. Una sola instrucción en SQL puede ejecutar la misma función que muchas líneas de código convencional.

SQL es soportado por dos productos IBM: IBM Database 2 (DB2) y Structured Query Language/Data System (SQL/DS), ambos en los IBM Sistema/370. SQL está también incluído en Systems Application Architecture (SAA), como el componente de la interfase de bases de datos de las interfases comunes de programación.

La herramienta interactiva para el usuario final, incluída en el Database Manager, llamada Query Manager permite ingreso de datos, edición, reportes, preguntas y aplicaciones propias del usuario. Query Manager soporta la interface de preguntas de SAA y es consistente con el producto Query Management Facilities (QMF) en los Sistemas/370, el cual da interfases de preguntas y reportes a DB2 y SQL/DS.

6.3.1.Lenguaje de definición de datos

Las funciones del lenguaje de definición de datos de SQL soportados por el Database Manager de OS/2 se listan a continuación:

- CREATE y DROP TABLE: Estas funciones permiten al usuario agregar o quitar una tabla de una base de datos. Cuando una nueva tabla es agregada con CREATE a la base, se le da un nombre y cada columna en la tabla es definida separadamente con un nombre y un tipo de dato. Los tipos de datos incluidos son CHAR, VARCHAR, LONG VARCHAR, INTEGER, SMALLINT, DECIMAL, FLOAT, DATE, TIME y TIMESTAMP. Una columna puede ser definida como nula o no nula. Una tabla puede tener hasta 255 columnas.

- ALTER TABLE: Esta función permite al usuario agregar nuevas columnas a una tabla. Las

columnas se agregan a la "derecha" de la tabla.

- CREATE y DROP VIEW: Las vistas pueden ser definidas en la parte superior de las tablas existentes o vistas. Cuando una tabla es eliminada, las vistas que dependen de la tabla son también eliminadas automáticamente.

- CREATE y DROP INDEX: Los índices pueden ser creados en base a las columnas frecuentemente accesadas a fin de aumentar el desempeño. Los índices son guardados en formato B-tree con características de acceso aleatorio y de actualización.

- COMMENT sobre tablas y columnas: Los comentarios pueden ser agregados a las tablas y a las columnas.

Las columnas LONG VARCHAR pueden ser de hasta 32700 bytes. Estos campos pueden ser usados para almacenar grandes cantidades de datos como gráficos, imágenes o audio.

Catálogos integrados del sistema, similares a los de DB2, son usados para guardar las definiciones y relaciones entre las bases de datos. Tales catálogos son tratados como tablas normales con la excepción de que solamente Database Manager puede actualizar dichos catálogos. Los usuarios pueden usar el Data Manipulation Language de SQL para hacer preguntas sobre los catálogos.

6.3.2.APIs de ambiente

Database Manager tiene funciones que no son parte de la interfase de bases de datos SAA. Estas funciones están en forma de APIs invocables por el administrador. Los programadores de aplicaciones pueden usar el API para controlar el ambiente y tener acceso a las funciones de utilitarios.

Las funciones del API de ambiente permiten a una aplicación crear una base de datos, cerrarla, conectarse (START USING) a una base y

desconectarse (STOP USING) de ella. Un proceso OS/2 debe ejecutar un START USING nombre de la base de datos antes que cualquier comando SQL pueda ser usado.

Otro grupo de APIs permite a las aplicaciones obtener información respecto a las bases de datos y cambiar los parámetros de configuración de las mismas, incluyendo el tamaño del buffer pool, máximo número de transacciones permitidas, entre otros. Estos son parámetros que los usuarios pueden manejar para afinar el desempeño y el uso de recursos del Database Manager. Por ejemplo, uno puede aumentar el tamaño del buffer pool para disminuir el número de operaciones de entrada/salida.

Las funciones del API de utilitarios permiten IMPORTAR datos generados por otros programas en una tabla o EXPORTAR una tabla en uno de los formatos soportados. Otras funciones permiten RESPALDAR y RESTAURAR bases de datos, reorganizar la estructura física de la base y actualizar las estadísticas usadas por el optimizador de desempeño.

6.3.3.Soporte a multiproceso y multiusuario

Database Manager soporta acceso concurrente a la misma base de datos desde diferentes procesos OS/2. Mecanismos de bloqueo, transacciones y rollback preservan la consistencia de los datos mientras múltiples procesos están leyendo o actualizando una base de datos. Los usuarios pueden tomar ventaja de este soporte ejecutando varias aplicaciones de bases de datos concurrentemente.

Este soporte a multiprocesamiento es el fundamento del Servicio Remoto el cual soporta varias estaciones de trabajo en una LAN. En la estación servidora, se crean múltiples procesos como "servidores" para servir a "clientes" en estaciones remotas. El Database Manager administra el acceso concurrente de los procesos y asegura consistencia de la base de datos usando técnicas de administración de transacciones. El soporte de multiprocesos es también usado para procesos concurrentes de tipo "demonio". Por

ejemplo, un demonio "detector de interbloqueo" en el Database Manager se despierta periodicamente para detectar interbloqueo entre transacciones concurrentes.

Las características de comunicación y sincronización de procesos en OS/2 son usadas extensivamente para el acceso a bases de datos por parte de múltiples procesos. Muchos bloques de control y buffers en RAM son compartidos y actualizados por diferentes procesos, con los semáforos de OS/2 siendo utilizados para serializar los cambios a estas estructuras de datos compartidas.

En la siguiente discusión, el término *proceso de aplicación* es usado para describir a cada proceso que efectúa comandos de bases de datos. En un ambiente de acceso de múltiples usuarios en una LAN, un proceso de aplicación puede ser un proceso creado por los Servicios Remotos de Database Manager o puede ser un proceso local.

6.3.4.Soporte a transacciones

Una transacción es una unidad de trabajo o unidad de recuperación que el usuario o el programa de la aplicación lo considera atómico. Un ejemplo de una transacción puede ser una transferencia de dinero de una cuenta corriente a una de ahorros. El débito de la cuenta corriente y el crédito a la de ahorros debe ser hecho como una sola acción; ambas forman parte permanente o no de la base de datos. Este es un ejemplo clásico del rol de una transacción.

Database Manager provee un soporte completo a las transacciones de manera muy similar a DB2 o SQL/DS. Cualquier lectura o escritura en la base de datos es hecha como una transacción. Una aplicación que empieza a usar una base de datos (comando START USING) y procede a ejecutar comandos SQL, automáticamente inicia una transacción. Esta transacción puede terminarse explícitamente con el comando COMMIT o ROLLBACK. El comando COMMIT hace permanente cualquier cambio hecho dentro de la transacción. El comando ROLLBACK elimina de la base de datos

todos los cambios hechos dentro de la transacción.

Si por alguna razón un proceso aplicativo termina de manera anormal mientras está en medio de una transacción, la transacción del proceso es restaurada a su estado previo automáticamente por Database Manager.

La consistencia de la base de datos es también protegida por el proceso de recuperación en el evento de una "caída" del sistema. Un archivo bitácora de recuperación de los cambios en la base es mantenido, a fin de que el proceso de recuperación pueda restaurar la base de datos a un estado consistente. El estado consistente se define a continuación:

- En el momento de una falla en el sistema, todas las transacciones que han terminado exitosamente son recuperadas a ese estado.

- Todas las transacciones que estaban en ejecución (transacciones que han hecho cambios

pero que no han concluído o no han "roll back") son "rolled back".

Por lo tanto, a pesar de que algún trabajo pueda perderse, el proceso de recuperación restaura la base de datos a un estado en el cual todos los cambios a los datos hechos por transacciones concluídas son reflejados en la base de datos.

Para regresar una base de datos a un estado consistente depués de una falla del sistema, algunos de los cambios reflejados en el archivo bitácora deben deshacerse, mientras que otros deben rehacerse. Database Manager es capaz de hacer un seguimiento del estado de cada transacción y es capaz de deshacer o de rehacer las operaciones necesarias. Esta recuperación de procesos es muy similar a la de DB2.

El archivo bitácora de recuperación es también usado para implementar puntos de seguridad internos durante la ejecución de una transacción. Los puntos de seguridad pueden ser usados por Database Manager; pero no están disponibles para las aplicaciones.

6.3.5.Concurrencia

Como se mencionó anteriormente, Database Manager permite que múltiples procesos puedan accesar una base de datos de manera concurrente. El sistema de seguridad usado para mantener la integridad de los datos durante procesos concurrentes se discute en esta sección.

El sistema de seguridad de Database Manager sigue los conceptos generales de seguridad en System R, SQL/DS y DB2. Database Manager mantiene el nivel de Lecturas Repetidas de aislamiento de datos. Esto significa que dentro la transacción de un proceso, todas las actualizaciones son aseguradas a fin de que otras transacciones no puedan accesarlas; este es el corazón del concepto de integridad de datos. A más de ello, todos los registros leídos son asegurados a fin de que otras transacciones no puedan alterarlos; esta es la esencia de las Lecturas Repetidas contra otros niveles de aislamiento de datos. Repetidas preguntas SELECT dentro de una transacción de

datos idénticos cada vez bajo Lecturas Repetidas, excepto cuando la transacción misma actualiza los datos. Todos los datos permanecen asegurados hasta que la transacción es concluída o "rolled back".

Database Manager asegura objetos de datos lógicos, tablas y registros de una manera jerárquica. Las seguridades a nivel de registro da una granulidad más fina y un mejor soporte a concurrencia. El comando SQL LOCK TABLE <nombre de tabla> es soportado externamente. Esta es la única manera explícita en que uno puede directamente asegurar datos dentro de una base de datos. Si no se ejecuta el LOCK TABLE, las seguridades implícitas se aplican automáticamente.

Database Manager intenta maximizar la concurrencia en una base de datos usando seguridades intencionalmente a nivel de tablas y asegurando los registros individualmente. El acceso concurrente a los índices es soportado por medio de algoritmos y estructuras de datos que permiten altos grados de concurrencia. El valor

y rango de las claves accesadas son mantenidos por la transacción que está accesando hasta que la misma concluya.

Las seguridades de los objetos lógicos son mantenidos en memoria con estructuras de datos especializadas para optimizar el ingreso y eliminación de registros. La cantidad de memoria dedicada para seguridades puede ser especificada en la configuración. Si se obtienen muchos seguros por los registros en una tabla, los seguros son "escalados" a nivel de seguro de tabla. Esto reduce los recursos del sistema para seguridades.

Puede ocurrir interbloqueo entre las transacciones que requieren o mantienen seguros. Estas son detectadas por un demonio llamado *detector de interbloqueos*, el cual está asociado a cada base de datos. En cualquier momento que una base de datos sea usada, el detector de interbloqueos es activado periódicamente. Este detector revisa los seguros en el sistema y determina si existe una situación de

interbloqueo. El usuario puede controlar la frecuencia de actividad del detector de interbloqueo cambiando un parámetro en el archivo de configuración de la base de datos.

Si existe interbloqueo, el detector de interbloqueo selecciona una "víctima" el cual es un miembro del ciclo de interbloqueo. La transacción víctima es sleccionada aleatoriamente y "rolled back", permitiendo a las otras transacciones involucradas en el interbloqueo proseguir.

6.3.6.Administración del almacenamiento

Database Manager usa el sistema de archivos de OS/2 para el manejo de las bases de datos. Generalmente cada tabla es almacenada en un solo archivo. Además, todos los índices de la misma tabla son almacenados en un archivo aparte. Cada uno de estos archivos está dividido internamente en páginas de 4 Kb, que es la unidad normalizada de entrada/salida del administrador de datos. La

ubicación del espacio se maneja a nivel de cada página. Los campos grandes son manejados separadamente para optimizar el alojamiento de tamaños de campos de hasta 32700 bytes en longitud. Todos los campos grandes de una tabla son almacenados juntos en un sólo archivo.

Cuando una base de datos es creada, un subdirectorio especial para esa base es creado y todos los archivos que la forman, son guardados en dicho subdirectorio. Los archivos de la base de datos no están considerados a ser usados por un ente diferente a Database Manager, ya que de ocurrir esto, tendría consecuencias catastróficas para la integridad de los datos.

Database Manager asigna un área separada de memoria, llamada *buffer pool*, para cada base de datos en uso. Un buffer pool consiste de varias páginas de 4 Kb y es usada por Database Manager para leer del o escribir en el disco. El buffer pool es usado como un área de *cache* para los datos de la base. En general, para el remplazo de las páginas en el buffer pool se usa el

algoritmo LRU (Least Recently Used). La más vieja, o menos recientemente usada página de datos es remplazada por una nueva. Las páginas frecuentemente usadas, tales como nodos de los índices, tienden a estar más tiempo en el buffer pool, reduciendo el número global de entradas/salidas del disco.

6.4. Múltiples tareas

En un sistema multitarea, el sistema operativo administra el microprocesador de manera tal que todos los elementos independientes aparecen ejecutándose simultáneamente. Tres tipos de multitarea están presentes en OS/2: sesiones, procesos e hilos. Las sesiones, el elemento más grande designa qué aparece en la pantalla y dónde es enviado lo que se ingresa por teclado. Los procesos incluyen programas y sus recursos de memoria. Los hilos son elementos individuales de multitarea.

6.4.1.Sesiones

Una sesión, el elemento de multitarea más alto, consiste de uno o más procesos que usan el mismo tipo de monitor y teclado. Por ejemplo, un usuario podría empezar a usar un procesador de palabras como una sesión, una hoja de cálculo como otra y un programa de base de datos como una tercera. Cada una de estas sesiones muestran información en un monitor lógico y aceptan el ingreso desde el teclado lógico.

Cuando el usuario conmuta a una nueva sesión, OS/2 escribe el nuevo monitor lógico de la sesión al monitor físico y dirige la información digitada en el teclado físico al teclado lógico de la sesión.

6.4.2.Procesos

Un proceso es una instancia de un programa en ejecución más todos los recursos usados por el programa. Cuando OS/2 comienza un proceso, éste carga el archivo que contiene el código del

proceso. La llamada *DosExecPgm* es usada para comenzar un proceso. Es similar a la función de DOS EXEC (interrupción 21, función 4BH), excepto que bajo OS/2, los procesos padre e hijo pueden correr concurrentemente.

Bajo DOS, cuando un programa invoca la función EXEC, el proceso padre espera en el limbo hasta que el proceso hijo concluya. Bajo OS/2, el padre tiene la opción de parar hasta que el proceso hijo termine, pero también puede correr en paralelo.

6.4.3.Hilos

Dentro de cada proceso hay uno o más hilos. Un hilo es una secuencia de ejecución dentro de un proceso y es el elemento de multitarea manejado por OS/2. Cuando un proceso comienza, sólo existe un hilo. OS/2 comienza la ejecución de este hilo en el punto de entrada del programa que está siendo ejecutado por el proceso. Este hilo inicial puede comenzar otros hilos para ganar más concurencia.

Para cada hilo, OS/2 mantiene una prioridad de hilo, una pila, un estado del procesador (copias de todos los registros del procesador) y un estado del hilo (bloqueado, listo o activo). Todos los otros recursos son apropiados por un proceso y son compartidos por todos los hilos de ese proceso. Por ejemplo, si un hilo abre un archivo y obtiene un manejador de archivo (file handle), otro hilo en ese proceso puede usar el manejador de archivo para accesar el archivo.

A pesar de que OS/2 es un sistema operativo multitarea, tiene acceso a sólo un microprocesador; por lo tanto, un sólo hilo puede estar en ejecución en un momento dado. Los otros hilos pueden estar: listos para ejecutar o bloqueados. Cuando el despachador de OS/2 determina que un nuevo hilo debería correr, éste guarda los contenidos de todos los registros del procesador en el área de estado del procesador para el hilo viejo. El despachador carga entonces los registros del área de estado del procesador del nuevo hilo. Esto permite al nuevo

hilo continuar justo donde se detuvo la última vez que fue interrumpido.

Los procesos pueden tener varios hilos ejecutando el mismo código. Cada uno de los hilos tendrá su propia pila, pero como los hilos tiene acceso a la misma memoria, deben tener precaución cuando usan variables estáticas y estructuras de datos. Mecanismos para la comunicación entre tareas, tales como semáforos RAM, pueden usarse para prevenir que otros hilos accesen estas áreas comunes hasta que el primer hilo termine.

6.4.4.Despachando hilos

Dado que un solo hilo a la vez puede ser ejecutado en el sistema, OS/2 usa un esquema basado en prioridad para determinar cuál hilo se ejecutará en un momento dado. Cuando se lo crea, a cada hilo se le asigna una clase prioridad así como un número de prioridad. Esta combinación determina la prioridad final del hilo. En todo momento, el hilo listo a ejecutar de mayor prioridad tiene acceso al procesador. Cuando el

hilo de mayor prioridad se convierte en no listo (cuando está esperando por I/O, cuando está esperando para sincronizarse con otro hilo o cuando su porción de tiempo termina), OS/2 entrega al procesador el siguiente hilo de más alta prioridad listo para ejecutarse. Cuando un hilo de mayor prioridad se convierte en listo otra vez (cuando la operación de I/O termina o cuando recibe sincronización de otro hilo), el hilo de menor prioridad abandona el control del procesador.

OS/2 usa "time slicing" para asegurar que todos los hilos de igual prioridad tiene una oportunidad de ejecutarse. Si los hilos que tienen una mayor prioridad están listos para ejecutarse al final de cada porción de tiempo, uno de ellos ganará acceso al procesador. Además, si otros hilos de igual prioridad están listos para ejecutarse, OS/2 termina el hilo actual y le da al siguiente hilo acceso al procesador durante su porción de tiempo. Si no hay otros hilos de igual o mayor prioridad listos para ejecutarse, OS/2 devuelve control del procesador al hilo original.

La prioridad de los hilos tiene tres clases, cada una de las cuales tiene 32 niveles de prioridad (0 a 31). La clase de más alta prioridad es la *tiempo crítico*. Todos los hilos en esta clase son de mayor prioridad que los hilos en las otras dos clases. Típicamente, los hilos en esta clase de prioridad serán aquellos que manejan áreas sensibles al tiempo, tales como los que sirven a dispositivos de comunicación de datos de alta velocidad.

La siguiente clase más alta es la *regular*. Esta clase es usada por la mayoría de los programas de aplicaciones y comandos de OS/2 invocados desde teclado. Por omisión, OS/2 provee opciones adicionales de despacho dentro de la clase regular para asegurar que todos los hilos en la clase obtengan una oportunidad para ejecutarse.

La clase con la prioridad más baja es la *idle time*. Los hilos en esta clase no se ejecutan a menos que no hayan hilos de cualquiera de las otras dos clases listos para ejecutarse. Esta

clase es típicamente usada para programas tales como respaldo automático de discos, spoolers de impresora o cualquier otro programa que se ejecute en el background.

6.5. Comunicación entre procesos

En cualquier sistema multitarea, deben estar disponibles mecanismos para que los elementos independientes se sincronicen y comuniquen entre sí. En OS/2, cada hilo corre como si estuviera apropiado de todo el procesador. Sin embargo, los hilos a menudo necesitan pasar información a otro o estar seguro de que un hilo ha completado su operación antes de que otro hilo comience la suya. Estos mecanismos son especialmente útiles entre hilos en diferentes procesos, porque los procesos están normalmente aislados el uno del otro en virtud del modo protegido del procesador.

OS/2 provee los siguientes mecanismos de comunicación entre procesos (IPC[5]): semáforos, pipes, memoria

[5] IPC: Interprocess communication.

compartida, colas y señales. En la mayoría de los casos, la comunicación entre procesos requiere que éstos estén de acuerdo en cuanto a los nombres y localizaciones de los recursos. Por lo tanto, los procesos deben ser designados y codificados especialmente para comunicarse el uno con el otro. La excepción es el pipe, el cual permite una comunicación transparente.

6.5.1.Semáforos

Los semáforos son objetos que pueden usar los hilos para coordinar el acceso a los recursos o para sincronizarse con otro. Los semáforos tienen dos estados: apropiado y no apropiado (o, dependiendo de cómo son usados, fijado y limpio).

Para coordinar el acceso a los recursos (tal como un área sensible de datos), un semáforo puede representar propiedad del recurso. Para hacer esto, todos los hilos deben adherirse a la convención de que antes de que cualquier hilo pueda accesar el recurso crítico, debe primero

pedir la propiedad del semáforo por medio de la llamada *DosSemRequest*. Si un hilo llama a *DosSemRequest* para solicitar un semáforo que ya tiene dueño, una de tres cosas puede suceder (dependiendo de las opciones que el hilo especificó en la llamada): la llamada puede quedar por siempre en espera a que el semáforo esté limpio; puede especificar un número especificado de milisegundos; o sencillamente no espera.

Varios hilos pueden requerir apropiarse del mismo semáforo. Si los hilos elijen esperar (en vez de retornar inmediatamente), aquellos que no ganan acceso inmediatamente al semáforo continuan esperando hasta que el hilo que es dueño libere al semáforo. Cuando esto sucede, los hilos en espera son despachados otra vez, y el hilo en espera con mayor prioridad se convierte en el nuevo propietario, incluso si un hilo de menor prioridad ha estado esperando más tiempo. Los otros hilos continuan esperando. Si hilos de igual prioridad están esperando, el hilo que ha

estado esperando por más tiempo se convierte en el nuevo propietario.

El uso de un semáforo para representar la propiedad de algún recurso permite a los hilos leer o escribir áreas de datos sensibles sin el temor de que otros hilos interrumpan antes de que terminen.

Otro uso de los semáforos es el de sincronizar las actividades de los hilos. En esta situación, un semáforo puede estar fijado o limpio. Por ejemplo, supongamos que un proceso consiste de un hilo de inicialización más varios otros hilos. El hilo de inicialización debe ejecutarse primero a fin de definir las estructuras de datos necesarias para los otros hilos. Después de que complete estas actividades, sin embargo, todos los otros hilos están en libertad de ejecutarse.

Para manejar esta actividad, el hilo de inicialización llamaría a *DosSemSet* para fijar el semáforo. Esto indicaría que la actividad de inicialización está ejecutándose y los otros

hilos no podrían correr. Cuando el hilo inicial complete su actividad, llamaría a *DosSemClear* a fin de limpiar el semáforo. Los otros hilos que dependen del inicial para terminar sus actividades llamarían a *DosSemWait* antes de hacer cualquier cosa. Esta llamada hace que los hilos esperen hasta que el semáforo sea limpiado antes de que puedan proceder.

OS/2 define dos tipos de semáforos:

- Semáforos del sistema, y
- Semáforos RAM.

Los semáforos del sistema son usados para la comunicación entre hilos de diferentes procesos. Los semáforos RAM son usados por hilos en el mismo proceso. La diferencia entre los dos está en cómo son creados y accesados, pero los semáforos del sistema también proveen unas características adicionales.

El uso de semáforos RAM depende de que los hilos sean capaces de accesar la memoria que contiene a los semáforos. Cuando dos hilos están en

diferentes procesos, ellos no pueden accesar la memoria del otro a menos que explícitamente cooperen para compartir un segmento de memoria. Luego, un mecanismo diferente debe ser usado.

Los semáforos del sistema usan un esquema basado en nombres para la creación y acceso. Este mecanismo es casi exactamente igual al empleado para crear y accesar archivos. Para crear un semáforo del sistema, un hilo llama a *DosCreateSem* y le da un nombre. En respuesta, OS/2 retorna un manejador que el hilo puede usar en otras llamadas para manejar al semáforo. Cuando otro hilo quiere usar el semáforo del sistema, debe primero llamar a *DosOpenSem*, especificando el nombre del semáforo. Si el semáforo está disponible, OS/2 retorna un manejador que se puede usar para otras llamadas. Cuando un hilo termina de usar un semáforo del sistema, éste llama a *DosCloseSem* para cerrar el semáforo. Cuando todos los hilos que usan un semáforo del sistema lo cierran, OS/2 lo borra automáticamente.

OS/2 mantiene un directorio para hacer un seguimiento a los semáforos del sistema. Este directorio se llama SEM y es mantenido en memoria en vez de en disco. En cualquier circunstancia en que los hilos se refieran a semáforos del sistema por el nombre, deben incluir \SEM\ como parte del nombre. Luego, el formato del nombre de un semáforo del sistema es

\SEM\NOMBRE.EXT

donde las reglas para definir a NOMBRE.EXT son las mismas que para los nombres de archivos.

A pesar de que un semáforo RAM colocado en memoria compartida aparentemente puede ofrecer la misma funcionabilidad que un semáforo del sistema, algunas diferencias sutiles pueden convertirse en importantes. OS/2 mantiene información acerca del propietario actual de un semáforo del sistema. Si un proceso sale y todavía es dueño de un semáforo del sistema, OS/2 notificará a cualquier hilo en otros procesos que están esperando a ese semáforo despertándolos y retornando un código de error indicando que el dueño del semáforo ha salido. Los semáforos RAM

ganan su simplicidad omitiendo este mecanismo de seguridad.

6.5.2.Pipes

Son otra clase de objeto que los hilos en diferentes procesos pueden usar para comunicarse con otros. Un pipe es simplemente un área de memoria usada para guardar datos, un buffer circular que es el sustituto en RAM de un archivo.

Usar un pipe es muy similar a usar un archivo. Un hilo escribe información a un pipe usando llamadas ordinarias de I/O. A menudo, los hilos que se comunican por pipes ni siquiera se dan cuenta de que están usando pipes en vez de archivos. Un hilo crea un pipe usando la función *DosMakePipe* y especificando el tamaño del pipe deseado. En respuesta a la llamada *DosMakePipe*, OS/2 retorna dos manejadores al pipe: el manejador de escritura y el de lectura.

Si un hilo intenta escribir a un pipe y éste está lleno, el hilo es detenido hasta que otro hilo lea suficiente información del pipe que permita al primero escribir lo que desea. Del mismo modo, si un hilo intenta leer de un pipe y no existen suficientes datos para satisfacer el requerimiento, el hilo es detenido hasta que otro escriba datos adicionales al pipe. Cuando un hilo termina de usar un pipe, éste cierra los manejadores con la llamada DosClose. Cuando no hay más manejadores de pipes, OS/2 automáticamente cierra el pipe.

6.5.3.Pipes nombrados

Es el más poderoso y completo medio de comunicación entre procesos (IPC) que ofrece OS/2, es además el único mecanismo de IPC que puede transferir datos a través de una red. Proveen un API de programación similar al de los archivos para intercambio de datos en ambos sentidos. Con los pipes nombrados, los procesos pueden intercambiar datos como si estuvieran

escribiendo a, o leyendo de, un archivo secuencial. Se ajustan muy bien para implementar programas servidores que requieren líneas de comunicación del tipo *muchos a uno*. Los pipes nombrados dan una plataforma robusta para el desarrollo de aplicaciones cliente/servidor. Lo que ofrecen es:

- Proveer un método para intercambiar datos e información de control entre procesos corriendo en diferentes computadores.

- Proveen llamadas en el API optimizadas para la implementación de *Llamadas Remotas de Procedimientos* (RPC) que son parejas de mensajes requerimiento/respuesta que pueden ser usados para la implementación de aplicaciones cliente/servidor usando llamadas similares a procedimientos. La invocación a un procedimiento remoto resultará en un intercambio requerimiento/respuesta sobre el pipe nombrado. El requerimiento que llega puede ser usado para disparar un procedimiento en un servidor y pasarle los parámeros que

requiera; los resultados, serán retornados por el servidor en un mensaje de respuesta sobre el mismo pipe.

Para establecer un pipe nombrado, un proceso servidor debe primero crearlo con la llamada *DosMakeNmPipe* la cual retorna un manejador que identifica el pipe. Un pipe nombrado es identificado por un nombre único que está de acuerdo con la convención de OS/2 para nombrar archivos, a pesar de que no se crea uno. Los pipes son llamados: \PIPE\nombre, y los pipes locales son conocidos por ese nombre. Los pipes remotos se los llama: \\server\PIPE\nombre, donde "server" es el nombre en la red de la máquina servidora. Un pipe local puede ser solamente usado por procesos en la misma máquina; mientras que un pipe remoto puede ser usado por procesos conectados por una LAN. Esto significa que una aplicación cliente puede comunicarse con un pipe nombrado desde cualquier máquina en la red.

Una vez que el pipe es creado en el servidor, debe estar en estado de *escuchar* antes de que un

cliente pueda conectarse a él. El servidor hace esto con la llamada *DosConnectNmPipe*. El pipe está ahora en modo ESCUCHA esperando a que un proceso cliente lo abra. En este punto, cualquier proceso cliente que conoce el nombre completo del pipe nombrado puede abrirlo con *DosOpen*. Esta es la misma llamada de OS/2 para abrir un archivo. Si el *DosOpen* es exitoso, retorna un manejador y el pipe esta en modo CONECTADO.

Si la llamada falla y retorna ERROR_PIPE_BUSY, el cliente puede hacer un *DosWaitNmPipe* lo cual bloqueará el proceso hasta que el pipe esté disponible. Qué sucede cuando varios clientes están esperando por el mismo pipe?. OS/2 despertará al proceso que más ha estado esperando, cuando el pipe esté disponible. Este proceso puede otra vez hacer un *DosOpen* para conectarse al pipe.

Una vez que un pipe está en el modo CONECTADO, los procesos servidor y cliente pueden usar las

llamadas del subsistema de archivos de OS/2 para comunicarse por medio de él usando:

- *DosWrite* o *DosWriteAsync* para escribir datos al pipe.
- *DosRead* o *DosReadAsyn* para leer datos del pipe.
- *DosBufReset* para sincronizar diálogos de lectura y escritura. Hace esto bloqueando el proceso que llama en un extremo del pipe hasta que todos los datos que haya escrito sean leídos en el otro extremo del pipe.

Cuando un proceso cliente desea cerrar el pipe realiza un *DosClose*. Esta es la misma llamada usada en OS/2 para cerrar archivos. El pipe está ahora en el estado CERRADO. El proceso servidor puede ahora ejecutar un *DosDisConnectNmPipe* lo que pone al pipe en modo DESCONECTADO. El servidor puede ahora hacer otro *DosConnectNmPipe* y esperar por el siguiente cliente, o puede cerrar el pipe nombredo con *DosClose*. Un servidor puede también hacer un *DosClose* sin un *DosDisConnectNmPipe* previo. Esto liberará al

manejador del pipe y permitirá al cliente leer cualquier dato que permanezca en el buffer. Sin embargo, un pipe cerrado no puede ser reusado otra vez sin un *DosMakeNmPipe*.

Los semáforos del sistema pueden ser usados en conjunto con un pipe nombrado local para controlar el acceso al pipe. Un proceso de lectura puede usar *DosSemWait* para esperar a que los datos lleguen a uno o más pipes. Esto evita métodos más costosos, tales como dedicar un hilo para cada pipe o "polear" cada pipe con el modo NOESPERA. La llamada *DosSetNmPipeSem* es usada para asociar un semáforo del sistema con una instancia de un pipe nombrado local. Hasta dos semáforos pueden ser unidos a un pipe, uno para el servidor y otro para el cliente. Un proceso puede revisar el estado de los semáforos del pipe nombrado con la llamada *DosQNmPipeSemState*, la cual retorna información acerca de los pipes nombrados locales conectados a un semáforo específico del sistema, así como información adicional de las operaciones de I/O que se pueden realizar en los pipes.

6.5.4.Memoria compartida

OS/2 provee varias funciones que habilitan a varios hilos de diferentes procesos para compartir los mismos segmentos de memoria. Un mecanismo para compartir segmentos involucra el uso de las funciones *DosAllocShrSeg* y *DosGetShrSeg*. Estas dos funciones permiten a los hilos accesar a los segmentos por nombre, parecido a como los hilos accesan a los semáforos del sistema. Cuando un hilo en otro proceso necesita accesar los segmentos compartidos, usa *DosGetShrSeg* especificando el nombre del segmento compartido que desea accesar. Este nombre debe incluir el directorio \SHAREMEN\.

Los segmentos compartidos pueden ser usados para transferir datos directamente de un proceso a otro. Para este tipo de actividad, un hilo llama a *DosAllocSeg* para asignar un segmento de memoria. Esta función no coloca un nombre para el segmento en el directorio SHAREMEN, sino un

parámetro en la función indica que el segmento puede ser compartido. Una vez que un hilo ha asignado un segmento compartido y colocado información en ese segmento, puede compartir el segmento con otro proceso con *DosGiveSeg* y especificando la identificación del proceso que compartirá el segmento.

6.5.5.Colas

Una cola es un lugar donde un hilo puede colocar un mensaje para otro hilo, muy similar a un casillero postal que es un lugar donde las personas pueden enviar mensajes a otras. Los hilos no necesitan sincronizarse para usar las colas como lo necesitarían si usaran pipes. Por el contrario, un hilo simplemente envía un mensaje a la cola para mantenerlo seguro y otro hilo inquiere por el mensaje después. Las colas son útiles para transferir memoria compartida entre los procesos, pero los hilos en el mismo proceso pueden también usar las colas para intercambiar información.

Las colas usan el mismo mecanismo para nombrarlas que el que usan los semáforos del sistema y los segmentos compartidos. Un hilo llama a *DosCreateQueue* y especifica un nombre para la cola. OS/2 mantiene este nombre en un directorio residente en memoria llamado QUEUES. El proceso que crea la cola es el dueño de la misma. Solamente los hilos en el proceso propietario pueden recibir mensajes de la cola, limpiar la cola o cerrar la cola. Cualquier hilo puede enviar mensajes a la cola. Otra vez, esto es análogo a un casillero postal, en donde el dueño del casillero es el único que puede recibir correspondencia o cancelar el casillero, pero cualquiera puede enviar correspondencia al casillero.

Antes de que un hilo en otro proceso pueda enviar mensajes a la cola, debe primero abrirla con *DosOpenQueue* y especificar el nombre de la cola. El nombre debe incluir el nombre de directorio \QUEUES\. En respuesta, *DosOpenQueue* retorna un manejador que el hilo puede usar cuando invoca

otras funciones de manejo de cola y la identificación del proceso propietario de la cola. Una vez que el hilo abre la cola, puede enviarle un mensaje con *DosWriteQueue*.

Múltiples mensajes pueden enviarse a la cola, el orden en el que los mensajes son almacenados depende de un parámetro que se especifica al momento de crearla. Los mensajes pueden mantenerse en orden FIFO, LIFO o prioritario (en el que el hilo que envía el mensaje especifica una prioridad de 0 a 15). Cualquier hilo propietario de la cola puede recibir mensajes de ella con *DosReadQueue*. Esta función quita un mensaje de la cola y le da su dirección al hilo. Si no hay mensajes en la cola, el hilo puede esperar hasta que llegue un mensaje o puede retornar sin recibir un mensaje. Una vez que el hilo receptor ha procesado el mensaje lo quita de la cola, éste es responsable de liberar el segmento que obtuvo; esto se hace con la función *DosFreeSeg*.

Los hilos de los procesos propietarios pueden usar llamadas adicionales. *DosPeekQueue* permite a un hilo leer un mensaje de la cola sin removerlo de la misma. Usando *DosPeekQueue* un hilo puede revisar una cola en busca de mensajes específicos, luego quitar esos mensajes con *DosReadQueue*. La función *DosQueryQueue* retorna el número de mensajes que han permanecido en la cola. *DosPurgeQueue* es usada para eliminar todos los mensajes de la cola. Por último, *DosCloseQueue* es usada para cerrar una cola y quitar su nombre del directorio QUEUES.

6.6. Un mejor DOS: Múltiples máquinas DOS virtuales

La mayoría de los usuarios que consideran cambiarse a OS/2 esperan que OS/2 provea un ambiente compatible DOS para ejecutar sus aplicaciones favoritas en DOS. También esperan que las capacidades de multitarea de OS/2, una de sus características más importantes, se aplique a sus actuales aplicaciones DOS. Dado que OS/2

está basado en el procesador 80386, estos requerimientos se ven satisfechos.

Con un diseño basado en el modo virtual 8086, OS/2 mantiene un ambiente de sistema protegido, inclusive cuando se ejecutan aplicaciones DOS. Esto mejora la integridad del sistema y hace posible ejecutar varias tareas DOS y OS/2 a la vez. El mecanismo de demanda de páginas de OS/2 de asignar memoria del sistema para cada máquina virtual DOS (VDM); por lo tanto, varias aplicaciones DOS corren en memoria virtual sin incrementar los requerimientos de memoria física del sistema.

6.6.1. Limitaciones de compatibilidad con DOS del OS/2 1.x

El diseño original de OS/2 estuvo basado en las capacidades del procesador Intel 80286; por lo tanto, la única manera práctica que ejecutar una aplicación DOS en versiones de 16 bits de OS/2 es en modo real. Esto se debe básicamente porque las aplicaciones DOS direccionan código y datos

en memoria usando el formato de direcciones *Segmento:Desplazamiento*, basado en el procesador Intel 8088/8086. En modo protegido, sin embargo, todos los programas deben usar el formato de direccionamiento *Selector:Desplazamiento*. Las aplicaciones OS/2 están escritas para correr en modo protegido y el diseño de multitarea permite que muchas aplicaciones OS/2 se ejecuten concurrentemente. Para ejecutar una aplicación DOS simultáneamente con muchas aplicaciones OS/2, el sistema conmuta entre el modo real y el modo protegido cuando es necesario.

Impacto en el desempeño al conmutar modos.

El Intel 80286 puede conmutar fácilmente del modo real al modo protegido, a pesar de que esto toma muchos ciclos de CPU. Sin embargo, no tiene provisión para conmutar de regreso directamente al modo real. Conmutar de regreso al modo real requiere reinicializar el procesador, en efecto, poniéndolo en un estado de recién encendido. Esto debe ser hecho a la vez que se mantiene toda la información de control del sistema, todos los

contextos de los procesos y estados de los recursos del sistema. Por esto, conmutar frecuentemente entre el modo real (donde un programa DOS se está ejecutando), y el modo protegido (donde varios programas OS/2 pueden estar corriendo), genera un alto grado de sobrecarga al sistema. Por lo tanto, el OS/2 de 16 bits soporta solamente una sola aplicación DOS corriendo en el foreground.

Pérdida de integridad del sistema en el modo real.

OS/2 da a cada proceso en modo protegido su espacio de direcciones independiente para código y objetos de datos. Esto protege al sistema de las aplicaciones y encapsula el código y datos de cada aplicación dentro de un único espacio de direcciones protegido de otras aplicaciones. Esto también previene que aplicaciones con errores inhiban al sistema, ya que los recursos de propiedad del sistema son accesibles sólo por código del sistema. Este alto nivel de protección, una característica de OS/2 no

existente en DOS, aumenta la integridad y confiabilidad del sistema.

Cuando OS/2 conmuta al modo real, una aplicación DOS puede accesar directamente cualquier objeto en memoria entre 0 y 1 Mb, incluyendo porciones del kernel de OS/2 y manejadores de dispositivos. Un programa DOS que inadvertidamente accese cualquier parte del código del sistema u objetos de datos bajo 1 Mb podría alterar el sistema. Además, un programa DOS podría directamente accesar cualquier dispositivo de E/S y hacer que éste entre en un estado desconocido. Esto podría llevar a que el dispositivo sea inútil en el ambiente de modo protegido de OS/2 y hacer que el manejador de dispositivos de OS/2 falle.

Aplicaciones DOS suspendidas en el background.

En OS/2 1.x, los programas son suspendidos cuando el usuario conmuta a una sesión en modo DOS. Esto es principalmente para evitar la sobrecarga asociado con conmutar entre el modo real y el protegido. En cualquier momento que un usuario OS/2 conmuta de una sesión DOS, toda ejecución en

esa sesión, incluyendo el manejo de interrupciones, se detienen. Por lo tanto, si la aplicación DOS instala un manejador de interrupciones (que previene conflictos con los manejadores de dispositivos de OS/2), éste solamente recibe las interrupciones mientras el programa DOS corra en el foreground.

Espacio disponible limitado para las aplicaciones.

El pequeño espacio disponible para las aplicaciones en OS/2 1.x, evita que algunas aplicaciones DOS puedan cargarse en memoria. OS/2 1.x instala parte de su kernel, código de manejadores de dispositivos y datos en memoria fija bajo el primer megabyte, a fin de las funciones del sistema se puedan ejecutar en modo real. Por lo tanto, las aplicaciones DOS tienen menos espacio para su código y datos que las que tienen bajo el sistema operativo DOS. OS/2 1.X da aproximadamente 520 Kb de memoria para las aplicaciones DOS cuando se usan las opciones de configuración por omisión del sistema. Este problema no es raro, incluso en el ambiente

nativo de DOS, ya que instalar TSRs y manejadores de dispositivos tienen el mismo efecto. DOS 4.01 ofrece aproximadamente 570 Kb libres usando la configuración por omisión del sistema.

Para obtener memoria más allá de los 640 Kb, las especificaciones XMS y EMS se usan como extensiones de DOS. Estas especificaciones permiten a una aplicación DOS accesar memoria física sobre 1 Mb. A pesar de que muchas aplicaciones DOS han sido mejoradas para usar estas técnicas, tanto XMS como EMS no son soportadas por OS/2 1.3.

6.6.2.Múltiples máquinas virtuales DOS

El sistema de múltiple máquinas virtuales DOS (MVDM) está diseñado para explotar una de las mayores innovaciones del 80386 llamado modo virtual 8086 (V86), que le permite a OS/2 ejecutar múltiples tareas DOS dentro del ambiente del modo protegido del 80386. A partir de OS/2 2.x, esta tarea DOS es una máquina virtual DOS

(VDM), el cual corre como un solo proceso del modo V86. Cada VDM en OS/2 2.x es manejado como una sesión de un solo proceso. El despachador de OS/2 controla la conmutación de tareas de las VDM de la misma forma que los procesos de las aplicaciones OS/2.

Conmutación rápida de modo.

El modo virtual 8086 es un superconjunto del modo protegido, el cual está operativo habilitando el bit VM del registro EFLAGS del 80386. El procesador provee soporte en hardware de alto rendimiento para conmutar entre el modo V86 y el modo protegido. Debido a esto y otros mecanismos de soporte en hardware, la mayoría de la sobrecarga en el sistema operativo asociado con la conmutación de modos ha sido eliminada. Cuando se conmuta entre el modo V86 (aplicaciones DOS) y el modo protegido (aplicación OS/2 o código del sistema), el hardware del 80386 automáticamente activa los mecanismos de protección apropiados no disponibles en el 80286.

La ejecución en modo real ha sido totalmente eliminada a partir de OS/2 2.x. Esto hace posible deshacer todo el código específico del modo real de los manejadores de dispositivos y módulos del kernel, los cuales eran ejecutables bimodales en versiones previas de OS/2. Por lo tanto, los nuevos manejadores de dispositivos de OS/2 2.x están escritos como ejecutables puramente en modo protegido. Esto simplifica la lógica y reduce el tamaño de los manejadores de dispositivos.

Integridad del sistema mejorada.

Corriendo aplicaciones DOS en el modo virtual 8086 y usando la características de paginamiento del 80386, es posible mantener un ambiente del sistema totalmente protegido. Cada programa DOS está encapsulado en su propio espacio de direcciones lineales de 1 Mb. Por lo tanto, un programa DOS no puede corromper ningún dato o código del sistema, o los datos o código de otra aplicación. Si la tarea DOS causa un trap o una excepción, será manejada completamente por el sistema operativo para mantener integridad.

El sistema selectivamente aisla las aplicaciones DOS de los dispositivos de E/S que son administrados exclusivamente por los manejadores de dispositivos de OS/2. Estos dispositivos son luego emulados o virtualizados para una o más aplicaciones DOS. Esta característica es nueva a partir de OS/2 2.x, el VDD provee a cada aplicación DOS con una instancia virtual del hardware real, el cual es controlado por un PDD. Si es necesario por desempeño u otras razones, un VDD-PDD puede cooperar para darle a un programa DOS acceso directo a un puerto en particular de E/S o un rango de puertos. Controlando el acceso al hardware de esta manera, las aplicaciones DOS no pueden corromper a los dispositivos que están ejecutando funciones de E/S para un proceso OS/2.

Múltiples sesiones DOS.

MVDM hace posible iniciar hasta 16 sesiones concurrentes DOS, cada una operando en su propio espacio de direcciones lineales independiente de 1 MB. Esto brinda verdadera multiprogramación al ambiente compatible de DOS en OS/2. El usuario

puede correr múltiples programas DOS de la misma manera que cuando corre múltiples aplicaciones OS/2.

Memoria disponible incrementada.

La memoria disponible base (bajo los 640 Kb) en cada VDM está sobre los 610 Kb, antes de cargar cualquier manejador de dispositivos DOS por el usuario y/o TSRs. Esto incrementa la memoria base disponible en OS/2. Ya que la memoria base es mayor que en el propio DOS, los usuarios de OS/2 están ahora en la capacidad de cargar algunos TSRs DOS con más grandes aplicaciones DOS, las cuales no cabrían juntas en la memoria base más pequeña disponible en DOS. Cualquier manejador de dispositvos de OS/2 instalado por el usuario no afectará la cantidad de espacio disponible para las aplicaciones DOS corriendo bajo MVDM.

Servicios de memoria EMS y XMS.

Muchas aplicaciones populares DOS están ahora usando EMS y/o XMS para ganar acceso a memoria en

modo protegido en los procesadores 80286 y 80386. Esto permite a las aplicaciones DOS accesar memoria sobre el límite de 1 MB del direccionamiento en modo real, para tener un espacio total de códigos y datos muy grande cargado en memoria para una velocidad de ejecución mejorada. La configuración normalizada de OS/2 2.x provee tanto funciones EMS y XMS.

EMS, basado en la versión 4.0 de EMS de (LIM) Lotus-Intel-Microsoft, permite a las aplicaciones DOS asignar y accesar hasta 32 Mb de memoria expandida en hasta 255 objetos EMS. Estos objetos pueden ser mapeados en el espacio de memoria base (bajo 1 Mb) a fin de que la aplicación DOS pueda accesar espacios de memoria muy grandes. Las funciones de administración de memoria virtual propias de OS/2 emulan la memoria expandida usando el mecanismo de paginamiento del 80386. Por lo tanto, el uso de EMS en una VDM no afecta la habilidad de las aplicaciones DOS en otras VDM para ejecutar funciones similares de EMS.

Las funciones XMS proveen otro mecanismo para las aplicaciones DOS para que accesen memoria sobre el límite de 1 Mb. XMS soporta el uso del área "HIMEM" - un área de 64 Kb justo sobre 1 MB - que puede ser usado para código y datos por la aplicación DOS. Otras funciones XMS permiten mover código y datos en memoria extendida y desde memoria extendida a memoria base.

Bajo MVDM, los posicionamientos en memorias XMS y EMS son admininistrados por la memoria virtual paginable de OS/2, no como memoria física fija. Por lo tanto, la memoria total posicionada puede exceder la cantidad de RAM del sistema.

BIBLIOGRAFIA

ABRAHAMS, John. *Integrated PBX Systems*. Manchester, UK: National Computing Centre.

ARMBRUST, Steven & FORGERON, Ted. *Multiple Tasks*. PC Tech Journal, November 1987, 90-106.

CAMPBELL, Joe. *C Programmer's Guide to Serial Communications*. Carmel, Indiana: Howard W. Sams & Co., 1987.

Cisco Systems. *Internetworking Technology Terms and Acronyms*. USA: Cisco Systems, Inc., 1992.

CONKLIN, Dick. *OS/2 Notebook: The best of the IBM Personal Systems Developer*. Redmon, Washington: Microsoft Press, 1990.

ERICSSON, ____. *INTERFACE DESCRIPTION*. Documento No. 1/155 19-CNA 103 321 Ue, 1987.

ERICSSON, ____. *PARAMETER DESCRIPTION*. Documentos No. 1/190 84-CNA 103 321 Ue y 1/190 84-CNA 103 423 Ue, 1987.

ERICSSON, ____. *DESCRIPTION*. Documento No. 1551-ROF 137 5215/2 Uen, 1988.

IBM Corporation. IBM OS/2 Extended Edition Version 1.3 *ACDI Programming Reference*. USA: IBM Corp., 1990.

IBM Corporation. *7171 ASCII Device Attachment Control Unit*. Reference Manual and Programming Guide. USA: IBM Corp., April 1989.

IBM Corporation. *IBM VS COBOL for DOS/VSE Release 3*. San José, California: IBM Corp., May 1981.

IBM Corporation. IBM Virtual Storage Extended Advanced Functions. *System Control Statements Version 2 Release 1*. USA: IBM Corp., 1987.

RANADE, Jay. *Introduction to SNA networking using VTAM/NCP*. New York, NY: McGraw-Hill Book Company, 1989.