Universidad Nacional Autónoma de México

ENEP ARAGON

"Mecanismos de Comunicación en Ambientes de Computación Distribuida"

TESIS

Que para obtener el título de:
INGENIERO EN COMPUTACIÓN
P r e s e n t a:

Adrián Erik Pérez Vargas

San Juan de Aragón 1995

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

VNIVERSIDAD NACIONAL
AVFNºMA DE
MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES ARAGON
JEFATURA DE CARRERA DE INGENIERIA EN COMPUTACION

ING. FERNANDO FLORES ZAVALETA

ING. SILVESTRE LOPEZ ABUNDIO

ING. RICARDO GUTIERREZ OROZCO

ING. BLANCA ESTELA CRUZ LUEVANO

ING. GABRIELA GONZALEZ HERNANDEZ

Informamos a ustedes de la autorización que se le concede al alumno ***ADRIAN ERIK PEREZ VARGAS***, para desarrollar el trabajo de tesis ***"MECANISMOS DE COMUNICACION EN AMBIENTES DE COMPUTACION DISTRIBUIDA"*** dirigida por el ***Ing. Fernando Flores Zavaleta***, solicitando a ustedes, sean tan amables de revisar el avance del mismo y hacer las observaciones que consideren pertinentes, o en su caso, indicar al alumno si dicha revisión se hará a la conclusión del trabajo de tesis.

Sin otro particular, aprovecho la ocasión para enviarles un cordial saludo.

A T E N T A M E N T E
"POR MI RAZA HABLARA EL ESPIRITU"
San Juan de Aragón Edo. de Méx., Agosto 9 de 1995.
EL JEFE DE LA CARRERA

ING. SILVIA VEGA MUYTOY

SVM/gga.

A los pocos que creyeron en mí,
por el apoyo que me dieron,
y a los muchos que no creyeron,
por el coraje que me infundieron.

Adrián Erik Pérez Vargas
Octubre de 1995.

INDICE

Objetivos.

Objetivo General.

Describir los mecanismos esenciales sobre los que se encuentra basada la rama de la computación conocida como computación distribuida, mostrando sus diferentes tipos, sus técnicas de programación y el papel que funge esta dentro de la computación empresarial.

Objetivos Específicos.

- Ubicar a un sistema distribuido dentro de los sistemas de redes de computadoras.
- Explicar el termino topología de red y analizar las topologías más comunes.
- Describir el papel de la computación distribuida dentro de la computación empresarial.
- Examinar las diferentes técnicas y mecanismos sobre las que se fundamentan los sistemas operativos de los ambientes distribuidos.
- Mencionar las diferencias entre estaciones de trabajo, servidores, clientes y terminales.
- Diferenciar entre los términos procesos distribuidos y procesos cooperativos.
- Mostrar las diferencias entre bases de datos distribuidas y bases de datos centralizadas.
- Destacar la importancia de la tecnología Cliente-Servidor en los sistemas distribuidos actuales.
- Comprender las diferentes maneras bajo las cuales se puede presentar un sistema Cliente-Servidor.
- Demostrar las ventajas de los sistemas distribuidos en cualquier tipo de sistemas multiusuario.

Prologo.

El significativo avance en la tecnologia ha venido a enriquecer el campo de uso de las computadoras de escritorio. La llegada de la tecnologia VLSI (Very Large Scale Integration) y de las redes de computadoras han hecho de las computadoras de escritorio poderosos centros de procesamiento y distribución masivo de datos. Ahora los recursos pueden ser concentrados en la computadora más poderosa, para que las demás computadoras puedan disponer de estos recursos por medio de una red.

La computación distribuida está tendiendo a ser la norma del futuro. Cada vez más y más usuarios están conectando sus computadoras sobre redes formándose con esto pequeñas redes locales (LAN). Las redes locales son interconectadas a fin de formar redes más grandes o de área amplia (WAN), estas a su vez, son expandidas cada vez más y colocadas en lugares tales como universidades, instituciones gubernamentales u otros centros que manejen gran cantidad de información. Las redes WAN, por último, también han sido interconectadas tendiéndose a formar así una inmensa red única universal. La computación distribuida da la posibilidad de construir un ambiente flexible de computación, permitiendo que la información fluya de computadora a computadora a través de toda la red. También da la posibilidad de acceder a recursos que bien pudieran no estar disponibles sobre un sola máquina y permite el intercambio electrónico de datos, cosa que antes se hacia por medios magnéticos, el cual generalmente era una cinta.

El presente escrito representa un estudio de los sistemas distribuidos. Se encuentra organizado en 5 capitulos, en los cuales se hace un estudio que va desde la evolución de los sistemas distribuidos hasta las técnicas empleadas por los modernos sistemas de computación distribuida.

El primer capitulo aborda el tema de la evolución de las comunicaciones hasta llegar a las redes computacionales; se analizan y describen las redes remotas o de área amplia (WAN) y las redes locales (LAN). Asi mismo se hace mención de algunos de los componentes físicos que integran un sistema distribuido a fin de dar una primera aproximación de estos. En el capitulo 2 se empieza por un estudio de las técnicas de programación y mecanismos en los que se basan las aplicaciones distribuidas tales como el paralelismo o el procesamiento en paralelo entre otros, finalizándose el capitulo con un código de programa ejemplo de tales técnicas. El tercer capitulo analiza las diferencias y semejanzas entre las tres formas en que comúnmente se llega a presentar un sistema distribuido procesos cooperativos, bases de datos distribuidas y tecnología cliente-servidor. El capitulo cuarto esta dedicado al estudio del principal mecanismo de comunicación del que se sirven la mayoria de los ambientes distribuidos, los RPC's. El capitulo se termina con una descripción del software que se encuentra entre las aplicaciones finales y el equipo, al que comúnmente se le conoce como middleware. Por último el capitulo quinto analiza el futuro de la computación distribuida, así como los requerimientos para la instalación de un sistema de éste tipo.

Justificación

Con la puesta en marcha del tratado de libre Comercio (TLC) de América del Norte el primero de enero de 1994, llegaron para nuestro país múltiples oportunidades de desarrollo, pero también un gran reto para el empresario mexicano. Mucho se discutió sobre los convenientes e inconvenientes que traería consigo el TLC. Durante las pláticas que se llevaron a cabo durante el último semestre de 1993 sobre la aprobación de dicho tratado, el principal inconveniente que se encontró fue la escasa tecnología y poca modernización de la industria mexicana, ya que de los tres países[1] que conforman este tratado (México, Estados Unidos y Canadá) el nuestro es el único que aún cuenta con industria artesanal.

En el campo de la informática también nuestro país presenta un severo atraso con respecto a sus dos socios comerciales. La industria informática es ya vital para el mundo moderno en que nos encontramos. Mucho tiempo ha pasado desde que surgieron los primeros modelos de computadora, ahora toda industria o empresa que quiera "sobrevivir" deberá contar con una infraestructura informática. Lo más común y recomendable es que esta infraestructura sea una red de computadoras. La tecnología en materia de redes computacionales ha ido cada vez más en aumento, actualmente encontramos sistemas que pueden intercambiar información estando a miles de kilómetros, o inclusive al otro lado del océano, en tan sólo unos cuantos minutos, e incluso segundos, gracias a las bondades que nos ofrece la tecnología de las telecomunicaciones.

Así mismo encontramos notables avances en las redes de área local (LAN) ya que con la introducción de sistemas de computo distribuido en los que podemos encontrar tecnologías tales como las de cliente-servidor las posibilidades para estas sean vuelto casi infinitas.

Ahora un sistema de red LAN bien puede formar parte de un sistema de redes remota, ya que tan sólo bastaría una solicitud de una de las computadoras hacia una computadora mayor conocida como servidor para tener acceso a información comercial, científica, tecnológica o de cualquier otro tipo. Imaginemos tan sólo por unos segundos el beneficio que esto puede traer para cualquier empresa, el tener datos actualizados de los movimientos del mercado tan sólo unos segundos después de que estos se hallan generado puede representar para el negocio o empresa la entrada hacia el mundo del éxito.

La introducción de los sistemas distribuidos, predominando los de tecnología Cliente-Servidor, ha ido en aumento durante los últimos dos años y hay quienes pronostican que para 1997, los servicios y el software para cliente-servidor constituirán el 20% de la industria del software en todo el mundo, cuyo valor asciende a $100,000 millones de dólares.

[1] **A fines de 1994 se anunció el ingreso al TLC de Chile, país que conformará parte de dicho tratado a partir del primero de enero de 1996.**

De lo anterior que se haya escogido el tema de la computación distribuida para este estudio. Las empresas (pequeña y mediana empresa principalmente) de nuestro país no pueden quedar al margen de esta nueva manera de concebir a la computación. Es imperioso la modernización de la empresa y tecnología mexicana, ya que solo así podremos hacer frente a los retos que nos prepara el futuro. Los sistemas distribuidos aparecen como un modelo idóneo que es posible ajustar a muchas empresas mexicanas, sobre todo si lo que maneja la empresa son bases de datos distribuidas.

La empresa nacional, con la llegada de empresas extranjeras, provenientes de los E.U. principalmente, tendrán que competir abiertamente, por ganar o incluso por no perder su mercado.

En este nuevo mercado, de competencia abierta, que apenas inicia, es imperioso que se tenga acceso a los grandes bancos de datos por lo cual la empresa deberá estar capacitada tecnológicamente, ya que solamente con los recursos humanos debidamente capacitados y con la infraestructura tecnológica necesaria se podrá hacer frente a estos retos que prepara el futuro.

Pero a pesar de las ventajas que nos presentan los sistemas distribuidos frente a otros sistemas de redes computacionales, muy pocas personas saben lo que estos son y lo que significan. Un sistema distribuido no requiere de grandes gastos en equipo para su instalación y las ventajas que se pueden obtener con éste tipo de sistemas son múltiples, como se verá a lo largo de este estudio.

También se analizan las perspectivas de estos sistemas y se mencionan algunos fabricantes de software y hardware para sistemas distribuidos. Se da mayor énfasis a los sistemas cliente-servidor ya que estos constituyen la mayor aplicación de un sistema distribuido. El hecho de que se mencionen algunos fabricantes tiene como objeto el auxiliar a aquellas personas que pudieran llegar a interesarse en la instalación de un sistemas distribuidos, para que con este estudio sepan a donde dirigirse y las posibilidades con que cuentan en cuanto a futuros desarrollos.

Así pues, este trabajo trata de dar un panorama de lo que serán los sistemas computacionales en un futuro, ya que la era de la computación distribuida, en muchos países como lo es el nuestro, se encuentra aun en sus inicios.

Es importante que se conozcan estas nuevas tecnologías, ya que como ya se dijo, nuestro país necesitará hacer uso de estas para no quedar en el rezago tecnológico y por ende económico

Este trabajo analiza a los sistemas distribuidos desde un punto de vista objetivo sin llegar a tocar puntos extremadamente técnicos a fin de que sea entendible por cualquier persona.

Los sistemas de computación distribuida se presentan como alternativa para aquellos usuarios que les sea vital el mantener actualizados sus bancos de datos, constituyen una tecnología que a pesar de que ya tiene algunos años, apenas ahora esta tomando fuerza. Cliente-Servidor, que es una de las tres formas que puede adoptar un sistema distribuido, solo requiere una inversión adicional a lo que se necesitaría en la instalación de un sistema de red normal y sus beneficios son mayores. Sin embargo, antes de decidirse por adoptar un sistema distribuido como el de cliente-servidor se deberá estar seguro si realmente es esto lo que se requiere.

Por regla general se deberá considerar que si en una empresa o lugar de trabajo se manejan grandes cúmulos de información y si esta información debe ser constantemente actualizada, o si se requieren múltiples paquetes computacionales así como compartir aplicaciones con otras empresas o instituciones, entonces sin duda alguna un sistema de cómputo distribuido es justamente lo que se necesita.

Capitulo 1

El desarrollo de la Computación en las Comunicaciones.

1.1 Los Inicios

1.1.1 Evolución de las comunicaciones

Desde los albores de la humanidad el hombre ha buscado siempre la mejor manera de poderse comunicar con sus semejantes, esto lo ha logrado a través de los siglos utilizando técnicas cada vez más sofisticadas. En un principio la comunicación se realizaba con mímica o dibujos rústicos, sin embargo la naturaleza creativa del hombre le permitió establecer códigos que le permitieran formar lenguas o idiomas para poder transmitir sus pensamientos. Con el tiempo, la necesidad de transmitir información a grandes distancias, lo llevó a crear diversos sistemas de comunicación como el establecido por los aztecas, quienes utilizaban los *tamemes*[1] como una forma de llevar información de un lugar a otro. La creatividad que caracteriza al hombre le permitió crear técnicas que facilitaron la transmisión de datos: señales de humo, sonidos de tambores, cuernos o caracoles, campanas o disparos de pistolas, el uso del destello del sol en los metales o espejos y el resplandor de

[1] Palabra de origen Nahuatl que significa mensajero

luces en la noche. Durante el siglo XIX se lograron grandes avances tecnológicos que facilitaron la comunicación a larga distancia.

El telégrafo eléctrico utilizado por los británicos fue el primer dispositivo que permitió la comunicación a larga distancia; esto se logró gracias a los progresos de la electricidad, al invento de la pila y del electroimán que aparecieron a principios del siglo XIX.

El sistema que acaba imponiéndose al anterior es el inventado por el norteamericano Samuel Morse, consistente de una aplicación muy sencilla de los electroimanes. En 1876 el norteamericano de origen británico, Alexander Graham Bell inventa el teléfono, acontecimiento que suscita una oleada de incredulidad a pesar de ser un gran adelanto en materia de electricidad, las vibraciones sonoras según esta nueva tecnología, se transmiten en forma de señales eléctricas por medio de una membrana metálica unida al núcleo de un electroimán, que está conectado a una batería y a un receptor. Otro de los grandes inventos realizados fue la telecomunicación, la cual no se hubiera realizado sin el descubrimiento de las ondas Hertzianas (1888) por el alemán Heinvich Hertz. El italiano Guglielmo Marconi, principal precursor de la telegrafía sin hilo o radiotelegrafía, establece en 1901 la primera comunicación trasatlántica entre Cornualles (Gran Bretaña) y la isla de Terranova (Canadá), iniciándose con este acontecimiento la era de las telecomunicaciones. En la figura No. 1.1 podemos observar una cronología de los avances más importantes en materia de comunicaciones.

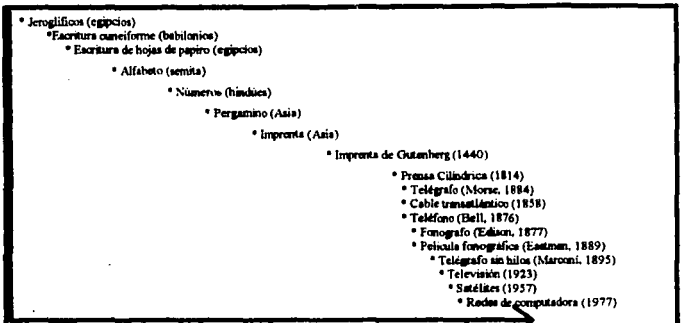


Fig. 1.1 Evolución de las técnicas de comunicación de la raza humana

1.1.2 Evolución de las computadoras

Por otro lado las computadoras, al igual que las formas y medios de comunicación, también han experimentado una constante evolución. En el proceso de la evolución histórica de la moderna computadora digital, se han diseñado y construido cientos de diferentes tipos de computadoras. La mayoría de ellas hace tiempo que pasó al olvido, pero algunas ejercieron un impacto significativo en las ideas modernas. Así mismo la forma de operar de estas máquinas o la tecnología en la que se encuentran basadas ha mostrado un sorprendente avance: desde los dispositivos que funcionaban a base de ruedas dentadas, comúnmente llamados engranes, hasta las modernas computadoras que se basan en la microtecnología.

La primera persona en construir una máquina calculadora que funcionara, fue el científico francés Blaise Pascal (1623-1662), en cuyo honor se llamó al lenguaje de programación Pascal. Este dispositivo, enteramente mecánico, sólo

podía hacer sumas y restas, pero 30 años después el matemático alemán, el barón Gottfried Wilhelm von Leibniz (1646-1716), perfeccionó la máquina de Pascal haciendo que también pudiera realizar multiplicaciones y divisiones. Los siguientes 150 años no fueron de gran avance, hasta que un profesor de matemáticas de la Universidad de Cambridge, Charles Babbage, diseño y construyó su máquina diferencial, la cual después perfeccionó y construyó su máquina analítica. Las principales características del diseño de Babbage fueron que en su diseño incluía secciones de entrada y salida de datos, una sección de almacenamiento de datos y una parte de cálculo. El gran avance de la máquina analítica, fue el de ser una máquina de propósito general; leía las instrucciones de las tarjetas perforadas y las ejecutaba. Puesto que la máquina analítica era programable en un lenguaje ensamblador sencillo, necesitaba de software; para producirlo, Babbage contrató a una joven llamada Ada Augusta Lovelace (hija del poeta británico Lord Byron), quien se convirtió en la primera programadora de computadoras y en cuyo honor recibió el nombre el moderno lenguaje de computadoras Ada.

Después de Babbage existieron muchos modelos y diseños de máquinas que pretendían realizar grandes cantidades de operaciones en poco tiempo, todos estos diseños tenían la característica de que tenían un funcionamiento principalmente mecánico aunque poco a poco se fue imponiendo la idea de utilizar relevadores en los diseños.

No fue sino hasta 1944 en que Howard Aiken construyó en la Universidad de Harvard la Mark I, la cual tuvo la característica de ser la primera máquina que funcionaba a base de principios electrónicos.

Debido al auge que despertó la segunda guerra mundial en cuanto al tener calculadoras mecánicas que permitieran tener una supremacía en el frente de batalla, varios científicos emprendieron nuevos trabajos algunos de ellos basándose en trabajos anteriores y así en 1946 surgió la ENIAC (Computadora e Integrador Numérico Electrónico) con la que inicia la primera generación de computadoras. Poco después Jonh von Neumann, uno de los involucrados en el proyecto ENIAC, construyó una versión de la máquina EDVAC (Computadora Electrónica Automática de Variable Discreta). De las máquinas construidas y diseñadas por Neumann destaca el hecho de que podían almacenar los programas en una memoria propia. Sus diseños fueron tan adelantados que aún en la actualidad las computadoras digitales se siguen basando en el modelo de von Neumann.

Con la invención del transistor en 1948 en los Laboratorio Bell, surge la segunda generación de computadoras. La primer computadora transistorizada, fue una máquina de 16 bits llamada TX-0 y sólo pretendía servir como dispositivo de prueba para una más completa TX-2, la cual también sirvió de modelo para la IBM 7090 la cual fue considerada como la computadora más rápida del mundo en su época.

La invención de los circuitos integrados en la década de los sesenta , permitió poner docenas de transistores en una sola pastilla. Este empaque hizo posible construir computadoras más pequeñas, rápidas y baratas que sus predecesoras transistorizadas. Así en 1964 IBM introduce el sistema /360, basado en circuitos integrados, diseñado para la computación científica y comercial. Entre las novedades de este tipo de máquinas se encuentran su compatibilidad para

correr programas de otros sistemas y la multiprogramación, que consiste en tener diversos programas en memoria, de modo que mientras uno este esperando para completar un proceso de entrada o salida, otro pueda ejecutarse. También la /360 fue la primera máquina en emular a otras computadoras.

En el transcurso de los años ochenta, la integración a muy grande escala (VLSI) hizo posible poner, primero decenas de miles, después cientos de miles y finalmente millones de transistores en una sola pastilla. Este desarrollo condujo a la fabricación de computadoras más pequeñas y rápidas. Para 1980, los precios habían disminuido tanto que era factible para un solo individuo, tener su propia computadora. Había comenzado la era de la computadora personal. La tabla 1.1 muestra una clasificación de las computadoras de acuerdo a sus generaciones y mostrando sus principales características.

Características Principales	Primera Generación	Segunda Generación	Tercera Generación	Cuarta Generación	Quinta Generación
ENTRA AL MERCADO	Aprox. (1950-1952)	Aprox. 1960	Entre 1968 y 1970	Entre 1977 y 1981	Hacia 1990
TECNOLOGÍA	Tubos de Vacío	Transistores, ferritas	Circuitos integrados (LSI) y memorias de películas magnéticas	Microelectrónica (VLSI), memorias MOS	Enjabres procesadores microscópicos (SVLSI), RISC
PERIFÉRICOS	Lectoras y perforadoras de tarjetas	Lectoras de tarjetas y primeras impresoras	Cintas y discos magnéticos, terminales de video	Terminales inteligentes, discos y cintas, equipos de graficación	Terminales con capacidad de inferencia, discos ópticos, digitalizadores, impresoras inteligentes
LENGUAJES	Lenguaje de máquina	Ensambladores y primeros compiladores (FORTRAN, ALGOL)	Lenguajes de alto nivel, COBOL, PL/1, base de datos (DMS)	Bases de datos distribuidas, lenguajes interactivos y procedurables	Lenguajes de cuarta generación (SQL) y orientados a objetos.
ALFABETO	Numérico	Números, letras y caracteres	Números letras y caracteres	Irrestricto, simbología variada.	Universal, simbología irrestricta
SISTEMA OPERATIVO	----------	Muy rudimentario	Manejo de discos multiproceso, memoria dinámica y virtual	Proceso sin interrupción, comunicación entre máquinas	Comunicación entra máquinas remotas, memoria dinámica, virtual, sistemas inteligentes
ASPECTO CUANTITATIVO	Memoria central 1000 a 8000 palabras; 10^4 ops/seg	MC: 8000 a 32000 palabras; 10^5 ops/seg	MC: 64 a 256 k palabras; 10^6 ops/seg	MC: 64k a 10^7 caracteres; 10^7 ops/seg	MC:64k a 10^9 caracteres; 10^9 ops/seg
MODELOS MACROS	IBM-650, UNIVAC	IBM 7090 BENDIX G-20	IBM-360 UNIVAC 1106	IBM 4330 UNIVAC 1100	----------
MODELOS MINIS	----------	----------	----------	PRIME 550 MP-3100	APPLE 2000
MODELOS MICRO	----------	----------	----------	APPLE, IBM PC	IBM PC y compatibles, MACINTOSH, POWER PC

Tabla 1.1 Generaciones de Computadoras

1.1.3 Máquinas Multinivel.

La mayoría de las computadoras modernas constan de dos o más niveles de comunicación de datos. En una máquina actual podemos diferenciar seis niveles perfectamente bien definidos como lo muestra la figura 1.2.

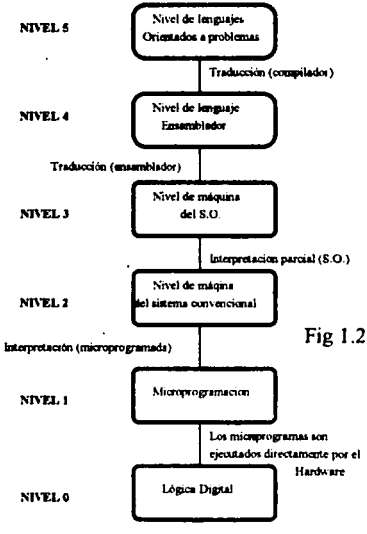


Fig 1.2 Niveles de máquinas Multinivel

El nivel 0, el más inferior, es realmente el hardware de la máquina. Sus circuitos ejecutan los programas escritos en el lenguaje de la máquina del nivel 1. En el nivel uno, existe un programa llamado microprograma, cuya función es interpretar las instrucciones del nivel 2; debido a esto es que a este nivel se

le llama nivel de microprogramación. Cada máquina de nivel 1 tiene uno o más microprogramas que pueden ejecutarse en ella.

Cada microprograma define implícitamente un lenguaje de nivel 2. El tercer nivel normalmente es un nivel híbrido. La mayoría de las instrucciones de su lenguaje están también en el lenguaje de nivel 2. Además existe un nuevo conjunto de instrucciones, una diferente organización de la memoria y la posibilidad de ejecutar dos o más programas en paralelo, entre otras cosas. Las nuevas posibilidades que se añaden al nivel 3 las lleva a cabo un intérprete que actúa en el nivel 2 al que tradicionalmente se le llama Sistema Operativo. Las instrucciones del nivel 3, idénticas a las del nivel 2, las lleva a cabo directamente el microprograma en lugar de efectuarlas el sistema operativo. En otras palabras, algunas de las instrucciones del nivel 3 las interpreta el sistema operativo y otras las interpreta directamente el microprograma. Esto es lo que se quiere decir con "híbrido" ; a este nivel también se le llama nivel de máquina del S.O.

Los niveles 4 y superiores difieren fundamentalmente con los anteriores ya que los primeros están pensados para los programadores de aplicaciones que tienen un problema que resolver. Los niveles 2 y 3 siempre se interpretan. Los niveles 4, 5 y superiores normalmente, aunque no siempre, se soportan por traducción. En los primeros niveles se utilizan lenguajes máquina numéricos, los programas escritos en ellos constan de largas series de números, buenos para las máquinas pero inadecuados para las personas.

A partir del nivel 4, los lenguajes contienen palabras y abreviaturas significativas para la gente. El nivel 4 o nivel de lenguaje ensamblador es mas bien una forma simbólica de uno de los lenguajes subyacentes. Los programas escritos en lenguaje ensamblador se traducen primero a un lenguaje de nivel 1,2 o 3 y luego se interpretan por la máquina real o virtual apropiada. El programa que realiza la traducción se llama ensamblador. Al nivel 5 pertenecen los lenguajes de alto nivel tales como BASIC. PASCAL, C, COBOL, etc.

El nivel 6 y siguientes consisten en colecciones de programas diseñados para correr bajo máquinas específicamente adaptadas a ciertas aplicaciones y que contienen gran cantidad de información acerca de ellas.

En resumen, lo esencial a recordar es que las computadoras están diseñadas como una serie de niveles, cada uno construido sobre su predecesor. Cada nivel representa una abstracción distinta, con objetivos y operaciones diferentes. Esto es importante que se tenga presente, ya que como se vera más adelante, los mecanismos de comunicación entre computadoras operan en los niveles inferiores manteniendo siempre una estrecha comunicación con la arquitectura de los niveles superiores. *Se denomina arquitectura al conjunto de tipos de datos, operaciones y características de cada nivel.* Se le llama arquitectura de computadoras, a su vez, al estudio del diseño de todas aquellas partes que son visibles a los programadores, tal como la cantidad de memoria con que se dispone y que es importante considerar al momento de programar. Esta denominación, en la práctica profesional significa lo mismo que *organización de computadoras.*

Las primera computadora digital, allá por los años cuarenta, tenía solamente dos niveles: el convencional, en el que se programaba, y el de lógica digital, que ejecutaba esos programas. En 1951 M. V. Wilke, en Inglaterra, concibió la idea de diseñar una computadora de tres niveles para simplificar drásticamente la electrónica. Esta máquina tendría incorporado un intérprete intercambiable, cuya función sería ejecutar los programas en lenguaje de máquina convencional en forma interpretativa. Se necesitarían menos circuitos electrónicos, debido a que el hardware tendría que ejecutar solamente microprogramas con un repertorio de instrucciones limitado, en lugar de programas en el lenguaje de máquina convencional que requieren un repertorio mucho mayor. Puesto que los circuitos electrónicos se hacían entonces con tubos de vacío, esta simplificación prometía reducir el número de bulbos y aumentar la confiabilidad.

Hasta antes de 1960 la mayoría de las computadoras funcionaban de modo abierto, es decir, el programador por si mismo tenía que operarlas. Con el surgimiento del sistema operativo esta tarea se facilitó, ahora el programador proporcionaba junto con el programa ciertas tarjetas de control que eran leídas y ejecutadas por el sistema operativo. En años sucesivos, los sistemas operativos se volvieron más complejos. Se añadieron niveles de instrucciones y potencialidades al nivel de máquina convencional hasta que empezó a tomar la forma de un nuevo nivel. Algunas de las instrucciones del nuevo nivel eran idénticas a las instrucciones del nivel de máquina convencional; pero otras , particularmente las instrucciones de entrada/salida, eran completamente diferentes (conocidas como macros del sistema operativo).

Los sistemas operativos se desarrollaron también en otros sentidos. Los primeros leían tarjetas e imprimían los resultados en la impresora. A esto se le llama *sistema por lotes* (batch). Normalmente había una espera de varias horas desde que se mandaba el programa hasta que los resultados estaban listos. A principios de la década de los sesenta, los investigadores del Dartmouth College en el MIT (Instituto Tecnológico de Massachusetts) y de otros lugares inventaron sistemas operativos que permitían a varios programadores a la vez comunicarse directamente con la computadora. En estos sistemas se le conectaron a la computadora central terminales remotas mediante líneas telefónicas. Un programador podía introducir por teclado su programa y obtener casi inmediatamente los resultados impresos. Esos sistemas se llamaron y todavía se llaman *sistemas de tiempo compartido*. Nuestro interés en los sistemas operativos se centra en la parte que permite realizar comunicación entre varias máquinas y los procesos o mecanismos comúnmente llamados RPC que son necesarios para tales comunicaciones.

1.1.4 La computación en las comunicaciones.

La primera liga de dispositivos de cómputo y de comunicaciones ocurrió en 1940 cuando el doctor George Stibitz utilizó líneas telegráficas para enviar datos desde Dartmouth College en New Hamphire hasta una calculadora de los laboratorios Bell en la ciudad de Nueva York. Pero fue hasta finales de la década de 1950 cuando la unión entre la computación y las comunicaciones comenzó en forma activa.

Al principio se emplearon líneas telegráficas para conectar las terminales de teleimpresoras con las computadoras, pero rápidamente se introdujeron las líneas telefónicas. Una pronta ampliación a gran escala fue el sistema de reservaciones de pasajeros, planeado y puesto en funcionamiento a principios de 1960 por las compañías American Airlines e IBM en que cientos de terminales dispersas fueron conectadas a un procesador central. El uso de las comunicaciones ha crecido constantemente desde entonces. Hoy, la mayoría de las microcomputadoras, minicomputadoras y macrocomputadoras son capaces de comunicarse con terminales distantes. La característica básica de las comunicaciones de computadoras es que el usuario utiliza una terminal sin percatarse de la conexión de diferentes sistemas de cómputo con servicios variados y diversas capacidades, conectados a la computadora central.

Dentro de estos diferentes sistemas de cómputo disponibles, el usuario deberá elegir explícitamente el sistema de cómputo en el cual quiere trabajar, estableciendo la conexión a través de una red con el sistema elegido y una vez hecho el enlace podrá comunicarse al sistema anfitrión como si fuera un usuario local.

Tratando de hacer una clasificación de los distintos tipos de soluciones que pueden encontrarse en la comunicación de computadoras, podemos formar la figura 1.3 en la que se observa una representación sobre una escala de distancias entre los elementos de tratamiento de información, el área geográfica a la que aproximadamente corresponden y algunas de las denominaciones utilizadas para identificar las soluciones desarrolladas. El presente estudio no pretende analizar a detalle cada una de estas áreas ya que sería necesario desarrollar un trabajo completo para cada una de ellas. Sin embargo, el tema que nos ocupa amerita que se introduzca principalmente a dos de estas áreas: las redes remotas y las redes locales (LAN) por lo que más adelante se discutirán brevemente estos dos puntos.

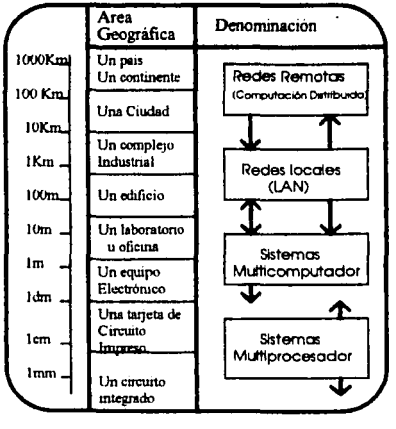


Fig. 1.3 Niveles de comunicación entre computadoras

Las computadoras surgen como una respuesta a los múltiples avances tecnológicos que se gestaron a mediados del presente siglo. Primeramente fueron utilizadas como grandes instrumentos de cálculo(durante la segunda guerra mundial) y posteriormente como grandes centros de procesamiento de datos.

Desde una perspectiva más amplia el desarrollo de la computación se ha debido a tres aspectos diferentes íntimamente relacionados:

a) Aspectos tecnológicos (hardware).

b) Aspectos de programación (software).

c) Aspectos del uso y aplicación de la computadora.

Al respecto nos dice Ayala San Martín *"Estos tres aspectos han interactuado entre sí, provocando el acelerado desarrollo de la computación de tal manera que, avances tecnológicos han permitido ampliar el uso y aplicaciones de la computadora; avances en la programación han permitido otros usos; necesidades específicas en el uso han provocado la investigación y desarrollo de la tecnología."*[2]

En cuanto al uso y aplicaciones de las computadoras, podemos distinguir cinco etapas :

1. **Instrumentos de cálculo.**
2. **Proceso electrónico de datos.**
3. **Sistemas de información.**
4. **Sistemas de comunicación.**
5. **Sistemas inteligentes de comunicación**

[2] Gerardo Ayala San Martín. *Introducción a la Computación*, UNAM 1991, Segunda edición.

Las cuales coinciden aproximadamente con la primera, segunda, tercera y cuarta generaciones de computadoras. La última etapa se ha prolongado hasta nuestros días y ha dado origen a una quinta etapa: la de sistemas inteligentes de comunicación que coincide con la quinta generación de computadoras.

La computadora como instrumento de cálculo sirvió en los institutos de investigación, en organismos militares, en sectores gubernamentales de estadística y en algunas grandes empresas.

Posteriormente la necesidad de manejar grandes volúmenes de información en procesos repetitivos provocó su automatización, lo que se denomina proceso electrónico de datos. En esta etapa las computadoras ampliaron su mercado, entrando de lleno a los sectores financieros, comerciales y de gobierno.

Debido a las innovaciones tecnológicas como, la introducción del disco magnético, la multiprogramación, terminales interactivas, etcétera, fue posible hacer uso de un acervo controlado de datos organizados de acuerdo con un esquema preconcebido que permitiera la consulta simultánea de la información por varios usuarios. Esto permitió el estudio de la organización misma, su planeación, la toma de decisiones de alto nivel, y no solamente la ejecución de tareas rutinarias. A esta etapa se le conoce como de los *sistemas de información*.

La cuarta etapa es la de los sistemas de comunicación la cual permite alimentar y actualizar la información de un acervo de datos disponible para todos los usuarios del mismo. En esta etapa los mensajes son enviados y recibidos sólo por los usuarios que cumplan con cierta condición (es selectiva); la distribución y recepción es instantánea y la información queda disponible

todo el tiempo. Ejemplo de ello se tiene en las reservaciones aéreas y los sistemas bancarios.

Al evolucionar la cuarta etapa y modernizarse la tecnología de las telecomunicaciones surge la ***telemática*** y con ella la quinta etapa, que es la de los sistemas inteligentes de comunicación. Esta etapa esta caracterizada por el uso de redes de computadoras tanto de área local (Redes LAN) como de área remota (Redes WAN). El estudio de las redes es muy importante para entender el tema que nos ocupa, es por ello que más adelante trataremos este tema más ampliamente. Es de destacar que a finales de la década de los 60´s y con la llegada de la cuarta generación de computadoras es que aparece lo que mas tarde seria conocido como redes de computadoras.

En la actualidad el simple hecho de procesar información no es suficiente. Además, es necesario disponer de la opción de compartir los datos con otras computadoras. Si una empresa con delegaciones por todo el mundo necesitara procesar los mismos datos en cada uno de sus centros, sería insólito plantearse la cuestión de introducir toda la información en cada uno de ellos. Para no caer en tan desaconsejable forma de trabajo, existen diversos dispositivos capaces de transmitir la información por una línea de comunicación (ejemplos de estos son los módem). A la ciencia que se encarga de estudiar la forma en que operan estos dispositivos y la forma en que se lleva a cabo la transmisión a través de dos o más computadoras a larga distancia de le conoce como **Telemática**.

En el mundo de la informática, cuando se habla con carácter general de transmisión de datos desde una computadora a otra se emplea el término comunicaciones y no comunicación, ya que son numerosas las formas de llevar a cabo dicha transferencia de datos. Por ejemplo en cuanto al método de transmisión de la información, encontramos los siguientes tipos:

- Transmisión simplex (la información viaja en un sólo sentido), semidúplex (la información viaja en dos sentidos pero no al mismo tiempo), dúplex (la información viaja en dos sentidos y simultáneamente).
- Transmisión serie (toda la información circula por un mismo canal) y paralelo (cada uno de los bits que forman el carácter circula por un canal independiente).
- Transmisión síncrona (en sincronía con el reloj) y asíncrona (no existe síncronía con el reloj).

De aquí que el estudio de la Telemática sea importante para manejar todos estos conceptos, por lo que una persona que se dedique al estudio de la comunicación remota entre computadoras deberá tener un buen conocimiento de ciencia telemática.

1.2. Las redes Computacionales.

1.2.1 Redes Remotas.

Las redes de computadoras surgen históricamente a finales de los años sesenta como una solución para la interconexión de computadoras situados en lugares dispersos con el objeto fundamental de compartir recursos, es decir, permitir a algún usuario de cualquier computador acceder y utilizar los recursos ya sean de hardware o software del conjunto de máquinas que constituyen la red. Las redes pueden ser tanto locales como remotas, estas últimas son las que se emplean en aquellas compañías que cuentan con una serie de sucursales y que requieren tener siempre actualizada su información de lo que sucede en cada una de sus filiales. Antes de estudiar más a detalle lo que son las redes remotas pasemos a definir los dos tipos de máquinas que pueden constituir a estas, nos referimos a las comúnmente llamadas terminales y a las computadoras tipo PC .

Cuando la cantidad de información que se maneja es muy importante, generalmente bases de datos, es necesario emplear las grandes computadoras, denominadas *mainframes* en inglés. Estas potentes computadoras, gracias a sus sistemas operativos multipuesto, disponen de un gran número de terminales que se emplean para introducir, visualizar o imprimir datos. Un ejemplo de este tipo de computadoras y sus terminales son empleadas por las grandes entidades bancarias, que nos resultan familiares cuando observamos sus dispositivos en oficinas y sucursales. Aunque con un aspecto totalmente distinto, otro ejemplo de este tipo de terminales son los populares cajeros

automáticos, compuestos de una sola terminal, un lector de banda magnética y una pequeña impresora. De lo anterior podemos deducir entonces que una terminal es una máquina que se encuentra conectada a un procesador central y que no tiene autonomía propia ya que depende totalmente del mainframe para funcionar.

Pero no debemos pensar que el uso de mainframes está limitado al colectivo de la banca, son numerosas las empresas que disponen de este tipo de computadora y por ende de los servicios de una red remota formada por terminales y mainframes. Entre los usuarios de estas máquinas es frecuente que, además de disponer de un terminal sobre su mesa, este instalada una PC dispuesta para tareas de otro ámbito, como pueden ser la creación de gráficos, bases de datos locales (o remotas como en el caso de los sistemas de computación distribuida y en especial en cliente-servidor), aplicaciones a medida, etc. Para estos usuarios el espacio puede ser un verdadero problema, que se resuelve instalando una tarjeta en la computadora PC de forma que nos permita disponer de las dos opciones. De esta manera podemos conmutar entre los servicios de una terminal o de una PC, a este tipo de computadoras se les conoce como *estación de trabajo*. La PC a su vez puede tener acceso a servicios de bases de datos distribuidos pasando de esta manera a ser lo que en tecnología cliente-servidor se conoce como un cliente y con lo cual se formaría parte de un sistema de computación distribuida.

A uno de los tipos más populares de tarjetas de emulación que se instala a las PC's para que puedan conmutar entre terminales y PC's se le conoce como

IRMA[3]. El número de tipos de redes de comunicaciones remotas y sus correspondientes recursos es enorme. Por este motivo, para el presente trabajo, nos basaremos en uno de los modelos más aceptados que es el modelo de IBM que comúnmente se conoce como redes SNA. Cabe mencionar que el modelo SNA se aplica únicamente para sistemas de redes de computadoras que funcionan bajo terminales y *mainframes* en sistemas de redes remotas, sin embargo, muchos de sus principios se han generalizado para otros tipos de sistemas, como es el caso de las redes LAN.

1.2.1.1 El modelo SNA.

En los inicios de la carrera entre fabricantes de sistemas informáticos que estamos viviendo, los sistemas crecían según los criterios de sus creadores. Esta "libertad de movimiento" ocasionó un sistema de comunicaciones caótico, en el cual cada aplicación controlaba los terminales que tenía asociados, así como toda gestión de las comunicaciones. Por este motivo IBM decidió estructurar su arquitectura y definir un sistema de comunicaciones, de forma que todos aquellos fabricantes que realizan productos para la conexión de sus sistemas estuvieran sometidos a esta normativa. El nombre que recibió fue SNA (System Netware Architecture). SNA no pretende ser el modelo universal de las comunicaciones, simplemente es una normativa pensada por IBM y para IBM y sus compatibles.

[3] Las tarjetas de emulación IRMAson llamadas así en honor a la novia de su inventor, un ingeniero estadunidense que fue el primero en tener la idea de hacer trabajar una PC como una terminal.

1.2.1.2 Arquitectura de las redes SNA.

SNA está estructurada en siete niveles que definen los servicios y protocolos para la conexión y operatividad de la red. Los tres primeros definen las funciones de la red. El primero se refiere al control físico encargado de las conexiones con los nodos contiguos. Es decir, son los aspectos eléctricos, mecánicos y funcionales de las conexiones de la red, tales como los niveles de las señales correspondientes a los valores "0" y "1", así como su duración; el modo de comunicación, etc. El segundo es el control del enlace de datos (data link control), que facilita el intercambio de datos entre dos nodos adyacentes. Se basa en la formación de grupos de información, las composiciones de ésta y un mecanismo de aceptación o rechazo de las mismas. El tercer nivel define el control de caminos (path control), y se encarga de encaminar los datos entre la fuente y el destino, así como del control del tráfico de datos en la red. Prevé la posibilidad de congestión de datos en un nodo de la red debido a una sobrecarga, así como la forma de resolverlos. Los siguientes cuatro niveles implementan las funciones NAU (Network Addressable Unit), las cuales permiten a los usuarios finales recibir y enviar mensajes a través de la red. El cuarto nivel define el control de la transmisión, es decir, ajusta el ritmo de intercambio de datos y cuando es preciso realiza las funciones de cifrado. El control y sincronismo del flujo de los datos, así como el intercambio de los mismos está definido en el quinto nivel. El sexto nivel define los servicios de presentación: ajusta los datos para los diferentes medios de presentación de la información, además coordina los recursos compartidos. Por último, el séptimo nivel define los servicios transaccionales, es decir, proporciona los servicios de

aplicación, como son el acceso a las bases de datos o el intercambio de documentos.

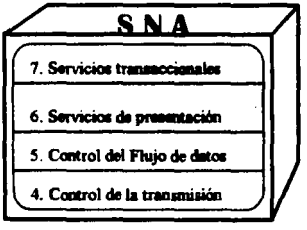


Fig. 1.4 Representaciones de los siete niveles definidos por IBM para su arquitectura de comunicaciones SNA.

1.2.1.3 Conceptos de las redes remotas.

Para lograr una mejor comprensión del tema que nos ocupa, es menester que definamos algunos términos, esto nos ayudará a entender de mejor manera el hardware sobre el que corren los sistemas distribuidos y los componentes de software que los forman.

Los usuarios finales. Son tanto las estaciones de trabajo como las referencias que identifican a las personas que las utilizan. Además de señalar a nivel físico las terminales que están conectadas a la red, se requiere que cada posible usuario de la red se identifique, de forma que sea factible conocer su permiso de acceso y estadísticas de uso.

Nodos. Conjunto de elementos hardware y software que realizan algunas de las funciones de la red y están conectados por enlaces. En general, nodo son todos aquellos elementos conectados a la red y que tienen la capacidad de manejar el protocolo de comunicaciones que se utiliza en el tipo específico de red al cual se encuentra conectada.

NAUs. (Network Addresable Unit). Se trata de elementos con direcciones de red que definen su localización dentro de la misma y el camino para poder llegar a ellas desde cualquier punto de la red. Pueden ser de tres tipos:

1. Unidad Lógica (LU, Logical Unit). Proporciona el acceso de los usuarios finales a los recursos de la red. Además gestiona la

transmisión de información entre ellos. Todo usuario de una red tiene que tener al menos una LU asignada.

2. Unidad Física (PU, Physical Unit). Pertenece a cada uno de los nodos y se emplea para manejar y monitorizar los recursos de ese nodo. Una unidad física puede tener y controlar a más de una unidad lógica.
3. Punto de Control de los Servicios del Sistema. Se les denominan SSCP (System Services Control Point). Son el punto central de control de un "dominio". Reside en el host y proporciona servicios para monitorizar y controlar los recursos de la red.

Dominios. Como lo ilustra la figura 1.5 están formados por un SSCP y todos los recursos de red que él controla.

Sesiones. Una sesión es una conexión lógica que permite a dos NAUs comunicarse entre sí. Pueden ser de los siguientes tipos:

1. SSCP-SSCP. Se utilizan para activar sesiones entre unidades lógicas situadas en dominios diferentes.
2. SSCP-PU. Se utilizan para controlar un nodo y sus recursos.
3. SSCP-LU. Interceden en la activación de sesiones LU-LU.
4. LU-LU. Se utiliza para que un usuario final pueda comunicarse con otro.
5. PU-PU. Sólo las funciones de control de la red. Por ejemplo, FEP-Host.

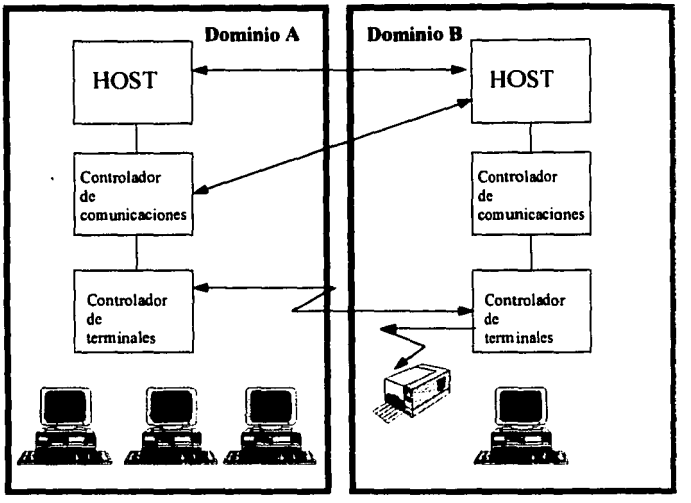


Fig. 1.5 Los dominios consisten en un SSPC y todos los recursos de la red que el controla.

1.2.1.4 Componentes de una red remota

Host. Es la computadora central que controla los procesos distribuidos. Si lo comparásemos con una PC sería la placa base. Su función es controlar la red. Los trabajos que desempeña son la ejecución de programas, el acceso a bases de datos, servicios de directorios y la propia gestión de la red.

Procesadores distribuidos. Sus funciones son similares a las de los hots, excepto que no suelen incluir funciones de gestión de la red. Para que los usuarios de la red puedan acceder a los hots deben pasar primero por estos procesadores.

Controladores de comunicaciones. Gestionan la red física , son los encargados de controlar los enlaces y encaminar los datos a través de la red. Si lo comparamos con el mundo de las PC´s es como si la computadora tuviera instalados un gran número de módem y una red local, además de un dispositivo externo que se encarga de controlar y gestionar todos estos recursos para que los distintos usuarios pudieran acceder a sus datos.

Controladores de cluster. Controlan las operaciones de entrada/salida de los dispositivos que tienen conectados. Como es de suponer, no siempre los terminales están ubicados en el mismo recinto físico que el host. Además pueden estar instalados otros dispositivos como son las impresoras, es decir, el número de periféricos y condiciones de trabajo es muy grande. Por este motivo es necesario agruparlos de forma tal que no operen todos al mismo tiempo.

Estaciones de trabajo. Proporcionan el acceso de los usuarios a la red para poder recibir y enviar información. El ejemplo más representativo son los terminales utilizados en entidades como la banca o aseguradoras y agencias de viaje.

Métodos de acceso. Se trata de programas que residen en el host para controlar que el acceso de los usuarios a la red sea sólo de los usuarios autorizados.

Subsistemas de aplicación. Residen en los procesadores. Sus principales actividades son el desarrollo interactivo de programas de aplicación, la

obtención y actualización de información, el procesamiento de trabajos por lotes remotos y la presentación gráfica en pantalla e impresoras. Uno de los ejemplos más significativos es el CICS (Customer Information Control System, sistema de control de la información personalizada).

Programas de aplicación. Podríamos decir que son los programas que utilizan los usuarios. Sus funciones son realizar cálculos, servicios transaccionales, edición de textos, etc. Se trata de programas como WINDOWS, WORD, LOTUS, etc., en el segmento de las PC.

Programas de gestión de red. Cooperan con las tareas de la red. Detectan e informan sobre los posibles problemas de la red y mantienen estadísticas del rendimiento de la red.

Programas de control de la red. Conducen y controlan los datos entre los controladores.

Enlaces. Un enlace es el camino entre dos nodos adyacentes. Cuando dos nodos están interconectados por más de un enlace físico se les define como multienlace. Los enlaces están formados por un medio transmisor de los datos y un programa de control del enlace de datos. Podríamos decir que se trata del cable por el cual se transmiten los datos y el programa de control de los mismos.

1.2.2 Redes de Área Local (LAN)

Originariamente las computadoras PC se diseñaron para atender al condicionante que su propio nombre indica, computadora personal. Pero actualmente esta propiedad se ha quedado pequeña frente a las avanzadas necesidades de los usuarios. Para aprovechar al máximo los recursos, tanto de hardware como de software, de las distintas computadoras ubicadas en el mismo entorno físico, es necesario interconectarlas. La forma de hacerlo es mediante las redes de área local. La finalidad de las redes locales es compartir recursos tanto físicos como lógicos. Como su propio nombre indica son locales, es decir, están pensadas para pequeñas superficies situadas en los edificios. Por ejemplo, supongamos que en una empresa dotada con una decena de computadoras se requiere que la calidad de impresión de sus documentos sea de muy alta resolución, y por tanto es necesario emplear una impresora de elevado precio. Si queremos que cada una de las computadoras pueda imprimir con esta calidad tenemos dos soluciones, o bien instalar una impresora de estas características en cada una de las computadoras o bien instalar una red de área local que permita compartir esta impresora con cada una de las computadoras conectadas a la red. Pero las posibilidades que nos ofrecen las redes locales no son solamente de carácter físico, también pueden ser rentables para compartir software. Por ejemplo, si en la empresa del modelo anterior todos los usuarios emplean el mismo programa, es necesario adquirir tantos programas o licencias de uso como computadoras existan. Además en algunas de estas computadoras se manejarán los mismos datos, con su correspondiente pérdida de tiempo.

Analizando detenidamente este caso, posiblemente sea más rentable instalar una red de área local, que permita a cada una de las computadoras conectadas a ella acceder a la información albergada en una de las computadoras, llamada servidor, que generalmente será de mayores funciones. Otra de las ventajas que nos ofrecen las redes locales es el correo electrónico. Esta característica permite enviar mensajes desde una de las computadoras conectadas a la red, a alguna o a todas las restantes. Por ejemplo en las empresas con un gran número de departamentos, la utilización del correo electrónico puede sustituir a los "comunicados internos", con el consiguiente ahorro de papel, tiempo y posibilidad de evitar extravío del documento. Además, si es necesario, se puede imprimir y obtener a ésta información en papel.

1.2.2.1 Topologías.

Se llama topología de red a la forma geométrica de la red, independientemente de los protocolos de comunicaciones que circulen en ella y de la topología lógica de la misma. Es diferente hablar de topología física o lógica ya que por ejemplo una red con protocolo paso de testigo (Token Ring) suele tener topología lógica de anillo, mientras que su topología física es de estrella. Encontramos diversos tipos de topologías, pero los tres tipos más usados son el bus, estrella y anillo, que a continuación se estudian.

Topología en BUS.

Consiste en un hilo único (bus) del cual se "cuelgan " cada una de las estaciones de la red como lo muestra la figura 1.6. El bus debe pasar cerca de todas y cada una de las estaciones.

Ventajas. No existen elementos centrales de los que dependa toda la red, cuyos fallos dejarían inoperantes a todas las estaciones. El cableado es de bajo costo, tanto por los materiales que se emplean como por su reducida complejidad de instalación. El momento de conexión/desconexión de las estaciones no afecta al funcionamiento de la red. Por último, el envío de información entre las estaciones es sencillo.

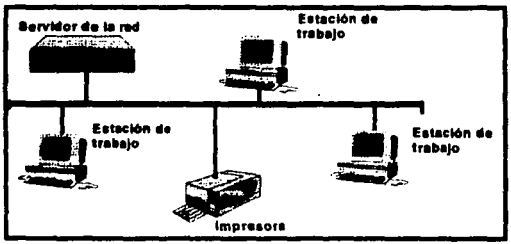


Fig. 1.6 Topología tipo BUS

Inconvenientes. Si se deteriora el cable se inutiliza la red por completo y además sólo se puede utilizar un medio de transmisión. En la actualidad es la topología más utilizada. Cuando la red es de grandes dimensiones se suele tener una estructura de múltiples buses interconectados todos ellos mediante un repetidor.

Topología en anillo.

Se construye un anillo físico tendiendo un conductor, generalmente de pares de hilos, desde cada estación a la siguiente. La información suele circular en un sólo sentido del anillo. Para que la información llegue a un nodo concreto debe pasar por todos los nodos anteriores, por lo que el envió de información a todas las estaciones resulta sencillo. A esta topología también se le conoce como "bucle". Para transmitir la información de un nodo a otro está se divide en paquetes que contienen la dirección del nodo que debe recibir la estación. La figura 1.7 muestra el diagrama de una configuración de red en anillo.

Ventajas. No existe dependencia de un nodo central. Es posible utilizar distintos medios de transmisión en diferentes sectores del anillo. El envió de información a todos los nodos es sencillo.

Inconvenientes. El cableado es caro, tanto por los materiales que se emplean como por la complejidad de su instalación. Una anomalía en el cable provoca la caída de toda la red. La fiabilidad de la red depende de todas y cada una de las estaciones, si cae una de ellas cae toda la red. Para añadir o retirar estaciones de la red es necesario detener la misma.

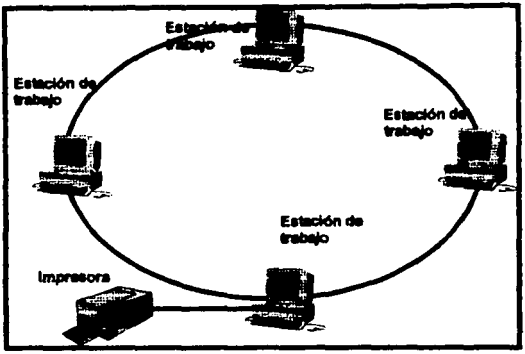


Fig. 1.7 Topología en anillo.

Topología en estrella.

Todas las estaciones de la red se unen mediante cables, generalmente. La unidad de control da turnos a las estaciones para utilizar la red. Este método se denomina polling. La unidad de control no tiene por que ser el servidor de ficheros, puede ser solamente un servidor de red, es decir la unidad encargada de gestionar el tráfico de información a través de la red. La figura 1.8 muestra el diagrama de una red conectada en estrella.

Ventajas. El protocolo de comunicación reside en la unidad central, por lo que se reducen las tareas de las estaciones y por tanto su costo. Las estaciones pueden tener diferentes velocidades de transmisión, medios y protocolos. Las averías son fáciles de localizar y es muy sencillo añadir o eliminar estaciones.

Inconvenientes. La unidad de control central es un punto crítico. Si éste cae toda la red cae. Para la instalación se requieren grandes cantidades de cable, ya que se deben unir cada una de las estaciones con la unidad central, por lo que el costo es elevado. El controlador central limita la relación entre el número de estaciones y las velocidades de éstas. En la implementación real se utilizan estrellas jerarquizadas, es decir, cada rama se ramifica de nuevo y se constituye una nueva red.

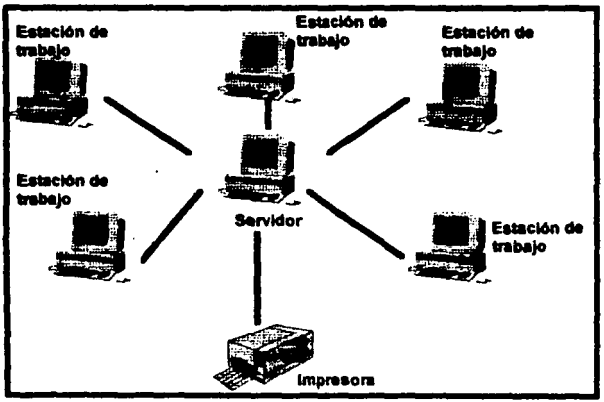


Fig. 1.8 Topología en estrella

Con los avanzados sistemas operativos y potentes equipos de comunicaciones que se comercializan actualmente, una de las topologías físicas más utilizadas es una mezcla de las tres que anteriormente se han expuesto. A medida que las necesidades de los usuarios crecen los diferentes tramos se construyen con

distintas tecnologías atendiendo a los criterios de longitudes, costos, velocidades, etc.

1.2.2.2 Técnicas y Medios de Transmisión.

Actualmente encontramos dos tipos de transmisiones que se llevan a cabo en las señales de las redes computacionales:

Transmisión en banda base. La señal procedente de la estación que desea comunicarse se entrega a la red en forma digital, sin modulación y ocupando todo el ancho de banda disponible en el medio. Es decir, empleando toda la capacidad del cable para transmitir la información.

Transmisión en banda ancha. La señal se modula y el ancho de banda disponible en el cable se divide en canales, por lo que el resto de los canales se pueden utilizar para otras comunicaciones de la red o para comunicaciones externas, por ejemplo, voz, otros datos, etc. Se podría decir que es como dividir el interior del cable en varios cables de menor sección y utilizar cada uno para un propósito. El inconveniente de utilizar banda ancha es que el hardware que se utiliza para esta técnica de transmisión tiene un costo muy elevado. Sin embargo, tiene la ventaja de que permite compartir los medios de transmisión con otros sistemas de comunicación, lo que la hace ideal cuando se tienen que instalar cables para sistemas de voz, vídeo y otros sistemas de datos.

Las diversas topologías se encuentran unidas a través de cables. Los tres tipos de cables que se utilizan en estas conexiones son:

-*Cables de pares trenzados*: Es el medio más barato. Tiene la gran ventaja de que en muchos de los edificios modernos ya está instalado para sus comunicaciones telefónicas, y se pueden utilizar como medio de la red local. El gran inconveniente que presentan es que son altamente sensibles a perturbaciones eléctricas del medio ambiente, por tanto la tasa de errores es alta y se deben reenviar los bloques de información, lo que se traduce en velocidades de transmisión muy limitadas.

-*Cable coaxial*: Es más caro que el anterior, pero tiene mejores características de transmisión de alta frecuencia, por lo que tiene la ventaja de ser menos sensible al ruido eléctrico, permitiendo velocidades de transmisión más elevadas. En el cable coaxial un hilo central transporta la señal. El hilo está protegido con un aislante y una camisa de hilos conductores en forma de malla que actúan como un escudo contra el ruido eléctrico. Este tipo de cable es el que se emplea para la instalación de la popular red local Ethernet.

-*Fibra óptica*: Es el medio de más reciente aparición. Se caracteriza por su altísima velocidad de transmisión, así como por su no menos elevadísimo costo (aunque actualmente tiende a bajar, a medida de que los medios para producirlo se hacen más sofisticados), tanto de los materiales como de instalación. El futuro de las redes LAN se ve plagado de cable de fibra óptica, pero mientras no se reduzcan los precios de éste se seguirán utilizando los dos tipos anteriores.

El estándar de redes locales por fibra óptica es el llamado FDDI (Fiber Distributed Data Interface/ Interfaz de distribución de datos de fibra óptica) y está normalizado por ANSI con el estándar X3T9.5. Entre sus características más importantes destacan su velocidad de transmisión de 100 Mbps y topología de doble anillo redundante (que envía datos simultáneamente en dos direcciones), así como los 2 KMs de distancia máxima entre estaciones y un máximo de 500 estaciones.

En una red normal de computadoras todas las estaciones están conectadas a un medio de comunicación único el cual sólo puede ser utilizado por una estación emisora a la vez y como este medio debe ser compartido de tal manera que se garantice que todas las estaciones tengan las mismas posibilidades de comunicarse a través de él se debe de seguir una técnica de red.

A estas técnicas de compartición del medio se les denomina *protocolos* de acceso, siendo los más populares:

A.- CSMA/CD. Acceso múltiple con escucha de portadora y detección de colisiones.
B.- Paso de testigo. (Token Pasing); utilizado en topologías bus (Token Bus) y anillo (Token Ring).
C.- Interrogación (polling).

En los ambientes de computación distribuida, como lo veremos en el siguiente capítulo, no es necesario que se tenga un medio de comunicación único, ya que precisamente el termino de *distribuida* viene de que se pueden tener varios medios de comunicación, los cuales pueden ser utilizados por más de una estación a la vez. Sin embargo lo anterior no libera a los sistemas distribuidos del uso de los protocolos, los cuales pueden ser los mismos que los empleados en las redes comunes.

1.2.2.3 Los estándares Ethernet y Token Ring.

Las redes Ethernet están basadas en el protocolo de acceso al medio CSMA/CD. En este protocolo la recepción de la información se realiza de una manera muy sencilla. Cada una de las estaciones conectadas a la red están siempre "escuchando" lo que transita por el cable, de forma que si detecta algún mensaje lo descifra y analiza a quien va dirigido. Si la dirección de destino del mensaje coincide con la dirección propia de la tarjeta, ésta almacena el mensaje en su memoria e informa a la computadora en la cual está instalada. Una vez que la computadora es informada, la tarjeta Ethernet espera a que ésta acepte el mensaje. Si por el contrario la dirección de destino no es la de la tarjeta, simplemente lo omite.

Por ejemplo, si una estación envía un mensaje a otra y la estación receptora está averiada o desconectada, la estación emisora realizará varios intentos para establecer comunicación. Si ésta sigue sin poderse establecer, la estación emisora dará como ausente a la estación receptora y el mensaje se perderá. Esto no afecta al resto de las estaciones de la red que pueden seguir operando normalmente. La velocidad máxima de transmisión en una red Ethernet, si no hay colisiones, es de 10 Megabits por segundo.

Token Ring atiende a las características del protocolo de paso de testigo. El medio de transmisión que se emplea es en anillo. Cuando una estación recibe el testigo por su lado "izquierdo", procedente de la estación anterior, debe pasarlo a la de su lado "derecho".

Si la estación que recibe el testigo no tiene nada que transmitir, sencillamente lo deja pasar. Cuando una estación emite un mensaje el testigo lleva la dirección de la estación de destino, que hasta llegar a esta última deberá pasar por todas las anteriores. Si a la estación transmisora le llega el testigo con la dirección de destino que ella había indicado, significa que la estación receptora está desconectada de la red.

Si una de las estaciones de la red se avería puede o bien desaparecer el testigo o bien enviar múltiples testigos en el anillo. Para resolver este problema una de las estaciones, la maestra, se encarga de volver a poner el testigo en marcha por la red o retirar los testigos sobrantes de la red. La velocidad de transmisión es de 4 Mbits por segundo (aunque esta no es la velocidad efectiva, ya que es necesario que el testigo pase por todas las estaciones).

1.3 Usos de las redes de computadoras

Vivimos en la era de las comunicaciones, en donde la computadora es, sin duda alguna, la herramienta que más ha propiciado el desarrollo actual de los medios de comunicación. Las redes de computadoras son en realidad un sistema de información electrónico. Actualmente es imposible concebir un mundo sin sistemas informáticos, ya que el papel que han jugado estos en las esferas gubernamentales, empresariales, en educación y hasta en entretenimiento ha sido determinante para alcanzar el grado de desarrollo en que en nuestros días se encuentran. Existen múltiples aplicaciones que se les han dado a las redes de computadoras, entre ellas podemos mencionar:

1. Correo electrónico; que consiste en enviar información de una a otra computadora obteniéndose resultados inmediatos.
2. Intercambio de archivos entre sistemas; que permiten distribuir las aplicaciones a diferentes puntos, con el consiguiente ahorro de tiempo.
3. Compartición de dispositivos periféricos; lo cual permite un gran ahorro en la compra de equipo y un incremento en la productividad.
4. Ejecución de un programa sobre otra computadora; hay casos en que es mejor correr una aplicación en una máquina diferente a donde el programa reside, debido a factores tales como el que el hardware de la máquina destino es mejor a el hardware de la máquina origen, o a que tan sólo se tiene una copia de un programa.
5. Accesos remotos; si dos computadoras están conectadas en red, se es posible acceder de una a otra máquina como si se tratara de una sola, lo cual hace posible el compartir las aplicaciones y la consulta de datos en diferentes lugares de almacenamiento.

Los puntos 4 y 5 serán discutidos más ampliamente en los capítulos siguientes, ya que son precisamente los dos aspectos más importantes sobre los que se basan los sistemas de computación distribuida.

Capitulo 2

Los sistemas de computación distribuida.

2.1 La Computación Distribuida

2.1.1 Concepto.

Las redes de computadoras han revolucionado el uso de la computadora. Ellas están presentes en nuestra vida diaria, desde cajeros automáticos hasta servicios electrónicos de mensajería; desde máquinas que hacen reservaciones aéreas hasta máquinas que hacen las veces de un mesero en un moderno restaurante. Hay muchas razones por las que se ha dado el explosivo crecimiento de las redes de computadora, entre ellas:

- El abaratamiento del hardware y la proliferación de computadoras personales.
- Los avances tecnológicos que han hecho posible que las computadoras traigan ya de fábrica software para implementación de redes como parte de su sistema operativo.
- El hecho de que nos encontramos en una era de la información y la computadora representa una parte integral para la diseminación de todo el cúmulo de acontecimientos.

Existen redes de computadoras en las que una o más máquinas se encuentran conectadas a un CPU central, en éste tipo de redes, todos los procesos y datos son cargados a partir del CPU que controla a la red, como un ejemplo de éste tipo de sistemas podemos mencionar a las populares redes LAN. Por otro lado también encontramos redes, generalmente de proporciones mucho mayores a las anteriores, en la que los procesos, datos o aplicaciones residen en diferentes CPU, a los que cada máquina puede acceder y procesar la información que resida en este lugar . A éste tipo de sistemas se les conoce como sistemas de *computación distribuida*.

La computación distribuida representa una estrategia para alcanzar los objetivos de la empresa del mundo de hoy. Los *procesos cooperativos*, la *tecnología cliente-servidor* y las *bases de datos distribuidas* son todas las variantes que presenta el campo de la computación conocido como computación distribuida, campo que también es conocido por el nombre de *procesos distribuidos* por algunos autores. Las tres tecnologías anteriores se caracterizan por correr bajo el mismo hardware y emplear mecanismos muy similares de software, sin embargo, la gran diferencia entre ellas, estriba en el hecho de como manejan y almacenan los datos. La figura 2.1 pretende ilustrar lo anterior, en ella podemos observar a la computación distribuida como parte de la computación empresarial y además los subsistemas de esta.

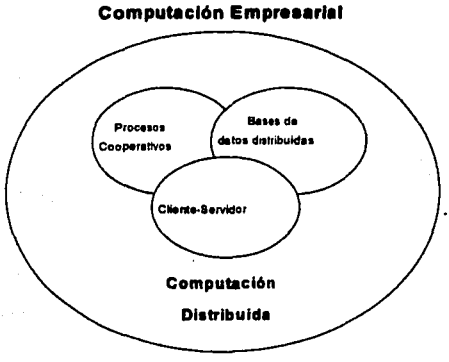


Fig. 2.1 Subsistemas de la computación distribuida

Los sistemas que operan bajo mecanismos de computación distribuida, contrastan con los sistemas centralizados en que en estos últimos, la totalidad de los datos aplicaciones y procesos se encuentran en un servidor principal regidos por una serie de características a fin de mantener una completa integridad de los datos, siendo su principal característica el echo de que sólo se permita una copia de los datos existentes, la cual reside en un sólo lugar, y en ese mismo lugar es actualizada y controlada.

En un sistema de computación distribuida, el usuario no hace distinción entre operaciones locales y remotas. Los programas no necesariamente se ejecutan en la estación de trabajo de donde fueron llamados. Sólo existe un archivo de sistema el cual es compartido por todos los usuarios. Los periféricos son compartidos. Existe más de un procesador central, por lo que la caída de uno no significa la parada del sistema.

La objetivo fundamental en un sistema de computación distribuida es compartir información. En su forma más básica, podemos decir que tenemos un sistema de computación distribuida cuando los procesos o aplicaciones corren en diferentes computadoras, sin la necesidad de una plataforma específica. La figura 2.2. muestra un ejemplo de un rudimentario sistema distribuido; en esta se puede observar a dos procesadores principales, los cuales se encuentran ligados el uno al otro. A su vez de cada procesador "cuelgan" varias estaciones de trabajo. La información se encuentra distribuida entre las dos minis, de tal forma que se pueda acceder a esta desde cualquier nodo del sistema. Aún suponiendo que no funcionara el procesador A, las estaciones que cuelgan de éste no se verían afectadas, ya que el procesador B tomaría su lugar, cosa que sería transparente para el usuario, siendo esta última la gran ventaja de los sistemas distribuidos.

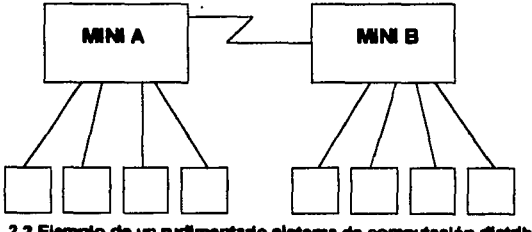


Fig. 2.2 Ejemplo de un rudimentario sistema de computación distribuido

2.1.2 Ventajas de los sistemas de computación distribuida.

Son obvias algunas de las ventajas de un sistema distribuido, como podría ser la compartición de recursos e información entre dos o mas servidores. Hay que tener en cuenta que la información generada en algún lugar frecuentemente es necesitada en otro; con un sistema distribuido esta información estará lista para compartirse casi desde el mismo momento en que se genere. Además de las ya descritas la computación distribuida presenta las siguientes ventajas:

Modularidad

En un sistema distribuido, las interfaces entre las partes del sistema son diseñadas con mucho más detalle que en un sistema centralizado . Como consecuencia, los sistemas distribuidos son construidos de una manera mucho más modular que en los sistemas comunes.

Expandibilidad

Como consecuencia de la modularidad tenemos a la expandibilidad. Los sistemas distribuidos son capaces de incrementar su capacidad de almacenamiento y/o procesamiento con el simple hecho de agregar nuevos servidores o procesadores al sistema.

Disponibilidad

El sistema distribuido siempre estará disponible, ya que la capacidad de procesamiento no depende de elementos centrales.

Escalabilidad

La capacidad de un componente de un sistema centralizado impone un límite para el crecimiento máximo del sistema. Como los sistemas distribuidos carecen de elementos centrales las restricciones de escalamiento prácticamente no existen.

Confiabilidad

Como parte de la confiabilidad del sistema tenemos a la disponibilidad. Un verdadero sistema confiable no sólo debe estar siempre disponible cuando se le necesite, sino que también debe de ser capaz de generar lo que se le pide, en el menor tiempo posible, aun bajo condiciones adversas. Los sistemas de computación distribuidos son diseñados atendiendo a estos aspectos, de tal forma que funcionen aún cuando algunos de sus componentes estén fallando.

2.1.3 El Sistema Operativo dentro de la computación distribuida.

El surgimiento de los sistemas operativos de computación distribuida constituye el siguiente paso lógico en la evolución de los sistemas operativos, evolución que ha sido posible gracias a una combinación de aspectos tanto tecnológicos como económicos. Así por ejemplo recordemos que los primeros sistemas operativos de tratamiento por lotes (Batch systems) pudieron ser desarrollados gracias a que las memorias de las computadoras llegaron a ser lo

suficientemente grandes como para poder soportar al sistema operativo y a las aplicaciones que constituyen un programa. Posteriormente debido a que los ciclos de operación de computadoras se abarataron y a que estas se hicieron más poderosas es que surgen los sistemas de tiempo compartido, los cuales permitieron a los programadores ser más productivos. El siguiente paso lógico, ha sido pues, el surgimiento de los sistemas de computación distribuida a los que podemos llamarlos los sistemas de la década de los 90`s

Para decirlo en pocas palabras, un sistema de computación distribuida es un sistema con muchos elementos de procesamiento y muchos dispositivos de almacenamiento, conectados todos en conjunto a través de una red. Potencialmente, un sistema distribuido es más poderoso que un sistema centralizado, principalmente debido a dos aspectos:

- *Son más confiables* a causa de que cada función es revisada varias veces. Cuando un procesador falla, otro toma de inmediato su lugar. Cada archivo puede ser almacenado en distintos discos, con el fin de que la falla de uno de ellos no destruya la información.

- *Son más rápidos*. Un sistema distribuido puede hacer más trabajo en la misma cantidad de tiempo que un sistema centralizado, a causa de que las computadoras tratan todos los procesos en paralelo.

Las dos características anteriores, tolerancia a las fallas y paralelismo, dan a los sistemas de computación distribuida el potencial para ser mucho más poderosos que un sistema de computación tradicional.

Las dos anteriores son las propiedades fundamentales de un sistema distribuido, las cuales sólo son posibles gracias a que se cuenta con un sistema operativo con características especiales. Todo sistema de computación distribuida deberá ser manejado por un sistema operativo distribuido, al respecto George A. Champine[1] da la siguiente definición refiriéndose a un sistema operativo de éste tipo: "*Un sistema operativo distribuido es un sistema que a la vista de los usuarios aparece como un sistema operativo centralizado ordinal, pero que corre sobre varios y a la vez independientes CPUs. La clave en estos sistemas se encuentra en la transparencia, en otras palabras, el uso de procesadores múltiples debe ser invisible (transparente) para el usuario. Otra manera de expresar la misma idea, es decir que a la vista de los usuarios, el sistema aparece como un único procesador virtual, y no como una colección de distintas máquinas.* " En capítulos siguientes se analizará más a detalle el funcionamiento de un sistema operativo distribuido, poniéndose especial atención a los procesos de comunicación entre máquinas conocidos comúnmente como RPC (Remote Procedure Call/ llamadas a procesos remotos).

A la parte fundamental del sistema operativo se le conoce como **kernel**, el cual provee de servicios tales como el manejador de archivos, administrador de memoria compartida, controladores de entrada y salida , etc. Comúnmente el **kernel** interactúa directamente con el hardware del sistema, pero en los ambientes distribuidos es necesario que el **kernel** pueda interactuar con otros sistemas operativos que en su turno controlan al hardware compartiéndose información que bien pudiera actuar bajo el control de sistemas operativos diferentes.

[1] George A. Champine, **Distributed Computer System**, North-Holland Publishing

2.1.4 El Modelo OSI

En un sistema distribuido se acostumbra a dividir a una determinada tarea en un conjunto de piezas, y solucionar en cada una de las computadoras del sistema, a cada una de estas piezas. Al final se vuelven a reconectar estas piezas a fin de obtener una solución parcial o total.

En el contexto de las redes de computadoras lo anterior es llamado encapamiento, lo cual no es otra cosa que dividir los problemas de comunicación en piezas (capas) y permitir que cada capa se especialice en una determinada función.

El punto de partida para describir las capas de una red es el modelo establecido por la agencia ISO (International Standards Organization), el cual recibe el nombre de **OSI** (Open Systems Interconeccion) para comunicación de computadoras, al cual podemos observar en la figura 2.3.

Fig. 2.3 Modelo OSI de 7 capas

El modelo OSI provee un estándar para describir a una red. La mayoría de las redes de computadoras están basadas en este modelo; para el tema que nos ocupa la capa de transporte es la más importante, ya que es la encargada de permitir el tránsito de datos entre más de dos sistemas.

La principal ventaja de tener este "encapamiento" es que se encuentran bien establecidas ciertas reglas conocidas como *protocolos* para permitir el transito de información entre cada nivel. Como todos los fabricantes de redes (tanto en software como en hardware) se tienen que apegar a estas reglas, es posible añadir a una red componentes de diversos fabricantes sin llegar a tener problemas de compatibilidad.

2.1.5 Técnicas empleadas en los sistemas distribuidos.

Las siguientes son algunas técnicas que son empleadas en la comunicación entre dos o más computadoras dentro de un sistema de computo distribuido. No se mencionan todas, pero sí las más usuales:

- Repetición de datos.

  Una transmisión de datos puede no ser suficiente para que la información sea recibida completa y libre de errores, por ello es que existe la repetición de información enviada a fin de reducir al mínimo los riesgos.

- Empleo de caches.

  Los caches son una técnica empleada para mejorar el rendimiento de un sistema, y que consiste en reservar un bloque especial de memoria donde estarán los datos utilizados con mayor frecuencia. Un cache para un sistema distribuido es una copia local de datos remotos. Es una técnica usada para mantener en cada procesador las palabras de memoria usadas con mayor frecuencia. El tener un cache permite al sistema accesar los datos remotos con la misma facilidad con que son accesados los datos locales. La función que realmente realiza el cache es almacenar la información que proviene del exterior y tenerla disponible para cuando una de las estaciones del sistema lo requiera esta información sea enviada, de tal manera como si todo fuera un sistema local.

- Uso de tiempos fuera.

  Hay muchos casos en los sistemas distribuidos en donde un proceso busca algún recurso que no se encuentra disponible porque este recurso esta siendo utilizado por algún otro proceso. Cuando los procesos o los host entran en conflicto, los recursos pueden aparecer como inaccesibles para otros procesos. Como regla general, en estos casos se usa una técnica conocida como tiempo fuera en la búsqueda de recursos. La técnica consiste en que los clientes (aquellas computadoras que solicitan los recursos) entren automáticamente en un tiempo de espera (timeout) al detectar un conflicto por compartición de recursos, lo cual provoca que no haya perdida de datos, ni caída temporal de algunos componentes del sistema.

- Uso de mecanismos estándar de invocación remota.

  Existen muchas maneras de hacer una petición a servicios remotos. La gran variedad de formas de invocar estos servicios llevó a buscar un estándar en los sistemas de computación distribuida. Actualmente los sistemas distribuidos sólo proveen algunas cuantas técnicas para hacer estas invocaciones, siendo las más populares los comúnmente llamados RPC (llamadas a procesos remotos) los cuales se estudiarán en el capítulo cuatro.

2.1.6 Paralelismo y procesamiento en línea (pipeline)

El paralelismo y el procesamiento en línea o pipeline también forman partes de las técnicas empleadas en los sistemas distribuidos, pero dada su importancia se ha decidido el tratar a estas dos por separado. Ya anteriormente hemos mencionado que el paralelismo es una de las principales características de los sistemas de computación distribuida. En esta ocasión vamos a centrar nuestro estudio en esta técnica que permite hacer de un sistema distribuido un sistema rápido y confiable.

Desde los albores de la computación, los diseñadores han tratado de hacer máquinas más rápidas. Hasta cierto punto, esto se podía lograr haciendo más veloz el hardware; sin embargo, empiezan a aparecer en el horizonte diversos límites físicos. Por un lado, las leyes de la física señalan que nada puede viajar más rápido que la velocidad de la luz, que es alrededor de 30 cm, por nanosegundo en el vacío y de 20 cm por nanosegundo a través del alambre de

cobre. Esto significa que para construir una computadora con un tiempo de instrucción de 1 nanosegundo, la distancia total que las señales eléctricas tuvieran que viajar entre la memoria, el CPU y de regreso, no debe exceder de 20 centímetros. De ahí que las computadoras demasiado rápidas tengan que ser muy pequeñas.

Desgraciadamente, las computadoras rápidas producen más calor que las lentas, y el construirlas en un volumen reducido hace difícil disipar dicho calor. En ocasiones, las supercomputadoras se sumergen en freón líquido, un refrigerante, a fin de eliminar el calor tan pronto como sea posible. Con todo, hacer más y más rápidas a las computadoras es cada día más difícil, así como muy caro. Por si todo esto fuera poco, imaginemos entonces las características que deberían tener las computadoras dentro de una red remota, en la que las estaciones pueden estar separadas varios kilómetros entre sí y donde los tiempos de respuesta que se requieren deben ser mínimos.

No obstante, es posible hacer a un sistema de computadoras más rápido utilizando una técnica diferente: en lugar de un sólo CPU que lleve a cabo las principales tareas del sistema, se puede construir un esquema computarizado con varias ALU más lentas (y más baratas), o hasta CPU completas, para obtener el mismo poder de cómputo a un costo más bajo. Así una tarea se puede dividir en varias subtareas y cada una de estas puede ser atendida por un CPU como si se tratara de un proceso independiente, trayendo por consecuencia la reducción del tiempo de procesamiento y el abaratamiento del proceso; a esta forma de trabajo es precisamente a lo que se le conoce como *paralelismo*.

Las máquinas paralelas pueden dividirse en tres categorías, basadas en el número de instrucciones y de datos que pueden procesar:

1. **SISD**: Corriente de instrucciones sencilla. Corriente de datos sencilla.
2. **SIMD**: Corriente de instrucciones sencilla. Corriente de datos múltiple.
3. **MIMD**: Corriente de instrucciones múltiple. Corriente de datos múltiple.

En la mayoría de los sistemas basados en máquinas tipo PC el procesamiento en paralelo es del tipo **SISD**, ya que sólo hay un programa y un conjunto de datos. En sistemas de este tipo se separa en partes la ejecución de cada instrucción, como el armado de un auto en una línea de ensamble. En la figura 2.4, se observa una CPU formada por cinco unidades de procesamiento de P1 a P5. Durante el primer intervalo del tiempo, P1 extrae de memoria la primera instrucción; en el segundo intervalo, la primera instrucción pasa a P2 para su análisis, mientras que P1 extrae la siguiente instrucción. En cada uno de los intervalos subsecuentes, P1 extrae una nueva instrucción y las anteriores pasan a la siguiente unidad, a lo largo de la trayectoria.

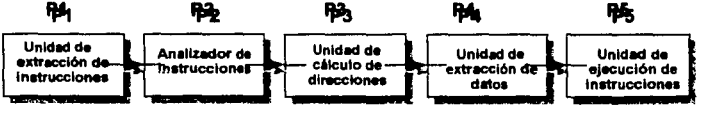


Fig. 2.4 Procesamiento en línea o escalonado (pipeline) de cinco unidades

A la organización de la figura anterior se le podría llamar *procesamiento entubado o procesamiento en línea* ***(pipeline)***. Si cada paso (intervalo de tiempo) dura n nanosegundos, se requieren 5n nanosegundos para ejecutar una instrucción. Sin embargo, cada n nanosegundos P5 termina de ejecutar una instrucción, lo que incrementa la velocidad en un factor de cinco. Nótese que aun cuando se emplea cierto paralelismo interno, el procesamiento entubado sigue siendo del tipo **SISD**, ya que sólo hay un programa y un conjunto de datos. Las máquinas de tipo **SIMD** operan en paralelo con múltiples conjuntos de datos. Una aplicación clásica de este tipo es el pronóstico del tiempo, donde hay que calcular la temperatura diaria promedio en diversos sitios, a partir de los promedios de las 24 horas del día. Para cada sitio, se deben hacer exactamente los mismos cálculos, pero con datos diferentes. En la computación distribuida se emplea generalmente el modelo **MIMD**, en la cual diferentes CPU manejan distintos programas compartiendo a veces, una memoria común. Por ejemplo, en un sistema de reservaciones de una línea aérea varias reservaciones simultáneas no se procesan en paralelo instrucción por instrucción, sino que se tienen múltiples flujos de instrucciones y datos los cuales son divididos y procesados de forma independiente juntándose al final el resultado de todos los procesos. En la figura 2.5 se muestra un **multiprocesador**, o sea una máquina **MIMD** con memoria compartida, a la cual tiene acceso cada procesador a través del bus.

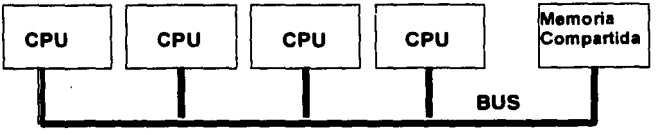


Fig. 2.5 Un multiprocesador básico

No se necesita mucha imaginación para darse cuenta de que habrá dificultades si un gran número de procesadores, pretende constántemente tener acceso a la memoria por medio de un mismo bus. Para reducir esta competencia y mejorar el desempeño, los diseñadores de multiprocesadores elaboraron varios esquemas; la figura 2.6 muestra uno de estos diseños, el cual es uno de los esquemas más usuales en plataformas cliente-servidor de índole pequeña. En la figura cada uno de los procesadores tiene cierta memoria local propia que no es accesible al resto. Esta puede ser usada para almacenar el código del programa así como aquellos datos que no necesitan compartirse. El acceso a esta memoria privada no utiliza el bus principal, con lo que el tráfico se reduce en gran medida.

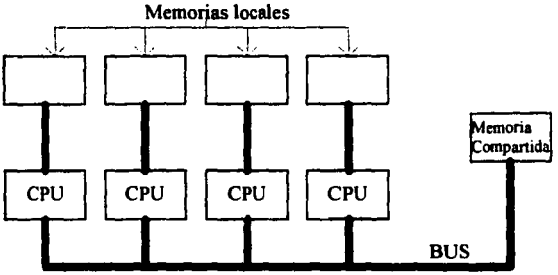


Fig. 2.6 Un multiprocesador con memorias locales

Resumiendo podemos decir que las máquinas **SISD** son las computadoras clásicas secuenciales de Von Neumann. Tienen un flujo de instrucciones, uno de datos y realizan una operación a la vez. Las máquinas **SIMD** tienen varias ALU para llevar a cabo una instrucción con diferentes conjuntos de datos en forma simultánea. La máquina *ILLIAC IV* es el prototipo de las máquinas **SIMD**.

Las máquinas **MISD** son en cierta forma una categoría extraña, con instrucciones múltiples operando sobre los mismos datos. No está claro si tales máquinas existen, aun cuando algunos clasifican a las que tienen procesamiento en serie en este tipo.

Por último tenemos a las **MIMD**, que son varias CPU independientes que operan como parte de un sistema más grande. La mayoría de los procesadores en paralelo caen dentro de esta categoría. Algunos expertos acostumbran a dividir a la categoría **MIMD** en máquinas denominadas multiprocesadores, que comparten una memoria primaria y otras que no lo hacen. Estas últimas han

recibido diversos nombres como *multicomputadoras, computadoras de memoria primaria o computadoras de memoria desarticulada.*

Retomemos ahora el concepto de procesamiento en línea o pipeline a fin de explicarlo más claramente. Como ya se menciono el pipeline es una forma para acelerar a las máquinas que trabajan dentro de sistemas de computación distribuida, el cual consiste en conformar el hardware con varias unidades funcionales y después unirlas para que trabajen de una manera escalonada. Dicho de otra manera *el procesamiento pipeline no es otra cosa mas que una técnica que suministra procesamiento paralelo dentro de un sistema de computadoras, en el cual se lleva a cabo una superposición de operaciones, llevando los datos o instrucciones a un "tubo conceptual", donde todas las etapas del "tubo" se procesan en forma simultánea.* Por ejemplo, mientras se ejecuta determinada instrucción, el computador decodifica la próxima instrucción.

En la figura 2.4 se observó cómo funciona una computadora con cinco unidades escalonadas, lo cual es lo mismo que ocurre cuando la tarea se divide entre cinco computadoras distribuidas. En la tabla 2.1 se muestra que si la instrucción 1 se extrae en el ciclo 1, se ejecutará en el ciclo 5. En forma parecida, si la instrucción 4 se extrae en el ciclo 8, se decodifica en el 9, etc., se ejecutará en el ciclo 12. Bajo condiciones óptimas, en cada ciclo de ahí en adelante se ejecuta una instrucción, dando un promedio de ejecución de una instrucción por ciclo, en lugar de una cada dos ciclos.

Desafortunadamente, hay estudios que han demostrado que alrededor del 30% de las instrucciones son saltos y estos hacen estragos en la línea de procesamiento. Los saltos pueden clasificarse en tres categorías: incondicionales, condicionales e iterativos. En la tabla 2.1 se puede observar una instrucción (señalada con B) de salto condicional. La siguiente instrucción a ejecutar debería ser la siguiente a la del salto, pero también podría ser la dirección a la cual saltó, llamada **blanco del salto**. En virtud de que el extractor de instrucciones no sabe cual sigue sino hasta que se ejecuta el salto, se demora y no puede continuar hasta la ejecución de éste. En consecuencia la línea se vacía. Sólo después de que termina el ciclo 7, se sabe que instrucción sigue. A la perdida de cuatro ciclos causada de hecho por el salto se le denomina **penalización por salto**. Resulta evidente que con un salto en una de cada tres instrucciones, la disminución en el desempeño es sustancial.

	CICLO									
	1	2	3	4	5	6	7	8	9	10
Extracción de instrucciones	1	2	B					4	5	6
Decodificación de instrucciones		1	2	B					4	5
Cálculo de direcciones			1	2	B					4
Extracción de operandos				1	2	B				
Ejecución					1	2	B			

Tabla 2.1 Unidad de procesamiento en línea de 5 etapas. Las instrucciones están etiquetadas con números. La instrucción marcada como B es un salto condicional.

Se han realizado una gran cantidad de investigaciones alrededor del problema de recuperar algo de este desempeño (al respecto podemos mencionar los trabajos de DeRosa y Levy, 1987; Hwu y col. 1990[2]). Lo más sencillo consiste en esperar que el salto no se realice y continuar procesando como si el salto fuera una simple instrucción aritmética. Si resulta que en efecto, el salto no se realiza, no se ha perdido nada. Si por el contrario el salto se efectúa, se deben eliminar las instrucciones que estén actualmente en la línea y empezar de nuevo.

2.2 Control de Procesos.

Ya se ha mencionado que las bases de datos distribuidas, los procesos cooperativos y la tecnología cliente/servidor son todas las formas bajo las cuales se nos puede presentar a la computación distribuida; en todas estas formas deberán existir mecanismos de comunicación entre procesos a fin de que las solicitudes de los clientes (las computadoras que hacen las peticiones sean atendidas). Estos mecanismos de comunicación deberán también ser controlados por rutinas especiales llamadas **control de procesos**. A continuación se examinará la manera en que estos programas son ejecutados y como los procesos son creados y terminados.

[2] DeRosa, J.A., y Levy, H. M.: "An Evaluation of Branch Arquitectures"; ACM, 1988.
Hwu, W.W. Et. Al.: "Comparing Software and Hardware Schemes for Reducing the cost of Branches"; ACM, 1989.

En los ambientes de computación distribuida siempre existirá un módulo de programa que llamará a otros módulos (llamados módulos de nivel inferior) a fin de llevar a cabo una tarea, dependiendo de la forma en que se comuniquen estos módulos entre sí podemos decir que tenemos dos tipos de comunicación: jerárquica y en pipeline, como se ilustra en la figura 2.7. En el modo jerárquico el módulo que llama envía un mensaje de petición al primero de los módulos inferiores, recibe un mensaje de recibido de este módulo, procesa el mensaje y en caso de que la petición no se haya satisfecho envía otro mensaje de petición al siguiente de los módulos inferiores. Este mecanismo se repite hasta que todos los módulos inferiores hayan sido cubiertos o la petición haya sido satisfecha.

En el modo pipeline, el mensaje fluye directamente desde el módulo que llama hasta los módulos que son llamados, esta técnica permite reducir el trafico de mensajes hasta en un 50% en relación con el modo jerárquico.

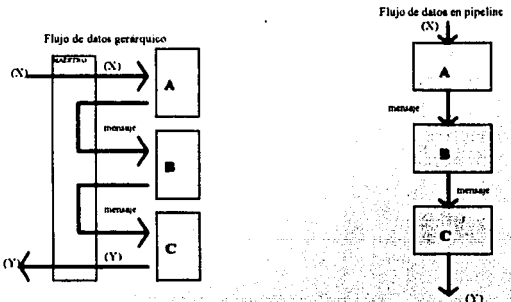


Fig. 2.7 Comparación de modos gerárquico y pipeline. En este ejemplo, el módulo maestro (en el modo gerárquico), envía y recibe varias veces un mensaje a fin de llegar al último módulo. En el modo pipeline se envía un mensaje y este fluye desde el primero hasta el último de los módulos.

2.2.1 Corrutinas y procedimientos.

Ya sea en forma jerárquica o en forma de pipelínea los programas se ejecutarán siempre a manera de llamadas de un módulo o procedimiento (el que llama) hacia otro módulo o procedimiento (el llamado). En la secuencia de llamada es clara la distinción entre el procedimiento que llama y el llamado. Considérese el procedimiento A, que llama al procedimiento B, en la fig. 2.8.

El procedimiento B hace cálculos un rato y retorna a A. A primera vista podría considerarse simétrica esta situación, ya que ni A ni B son el programa principal, sino que son procedimientos (el procedimiento A podría haber sido llamado por el programa principal, pero este hecho es irrelevante). Además, primero hay una transferencia de control de A a B (la llamada) y más tarde una transferencia de control de B a A (el retorno).

La simetría viene del hecho de que cuando pasa el control de A a B, B comienza a ejecutarse desde el principio. Cuando B retorna a A, la ejecución no comienza al principio de A, sino en la sentencia que sigue a la llamada. Si A se ejecutará durante un rato y llamará a B otra vez, la ejecución comenzaría al principio de B de nuevo, no en la sentencia que sigue al retorno anterior. Si, en el transcurso de la ejecución, A hiciera referencia a B muchas veces, cada una de las veces que arrancara B, éste comenzaría por el principio, mientras que A nunca lo hace.

Esta diferencia se refleja en el método por el que se pasa el control de A a B. Cuando A llama a B, usa la instrucción de llamada a procedimiento, que pone la dirección de retorno (es decir, la de la primera sentencia que sigue a la llamada) en algún sitio útil, por ejemplo en la cima de la pila. Pone entonces la

dirección de B en el contador de programa para completar la llamada. Cuando B retorna, no usa la instrucción de llamada, sino la de retorno, que simplemente desapila la dirección de retorno y la mete en el contador del programa. En los sistemas distribuidos se tendrán por lo general, dos procedimientos, A y B, cada uno de los cuales llamará al otro como a un procedimiento. Cuando B retorna a A, salta la sentencia que sigue a la llamada a B, como antes. Cuando A transfiere control a B, no va al principio (excepto la primera vez) sino a la sentencia que sigue al "retorno" más reciente, es decir, a la más reciente llamada a A. Dos procedimientos, cada uno de los cuales considera el otro como un procedimiento (en el sentido de que es llamado, realiza cierto trabajo y retorna a la sentencia que sigue a la llamada), se llaman **corrutinas** y la instrucción que llama a la corrutina se denomina, instrucción de **reanudar**.

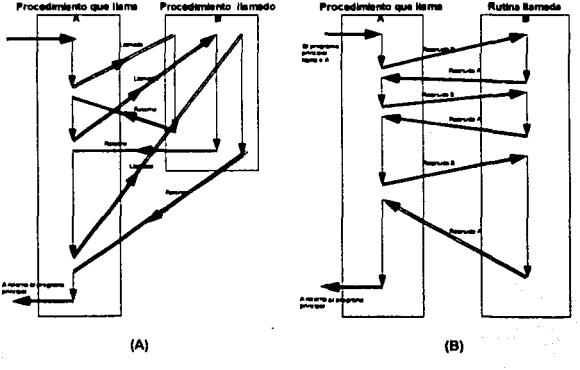


Fig. 2.8 (A) Cuando se llama a un procedimiento, su ejecución comienza siempre en la primera sentencia. (B) Cuando se reanuda una corrutina, su ejecución comienza por la primera sentencia que no ejecutó la vez anterior, no por el principio

2.2.2 Creación de procesos.

El sistema operativo de los sistemas de computación distribuida permite la creación y la destrucción de procesos sin la necesidad de parar a una o más de las computadoras del sistema. Las computadoras que forman parte del sistema distribuido pueden soportar un número variable de máquinas de nivel 3, de manera que a cada proceso le corresponde una máquina virtual[3]. Para obtener todas las ventajas del procesamiento en paralelo, un programa del nivel 3 necesita una instrucción virtual para crear nuevos procesos en los que delegue trabajo. Los sistemas operativos distribuidos disponen de una instrucción de nivel 3 para crear un nuevo proceso, la cual permite que el proceso creador especifique el estado inicial del nuevo, incluyendo su programa, datos y dirección de comienzo.

En algunos sistemas operativos, el proceso creador (padre) mantiene un control total sobre el creado (hijo). En este caso existen instrucciones virtuales para que el padre detenga, haga continuar, examine y haga terminar a su hijo. En otros casos, el padre tiene menos control sobre sus hijos: una vez que un proceso ha sido creado, no hay manera de que el padre lo detenga, haga continuar, examine o haga terminar. Los dos procesos se ejecutan con total independencia de uno respecto del otro.

Hay ocasiones en que dos procesos independientes comunicados por medio de un registro compartido en la memoria de alguna de las estaciones del sistema y que se ejecutan en paralelo y de manera asíncrona se encuentran en la

[3] Una máquina virtual es el sistema operativo de los *mainframe* de IBM, originalmente desarrollado por sus clientes y finalmente adoptado como un producto de sistemas de IBM (VM/SP). VM puede correr múltiples sistemas operativos dentro del CPU al mismo tiempo, ejecutando cada uno sus propios programas.

necesidad de utilizar un mismo recurso o una misma subrutina. Por razones de simplicidad llamaremos al proceso 1 **productor** y al proceso 2 **consumidor**. Como es lógico uno de los procesos (generalmente el de mayor velocidad, el productor en este caso) ganará el uso del recurso o subrutina que se requiere. Pero puede darse el caso en que sea el consumidor el que gane el uso del recurso, lo que ocasionará que el otro proceso, el productor, entre en un estado de espera para poder disponer del recurso ocupado por el consumidor. Si el proceso consumidor termina de utilizar el recurso y para continuar su funcionamiento precisa de los resultados que haya obtenido el productor y como éste último se encuentra en estado de espera, ocasionará esto que el consumidor entre también a un ciclo de espera, con lo que ambos procesos estarán interrumpidos y continuarán siempre así, lo que podría ocasionar la caída del sistema. A esto se le llama **condición de carrera**, puesto que el éxito del método depende de quien gana la carrera para el uso del recurso necesitado.

Una de las soluciones más aceptables para el problema de la condición de carrera es la propuesta por el matemático Dijkstra (1986), la cual pretende lograr la perfecta sincronía en los procesos paralelos. En alguna parte de la memoria hay dos enteros no negativos llamados **semáforos**. El sistema operativo soporta dos instrucciones de nivel 3 que operan sobre los semáforos, SUBIR (UP) y BAJAR (DOWN). SUBIR añade 1 a un semáforo y BAJAR le resta 1.

Si se aplica una instrucción BAJAR a un semáforo mayor que 0, éste es decrementado en 1 y el proceso que la ejecuta continúa. Pero si el semáforo es 0, la instrucción bajar no puede completarse y el proceso que llamó a BAJAR entra en estado de espera y permanece en este estado hasta que otro proceso haga un SUBIR sobre el mismo semáforo. La instrucción SUBIR mira si el semáforo es cero. Si lo es y hay otro proceso detenido en él, el semáforo se incrementa en 1. El proceso en espera puede ahora completar la operación BAJAR en la que se había detenido, poniendo de nuevo el semáforo a 0 y permitiendo a ambos procesos continuar el cálculo. Una propiedad esencial de las instrucciones de semáforo es que, una vez que un proceso ha iniciado una instrucción sobre un semáforo, ningún otro proceso puede acceder al mismo hasta que el primero haya completado la instrucción o se haya parado al intentar un BAJAR en un cero. La figura 2.9 resume las propiedades de las instrucciones de semáforo.

Instrucción	Semáforo = 0	Semáforo > 0
SUBIR	Semáforo = semáforo+1 Si el otro proceso se ha interrumpido al intentar completar una instrucción de BAJAR en este semáforo, puede ahora completarla y continuar ejecutándose Semáforo = semáforo +1	Semáforo = semáforo +1
BAJAR	El proceso se para hasta que otro ejecute un SUBIR sobre este semáforo	Semáforo = semáforo -1

FIG. 2.9 Efecto de una instrucción semáforo.

La propiedad esencial de las operaciones de semáforo es que son indivisibles. Una vez que se ha iniciado una operación de semáforo, ningún otro proceso puede usarlo hasta que el primero haya completado la operación o se haya suspendido mientras lo intentaba. La sincronización mediante los semáforos es una técnica que funciona en un número arbitrariamente grande de procesos. Varios procesos pueden detenerse cuando intentan completar una instrucción BAJAR en el mismo semáforo. Si algún otro proceso hace por fin un SUBIR en ese semáforo, uno de los que esperan puede completar su instrucción bajar y continuar. El valor del semáforo continúa siendo 0 y los otros procesos continúan esperando.

2.2.3 El Sistema Operativo UNIX

El sistema operativo más popular en ambientes de computación distribuida es el UNIX, ya que éste cuenta con muchas de las características de las que hemos venido hablando a lo largo de este capítulo. UNIX corre casi independientemente del equipo sobre la que se encuentre instalado, cosa que no hacen sistemas operativos tales como MS-DOS. UNIX poco a poco a venido acrecentando su popularidad debido principalmente a que puede ser adaptado a muchas máquinas, desde computadoras personales hasta supercomputadoras. La amplia disponibilidad de UNIX significa que puede escribirse una sola vez un programa de aplicación y después recompilarse por una docena de máquinas diferentes sin ser modificado. Esta característica sólo ha sido posible porque UNIX hace muy pocas suposiciones acerca del hardware y muy pocos requerimientos de éste.

El modelo de memoria de UNIX es simple. Cada proceso como se puede ver en la figura 2.10, tiene tres segmentos: de código, de datos y de pila. El código tiene una dimensión fija, pero tanto los datos como la pila pueden crecer en direcciones opuestas. Este modelo es fácil de implantar en casi cualquier máquina.

Además, si la máquina posee paginación se puede paginar el espacio completo de direcciones, sin que el programa de usuario ni siquiera se dé cuenta. Lo único que se nota es que se permite tener programas más grandes que la memoria física de la máquina. Los sistemas UNIX que no paginan por lo general intercambian procesos entre la memoria y el disco, para permitir que un gran número de procesos sean de tiempo compartido.

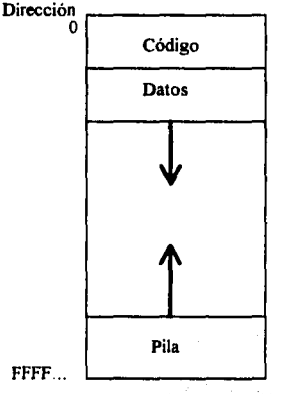


Fig. 2.10 Espacio de direcciones de un solo proceso UNIX.

Gran parte de la popularidad de UNIX se debe a su simplicidad, que a su vez es resultado directo de la organización de su sistema de archivos. A continuación se muestra un fragmento de programa C que ilustra cómo funcionan las llamadas de E/S más importantes. Hemos decidido emplear código C en todos los ejemplos que se presentarán de aquí en adelante ya que el

lenguaje C es un lenguaje nativo de UNIX, de hecho UNIX esta echo en lenguaje C, de ahí la gran potencia que alcanzan las aplicaciones desarrolladas en C en entornos de UNIX. Además el código en C es muy portable en todos los sistemas UNIX y los programas que se presentarán a lo largo de los siguientes capítulos han sido desarrollados para que puedan ser manipulados en diferentes máquinas y con los cambios mínimos si es que lo llegaran a necesitar.

En el fragmento de programa observamos que antes de entrar a la interacción, el programa abre un archivo existente, *datos*, y crea uno nuevo *nuevof*. Los segundos parámetros de las dos llamadas especifican los archivos que han de leerse y escribirse, respectivamente. Ambas llamadas devuelven un pequeño entero positivo denominado *descriptor del archivo*, con el cual se localiza el archivo en llamadas posteriores. Si falla la apertura o la creación de archivos, se devuelve un descriptor de archivo negativo, lo cual indica que la llamada falló.

```
entd=open("datos",O_RDONLY);  /*abre archivo existente datos*/
said= creat ("nuevof",BitsProtec); /*crea nuevo archivo nuevof*/
do
{
        cuenta=read (entd, buffer, byte);     /*lee el buffer*/
        if cuenta > 0
          write (said, buffer, byte);   /*escribe buffer*/
} while (cuenta > = 0);
close (entd);
close(said);
```

2.2.4 Administración de procesos en UNIX

Un proceso UNIX puede crear en cualquier momento, un subproceso que sea una réplica exacta del anterior, mediante la ejecución de una llamada de sistema **FORK** (bifurcar). Al proceso original se le denomina padre y al nuevo, hijo. Inmediatamente después de la ejecución de la llamada, los dos procesos son idénticos y hasta comparten los mismos descriptores de archivos. En seguida cada uno toma su propio camino y realiza sus funciones en forma independiente del otro.

En muchos casos, el proceso-hijo manipula de algún modo los descriptores de archivos, ejecutando después la llamada del sistema **EXEC**, la cual sustituye su programa y sus datos con los que encuentra en el archivo ejecutable que se especificó como parámetro de la llamada EXEC. Por ejemplo, cuando un usuario teclea en una terminal un comando xyz, el intérprete de comandos denominado shell ejecuta la llamada FORK para crear un proceso hijo, el que a su vez ejecuta EXEC para correr el programa xyz.

Ambos procesos corren en paralelo (con o sin EXEC), a menos que el padre desee esperar a que el hijo termine, antes de continuar. Si esto sucede, el padre ejecuta la llamada **WAIT** (espera), la cual suspende el proceso hasta que el hijo concluye ejecutando **EXIT**, después de lo cual, el padre continua. Los procesos pueden ejecutar estas bifurcaciones cuantas veces lo deseen , haciendo crear así un árbol de procesos.

Por ejemplo en la figura 2.11, el proceso A ha ejecutado FORK en dos ocasiones, creando los hijos B y C. Luego B ha ejecutado también dos veces la llamada y C en una ocasión, dando como resultado un árbol de 6 procesos.

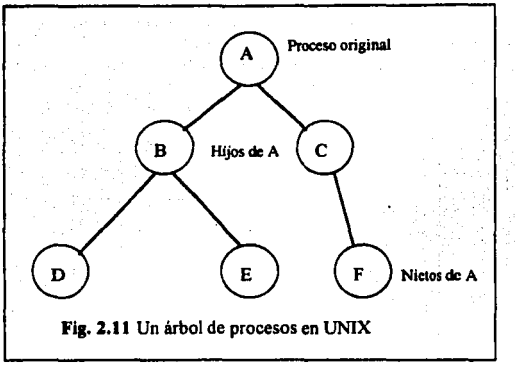


Fig. 2.11 Un árbol de procesos en UNIX

En UNIX, los procesos se pueden comunicar entre sí por medio de una estructura denominada conducto, que es una especie de buffer en el que un proceso puede escribir una serie de datos y otro recuperarla. En un conducto, los bytes siempre se recuperan en el orden en que se escribieron. No es posible realizar acceso aleatorio.

La programación en ambientes distribuidos involucra la interacción de dos o más procesos. Ya hemos mencionado la manera en que estos programas son creados y ejecutados; a continuación ejemplificaremos el uso de la sentencia fork, la cual es la única manera por la cual puede ser creado un nuevo proceso en UNIX a partir de un proceso ya existente.

La llamada al sistema fork crea una copia de el proceso que fue ejecutado. El proceso que ejecuta el fork es llamado el proceso padre y el nuevo proceso es el

proceso hijo. La sentencia fork es llamada una sola vez (por el proceso padre), pero esta retorna dos valores, un valor es para el padre que es la dirección del nuevo proceso hijo y el otro valor es para el hijo el cual siempre es cero. Si la llamada a la sentencia fork no tiene éxito, se retorna un valor -1. Para obtener la dirección del proceso padre el hijo puede ejecutar la llamada getppid(). Un ejemplo de la llamada del sistema fork se muestra a continuación:

```
main( )
{
int childpid;

  if (( chilpid = fork( ) == -1)  {
        perror("ERROR en llamada fork");
         exit(1);
  } else if (chilpid == 0) {
        /*proceso hijo*/
        printf("Hijo: hijo pid = %d, padre pid= %d \n", getpid( ), getppid( ));
        exit(0);
  } else {
       /* proceso padre*/
       printf("Padre: hijo pid = %d, padre pid = %d \n", childpid, getpid( ));
        exit(0);
  }
}
```

Podemos representar la operación fork como lo muestra la figura 2.12.

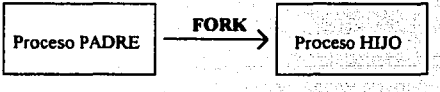


FIG. 2.12 Representación de una sentencia FORK

Si un segmento de texto puede ser compartido, entonces tanto el padre como el hijo pueden compartir este segmento después de la llamada a un fork como lo muestra la figura 2.13.

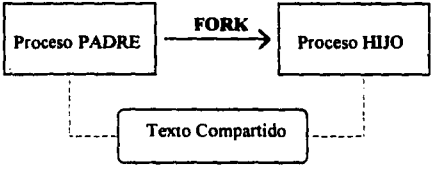


FIG. 2.13 Representación de una sentencia FORK con texto compartido.

Un hecho importante en una operación fork es que los archivos que fueron abiertos por el proceso padre antes de un fork son compartidos por su proceso hijo después del fork. Los archivos o dispositivos que lleguen a ser requeridos por un proceso hijo son liberados después de su uso por el proceso padre.

Un proceso hijo difiere del proceso padre en los siguientes aspectos:

1. El proceso hijo tiene un nuevo y único proceso ID[4]
2. El proceso hijo tiene un padre diferente al proceso ID
3. El proceso hijo tiene su propia copia de los descriptores del padre.

[4] En UNIX a cada proceso, al momento de ser creado, se le asigna un número entero, generalmente entre 0 y 30000 que es conocido como ID. El proceso con ID=1 es el proceso inicial. También existen diferentes ID's empleados para identificar a los procesos padres, a grupos de terminales y a usuarios del sistema.

Hay dos usos para una operación fork:

- Cuando un proceso requiere hacer una copia de sí mismo, a fin de que una copia haga una operación y la otra pueda realizar una segunda tarea. Este es un caso típico en los servidores de redes.

- Cuando un proceso requiere ejecutar otro programa. Debido a que la única manera de crear un nuevo proceso es con la operación fork, el proceso debe primero ejecutar un fork sobre sí mismo, y entonces, una de las copias (generalmente el proceso hijo) invoca un **exec** para ejecutar el nuevo programa. Este es un caso común para programas tales como los shells.

La única manera en la cual un programa es ejecutado por UNIX es mediante un proceso existente que haga una llamada a la función exec. La función exec reemplaza el proceso actual con el nuevo programa. El proceso ID no sufre cambios. Al proceso que requirió de la función exec se le conoce como el proceso llamante y al programa que es ejecutado se le conoce como el proceso llamado o nuevo programa.

2.2.5 Creación de pipas en UNIX

Una de las grandes facilidades que nos ofrece UNIX es en la creación y manejo de pipas. Una pipa es un factor importante en un sistema distribuido debido a que nos proporciona una fácil manera de llevar a cabo el flujo de datos. La instrucción que empleamos en UNIX para la creación de pipas es:

int pipe(int *filedes);

la cual regresa dos descritores de archivo - filedes[0] archivo de lectura, filedes[1] archivo de escritura-. El siguiente programa muestra la forma de crear y usar una pipa dentro de UNIX:

```
main( )
{
     int pipefd[2], n;
     char buff[100];

     if (pipe(pipefd) < 0)
          err_sys("ERROR DE PIPAS");
     printf("Lee fd= %d, Escribe fd= %d \n" pipefd[0], pipefd[1]);
     if (write(pipefd[1], "hola mundo \n", 12) !=12)
          err_sys ("ERROR DE ESCRITURA");

     if ( (n=read(pipefd[0], buff, sizeof(buff))) <= 0)
          err_sys("ERROR DE LECTURA");

     write(1,buff,n);
      exit(0);
}
```

Las pipas en los ambientes distribuidos son la manera más común de llevar a cabo la comunicación entre dos procesos diferentes. Por ejemplo supongamos que un proceso desea realizar dos tareas simultáneamente, para ello primeramente tendrá que crear una pipa y posteriormente deberá hacer una llamada a un fork para crear una copia de sí mismo, de tal manera que se tenga una estructura como la mostrada en la figura 2.14:

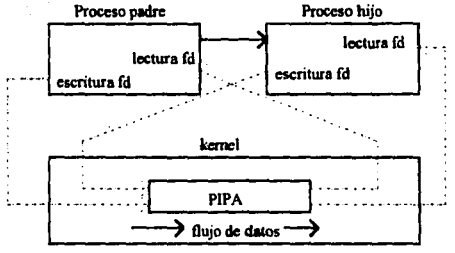


Fig. 2.14 Una pipa inmediatamente después de una llamada a fork.

Posteriormente el proceso padre podrá mandar la información necesaria a través de la pipa y hacia el proceso hijo, mandando al final de la información una señal de fin de envío; el proceso hijo leerá la información y al final mandará una señal de fin de lectura a fin de que ambos procesos estén perfectamente coordinados. Esto constituye una manera fácil de intercambio de datos entre los dos procesos, teniéndose una estructura similar a la de la figura 2.15.

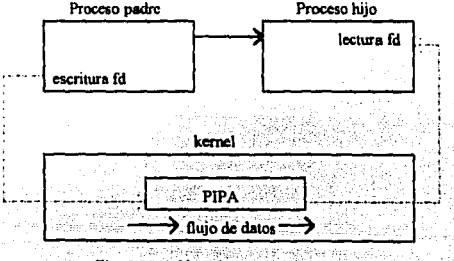


Fig. 2.15 Una pipa entre dos procesos.

En ocasiones se desean tener pipas bidireccionales, para lo cual es preciso crear dos pipas y usar cada una para cada dirección, para ello los pasos que se deberán seguir son:

1. crear la pipa1 y pipa2
2. ejecutar la sentencia fork
3. el padre envía la información a través de la pipa1 y envía la señal de termino de información.
4. si se ha recibido información a través de la pipa2, el padre envía una señal de recibido.
5. el hijo lee la información de la pipa1
6. el hijo envía la señal de termino de lectura a través de la pipa2.

Los anteriores pasos nos generarán una estructura similar a la de la figura 2.16.

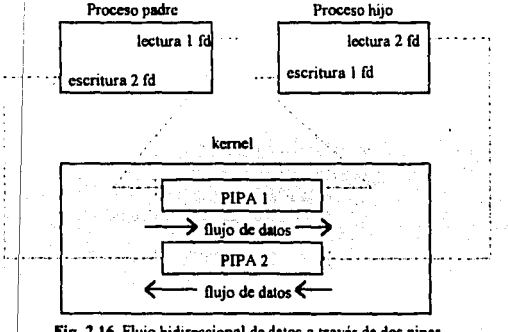


Fig. 2.16 Flujo bidireccional de datos a través de dos pipas.

A lo largo de este capítulo hemos venido mencionando las principales técnicas que son empleadas en los sistemas de computación distribuidos. Hasta ahora no hemos redundado en los tres tipos comunes bajo los cuales se presenta la computación distribuida; en el capítulo tres abordaremos precisamente estos tres puntos, destacando lo más imporatante de cada uno de ellos. También en el capítulo cuatro mostraremos un programa donde hacemos uso de algunas de las técnicas anteriores en conjunto con RPCs para comunicaciones en ambientes cliente-servidor.

Capitulo 3

Procesos Cooperativos. Bases de Datos Distribuidas. y Tecnología Cliente o Servidor.

En el capítulo anterior se discutieron los principales mecanismos que son comunes a los sistemas de computación distribuida. En sí, los puntos ya discutidos, forman la base para la operación de las tres formas bajo las cuales se nos puede presentar un sistema distribuido. A pesar de que estas tres formas tienen muchas semejanzas, también son importantes sus diferencias. A lo largo de este capítulo estudiaremos a cada una de estas formas por separado, resaltando sus diferencias, sus rasgos comunes y principios de funcionamiento. Haremos mayor hincapié el los sistemas bajo plataforma cliente-servidor ya que estos constituyen la aplicación total de lo que es un sistema distribuido aparte de que el concepto cliente-servidor forma parte de los procesos cooperativos y emplea lo que son las bases distribuidas.

3.1 Procesos Cooperativos

Los mainframes, minicomputadoras y redes de área local (LANs) continúan coexistiendo y aumentando día con día. A pesar de ello, la infraestructura existente para soportar estos sistemas de información es cada vez más costosa e ineficiente. Los procesos cooperativos, la interacción de diversos sistemas de computación a través de datos compartidos (bases de datos distribuidas) y aplicaciones cliente/servidor, toman cada día más auge, aunque aun no se ha logrado explotar completamente los alcances de esta nueva tecnología. Hoy en día son muchas las firmas que han involucrado en tareas para el desarrollo de arquitecturas de procesos distribuidos y se espera que para el año 2000 la totalidad de los sistemas de computo del mundo estén operando en plataformas de este tipo.

Una de las características de los procesos cooperativos es que son simbióticos; las aplicaciones lógicas para cualquier operación "viven" en múltiples lugares, y los datos usualmente se encuentran distribuidos de manera tal que puedan atender, en lo mejor posible, a las necesidades de cada aplicación. Por ejemplo, el programa de aplicación A tiene dos componentes, A1 y A2, diseñados de forma tal que se les permita correr en diferentes microprocesadores. A1 realiza su porción de la tarea, y requiere de A2 para que realice su parte del proceso. Aquí encontramos la simbiosis ya que ninguno podrá concluir su trabajo si no recibe ayuda del otro. Ambos procesos son sinérgicos a causa de que los diseñadores de sistemas han posicionado el código de aplicación en el nodo de

computadora mejor equipado a fin de realizar la tarea lo más eficientemente posible o en el nodo de computadora donde residan las bases de datos que serán empleadas por todas o la mayoría de las aplicaciones.

La característica esencial de los ambientes que operan bajo procesos cooperativos es el hecho de que *tanto los datos como las aplicaciones se encuentran dispersas a través de todo el hardware del sistema* existiendo entre ellos una relación de interdependencia.

Los procesos cooperativos son implementados mediante llamadas a procesos remotos (RPC/ *Remote Procedure Call*), los cuales se encuentran soportados en sistemas relaciónales de bases de datos o en sistemas de comunicaciones interactivas tales como APPC, LU2 y LU6.2. Por lo pronto no abordaremos el estudio de los RPC, los lectores interesados en el tema pueden pasar directamente al capítulo 4 en donde se discuten a detalle estos mecanismos de comunicación.

En un principio la maraña existente de técnicas y formas de comunicación entre los sistemas de información de las diferentes firmas constituía una limitante para el desarrollo de los sistemas cooperativos, conforme estas técnicas se fueron estándarizando y regulando las barreras para el desarrollo de los procesos cooperativos se fueron superando y estos se volvieron cada vez más y más poderosos.

3.1.1 PC y WORKSTATION

A lo largo de este capítulo haremos mucha referencia a los términos *estación de trabajo o workstation y a computadoras tipo PC*, por lo que a continuación estableceremos la diferencia entre estos dos sistemas.

Una *computadora tipo PC o PC simplemente*, es una máquina manipuladora de información que cuenta con unidades de almacenamiento local tales como disco duro o floppys, memoria principal (RAM), dispositivos periféricos (impresora, mouse, etc.) pero que no esta especialmente adaptada para trabajar en un ambiente de red, es decir no cuenta con los elementos necesarios para conectarse a una red de computadoras.

Por otro lado una *estación de trabajo* es una máquina que aparte de contar con las características de la PC trae una tarjeta de red integrada, una mayor capacidad de almacenamiento en disco duro, mayor capacidad de memoria RAM y generalmente un monitor de alta resolución.

De lo que hemos venido mencionando se podrá haber notado que cuando hablamos de una estación de trabajo, nos estamos refiriendo al lugar en donde son hechas las solicitudes, es decir donde el trabajo comienza o donde el usuario hace sus peticiones. Algunos autores sugieren que la estación de trabajo o workstation que generalmente es la computadora más pequeña sea llamada "cliente" y el host o la computadora principal sea llamada "servidor". No se tome lo anterior como un estándar ya que en realidad los procesos cooperativos pueden involucrar dos estaciones de trabajo de igual capacidad,

actuando una de ellas como el cliente y la otra actuando como el servidor para una unidad de trabajo. Pero para otra unidad de trabajo el papel se puede intercambiar y los papeles entonces se invertirían, es decir, el cliente podría pasar a ser el servidor y el servidor a ser el cliente o inclusive se podría llegar a un estado bidireccional en el que ambas computadoras actuarán en cada ciclo de una manera determinada.

Tabla 3.1 Diferencia entre PC y Workstation

	PC	WORKSTATION
Existencia de Floppys	En todos los casos	Puede o no tenerlos
Disco Duro	A partir de 20 MB	A partir de 120 MB
Memoria RAM	640K en adelante	2 MB en adelante
Microprocesador	8086, 8088, 286, ...	286, 386, 486, ...
Tipo de monitor	De alta o baja resolución	Por lo general alta resolución
Interconección a red	No existe	Indispensable

3.1.2 El porque de los procesos cooperativos.

Ya hemos descrito lo que son los procesos cooperativos y también hemos señalado el importante crecimiento que estos han experimentado a últimas fechas. Pero ¿que factores han provocado que tal tecnología haya tenido un crecimiento tan impresionante?. Encontramos principalmente tres factores que responden a esta pregunta: *facilidad en la utilización de recursos, productividad e innovación.*

- Facilidad en la utilización de recursos.

A medida que avanza la tecnología los precios de los equipos de cómputo se han vuelto cada vez más bajos, por lo cual no será difícil que en el futuro se cuente con computadoras tipo mainframe en cada escritorio de una oficina. Cosa que actualmente ya ha comenzado a suceder, tomemos como ejemplo el hecho de que con tan sólo una tarjeta que se conecta a la placa madre puede dar aun equipo convencional la capacidad de comunicarse con otras computadoras y por ende compartir sus recursos. Ahora si además está computadora puede comunicarse con computadoras de otros sistemas similares se tendrá una infraestructura robusta de un sistema de cómputo y las ventajas de compartir los recursos, tanto físicos como lógicos, serán invaluables. En sistemas como los descritos encontraremos a los procesos cooperativos en las peticiones que se realizan entre máquinas para la compartición de estos recursos. Futurólogos de notable prestigio tales como el norteamericano Isaac Asimov han llegado a pronosticar que en un futuro no muy lejano, todas las máquinas computadoras, tanto en hogares como en oficinas o centros de investigación, estarán interconectadas entre sí pudiéndose consultar determinada información, por poner un ejemplo, desde la computadora de una casa habitación hasta la computadora de una biblioteca, ya que se contara con multitud de recursos compartidos, es decir, viviremos en un mundo plagado de procesos cooperativos.

- Productividad.

  La tendencia en la industria indica que las terminales tipo mainframe llegarán a convertirse en máquinas tipo PC en un futuro. Lo mismo provocará que cada vez más y más usuarios finales se acerquen a este tipo de computadoras a fin de satisfacer sus requerimientos, provocándose con ello que se tenga una mayor productividad en los negocios y cualquier otro tipo de operaciones. Al estandarizarse los sistemas los programas serán más fáciles de usar y por ende la curva de aprendizaje de los mismos se verá reducida.

- Innovación.

  Mediante los procesos cooperativos, tanto los procesos de aplicaciones y de negocios pueden ser rediseñados de tal manera que puedan ser mucho más efectivos. Por ejemplo, utilizando las capacidades multitareas de una workstation, una aplicación puede comenzar una verificación de datos o acceder a otros procesos, mientras que el usuario consulta información de un banco de datos remoto.

Aparte de las ya mencionadas existen algunas otras razones para considerar la implementación de procesos cooperativos como podrían ser razones de organización. Consideremos que existen muchas compañías que se encuentran separadas en departamentos o sucursales por diversos factores o estrategias, los procesos cooperativos pueden fungir aquí un papel importante manteniendo actualizada la información que se genere en cada una de las computadoras de

la compañía. De la misma manera habrá un ahorro en el costo de las comunicaciones ya que los datos podrán ser almacenados y accesados localmente, con lo cual también se evitará la necesidad de adquirir equipos de cómputo costosos.

3.1.3 Modelo de aplicación de los Procesos Cooperativos.

El modelo tradicional sobre el que se basan las aplicaciones que corren en el mismo procesador consiste de tres componentes como lo muestra la figura:

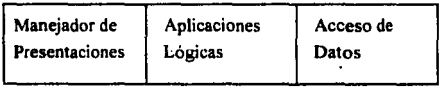


Fig. 3.1 Modelo Stand-alone

Por otro lado las aplicaciones cooperativas, las cuales pueden correr en más de un procesador, se basan en el modelo que muestra la figura 3.2:

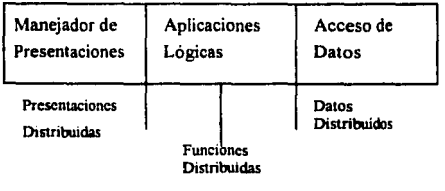


Fig. 3.2 Modelo para Aplicaciones Cooperativas

Tales aplicaciones cooperativas se pueden presentar bajo tres diferentes rubros: *presentaciones distribuidas, funciones distribuidas y acceso a datos distribuidos.*

Presentaciones distribuidas.- Se refiere a la forma en que los datos son presentados al usuario. Generalmente la interface de usuario y un nivel de edición local se encuentran en la estación donde se realiza el trabajo, mientras que las aplicaciones lógicas y los accesos a los datos son principalmente proporcionadas por el host o cualquier otro procesador.

Funciones distribuidas.- Son funciones especializadas que pueden residir tanto en las estaciones de trabajo como en otros procesadores. Las funciones distribuidas ayudan a que los procesos que requieren de datos o aplicaciones residentes en otras máquinas se ejecuten correctamente.

Datos distribuidos.- Las bases de datos sobre las que reside la información en un sistema distribuido son conocidas como bases de datos distribuidas, estos datos pueden ser almacenados tanto de forma relacional o no relacional. De las bases de datos distribuidas hablaremos más adelante.

En la implementación de los procesos cooperativos deberemos atender principalmente a tres factores importantes:

- **La distribución del software**, es de gran importancia debido a que las aplicaciones deberán estar distribuidas sobre los sistemas que las requieran con mayor frecuencia. Las actualizaciones que se hagan sobre los servidores

deberán de reflejarse sobre los clientes ya que de lo contrario se perdería integridad en los datos.

- **Consistencia de datos**, todas las máquinas que funciones dentro del mismo sistema deberán manejar la misma estructura en cuanto a la presentación de los datos a fin de evitar perdidas de información.

- **Operaciones remotas** , deberán funcionar bajo los mismos protocolos de comunicación a fin de que se tenga acceso a todas las aplicaciones del sistema.

3.2 Bases de Datos Distribuidas

3.2.1 Concepto.

Una *base de datos distribuida* es una colección de bases de datos almacenadas en más de un CPU y en la cual los usuarios reciben la información como si se tratará de una gran base de datos, cuando de hecho la información esta compuesta por un conjunto de pequeñas bases de datos. Cada base de datos local es controlada por su DBA (el cual se encarga de realizar mantenimiento, respaldos, indexados, etc.). El DBA (Database Administrator/ Administrador de Base de Datos) es la persona responsable del diseño físico y de la administración de la base de datos además de la evaluación, selección e implementación del DBMS que es el software que controla la organización, el almacenamiento, la recuperación, la seguridad y la integridad de los datos en

una base de datos. El DBMS también es el encargado de aceptar solicitudes de la aplicación y generar las órdenes al sistema operativo para que transfiera los datos apropiados.

3.2.2 Ventajas de los sistemas de bases de datos distribuidas.

Es importante hacer notar que en estos sistemas de bases de datos distribuidos cada CPU corre el software para accesar a las otras bases de datos; no existe un software central el cual pueda coordinar todas las actividades participantes. Esto proporciona la ventaja de dar cierta autonomía y evita una potencial falla del software central.

Un sistema de bases de datos distribuidas incluye una base de datos y herramientas de aplicación, estas últimas pueden ser localizadas en un CPU diferente a los que se hayan colocado las bases de datos.

La mayoría de los sistemas que operan bajo procesos distribuidos ofrecen las siguientes ventajas:

Transparencia de localidad: En un sistema de bases de datos distribuidas puede ser demasiado costoso el cruzar a través de varios datos para localizar una información específica. Para evitar esta perdida de tiempo y de recursos informáticos, se emplean tablas de localidad en las que se encuentran especificados las direcciones de estos datos. Así por ejemplo podemos tener una cierta información que se encuentre particionada en

varias bases de datos (supongamos en México y Nueva York), si un usuario necesita ciertos datos específicos únicamente tendrá que especificar el nombre del asunto o tema que requiere y la tabla de localidad encontrará esta información evitándose de esta manera una búsqueda larga y tediosa. De esta forma encontramos que los dos principales beneficios que nos proporciona la transparencia de localidad son:

- No se necesita conocer o recordar donde se encuentran los datos almacenados
- Los objetos pueden ser movidos sin provocar alteraciones en aplicaciones de otros usuarios.

Autonomía de sitio: La autonomía de sitio significa que cada base de datos que forme parte de un sistema de bases de datos distribuidas sea administrada separadamente e independientemente de las otras bases de datos, los beneficios que incluye esta autonomía son:

- Una mejor resistencia a errores o a perdida de datos,
- Una falla en el administrador central no necesariamente paraliza todo el sistema,
- Cada nodo continua operando aún cuando la red este fuera de servicio,
- Se tiene un control local,

- **Se tiene un diccionario de datos[1] para cada base de datos local**
- **Cada estación o sitio de trabajo puede actualizar su software independientemente de la red.**
- **Los cambios a una base de datos tan sólo afectan a ella misma y sólo necesitan ser validados de manera local.**

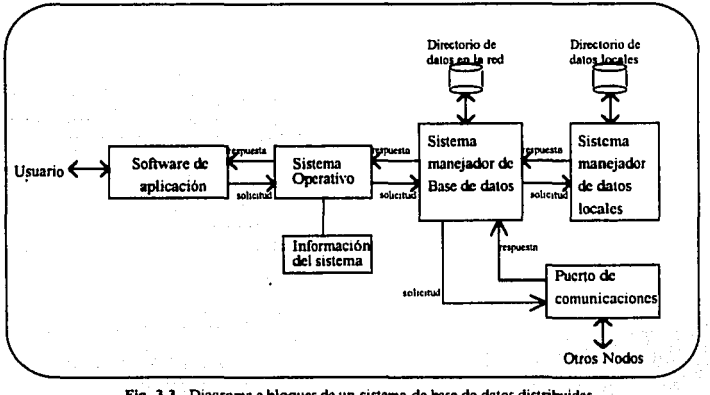


Fig. 3.3 Diagrama a bloques de un sistema de base de datos distribuidas.

[1] Un diccionario de datos es una base de datos acerca de datos y bases de datos. Contiene el nombre, tipo, rango de valores, fuente y autorización para el acceso a cada elemento de datos en los archivos y bases de datos de la organización. Indica también que programas de aplicación utilizan dichos datos de tal manera que cuando se observa un cambio en una estructura de datos, puede generarse una lista de programas afectados. El diccionario de datos puede ser un sistema independiente o parte integral del DBMS utilizado para el control. La integridad y exactitud se garantizan mejor en el último caso.

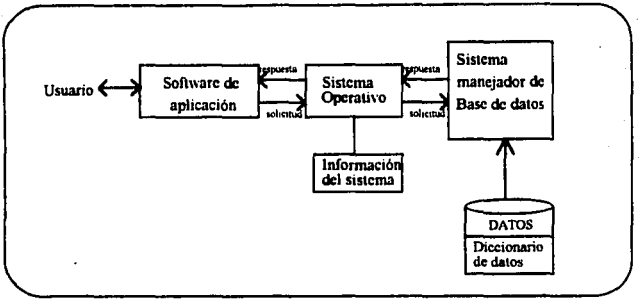


Fig. 3.4 Diagrama a bloques de un sistema de base de datos centralizado.

3.3 La Tecnología Cliente-Servidor

3.3.1 Qué es Cliente-Servidor

El último caso de computación distribuida que estudiaremos es el de cliente-servidor. La razón de que se le halla dejado hasta el último es que cliente-servidor es una tecnología que se sirve de lo que son las bases de datos distribuidas y toma algunos conceptos de los procesos cooperativos.

En una implementación bajo cliente-servidor la aplicación reside en el procesador "cliente" y los datos en la computadora conocida como "servidor". Tanto cliente-servidor como los procesos cooperativos corren bajo el mismo hardware, su diferencia se encuentra en el hecho de que en cliente-servidor las aplicaciones se encuentran, de manera completa, en un sólo CPU; mientras que en un proceso cooperativo tanto las aplicaciones como los datos pueden estar

dispersas en cada máquina a lo largo de todo el sistema. De esta manera las aplicaciones lógicas se ejecutarán solamente en el propio procesador cliente, mientras que el servidor realizará funciones tales como las de manejo y administración de archivos y datos. El cliente tiene acceso a estos datos a través de un sistema de red.

Para decirlo con pocas palabras, en un ambiente computacional con cliente-servidor, los empleados utilizan sus PC, conocidas como clientes, para ejecutar software de productividad personal, como hojas de cálculo. Si, por ejemplo, necesitan datos sobre la empresa, solicitan acceso a un servidor. Podrían necesitar estos datos y la hoja de cálculo para analizar en que zonas tiene mayores ventas la empresa.

El software de productividad en grupo corre tanto en servidores como en máquinas de escritorio. Un fabricante podría tener software especial para que los empleados que solicitan suministros tengan acceso a los sistemas de facturación e inventario que corren en el servidor.

Otro software de productividad en grupo hace posible que los empleados intercambien información y documentos. Por ejemplo, los compañeros de trabajo podrían obtener copias electrónicas de los análisis de ventas por regiones y un informe sobre el inventario.

Antes, el software para realizar labores relacionadas con la contabilidad, el presupuesto y la administración de proyectos se encontraba todo en una mainframe enorme y costosa. Ahora, esas mismas aplicaciones están a disposición de los usuarios de LAN en una instalación cliente-servidor.
En computación con cliente-servidor, una computadora poderosa actúa como la central de conexiones de un grupo de usuarios que necesitan trabajar conjuntamente. En el nivel más bajo del ambiente cliente-servidor podemos usar un servidor Compaq ProLiant en una red Net Ware para que grupos de hasta 50 usuarios compartan archivos e impresoras.

A diferencia del software para un sólo usuario, el software para un ambiente cliente-servidor debe dar servicio tanto a personas en lo individual como en grupo. Asimismo, debe tener gráficas y características que simplifiquen su uso. como las que la gente espera encontrar en los programas para PC.

3.3.2 Clientes y Servidores.

La costumbre dicta que el cliente sea una microcomputadora, pero un cliente bien puede ser una workstations, un mainframe o cualquier otro tipo de máquina. Esta tecnología fue la pionera en su tipo y es también la que ha alcanzado mayor desarrollo y aceptación. En algunos ejemplos de operación bajo cliente-servidor, un subconjunto de aplicaciones o datos (tales como las tablas de edición) pueden estar residentes sobre la máquina cliente sin llegar

a invadir la premisa básica del modo de procesamiento. Claro esta que si la máquina cliente es una computadora personal, esta seguirá corriendo aplicaciones comunes tales como procesadores de texto u hojas de cálculo sin interrumpir en los aspectos de la red. Entre las funciones de clientes y servidores observamos importantes diferencias como se discute a continuación:

a) El papel del cliente

En una arquitectura cliente-servidor el cliente es de manera primaria un consumidor de servicios proporcionados por uno o más servidores. La arquitectura provee una clara separación de funciones basadas en la idea de que el objetivo principal del servidor es "servir" en lo mejor posible al cliente. El cliente puede solicitar los servicios de un determinado servidor, sin importar de si éste esta en la misma red que el cliente o en una red remota. La forma en que se realiza esta comunicación entre servidor y cliente es mediante llamadas a RPC.

Existen variedad de servicios de que el cliente puede disponer. Por ejemplo, los clientes pueden manejar servicios de ventanas que proveen la habilidad para activar, mover, cambiar tamaño u ocultar una ventana en particular. Estos servicios son esenciales en una implementación cliente-servidor debido a que estos interactúan con mensajes que son enviados por el servidor a fin de mantener informado al cliente sobre el estado del sistema. Las aplicaciones que se corren en el cliente generalmente utilizan software GUI (Grafical User Interface), un ambiente común en estas aplicaciones es Windows NT.

Los clientes también se comunican con los servidores a través de un conjunto de servicios de red que crean, envían y reciben mensajes, entre ellos se encuentran Protocolos de Control de Transmisiones/ Protocolos Internet, protocolos IPX (Internetwork Packet Exchange) y Sistemas Básicos I/O de redes (Network Basic I/O System).

b) El papel del servidor

Los servidores son aquellas partes del sistema que se encargan de satisfacer los pedidos hechos por un cliente. En una arquitectura cliente-servidor estos servicios pueden ser proporcionados para realizar una compleja tarea o solamente una parte de ella. Un conjunto de servidores pueden trabajar juntos a fin de proveer una funcionalidad mayor y atender una tarea en el menor tiempo posible. Estos servidores pueden correr cada uno con diferentes sistemas operativos de red (NOS/Netwok Operating System) y sobre diferentes plataformas de hardware. El cliente invocara estos servicios sin considerar la tecnología que emplee cada servidor.

En un escenario cliente-servidor, los servidores atienden a las solicitudes hechas por los clientes a el NOS shell, el cual se encuentra residente sobre el mismo cliente y es la parte encargada del establecimiento de comunicación entre cliente y servidor. El NOS shell transforma la solicitud hecha por el cliente a manera de que pueda ser enviada a través del sistema hacia el servidor, siguiendo una serie de protocolos previamente establecidos. Una vez que el mensaje se encuentra en el servidor, éste es transformado de nueva cuenta y procesado, enviándose una respuesta al cliente.

El servidor también provee los servicios de archivo que manejan el acceso a los directorios virtuales y a los archivos localizados sobre el servidor . Para minimizar los costos de instalación y mantenimiento de software, muchas aplicaciones son recuperadas desde el servidor y ejecutadas en el cliente. De esta manera las actualizaciones se hacen en el servidor teniéndolas disponibles en el cliente de manera inmediata.

Los fabricantes de software para cliente servidor tales como Sybase, Inc´s SQL Server, ORACLE Corp.´s, Grupta Corp.´s SQLbase, Informix Software e Inc.´s Informix proveen soporte al servidor para ejecutar directamente aplicaciones SQL provenientes de los clientes.

Los servicios de conexión a través de sistemas LAN/WAN tales como puentes, ruteadortes y gateways son proporcionados por el NOS. También los servidores se encargan de proporcionar seguridad al sistema solicitándole a un usuario su login y password para tener acceso al mismo.

En una red LAN cuando la computadora personal accede a una base de datos localizada en un disco compartido, la base entera es recuperada y cargada en la memoria principal de la PC. Esto significa que si se desea desplegar información a cerca de un tema en particular, se tendrá que hacer una búsqueda de este tema a través de toda la información que haya sido enviada a la PC y si la memoria de la máquina es demasiado pequeña también se correrá el riesgo de tener perdida o inconsistencia de datos.

Por otro lado en un ambiente cliente-servidor, el tema requerido será buscado sólo en una base de datos dedicada o en un servidor y solamente la información concerniente a éste será enviada a la computadora personal. Esta técnica incrementa la velocidad y eficiencia a causa de que menor cantidad de información es transferida sobre la red.

Un sistema basado en cliente-servidor tendrá cuatro componentes básicos: *los procesos de aplicación*, los cuales comúnmente interactúan entre dos o más plataformas de hardware; *el sistema manejador de base de datos*; *los componentes de hardware* que típicamente son pequeñas máquinas, tipo PC, y *uno o varios mainframe y componentes de la red*, los cuales consisten en cableado, trayectorias o rutas y protocolos. Estos últimos se analizaron ya en el primer capítulo de esta obra, de las bases de datos distribuidas se habló anteriormente y ahora sólo resta hablar de otro de los mecanismos esenciales de cliente-servidor: los procesos distribuidos.

3.3.3 Procesos Distribuidos

Un proceso distribuido ocurre cuando una aplicación en un CPU accesa a una base de datos en otro CPU. En otras palabras, si tenemos una aplicación que esta corriendo en nuestra computadora, pero en determinado momento necesitamos información que en nuestra máquina no esta disponible, podemos hacer uso de los servicios de la red y enlazarnos o conectarnos a otra computadora, en la que si este esta información disponible, para leer la

información necesaria y traerla a nuestra máquina. De esta forma un proceso distribuido es la ejecución de uno o más procesos o programas, independientes entre sí, en diferentes estaciones de la red.

Algunos de los beneficios de los procesos distribuidos son:

- Permite utilizar el hardware más apropiado para un trabajo específico. Por ejemplo las computadoras personales (PC's) pueden ser usadas para obtener un procesamiento rápido y barato, mientras que las minicomputadoras o los mainframe pueden ser usados para correr aquellos procedimientos que requieran accesos rápidos de entrada/salida y gran capacidad de espacio libre en disco para almacenar los datos provenientes de las PC's..

- Permite apoyar a un mayor número de usuarios con un hardware más barato. Mayor número de procesos pueden ocurrir en una PC, y la capacidad de procesamiento puede ser incrementada al comprar más PC's como parte del hardware para el mainframe.

- Permite realizar en un mismo tiempo, tanto el procesamiento de los datos como la administración de estos. De esta manera, los datos pueden ser administrados por el DBA, mientras que las aplicaciones y herramientas de procesamiento son controladas localmente por el usuario de cada estación de trabajo.

Mucha gente en el mundo de la informática acostumbra a confundir los términos *procesos cooperativos* y *procesos distribuidos* siendo que los segundos son realmente una variación de los primeros. Hablamos de procesos cooperativos cuando se comparten tareas y aplicaciones entre dos o más procesadores y en donde por lo menos un procesador es una workstation. Por otro lado un proceso distribuido es la dispersión de recursos y funciones a través de dos o más sistemas de cómputo. En un proceso cooperativo se tiene una función de interdependencia entre las aplicaciones, es decir, un programa que se ha ejecutado en una máquina necesita de los resultados generados por otro programa que se ejecutó en otra máquina a fin de poder continuar. En un proceso distribuido se tienen acomodadas las aplicaciones de tal manera que se ubiquen en donde más frecuentemente se les requiere, las cuales pueden ser ejecutadas, cada una, de una manera libre e independiente de otras.

3.3.4 El papel de procesos distribuidos y bases de datos distribuidas en tecnología cliente-servidor.

Hemos explicado hasta el momento lo que son las bases en que se apoya la tecnología cliente servidor. La anterior discusión nos servirá como fondo para comprender más enteramente lo que es un sistema formado por clientes y servidores. Recuérdese que en un sistema de bases de datos distribuidas se tiene dos o más computadoras en las que se encuentra almacenada la información y el resto de las computadoras (también llamadas estaciones de

trabajo) acceden a la información almacenadas en las primeras; a partir de esto se deriva lo que es el concepto de servidor y cliente respectivamente. Realmente un sistema de bases de datos distribuidas esta conformado por clientes y servidores.

De esta manera un cliente puede ser una PC, una minicomputadora o un mainframe que accede a cierta información que se encuentra almacenada en otro nodo. Un servidor es cualquier nodo con la base de datos de la cual estos datos pueden ser requeridos. Los servidores deben funcionar con sistemas operativos multiusuarios. En un nodo se pueden tener muchas transacciones ocurriendo a un mismo tiempo; estas pueden ser de un cliente o de algún servidor.

Aunque en sistemas muy sofisticados de plataformas cliente-servidor podemos tener aplicaciones ejecutándose mediante la técnica de los procesos cooperativos, *lo común para cliente-servidor es que se empleen procesos distribuidos en sus operaciones*, ya que los servidores son sólo el banco de datos de donde se toma la información y por lo mismo, no constituyen una parte del proceso que sea neccsaria para que la aplicación que se corre en el cliente pueda continuar antes de seguir a otra.

Un cliente es el encargado de ejecutar las aplicaciones. Este envía sentencias (generalmente en algún tipo de SQL) de petición a través de la red a el servidor. El servidor recibe y ejecuta las instrucciones recibidas y devuelve los datos pedidos por el cliente. Una máquina cliente no es capaz de interpretar

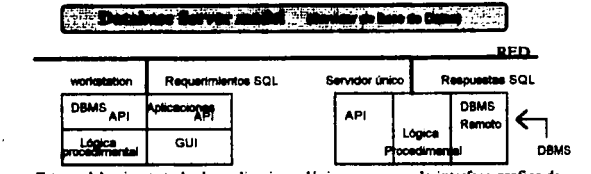
instrucciones en SQL ya que el cliente no cuenta con el diccionario de datos ni con el núcleo o kernel necesarios para interpretar las instrucciones, por ello el servidor debe enviar instrucciones claras y precisas al cliente.

3.3.5 Los Cuatro Modelos Básicos de Cliente-Servidor.

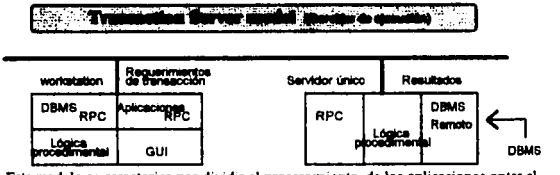
En cualquier arquitectura cliente-servidor, los sistemas clientes y servidores son construidos a partir de un número determinado de componentes interconectados. La coordinación y cooperación entre los componentes es proporcionada por funciones sincronizadas corriendo tanto en el cliente como en el servidor. Las aplicaciones comerciales pueden ser visualizadas como un conjunto de funciones componentes para el acceso de datos, aplicaciones lógicas e interface de usuario.

Existen cuatro modelos para procesar las transacciones en una arquitectura cliente servidor; el seleccionar la opción que más se adecue a las necesidades de determinada institución no es tarea fácil aún para el más experimentado diseñador de sistemas. Estos modelos son, respetando sus nombres en inglés: el modelo Database Server, el modelo Transaction Server, el modelo Peer to Peer y el modelo Distributed Presentation los cuales se ilustran en la figura 3.5.

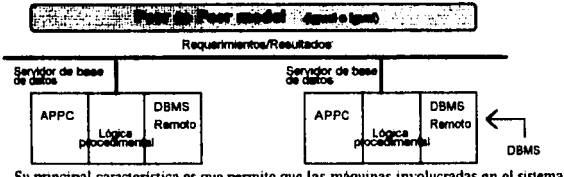
Cuatro Caras de Cliente Servidor



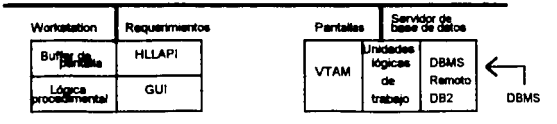
Este modelo ejecuta todas las aplicaciones lógicas y procesa la interface grafica de usuario (GUI) sobre la máquina cliente, mientras los requerimientos del DBMS son procesados y ejecutados en el servidor.



Este modelo se caracteriza por dividir el procesamiento de las aplicaciones entre el cliente y el servidor.



Su principal característica es que permite que las máquinas involucradas en el sistema puedan desempeñar papeles de clientes o de servidores, según la aplicación que procesen.



Se caracteriza por aprovechar todas las características de las aplicaciones residentes en la máquina host a través de una red LAN basada en aplicaciones GUI.

Fig. 3.5 Las cuatro caras de cliente-servidor

El modelo Database Server (servidor de base de datos) representa la forma más común de un procesamiento cliente-servidor, y la mayoría de las aplicaciones existentes probablemente caerán dentro de esta categoría. Dentro de este modelo todas las aplicaciones lógicas y las funciones de usuario son ejecutadas en el cliente. Las funciones a acceso de datos se ejecutan por lo común sobre el servidor, estos datos estarán implementados generalmente bajo un esquema relacional del DBMS[2].

Este modelo procesa solo los requerimientos basados en SQL -cualquier otra función que no sea basada en SQL se ejecutará en la máquina cliente. Los clientes se comunican con los servidores vía API (Aplications Program Interface/ Interfaz de programa de Aplicación) el cual es un lenguaje y formato de mensaje utilizados por un programa para activar e interactuar con las funciones de otro programa o de hardware. Este API formatea y envía la sintaxis SQL al servidor para su procesamiento. Comúnmente los vendedores de software para DBMS proporciona al API bajo la forma de un módulo de librería ejecutable.

El modelo database server puede residir sobre varias plataformas de hardware y soportar numerosos sistemas operativos. Por ejemplo, Oracle's APIs permite a los usuarios comunicarse con sistemas que operen bajo OS/2, Unixbased, VAX VMS y ambientes de servidores IBM MVS. Un modelo cliente-servidor de este tipo es muy aceptado en sistemas de información para ejecutivos, en

[2] El DBMS (DataBase Management System/ Sistema de administración de base de datos) es el software que controla la organización, el almacenamiento, la recuperación, la seguridad y la integridad de los datos en una base de datos. Acepta solicitudes de aplicación y genera las órdenes al sistema operativo para que transfiera los datos apropiados.

sistemas de toma de decisiones y para sistemas en los que en general se tenga un bajo tráfico de entrada y salida de datos. El principal problema que presenta este modelo son los bajos tiempos de respuesta, los cuales se deben principalmente a limitaciones en su arquitectura.

Una alternativa al modelo database server es el modelo transaction server (modelo de ejecución), el cual divide a las aplicaciones lógicas entre clientes y servidores, con la característica de que la mayor parte del procesamiento de la aplicación ocurre sobre el servidor. Este modelo utiliza unidades lógicas de trabajo, las cuales son demandadas por el cliente y ejecutadas por el servidor. Es altamente recomendable en sistemas donde se requiera realizar transacciones de índole compleja o que requieran de una intensiva manipulación de datos. Este modelo permite que los mensajes a ser enviados al servidor vayan acompañados tanto por datos como por instrucciones de manipulación de los mismos y sólo el resultado es devuelto al cliente.
Este modelo también tiene las ventajas de que permite balancear la carga de trabajo entre clientes y servidores con lo cual se reduce el tráfico de la red. Otra ventaja es que para aplicaciones de gran magnitud se permite dividir las tareas de tal manera que mientras que en el servidor se encarga de procesar los datos requeridos, el cliente se encargara de dar formato a estos datos a fin de ser presentados.

Encontramos tres maneras principales para construir aplicaciones cliente-servidor: **llamadas a procesos remotos, procedimientos almacenados y**

procesos de transacción en línea. Las llamadas a procesos remotos (RPC) son fundamentales en la computación distribuida y son el punto de partida de los procedimientos almacenados y de los de transacción en línea. Sistemas tales como Inc.'s Netware Core Protocol (NCP) de Novell y Inc.'s Network File System de Sun Microsystems fueron diseñados basándose a llamadas a RPC's. En aplicaciones DBMS se utiliza la tecnología de procedimientos almacenados. Estos procedimientos son una colección de sentencias SQL y de directivas de control de flujos de datos, tales como *IF, THEN, ELSE*; los cuales son verificados, compilados, ligados y almacenados en el servidor del DBMS. Los procedimientos aquí almacenados pueden llamar a otros procedimientos y ser ejecutados sobre servidores remotos. Las sentencias en SQL pueden ser estáticas, lo cual significa que estas son verificadas, ligadas y almacenadas en el servidor del DBMS antes de que sean ejecutadas; así mismo también pueden ser dinámicas, es decir, las sentencias SQL son tratadas conforme se vayan ejecutando, aunque con esta última técnica el desempeño general del sistema puede decrementar drásticamente.

El modelo peer-to-peer (igual a igual) es similar al modelo de transacción y ofrece muchas de las ventajas de este modelo. El modelo Peer además ofrece la ventaja de que permite, como su nombre lo indica, establecer una similitud de papeles entre sus componentes. Esto significa que las máquinas que lo integran pueden fluctuar entre ser clientes o servidores según la aplicación que procesan. El cliente y el servidor se comunican mediante el empleo de técnicas

de comunicación de programa a programa, por TCP/IP o mediante protocolos NETBIOS.

El modelo igual a igual es recomendable cuando las aplicaciones deben responder a eventos externos que no queden bajo el control del usuario, tales como respaldos automáticos programados. Este modelo fue diseñado para proveer a los programas con funciones de soporte a través de comunicaciones programa a programa; así mismo el modelo también provee capacidad para efectuar procesos de transacciones distribuidas (DTP/ Distributed Transaction Processing), donde dos o más programas, residentes en diferentes plataformas de hardware , cooperan a fin de realizar una determinada tarea.

Por último el modelo de presentación distribuida (distributed presentation) tiene la gran ventaja de que posibilita al usuario ligar, desde principio hasta el fin, aplicaciones LAN basadas en otra aplicación existente en el host, sin modificar el código residente en el mismo host. Este modelo emplea interfaces GUI, es decir, que se trabajará en la máquina cliente en ambientes gráficos similares al ambiente windows. Ejemplo de esto es el servidor IBM 3270 (diseñado específicamente para trabajar en modo de presentación distribuida de cliente-servidor), el cual permite que desde el cliente se diseñen menús del tipo pull-down, cajas de dialogo, botones de comandos, etc., todo ello como parte de una aplicación.

Así estos cuatro modelos forman las cuatro caras en las que se puede presentar un ambiente basado en cliente-servidor. Comparando estos modelos con las

topologías vistas en el capítulo uno se habrá notado como la infraestructura de las redes ha evolucionado drásticamente y ha llegado a ser cada día más compleja. La arquitectura cliente-servidor deberá ser manejada desde una perspectiva empresarial. El cambio de un modelo de red de computadoras a otro no es sencillo ya que no existen aún normas claras para este caso. El integrar diferentes tecnologías de múltiples vendedores requiere un tremendo esfuerzo. En su mayoría las herramientas para construir multiprotocolos no existen. Por ello, el elegir el sistema de computadoras que más se necesite debe ser una tarea planeada y que involucre los esfuerzos de todos aquellos que se servirán de la misma.

3.3.6 El administrador de base de datos en cliente-servidor.

En un sistema cliente-servidor encontramos que el DBA juega un papel muy importante. Ya hemos hablado de las características que debe presentar el administrador de base de datos (ver sección 3.2). El DBA dentro de un sistema de bases distribuidas es el encargado de asignar responsabilidad para cada base de datos participante de tal manera , como si se tratara de una base de datos independiente. De echo, un sólo DBA puede ser responsable de todas las bases de datos o cada base de datos pude tener su propio DBA, si se tratara de múltiples DBA´s, cada uno de estos deberá estar coordinado con el otro principalmente en aquellos aspectos que se refieran al sistema operativo de la red.

Cuando se tienen múltiples CPU's conectados a través de una red con la intensión de permitir a usuarios y aplicaciones acceder a diferentes nodos, entonces se recomienda tomar en cuenta algunas consideraciones para el DBA. Entre estas consideraciones encontramos a las siguientes:

- Se deberá de ser capaz de tomar decisiones acerca de la instalación (dónde se instalará, qué software y a qué máquinas se tendrá acceso).
- Tomar decisiones acerca de la integridad y seguridad de los datos al cruzar de un lado a otro de la red.
- Garantizar el buen funcionamiento del sistema en todos los nodos a través de la red.

También es importante mencionar que en un ambiente de procesos distribuidos, la decisión de donde colocar a el software de aplicación deberá ser independiente del lugar donde los datos serán almacenados, en otras palabras, software y datos deberán estar en CPU's distintos. Un factor importante es escoger el CPU donde residirá nuestro software, para tomar tal decisión el administrador del sistema deberá considerar diversos factores tale como son el tiempo de respuesta que requiere el sistema, las aplicaciones que serán más usadas y el espacio libre con que cuente la máquina para la instalación de nuevas aplicaciones.

En un sistema de bases de datos distribuidas, los DBA's tiene mucha más flexibilidad para decidir en donde se almacenarán los datos; el único hecho que

se debe tener en cuenta es que las máquinas en las que se encuentre la información deberán ser capaces de soportar los mecanismos necesarios de seguridad, esto es, que un usuario pueda tener acceso a los datos que necesite, pero que no pueda acceder a más de los que él requiere.

Si se tiene una buena distribución de los datos y las aplicaciones, los beneficios serán múltiples:

- Reducción del tráfico de red,
- Mejor tiempo de respuesta,
- Reducción en la redundancia de datos
- Control local de datos y
- Abaratamiento de los costos al acceder a bases de datos locales o cercanas en lugar de bases de datos remotas.

3.3.7 Los componentes de un Sistema Cliente-Servidor.

Hemos dado ya una definición de tecnología cliente-servidor, en este apartado vamos a completar la definición ya dada y a analizar los elementos componentes de un sistema que opere bajo cliente-servidor. Una definición completa de cliente-servidor debe incluir a los siguientes aspectos:

1. Protocolos de comunicación y facilidades de red que pueden ser obtenidos al establecer el cliente y el servidor.
2. La manera en que los programas de aplicación generan los servicios que les son requeridos. Esto es normalmente en la forma de un programa de interface de aplicación (API/Application Program Interface).

3. Una explicación de la manera en que los servidores reciben y responden a los requerimientos de los clientes, lo cual es comúnmente un API.
4. Un conjunto de explicaciones de los requerimientos del sistema (tanto de software como de hardware), que especifiquen las herramientas necesarias para correr cualquier aplicación en la arquitectura definida.

Teóricamente, hay un número infinito de posibilidades para implementar un sistema con arquitectura cliente-servidor. En general estas arquitecturas deberán enfocarse a tener lo siguiente:

- Ser diseñados de una manera transparente a fin de permitir la operación, locales que puedan ser direccionadas hacia servidores remotos para poder utilizar servicios tales como la impresora o aplicaciones del servidor accesado.
- Tener las facilidades y especificaciones que describan como establecer un enlace de comunicación entre dos o más sistemas, a través de las cuales se pueda desarrollar una relación de cliente-servidor; como por ejemplo el sistema CPIC (Common Programming Interface for Communications).
- Especificaciones para la distribución de los elementos de un programa individual entre múltiples sistemas, a manera de que los servicios proporcionados por un servidor sean utilizados por todos ellos y como si se tratara simplemente una parte local de la aplicación del cliente. Esto es lo básico en el desarrollo de cliente-servidor.

3. Una explicación de la manera en que los servidores reciben y responden a los requerimientos de los clientes, lo cual es comúnmente un API.
4. Un conjunto de explicaciones de los requerimientos del sistema (tanto de software como de hardware), que especifiquen las herramientas necesarias para correr cualquier aplicación en la arquitectura definida.

Teóricamente, hay un número infinito de posibilidades para implementar un sistema con arquitectura cliente-servidor. En general estas arquitecturas deberán enfocarse a tener lo siguiente:

- Ser diseñados de una manera transparente a fin de permitir la operación, locales que puedan ser direccionadas hacia servidores remotos para poder utilizar servicios tales como la impresora o aplicaciones del servidor accesado.
- Tener las facilidades y especificaciones que describan como establecer un enlace de comunicación entre dos o más sistemas, a través de las cuales se pueda desarrollar una relación de cliente-servidor; como por ejemplo el sistema CPIC (Common Programming Interface for Communications).
- Especificaciones para la distribución de los elementos de un programa individual entre múltiples sistemas, a manera de que los servicios proporcionados por un servidor sean utilizados por todos ellos y como si se tratara simplemente una parte local de la aplicación del cliente. Esto es lo básico en el desarrollo de cliente-servidor.

3.3.8 Conexión a sistemas externos.

Ya hemos visto lo que es y como actúa un sistema bajo el concepto de cliente-servidor. En un sistema bajo este concepto una computadora cliente puede solicitar información de un banco de datos que se encuentre en otra ciudad y recibir la información requerida en un lapso muy breve. Esta computadora cliente puede formar parte de una red local común, como podría ser una red con topología en bus, y mediante algún dispositivo, como por ejemplo un módem, ponerse en contacto con otra red en donde se encuentre la computadora servidor a fin de solicitar la información deseada. Pero hay ocasiones en que se tiene que establecer contacto con otro servidor que podría estar, por ejemplo, en el mismo edificio o en un lugar muy cercano al de la computadora que hace una petición (cliente) y a su vez este servidor puede formar parte de una red local con diferente topología, para estos casos existen aparatos conocidos como conectores y que tienen el propósito de ayudar a que se "entiendan" estos sistemas. Los principales son:

Repetidores: Básicamente se trata de amplificadores de señal que permiten sobrepasar las longitudes máximas del cable definidas por el fabricante. Su única limitación de uso es el número que se puede instalar por tramo de cable.

Puentes (bridges): Permiten interconectar dos o más redes que empleen el mismo sistema operativo. Por ejemplo, Ethernet con Ethernet. Como su función es simplemente desviar el tráfico de la información de una red a

otra, no utilizan técnicas complejas y su instalación es sencilla. Existen puentes que se comportan de forma transparente y pasan cualquier información que detectan en un cable a otro. Sin embargo, también los hay que pueden "aprender" en que red está instalada cada estación y filtrar la información, de forma que sólo pase la información con destino a las estaciones instaladas en esa red.

Encaminadores (routers): Los encaminadores permiten interconectar redes locales diferentes o iguales entre sí. A efectos de direccionamiento y tráfico cada red tiene identidad propia y diferente. Los encaminadores se instalan en una de las estaciones de la red y, formados por hardware y software, convierten los protocolos de una red a los de otra.

Pasarela o puerta de acceso (gateway): Permite conectar una red local a una red de otro tipo. Por ejemplo, conectar una red Ethernet a un host 3090 de IBM. La pasarela es, normalmente, una de las estaciones de trabajo de la red. Esta estación está conectada a ambas redes, mediante el hardware y el software necesario convierte los protocolos de ambas redes, de forma que los usuarios de una red vean a los de la otra como si estuvieran conectados a la suya.

Servidor de comunicaciones. Es una de las estaciones de la red que se encarga de gestionar las comunicaciones del resto de los usuarios de la red. Los

servidores de comunicaciones permiten compartir módems, tarjetas fax, etc.

3.4 Revisión.

Para terminar este capítulo resaltaremos los puntos más importantes de la computación distribuida y de las tres áreas que la forman. Un sistema distribuido trata de obtener lo mejor de dos mundos: el de los mainframes y el de las PC's. Los mainframes son muy poderosos pero poco necesarios si la tarea que se tiene a la mano es muy simple, por ejemplo, el actualizar registros individuales puede ser mucho más fácil en una estación de trabajo que sobre un mainframe, pero el poder de un mainframe es indispensable si lo que se desea es realizar un complejo procesamiento por lotes. De cualquier manera, ambos sistemas tienen sus pros y sus contras, por lo que si tomamos lo mejor de cada uno y lo ensamblamos en otro sistema obtendremos lo que se conoce como un sistema distribuido.

La computación distribuida no es otra cosa mas que *la dispersión de los recursos de computación o las funciones de estos entre dos o más sistemas.* Tales sistema se pueden presentar de tres maneras distintas: Procesos cooperativos, Bases de datos distribuidas y tecnología cliente-servidor. Aunque todas ellas básicamente persiguen el mismo objetivo que es el tener el poder de procesamiento de los sistemas grandes pero con la facilidad de manejo de un sistema pequeño y el poder compartir información en el momento que esta sea

requerida, tienen algunas diferencias como lo mencionan las siguientes definiciones:

- Procesos Cooperativos.- Pueden ser definidos como la compartición de la ejecución de una tarea entre dos o más procesadores, uno de los cuales es generalmente una estación de trabajo. Sobre un procesador una aplicación puede ser escrita y sobre otro u otros puede ser ejecutada.

- Bases de datos distribuidas.- Forman parte de los procesos cooperativos y de las aplicaciones cliente-servidor; básicamente son un conjunto de datos almacenados sobre varías computadoras y distribuidos de tal manera que tengan una congruencia entre sí, y ubicados sobre las computadoras que más frecuente uso hacen de ellos.

- Tecnología cliente-servidor.- Es una técnica parecida a los procesos cooperativos que provee las mismas facilidades pero sobre un nivel más localizado, tal como una empresa o un departamento dentro de esta empresa. El cliente es la computadora que hace la petición y el servidor la que da la respuesta; en el cliente se encuentran almacenadas las aplicaciones y en el servidor los datos y archivos secundarios. Esta variante de los sistemas distribuidos es la que más auge ha encontrado en las modernas empresas del mundo de hoy, debido precisamente a que su carácter de local la han hecho accesible a diversas corporaciones, con lo que pueden compartir recursos y datos sobre cada máquina de la institución.

Capitulo 4

Llamadas a Procesos Remotos (RPC).

4.1 RPC´s

4.1.1 ¿Qué son los RPCs?

Uno de los principales problemas a los que se tuvieron que enfrentar los programadores de los primeros sistemas de computación distribuida, fue a la ausencia de una plataforma que les permitiera escribir software para los sistemas distribuidos. Una plataforma es un conjunto de rutinas, comúnmente agrupadas en una librería, que provee funciones específicas y sobre la cual se puede construir una aplicación. Idealmente una plataforma para un sistema distribuido no debe ser específica de un cierto sistema operativo o de cierto hardware, ya que debe funcionar bajo diferentes tipos de estos. La portabilidad de una plataforma incrementa la portabilidad de la aplicación que usa la plataforma. De esta manera, la búsqueda de una plataforma sobre la cual se pudieran crear aplicaciones distribuidas dio origen a los RPCs.

Los RPC´s (Remote Procedure Calls/Llamadas a Procesos Remotos) son el mecanismo básico de comunicación para los ambientes que operan bajo el concepto de computación distribuida. Estos forman un mecanismo estándar para transferir el código a un servidor y regresar el resultado a un cliente. Su ejecución puede ser tanto de manera local (cuando se comunican entre sí

máquinas que se encuentran en un mismo lugar) como remota (comunicación entre máquinas que se pueden ubicar en diferentes instalaciones o poblaciones).

Los RPC dan la posibilidad de hacer una llamada a un procedimiento que no reside en el espacio de direcciones del proceso llamante. El procedimiento llamado se puede ubicar en la misma computadora en la que se encuentra el procedimiento que llama, o en una máquina diferente. Una llamada a un proceso remoto es análoga a un salto remoto a una subrutina llamada. La figura 4.1 muestra como una aplicación local hace una solicitud a un proceso remoto pasándole ciertos parámetros y el control de la aplicación a fin de que lleve a cabo cierto proceso y regresándose después los resultados.

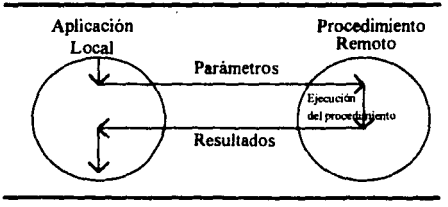


Fig. 4.1 Una ejecución remota

Los RPC's conforman un mecanismo de llamadas a funciones para interprocesos de comunicación locales o remotos. Este mecanismo funciona bajo un esquema de paso de mensajes, en el cual los argumentos de un procedimiento son enviados a un proceso remoto, retornando este procedimiento los resultados generados al primer procedimiento.

El usuario no necesitará saber como los mensajes son enviados o recibidos ya que el mecanismo de los RPC´s proveen todo lo necesario para ello.

La idea básica de una llamada a un proceso remoto es extender el uso de las llamadas a procedimientos en los ambientes de computación distribuida. Visto de una manera muy simple, cuando un procedimiento remoto es invocado, el procedimiento llamante es suspendido, y sólo un mensaje conteniendo algunos argumentos es pasado a la máquina remota, donde es ejecutado un nuevo procedimiento. Cuando el procedimiento termina, los resultados son regresados a la máquina llamante restaurándose el proceso que había quedado suspendido y su ejecución continua como si todo se hubiera hecho sobre la misma máquina.

Los RPC son considerados como el mecanismo principal de la computación distribuida a causa de que son simples, familiares (debido a su similitud con llamadas locales), generales (ya que consideran a las aplicaciones que son implementadas usando llamadas a procedimientos locales como el mecanismo primario de control) y a que pueden ser implementados eficientemente.

Los RPC´s pueden simplificar la construcción de programas distribuidos ya que resumen los detalles de comunicación, transmisión de errores y fallas entre las máquinas. Idealmente una llamada remota debería tener la misma semántica que una llamada local, a fin de que la naturaleza distribuida del programa no afectara su funcionalidad (lo cual traería un aumento en el desempeño). En la práctica, este ideal es difícil de alcanzar. Una razón para esto es que es difícil ocultar las fallas completamente; de esta forma las diferencias en los tipos de datos y las necesidades para enviar códigos especiales en mensajes (como el

paso de un puntero a una pila) comúnmente resultan en RPC´s que llevan parámetros extras lo que los hace diferentes a sus similares locales.

Si el RPC encuentra que se desea redireccionar la información que se encuentra enviando o recibiendo, entonces se solicita la ayuda de un servidor y un redirector. Un redirector es un programa elemental que se encuentra residente en la memoria de un sistema cliente y que intercepta los requerimientos hechos por programas de aplicación. Cada intersección es examinada por el redirector a fin de determinar si esta se aplica a un recurso local o remoto. Si se tratase de un requerimiento local se regresa al modo normal de operación y se continua la transmisión como estaba. Si se trata de un requerimiento remoto, éste será enviado a través de la red a un servidor a fin de que la llamada quede satisfecha.

En la figura 4.2 se observa como una función llama a una subrutina en una máquina diferente como si se tratara de una subrutina local. Aquí el código RPC utilizando Stubs, cuya función se verá más adelante, se encarga de establecer la comunicación entre las dos máquinas a través de la red, enviando los requerimientos y regresando resultados. Los RPCs crean una función dedicada entre dos componentes de una red, de manera similar a como dos PCs se comunican a través de un sistema de bus.

Los RPCs permiten a las aplicaciones ejecutar instrucciones sobre una máquina remota

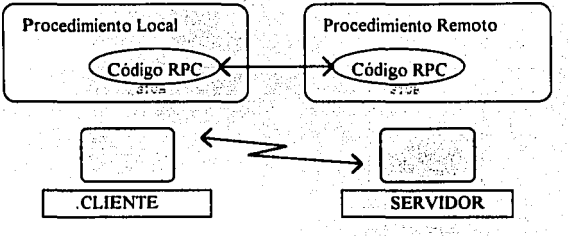


Fig. 4.2 Llamada a Procesos Remotos (RPCs)

Un problema común que se presenta con los RPC en el paso de argumentos y resultados es la diferencia que pudiera existir entre las arquitecturas de las máquinas locales y remotas. Esto puede ocasionar que el proceso remoto pueda no interpretar correctamente los argumentos pasados a éste, o que la aplicación local malinterprete los resultados generados. Para evitar estos problemas existen técnicas tales como la desarrollada por Sun Microsystems la cual lleva el nombre de XDR (eXternal Data Representation/Representación externa de datos) y que permite describir los argumentos y resultados de un procedimiento remoto sobre cualquier máquina sin importar la diferencia en sus arquitecturas.

Una de la compañías que más se ha destacado en el desarrollo de plataformas que operen en ambientes de cómputo distribuido es la OSF (Open Software Fundation/ Fundación Abierta de Software) la cual es creadora del esquema DCE (Distributed Computer Enviroment/ Ambiente de Computación Distribuida) el cual es todo un conjunto integrado de servicios que soportan el desarrollo, uso y mantenimiento de aplicaciones distribuidas.

4.1.2 Historia de los RPCs

Los RPC empezaron a ser utilizados desde los primeros años de la década de los 80. En un principio eran utilizados para ligar aplicaciones escritas en lenguaje C y que corrían en plataformas UNIX; con el tiempo su uso se extendió hacia una variedad de aplicaciones entre las que podemos mencionar sistemas operativos, lenguajes de programación y protocolos de red. El principal factor de este crecimiento es que los RPCs permiten que las aplicaciones ejecuten un conjunto de instrucciones sobre una máquina diferente a donde se ejecuta la aplicación principal, de una manera similar a como se hace una llamada a un procedimiento que se encuentra en la misma máquina.

Uno de los primeros trabajos de llamadas a procesos remotos se debe al Ing. Jim White, el cual es considerado como el padre de los RPCs. Su trabajo en la compañía norteamericana SRI durante los años 70's, dio lugar a lo que más tarde se conocería como WHIT 76 (que es en realidad un primitivo mecanismo RPC), desgraciadamente aun era muy pronto para alcanzar a ver la verdadera potencialidad de esta nueva tecnología, por lo que White cambió de compañía y fue en la XEROX Company donde los RPCs se dieron a conocer al mundo. Trabajando en la XEROX, White junto con los ingenieros Alan Frieier y Bob Lyon sacaron a la luz el primer RPC comercial: el XERO 81. En 1983 Lyon dejó la XEROX para trabajar en una compañía que para entonces apenas comenzaba a crecer: Sun Microsystems. Para 1985 Lyon había desarrollado uno de los mecanismos de RPCs más aceptados hoy en día: *Sun's version RPC*.

Otra de las contribuciones de Lyon fue el desarrollo del sistema NFS (Network File System) el cual emplea mecanismos RPC para dar transparencia en el acceso de archivos en los ambientes de computación distribuida. La flexibilidad de los sistemas de Lyon ha sido confirmada por el gran número de plataformas sobre las cuales las librerías RPCs han sido instaladas. A partir de los RPCs desarrollados por White y por Lyon, se han desarrollado nuevas versiones de RPCs los cuales, con el tiempo, han pasado a ser la principal técnica empleada para comunicar a dos o más computadoras que operen dentro de un ambiente distribuido.

4.1.3 Consideraciones en el diseño de los RPC.

Todo RPC debe ser diseñado atendiendo a cuatro aspectos básicos: la forma en que se enlazarán las aplicaciones, la heterogeneidad, la transparencia y la concurrencia; los cuales dirán la forma en que un proceso se comunicará con otro, la forma en que se entienden las aplicaciones escritas en plataformas diferentes, la manera en que se ocultan los detalles del sistema operativo y la manera en que cooperan las aplicaciones para el logro del objetivo, respectivamente.

a) El enlace de las aplicaciones. Al igual que los programas que se ejecutan sobre una sola máquina hacen uso de pilas y colas para ejecutar otros procedimientos, los RPC también hacen uso de estas a fin de ejecutar códigos de programa sobre otra máquina. Todo sistema RPC debe contener mecanismos para conectar el programa llamante con el llamado. La manera más común que se emplea para llamar a un

programa sobre otra máquina, es a través de un nombre, tal y como se hace para llamar a un procedimiento local. Lo único que se debe tener en cuenta es que tanto el procedimiento llamante como el llamado deberán manejar los mismos tipos de datos. También para establecer un enlace entre procedimiento llamante y llamado se utiliza una secuencia de protocolos, la cual es un tipo de RPC que contiene una combinación de protocolos de comunicación que establece la forma de comunicarse entre cliente y servidor. Por ejemplo, ncacn_ip_tcp representa una combinación de protocolos sobre una misma red de computadoras, la instrucción viene del inglés *"Network Computing Architecture Conection-oriented protocol, over a network with the Internet Protocol and the Transmission Control Protocol for transport"*.

b) **Heterogeneidad**: Los sistemas distribuidos se caracterizan por su alto grado de heterogeneidad: muchas clase de máquinas se encuentran conectadas a la misma red, y los programas que corren en estas máquinas frecuentemente son escritos en diferentes lenguajes de programación y con diferentes sistemas operativos. Los programas deben correr en este ambiente heterogéneo, por lo que es necesario definir la semántica de comunicación del lenguaje y de la máquina de manera independiente. Muchos RPCs permiten la comunicación entre los componentes heterogéneos permitiendo una declaración estática de la interfaz de procedimientos remotos, esto significa que tanto el procedimiento que llama como el que es llamado deben de estar de

acuerdo en los tipos de argumentos y en los resultados (si se manejan enteros se deben regresar enteros, si son flotantes se regresan flotantes), así como en la representación de los datos (ya que, por ejemplo, un entero puede ser representado por 16 o por 32 bits dependiendo del sistema en que se trabaje). Para asegurarse de que tanto los tipos como la representación de los datos sea la misma para todo el sistema, los RPCs emplean rutinas conocidas como **"Stubs"** que no son otra cosa más que convertidores de formato de datos. Es importante que un RPC cuente con rutinas de conversión de éste tipo ya que aún cuando los tipos sean los mismos, se pueden tener diferentes módulos de comunicación que pueden emplear diferentes representaciones para el mismo tipo. Como se observó en la figura 4.2 para llamar al procedimiento remoto, el programa cliente hace una solicitud a un stub local(o stub cliente), el cual establece una representación adecuada de los argumentos en el mensaje e inicia comunicación con la máquina servidor. En el servidor los mensajes son recibidos por un stub servidor el cual reconstruye los argumentos y los pasa al procedimiento deseado. Cuando el procedimiento retorna los resultados, el proceso se invierte: el stub servidor convierte los datos a un formato adecuado y los envía de regreso al stub cliente; el stub cliente reconstruye los resultados y los envía al programa que hizo la llamada. Los stubs ocultan todos los detalles de conversión y comunicación; el procedimiento que hace la llamada se comunica de

manera local con el stub cliente y el proceso que regresa los resultados lo hace también de manera local con el stub servidor. En la figura 4.3 se muestra más a detalle lo expuesto anteriormente.

c) **Transparencia.** Uno de los objetivos básicos de cualquier sistema RPC es el hacer la semántica de una llamada remota tan parecida como sea posible a una llamada local. Sin embargo, existen algunas diferencias, que son imposibles de evitar. En primer lugar tenemos el hecho de que cada sistema maneja las fallas que pudiesen ocurrir de diferente manera, resultando en una gran variedad de diversas semánticas. Pudiéndose dar el caso, por ejemplo, de que una máquina, a la que se le ha enviado un mensaje, estuviera fuera de línea; pero por tenerse diferentes significados en los códigos de error, la máquina que envía el mensaje no se de cuenta de esto y siga mandando la información, trayendo por consecuencia una perdida de datos y de recursos del sistema. Un segundo problema que llega a afectar la transparencia de una llamada remota, consiste en la diferencia que existe entre los tipos de datos usados en los mensajes y en los programas; este problema ha sido resuelto, como se expuso anteriormente, por el empleo de stubs o por la restricción de ciertos tipos de datos tales como los punteros.

d) **Concurrencia.** Mucho se ha dicho sobre que los RPCs son poco concurrentes. Esto es debido a que en los sistemas RPC el procedimiento que hace una petición es suspendido hasta que el procedimiento que es llamado retorne una respuesta. Lo anterior puede

causar graves problemas ya que, por mencionar algo, el cliente bien podría hacer otro trabajo mientras que el RPC se ejecuta en el servidor. Además, el servidor puede hacer llamadas a otros servidores como parte del servicio para el cliente y entrar, a su vez , en un tiempo de espera, tiempo en el que podría atender a otras solicitudes. Una solución para tratar de hacer a los RPCs más concurrentes consiste en procesarlos en paralelo (tal como se explicó en el capítulo 2), lo cual se logra dividiendo a estos en pequeños bloques, así mientras un bloque se ejecuta en el servidor, los otros bloques son procesados en la máquina local.

4.1.4 Secuencia de una llamada a un proceso remoto.

Hemos ya mencionado que aunque aparente para el usuario la llamada a un proceso remoto ocurre tal y como si fuera una llamada aun proceso local, para el sistema operativo una llamada remota y una llamada local son cosas muy distintas. La figura 4.3 muestra detalladamente los pasos que conforman la ejecución de un RPC; estos pasos por su orden de ejecución son:

1. El cliente llama a un procedimiento, conocido como stub del cliente, el cual aparece ante el cliente como si fuera el servidor al que desea llamar. El propósito del stub es convertir los argumentos del cliente a un formato estándar y colocar a estos argumentos en uno o más paquetes para enviarlos a través de la red. A este proceso de empaquetar los argumentos y enviarlos por la red se le conoce como "*marshaling*".

2. Los argumentos empaquetados o mensajes de la red son enviados al sistema remoto por el stub cliente. Para hacer esto se requiere hacer una llamada de sistema a través del kernel local.
3. El kernel local ubica al kernel remoto e inicia el envío de información.
4. En el sistema remoto se encuentra el stub servidor esperando por los requerimientos del cliente. El stub servidor desempaquetará los argumentos recibidos a través de la red y posibilitará su conversión.
5. El stub servidor ejecuta una llamada a un procedimiento local, pasándole los argumentos que recibió a través de la red y provenientes del stub cliente.
6. Cuando el procedimiento en el servidor termina de procesar los requerimientos del cliente, retorna al stub servidor los resultados generados.
7. El stub servidor convierte los valores regresados, si es necesario, y empaqueta a estos en uno o mas mensajes de red para ser enviados al stub cliente.
8. El mensaje viaja de regreso hacía la máquina que hizo la llamada.
9. El stub cliente interpreta el mensaje de la red recibido por el kernel local.
10. En caso de necesitarlo el stub cliente convierte los resultados a un formato que pueda ser entendido por el procedimiento que hizo la llamada y los envía hacia éste. Para el procedimiento cliente todo ha transcurrido tal y como si todas las operaciones hubieran sido hechas localmente.

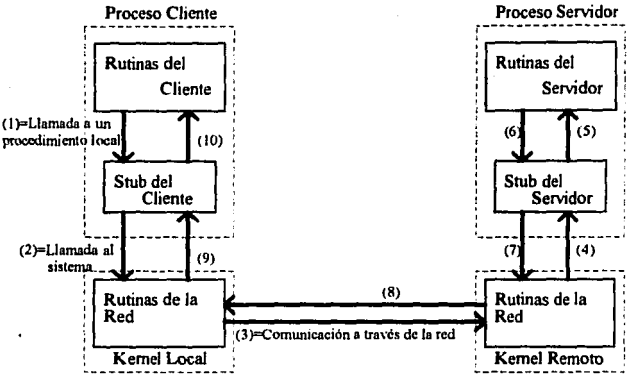


Fig. 4.3 Pasos de una llamada a un proceso remoto.

4.1.5 El modelo OSI y los RPCs.

En 1984, ISO desarrollo el modelo OSI, el cual ya se explicó en el capítulo 2. Este modelo contiene siete capas conceptuales como lo ilustró la figura 2.3. La capa siete, la de aplicación, es la ventana entre el modelo OSI y el ambiente de los sistemas locales y en donde residen las aplicaciones distribuidas. La capa seis, la de presentación, proporciona independencia a la aplicación ya que permite correr a las diferentes aplicaciones independientemente del tipo de datos que manejen; los RPCs caen en algún lugar entre estas dos capas. Comúnmente son considerados como parte de la capa seis. Ya que el objetivo de los RPC es el ocultar la capa de aplicación de algunos de los detalles de la red, estos suelen normalmente incluir una especificación para el intercambio de argumentos y resultados entre el cliente y el servidor en algún formato estándar. Esto amplía el rango de portabilidad entre diferentes sistemas y

evita que los programas de aplicación tengan que preocuparse de detalles tales como el orden de los bytes. La figura 4.4 muestra donde se incluyen los RPCs dentro del modelo OSI.

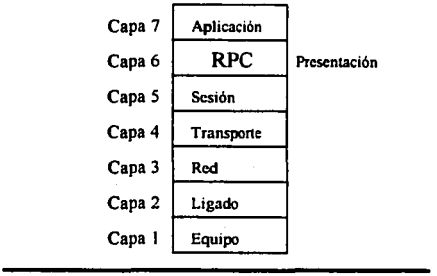


Fig. 4.4 RPCs en el modelo OSI

4.1.6 Implementaciones de RPCs.

Como ya se mencionó cuando hablamos de historia de los RPC, el diseño original de un RPC creado por Jim White, fue retomado por diversas compañías y modificado según diversas conveniencias. Hoy en día encontramos tres diferentes implementaciones de RPC que dominan prácticamente todo el mercado de las aplicaciones distribuidas, estas son:

1. **Computación de Red Abierta** (ONC/Open Network Computing) de Sun Microsystems, que divide a los RPC en dos partes principales: las especificaciones a llamadas remotas y la representación externa de datos (XDR/ eXternal Data Representation).

Estos RPC comúnmente utilizan UDP o TCP[1] como su protocolo de transporte. El tamaño de palabra es de 32 bits, lo cual significa que por ejemplo, cuando un cliente pasa un entero de 16 bits a un servidor, el entero de 16 bits es primeramente convertido a 32 bits por el cliente y reconvertido a 16 bits por el servidor.

2. **Xerox Courier**. La cual fue la primera aplicación de RPCs desarrollada, incluye un modelo construido sobre XNS/SPP (Xerox Network Systems/Sequenced Packed Protocol <Secuencia de Protocolos en Paquete Xerox>)[2] como su protocolo de transporte y una representación estándar de datos. Esta implementación impone un estándar que tanto el cliente como el servidor deben usar. Maneja un tamaño de palabra de 16 bits y utiliza código ASCII para la representación de caracteres.
3. **Arquitectura de Red de Computadoras Apollo's** (NCA/Network Computing Architecture), la cual maneja el protocolo UDP y una técnica conocida como NDR (Network Data Representations/ Representación de Datos en Red) que permite definir una representación estándar de datos. Es la representación más moderna de un RPC, la cual surge como un intento de establecer un estándar en el manejo de estos mecanismos.

[1] El protocolo TCP también conocido como UDP o simplemente TCP/IP (Transmission Control Protocol/Internet Protocol -protocolo de control de transmisiones/protocolo Internet) es un protocolo desarrollado bajo contrato del Departamento de Defensa de los Estados Unidos para intercomunicar sistemas diferentes. Es un estándar UNIX de hecho, pero está respaldado por sistemas operativos de cinco mainframe. Es utilizado por muchas corporaciones y casi todas las Universidades y entidades federales.

[2] El protocolo XNS es en realidad una arquitectura de redes desarrollada por Xerox Corporation durante la década de los 70's para integrar sus productos de oficina y sistemas de computación.

La tabla 4.1 compara los tipos de datos y formatos soportados por estas tres implementaciones de RPC. En la columna de cada representación se muestra un • para indicar que se soporta el correspondiente tipo de dato. Si el espacio esta en blanco, entonces el tipo de dato no es soportado por la implementación.

Tipo de Dato	*Sun RPC*	*Xerox Courier*	*Apollo NDR*
lógica de 8-bit			•
lógica de 16-bit		•	
lógica de 32-bit	•		
entero con signo 8-bit			•
entero sin signo 8-bit			•
entero con signo 16-bit		•	•
entero sin signo 16-bit		•	•
entero con signo 32-bit	•	•	•
entero sin signo 32-bit	•	•	•
entero con signo 64-bit	•		•
entero sin signo 64-bit	•		•
orden del byte	grande	grande	grande o pequeño
formato de enteros signados	complemento a 2	complemento a 2	complemento a 2
punto flotante 32-bit	•		•
punto flotante 64-bit	•		•
formato del punto flotante		IEEE	IEEE, VAX, IBM o Cray
tipo de caracter	ASCII	16-bit NS	ASCII o EBCDIC
enumeración	•	•	•
estructura (de registros)	•	•	•
arreglos Dim. simple y complejo	•	•	•
arreglo Dim. de Estruc. fija y variable	•	•	•
multidimensiones de tamaño fijo			•
multidimensiones de tamaño variable			•
distinción entre uniones			•
ocultamiento de datos de largo fijo	•	•	•
ocultamiento de datos de largo variable	•	•	•

Las tres implementaciones anteriores utilizan lo que se conoce como tipeo implícito. Un tipeo implícito es cuando sólo el valor de la variable es transmitido a través de la red y no el tipo de la variable. En contraste a esto, la técnica ISO para representación de datos, denominada ASN.1 (Abstract Syntax Notation 1), utiliza tipeo explícito en el que se transmite el tipo de cada dato en un mensaje (comúnmente en código de 8 bits) junto con su valor.

4.1.7 Desarrollo de una aplicación con llamadas a procesos remotos.

El desarrollo de aplicaciones distribuidas mediante RPC puede resultar una tarea bastante ardua aún para un programador experimentado. El desarrollo de éste tipo de aplicaciones involucra tres pasos como lo muestra la figura 4.5. Afortunadamente los tres modelos de RPCs vistos anteriormente incluyen ya todas las especificaciones y mecanismos necesarios para establecer una comunicación. Con esto únicamente es necesario escribir el código para el cliente ya que tanto la interface (el conjunto de reglas que indicarán la manera en que se establecerá la comunicación) como el código del cliente son provistos por la implementación del tipo de RPC.

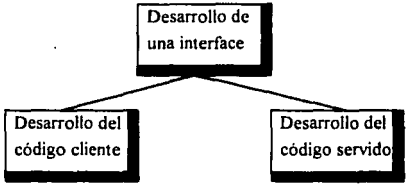


Fig. 4.5 Pasos involucrados en el desarrollo de una aplicación distribuida

A continuación desarrollaremos una aplicación distribuida empleando RPCs, naturalmente el diseño se ha tratado de simplificar para un mejor entendimiento de sus partes. Aunque normalmente sólo es necesario escribir la parte del cliente, en el ejemplo desarrollaremos las tres partes a fin de que el lector tenga un mejor conocimiento de una aplicación de éste tipo; por lo mismo

no nos basaremos en ninguna de las implementaciones antes descritas, ya que estas traen su propia interface ya desarrollada y el propósito del ejemplo es desarrollar el mecanismo completo. Las librerías que se llegan a emplear son provistas por el mecanismo DCE de Open Software Fundation.

El ejemplo supone una comunicación establecida entre una sola máquina cliente y una sola máquina servidor, también se supone que el servidor cuenta con un hardware especial tal como un procesador de arreglos (el procesador de arreglos es una extensión de la unidad aritmética, que es capaz de ejecutar operaciones simultáneas sobre elementos de un arreglo de datos en una cantidad de dimensiones.). En este ejemplo el cliente realiza una operación aritmética basada en arreglos para lo que hace una llamada a un proceso remoto que utiliza el procesador de arreglos. El proceso remoto, ejecutado en el servidor, toma dos arreglos como argumentos, suma sus elementos entre sí y retorna el resultado al cliente en un tercer arreglo. Por último el resultado es desplegado en el sistema cliente.

4.1.7.1 Creación de la interface.

Crear la interface significa establecer las normas comunes entre servidor y cliente, estableciendo que tipos de datos serán manejados, como serán estos enviados y como regresados, así como todo un conjunto de declaraciones acerca de como un procedimiento será tratado.

La creación de la interface también incluye la generación de un UUID (Universal Unique Identifier/ Identificador Único Universal), que es simplemente un número único que identifica a la interface de cualquier otra interface de la red.

La definición de la interface incluye elementos sintácticos llamados atributos, los cuales dan información al sistema acerca de los tipos de datos, arreglos,

punteros, nombres de funciones, etc.. El código que se presenta a continuación representa una muy simple definición de interface que se utilizará para el ejemplo que estamos desarrollando. El texto consiste de un **encabezado** y un **cuerpo**. El encabezado contiene un atributo **UUID** y el nombre asignado a la interface. El cuerpo especifica los procedimientos que utiliza la interface:

```
/* Nombre de programa: arithmetic.idl */
/* Esta definición de interface representa un procedimiento con una aritmética
/*básica que una llamada a un proceso remoto puede utilizar*/

uuid(C985A380-2552-11C9-A50B-08002BOECEFI) /* Identificador único*/  <-1
                                           /* universal*/
interface arithmetic          /* nombre de la interface*/            <- 2
{
const unsigned short ARRAY_SIZE = 10; /* constante entero sin signo*/  <- 3
typedef long long_array[ARRAY_SIZE] /*Se define un tipo de dato*/     <- 4

void sum_arrays (             /*Definición de la función y sus parámetros*/ <- 5
[in] long_array a,            /* Primer parámetro in */
[in] long_array b,            /* Segundo parámetro in */
[out] long_array c            /* Tercer parámetro out*/
                );
}
```

1. El atributo uuid especifica la interface UUID, el formato de éste número puede variar de un sistema a otro, por lo que se recomienda consultar los manuales del sistema sobre el que se trabaje.
2. Aquí aparece la palabra reservada **interface** seguida por el nombre de la misma.
3. Se define una constante, en este caso el tamaño del arreglo.
4. Se define un tipo de datos a fin de usarlo en otras aplicaciones que funcionen bajo esta interface. Para este caso, se define un arreglo de 10 enteros largos.
5. El resto de esta definición de interface es la declaración de procedimientos. Los atributos in y out son necesarios a fin de que el compilador conozca cuales datos deben de ser enviados o recibidos a través de la red (in pasa un valor al procedimiento remoto, out regresa un valor desde el procedimiento remoto).

Cuando el código de interface anterior sea compilado se creará un archivo de cabecera en lenguaje C necesario para crear los stubs cliente y servidor. Un compilador IDL (Interface Direct Lenguaje/ Interfaz de Lenguaje Directo) puede servir muy bien para este propósito (el compilador IDL provee una

sintaxis y semántica muy similar a C, pero con características especiales para poder operar en una red).

4.1.7.2 Creación del código cliente.

El código del cliente es el más simple de todos, de hecho no se encontrará diferencia entre este programa y otro cualquiera. Las librerías para establecer comunicación remota son proporcionadas por el ambiente DCE. El código es el siguiente:

```
/* NOMBRE DE ARCHIVO: client.c */
/* Este es el modulo cliente para el ejemplo arithmetic */
#include <stdio.h>
#include "arithmetic.h"  /*Archivo de cabecera creado mediante un compilador
IDL*/                                                          <- 1

long_array a ={100,200,345,23,67,65,0,0,0,0};
long_array b ={4,0,2,3,1,7,5,9,6,8};

main()
{
   long_array result;
   int   i;

   sum_arrays(a, b, result); /*Una llamada a un proceso remoto*/      <- 2
   puts ( "Suma: ") ;
   for(i = 0; i < ARRAY_SIZE; i++)
   printf("%ld\n", result[i]);
}
```

1. Se incluye el archivo de cabecera producido mediante el compilador IDL.
2. El cliente hace una llamada al proceso remoto sum_arrays usando los dos arreglos previamente inicializados como parámetros. Desplegándose después los elementos del arreglo resultante.

4.1.7.3 Creación del código servidor.

El desarrollo del código servidor requiere que se conozcan la definición de interface y algunas rutinas de RPCs. Aquí también utilizamos el archivo de cabecera arithmetic.h debido a que contiene la definición de requerimientos para el procedimiento remoto y las llamadas al sistema. Escribiremos dos distintas porciones de código: el procedimiento remoto y el código para inicializar al servidor.

El siguiente es el código remoto, el cual también utiliza el archivo de cabecera **arithmetic.h.**

```
/*NOMBRE DE ARCHIVO: procedure.c*/
/*Implementación de procedimiento definido en la interface arithmetic*/
#include <stdio.h>
#include "arithmetic.h"          /*archivo de cabecera creado con un compilador
IDL*/                                                              <- 1
void sum_arrays(a,b,c)                                             <- 2
long_array a;
long_array b;
long_array c;
{
int i;
for(i=0; i<ARRAY_SIZE; i++)
    c[i]=a[i] + b[i];    /*suma de los arreglos de elementos*/       <-3
}
```

1. Se incluye el archivo de cabecera creado con el compilador IDL.
2. Se declara el procedimiento tal y como aparece en la definición de la interface.
3. Se realiza la operación de suma entre los elementos del arreglo.

Ninguno de los códigos cliente o servidor expuestos anteriormente presentan diferencias en relación con otros programas escritos en lenguaje C; incluso estos pueden ser compilados y ejecutados localmente sin ningún problema y sobre cualquier máquina con un compilador C. Sin embargo, el programa que a continuación se mostrará es todo lo contrario, ya que éste hace uso de los servicios de la red y por lo tanto emplea librerías de los ambientes DCE. El estudio de las librerías y procedimientos DCE queda fuera de los alcances del presente estudio, los lectores interesados en el tema pueden consultar la sección bibliográfica en donde se mencionan varios libros que abordan este tema.

Para facilitar el que el cliente encuentre al servidor adecuado en el programa siguiente se usa lo que en DCE se conoce como **servicio de nombre** (Name service, también conocido como CDS/Cell Directory Service), el cual es un término empleado para denotar un software que convierte un nombre en una dirección física almacenada en una base de datos, esto es, un servicio que ofrece la misma información a las aplicaciones que se encuentran ejecutándose sobre diferentes sistemas.

El código de inicialización del servidor es el siguiente:

```
/* NOMBRE DE ARCHIVO: server.c*/
#include <stdio.h>
#include "arithmetic.h"          /*Cabecera creada con un compilador IDL*/
#include "check_status.h"        /*Cabecera con la macro CHECK_STATUS*/
main()
{
 unsigned32              status;                 /* error status (nbase.h) */
 rpc_binding_vector_t  *binding_vector;          /*se establece un enlace (rpcbase.h)*/
 unsigned_char_t       *entry_name;              /*referencia al servicio de nombre (lbase.h)*/
 char *getenv();
 rpc_server_register_if(                 /* registra la interface con la rutina RPC*/     <-1
     arithmetic_v0_0_s_ifspec,           /* especifica la interface(arithmetic.h)*/
     NULL,
     NULL,
     &status
                        );
 CHECK_STATUS(status, "No se puede registrar la interface\n", ABORT);
 rpc_server_use_all_protseqs(            /*Se crean vínculos de información*/          <- 2
     rpc_c_protseq_max_reqs_default, /*tamaño del mensaje de llamada ( rpcbase.h)*/
     &status
                             );
 CHECK_STATUS(status,"No se puede crear vínculo de información"\n", ABORT);
 rpc_server_inq_bindings(                /* obtiene información del servidor*/           <- 3
   &binding_vector,
   &status
                         );
 CHECK_STATUS(status, "No se puede obtener vínculo de información\n", ABORT);
 entry_name = (unsigned_char_t *)getenv("ARITHMETIC_SERVER_ENTRY");
 rpc_ns_binding_export(          /*exporta una entrada a un servicio de nombre*/ <- 4
     rpc_c_ns_syntax_default,    /* sintaxis del servicio de  nombre (rpcbase.h)*/
     entry_name,
     arithmetic_v0_0_s_ifspec,
     binding_vector,
     NULL,
     &status
                       );
 CHECK_STATUS(status, "No se puede exportar el servicio de nombre", ABORT);
 rpc_ep_register(                        /*registra el endpoint*/                      <- 5
     arithmetic_v0_0_s_ifspec,       /*especificación de interface (arithmetic.h)*/
     binding_vector,
     NULL,
     NULL,
     &status
                 );
 CHECK_STATUS(status, "No se puede agregar la dirección al endpoint\n",
          ABORT);
 rpc_binding_vector_free(            /*libera los enlaces de la memoria*/             <- 6
 &binding_vector,
 &status                     );
```

```
CHECK_STATUS(status, "No se pueden liberar los enlaces y vectores\n", ABORT);
  puts("Recibiendo llamada remota...");
  rpc_server_listen(              /*se recibe la llamada remota*/          <- 7
       rpc_c_listen_max_calls_default,
       &status
                  );
  CHECK_STATUS(status, "Recepción de RPC fallada\n", ABORT);
}
```

1. Se registra la interface con las librerías RPC del sistema. La instrucción *arithmetic_v0_0_s_ifspec* es conocida como manejador de interface y hace referencia a la información que la aplicación necesita, tal como el UUID.
2. Se crean los vínculos de información; para vincular la información se deberá escoger una o más secuencias de protocolos. En este caso se emplea la instrucción *rpc_server_use_all_protseqs* a fin de que el cliente pueda usar todos los protocolos disponibles. La llamada *rpc_c_protseq_max_calls_default* posibilita al DCE a establecer el tamaño del buffer.
3. Se obtienen los vínculos obtenidos en el paso dos, esto es importante ya que se verifica que exista una correcta comunicación entre cliente y servidor.
4. El servidor exporta toda la información de los enlaces a la base de datos del servicio de nombre. Con la sentencia *rpc_c_ns_syntax_default* se le indica a la rutina la sintaxis con la que se almacenarán los datos.
5. La rutina RPC asigna endpoints a un servidor como parte de crear enlaces de información. Un endpoint es un número que representa a un proceso específico que se ejecuta en un sistema.
6. Se libera la memoria que fue ocupada al hacer la llamada a la rutina *rpc_server_inq_bindings*.
7. Finalmente, el servidor debe esperar a que las llamadas arriben. Cada sistema tiene un número máximo de llamadas que el servidor puede aceptar en un mismo instante de tiempo. DCE establece éste máximo al usar la sentencia *rpc_c_listen_max_calls_default*.

4.1.8 Las Ventajas de usar RPCs

Los RPCs dan la posibilidad de escribir aplicaciones distribuidas, que como ya se mencionó a lo largo de los capítulos anteriores, son un conjunto de procedimientos dispersos en diferentes computadoras interconectadas por

medio de una red. Estos procedimientos son invocados usando RPCs , los cuales no distinguen entre arquitecturas o sistemas operativos, por lo que pueden ejecutarse sobre cualquier plataforma de hardware y/o software razón por la que son ampliamente utilizados.

Cuando se programa utilizando RPCs, se programa utilizando las mismas técnicas que los programadores experimentados manejan, sin tener que llegar a preocuparse de que se esta programando bajo un ambiente de red y que por lo tanto la aplicación que se esta desarrollando deberá de ser capaz de ejecutarse sobre cualquier máquina del sistema. Dicho de otra forma, los RPCs ocultan los detalles de la programación a nivel de la red, permitiendo al desarrollador concentrarse totalmente en su aplicación.

Los RPCs ocultan la dependencia que cualquier programa tiene del sistema operativo. El único requerimiento para que una aplicación basada en RPCs sea portable, es que la computadora en la que se ejecutará la aplicación, contenga una librería RPC, que sea compatible con aquella sobre la cual la aplicación fue originalmente desarrollada. Esto es debido a que no existen protocolos estándar para el desarrollo de los RPC, lo cual provoca que se dificulte la portabilidad de aplicaciones sobre computadoras que tienen diferentes librerías de RPCs.

Las llamadas a procesos remotos soportan una amplia variedad en el diseño de aplicaciones. Debido a que estructuran a una aplicación distribuida como un conjunto de procedimientos que definen a un servicio, las librerías RPC pueden ser usadas para construir bloques de aplicaciones muy sofisticadas.

Los procedimientos remotos pueden ser utilizados en diferentes aplicaciones por varios clientes y los servidores podrán llamar a otros servidores para completar su trabajo. La naturaleza modular que proveen los RPCs en aplicaciones distribuidas, puede ser de gran utilidad al dividir aplicaciones demasiado largas para que se ejecuten en diferentes procesadores, disminuyéndose el tiempo de ejecución y teniéndose tiempo libre para otras actividades.

Los RPCs tienen un amplio uso, que va desde la ejecución remota hasta la realización de respaldos de información sobre otras máquinas, desde mecanismos de manejo y control de redes hasta el procesamiento distribuido de imágenes.

4.1.9 Desventajas de los RPCs.

Quizás el principal problema al que se enfrentan las aplicaciones que emplean RPCs como mecanismo de comunicación sea el tiempo de respuesta (ya que la computadora cliente hace un requerimiento y debe de esperar hasta que el servidor procese ese requerimiento y regrese un resultado). Por tal motivo los RPCs deben ser desarrollados en un ambiente de red de alta velocidad en donde haya suficiente ancho de banda a fin de asegurar tiempos de respuesta adecuados. A pesar de los múltiples esfuerzos que se han hecho para minimizar el tiempo que debe esperar la máquina workstations para recibir la respuesta por parte del servidor (tiempo en el que la workstations no realiza ninguna otra

actividad), este tiempo de espera continua siendo la desventaja primaria en el empleo de los RPCs. Otras de las desventajas de los RPCs es que son muy pobres en el envío de mensajes del estado del sistema ya que consideran que las máquinas de la red están siempre disponibles.

También es problema importante la ausencia de protocolos que normalicen la forma en que las llamadas a procesos remotos serán efectuadas o la forma en que los RPCs se deben relacionar con las interfaces de programa de aplicación (API/ Aplication Program Interface). Lo que provoca que cada fabricante haya adoptado algún estándar limitando la compatibilidad de las aplicaciones.

Por lo anterior se han dedicado cientos de horas de trabajo para buscar tecnologías que puedan llegar a superar a las llamadas a procesos remotos, sin embargo, en la mayoría de los casos, los resultados sólo han generado técnicas para hacer más veloces a los RPCs y es que no es fácil sustituir una tecnología de la que se tienen muchas más ventajas que desventajas. En la implementación del middleware, del que hablaremos a continuación, los RPCs han encontrado una fuerte competencia por parte de una técnica conocida simplemente como "*mensajería*"; pero con la fusión de los RPCs y de las técnicas de programación orientadas a objetos parece ser que a los RPCs aún les queda mucho futuro.

4.2 El middleware.

Debido a la gran cantidad de aplicaciones distribuidas (predominando las de cliente-servidor) que existen en el mercado, las compañías enfrentan el dilema de lograr que sus aplicaciones trabajen en un ambiente distribuido heterogéneo, es decir, que sus programas desarrollados, por ejemplo, bajo cliente-servidor, corran lo mismo bajo el software y/o hardware de un fabricante como en el de otro, y que además sus programas sean capaces de trabajar con diferentes protocolos de red, sistemas operativos y bases de datos.

La solución a este problema parece ser el llamado middleware, una capa que se encuentra entre las aplicaciones y los sistemas operativos, las bases de datos y los protocolos de red.

El middleware hace que los sistemas operativos y protocolos dispares sean invisibles para el desarrollador de aplicaciones y la aplicación resultante. El middleware es transparente para el usuario final, y está diseñado fundamentalmente para desarrolladores de aplicaciones y programadores.

4.2.1 El middleware y la tecnología cliente-servidor.

Definir el middleware es definir el mecanismo fundamental de cliente-servidor. Como lo muestra la figura 4.6, cliente-servidor implica un conjunto de requerimientos hechos por un cliente (requeridor) a un servidor el cual procesa los requerimientos y retorna los resultados al cliente. En la figura un usuario que opera una IBM PC hace una solicitud a un servidor (Sun Server) que a su

vez es cliente del HOST IBM 3090. Obsérvese que cada máquina opera bajo su propio modelo de base de datos y bajo su propio protocolo. El factor principal de una implementación bajo cliente-servidor es su gran flexibilidad para adaptarse a las diferentes necesidades de un conjunto de usuarios; con cliente servidor un usuario no tiene porque depender de un sólo proveedor sino que pueden escoger de entre un sin número de tecnologías existentes la que más se adapte a sus necesidades.

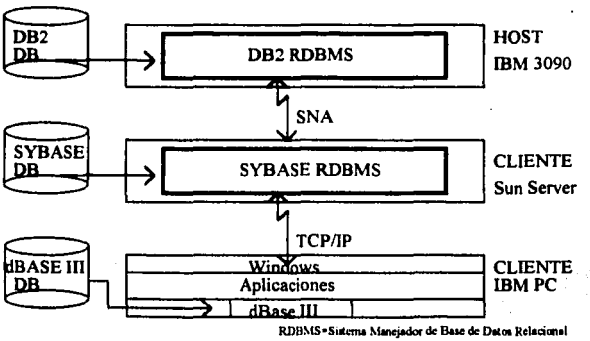


Fig. 4.6 Arquitectura de un sistema Cliente-Servidor

El middleware es un componente esencial de cliente-servidor; es la parte encargada de resolver dos de los problemas más importantes a los que se enfrentan los diseñadores de cliente-servidor: la heterogeneidad en redes y en

los sistemas operativos, y la falta de facilidades del sistema operativo para el desarrollo de aplicaciones que corran en la red.

La figura 4.7.a muestra un ambiente sin middleware. En este ambiente son necesarios especialistas en redes y sistemas operativos para implementar y mantener aún las aplicaciones distribuidas más sencillas. Lo anterior es debido a que en los ambientes sin middleware los sistemas operativos proveen sólo las facilidades para el desarrollo de aplicaciones en red, estas facilidades están principalmente limitadas a enviar y recibir mensajes entre computadoras; pero comúnmente no existen facilidades para manejar y rastrear los centenares de conversaciones que ocurren entre los componentes de un sistema distribuido.

Contrastando a esta situación, la figura 4.7.b muestra un ambiente con middleware. Aquí se reduce la complejidad para desarrollar y mantener el software de la red. *El middleware es la capa de software que reside entre una aplicación y la red, y se encarga de manejar la interacción entre dos aplicaciones activas a través de una plataforma heterogénea de computadoras.* De aquí que el middleware ayude a los desarrolladores de aplicaciones con la carga de tratar de conciliar códigos incompatibles e interfaces APIs[3] de vendedores específicos. De esta manera los desarrolladores se liberan de trabajar con los complejos mecanismos que constituyen los fundamentos de

[3] Las interfaces de software o programación (API) son los lenguajes, códigos y mensajes que utilizan los programas para comunicarse entre sí y con el hardware. A veces cada fabricante tiene sus propias interfaces API, de ahí que el papel del middleware de conciliar estas interfaces sea muy importante.

las redes y sus sistemas operativos, enfocando su atención al desarrollo de aplicaciones más robustas.

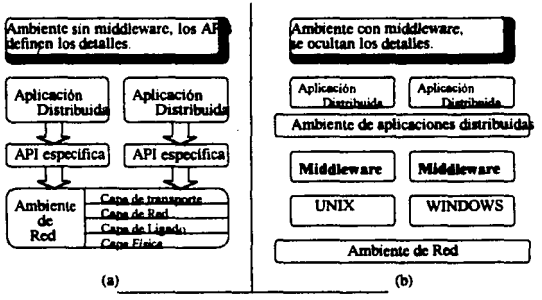


Fig 4.7 Un ambiente distribuido con y sin capa de Middleware

4.2.2 Tipos de Middleware.

En general el middleware esta compuesto por tres capas:

1. La primera capa conocida como capa API (application programmer interface) es utilizada por los programadores para el desarrollo de aplicaciones distribuidas.
2. La segunda capa se encuentra situada después de la capa API y que se encarga de comunicarse con otras aplicaciones basadas en middleware que corran en la red.
3. La tercera capa, que puede ser una simple lista o toda una estructura de datos, se usa para asignar tareas a las máquinas que tengan un mejor desempeño en la realización de determinada actividad. En esta capa se lleva

un registro del número de máquinas que conforman la red, de la configuración que tienen y de las tareas que pueden realizar.

El middleware se puede implementar basándose en tres mecanismos diferentes. Dependiendo del mecanismo que se utilice para implementarlo serán sus características; estos tres mecanismos son:

- Llamadas a procesos remotos (RPCs),
- Mensajes de software en fila o middleware de mensajería,
- Corredores de requerimientos a objetos (ORBs/Object Request Brokers).

Lo que distingue a cada uno de estos tres tipos es la manera mediante la cual los datos son transmitidos a través del ambiente distribuido. La mayoría del middleware comercial esta construido basándose en alguna de estás tecnologías, entre este middleware comercial podemos mencionar:

- **Middleware de red.**- Conjunto de herramientas las cuales permiten que los desarrolladores de aplicaciones construyan sus aplicaciones sin tener que programar en alguna determinada capa de la red. Esta categoría puede emplear RPCs, mensajes en fila u ORBs.

- **Middleware de base de datos.**- Interface estándar de bases de datos y aplicaciones SQL; puede también incluir bases de datos frontales[4] que los

[4] Una base de datos frontal (database front ends) es aquella que se encuentra conectada a las líneas de comunicaciones en un extremo y al mainframe o host en el otro extremo. Es capaz de almacenar y recibir mensajes y códigos de error.

desarrolladores de aplicaciones usan con una aplicación, por ejemplo, las herramientas que permiten a las aplicaciones cliente-servidor establecer comunicación con un mainframe. Tanto los sistemas de bases de datos relaciónales como los sistemas de bases de datos orientadas a objetos pueden ser consideradas como una forma de middleware de base de datos.

- **Middleware de conversión.**- Libera a los usuarios de decidir que datos serán desplegados y cuales almacenados. Incluye productos para la conversión de texto, gráficas y cualquier elemento usado en una aplicación; ejemplo de éste es el Intercambio Electrónico de Datos (EDI/ Electronic Data Interchange).

- **Middleware GUI.**- Incluye herramientas CASE que permiten multiplicar los GUIs (Graphics User Interface/Interfaz Unica de Usuario), es decir, aumentar los menúes, iconos y otras características gráficas utilizadas en alguna aplicación. Ejemplo son los productos XVT de XVT Corporation.

- **Software desarrollador de middleware.**- Conjunto de herramientas CASE basadas en 4GLs, que los desarrolladores pueden utilizar para acortar los tiempos de desarrollo.

Como existen tres diferentes métodos para la implementación del middleware también existirán tres diferentes tipos de éste, cada uno con sus propias ventajas y desventajas como se vera a continuación.

a) Implementación del middleware por medio de RPCs.

Si bien es cierto que los RPCs presentan algunos inconvenientes, también es cierto que sus bondades son aún mucho mayores. Como una tecnología madura y que provee un ambiente de programación común para los desarrolladores de aplicaciones así como servicios de directorio, seguridad y servicios de tiempo, el empleo de los RPCs es la técnica predilecta para la implementación del middleware. En la mayoría de las arquitecturas de aplicaciones distribuidas, encontramos middleware basado en RPCs. La firma que más se destaca en la producción de middleware basado en el empleo de RPC es la Open Software Foundation. Otros desarrolladores de middleware de éste tipo son: HP, NetWise, NobleSet y SunSoft.

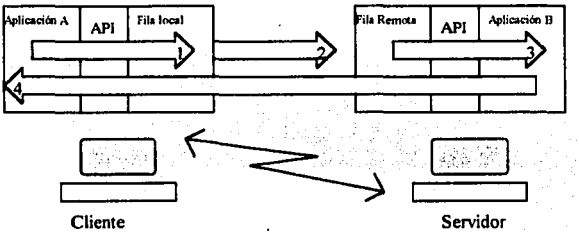
b) Implementación de middleware por medio de mensajes en fila.

Este tipo de middleware presenta menos desarrollo que el middleware basado en RPCs debido a que es una tecnología muy novedosa y no tan conocida. En vez de emplear llamadas a procesos remotos, el middleware de mensajería brinda conectividad, para lo cual elabora un conjunto de APIs para las que tienen que escribir los desarrolladores. Los mensajes en fila envuelven un intercambio asincrónico de datos. La figura No. 4.8 muestra la manera en que dos programas establecen una comunicación con el empleo de la técnica de mensajes en fila y mediante los comandos SEND y RECEIVE. Esta técnica permite a los programas que continúen procesando mientras esperan una respuesta a un requerimiento hecho con

antelación. De esta manera se elimina la total dependencia de la red para continuar un proceso y los tiempos de espera de los RPCs. Al igual que los RPCs los mensajes en fila liberan al programador de la compleja tarea de tener que programar en una capa específica de la red y de entender los complejos mecanismos de cada sistema operativo. También tienen la ventaja de que permiten al usuario mezclar diversos modos de comunicación (asincrónico, sincrónico y modo conversacional).

La diferencia fundamental entre el middleware de RPC y el middleware de mensajería es que la primera es una tecnología "síncrona" mientras que la segunda es, como ya se dijo, "asíncrona". En las aplicaciones sincrónicas, cuando se activa una RPC las aplicaciones envían una llamada para datos y esperan a que llegue la respuesta. Si se produce algún error o alguna falla, se notifica a los usuarios de forma inmediata y se vuelve a efectuar la RPC. En las aplicaciones asincrónicas, si el servidor está ocupado al momento de llegar la llamada, los usuarios no se tienen que quedar esperando a que quede libre la vía. Lo que sucede es que los usuarios pueden moverse a la siguiente fase de su programa y el mensaje regresa diciendo que el servidor estaba ocupado. Inmediatamente, se vuelve a hacer otro intento para establecer una conexión. Mientras los RPCs tiene la ventaja de que emplean llamadas procedimentales que son familiares a los programadores, los mensajes en fila demandan un nuevo modelo de programación. A causa de que esta es una tecnología de reciente aparición, se encuentran pocas aplicaciones

que operen en ella. Algunos de los fabricantes que han lanzado al mercado middleware de mensajería son: COVIA Technologies, Digital Equipment Corp., IBM, Peerlogic, Horizon, Momentum y Systems Strategies.



(1) La aplicación manda los requerimientos a una fila local. (2) El mensaje viaja de una fila local a una fila remota. (3) La fila remota pasa el mensaje a la aplicación a fin de que sea procesada. (4) La aplicación regresa una respuesta.

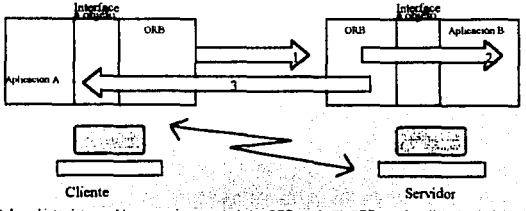
Fig. 4.8 Rutina de envio de mensajes en fila

c) Implementación de Middleware por medio de corredores de requerimientos a objetos (ORB).

La tecnología de más reciente desarrollo aplicada al middleware es la Object Request Broker (ORB). Un ORB es un software que maneja el intercambio de mensajes entre objetos a través de la red actuando como agente intermediario entre dos entidades. Como lo muestra la figura 4.9, el campo ORB requerido desde la aplicación cliente, encuentra e invoca al objeto remoto que satisface todos los requerimientos del objeto cliente, pasa la solicitud a éste y después regresa el resultado a el respectivo objeto cliente.

Los ORBs ocultan la complejidad de accesar a las aplicaciones remotas u objetos. Las aplicaciones sólo necesitarán saber el nombre del objeto al cual invocarán y los parámetros que se sustituirán dejándose todo lo demás a los ORBs. Los ORBs usan RPCs como un mecanismo de transporte para manejar interacciones entre objetos de la red. De manera semejante a los mensajes en fila, los ORBs soportan un alto nivel de APIs los cuales ocultan las complejidades de la red y de su sistema operativo de los ojos de los programadores. La principal ventaja de esta tecnología es la misma que la de todas las aplicaciones orientadas a objetos: mayor facilidad y rapidez en la actualización y mantenimiento de sus componentes. Sin embargo a causa de que su uso se basa en el manejo y manipulación de objetos son difíciles de entender, sobre todo para los desarrolladores que no estén familiarizados con la programación orientada a objetos (POO).

Los principales proveedores de middleware basado en ORBs son: HP, Digital Equipment Corp., IBM, Hyperdesk Corp., Iona Technologies y SUNSOFT.



(1) Los objetos intercambian mensajes a través de los ORBs. (2) El ORB pasa la solicitud a la aplicación remota. (3) Se retorna el resultado a la aplicación llamante.

Fig. 4.9 Object Request Broker (ORB)

La tabla 4.2 resume todo lo expuesto anteriormente:

Tipos de Middleware	Ventajas	Desventajas
Por RPC	• Excelente en el manejo de paquetes de aplicaciones. • Utiliza programación procedural. • Provee facilidades de programación y de seguridad. • Es una tecnología madura.	• Tiempo de respuesta (se debe esperar una respuesta para continuar un proceso). • Muy pobres en el envío de mensajes del estado del sistema. • Asume que los recursos de la red se encuentran siempre disponibles.
Por Mensajería	• No existe el tiempo de respuesta de los RPC. • Uso de un alto nivel de APIs • Permite manejar prioridades. • Da la posibilidad de mezclar diversos modos de comunicación (asincrónica, sincrónica, conversacional).	• Constituyen un modelo totalmente nuevo de programación. • Es una tecnología poco desarrollada. • Existen pocas aplicaciones que trabajen bajo esta técnica.
Por medio de ORBs	• Soporta un alto nivel de APIs • Manejo de un API estándar por el que el cliente puede acceder a un ORB. • Maneja RPC como mecanismo de comunicación. • Fácil de mantener.	• Tecnología poco desarrollada. • Difícil de entender; se necesita conocer métodos de desarrollo orientado a objetos. • Existen pocas aplicaciones que trabajen bajo esta técnica. • Puede haber efectos secundarios en el desempeño de la red.

4.2.3 Consideraciones para la implementación del Middleware.

El middleware no es la panacea para todas las necesidades en los ambientes de computación distribuida. En la mayoría de los casos el middleware permanece aún inexplotado, causa que origina que no existan muchos estándares para el mismo. Las dos instituciones que más se han destacado en la adopción y desarrollo de estándares para el middleware son:

- La Open Software Foundation (OSF), la cual es desarrolladora de sistemas para ambientes de computación distribuida (DCE/Distributed Computing Enviroment) y

de manejadores de ambientes distribuidos (DME/Distributed Management Environment).

- La Object Management Group (OMG/ Grupo de Administración de Objetos), que se caracteriza por ser una de las instituciones que más ha contribuido al desarrollo de estándares para mecanismos de computación distribuida basados en objetos.

En la selección de un middleware el usuario deberá distinguir de entre los tipos de middleware existentes aquel que mejor se ajuste a las necesidades de su organización; y lo más importante, el impacto que puede tener sobre el desempeño de la red. Los productos de middleware deben de ser capaces de liberar a los desarrolladores de tener que preocuparse por los intrincados mecanismos que emplean las redes en su comunicación. Lo anterior se puede llevar a cabo estableciendo una interface estándar para intercambiar datos entre todas las aplicaciones. Por ejemplo, el middleware de mensajería provee cuatro comandos básicos (SEND, RECEIVE, CONNECT y DISCONNECT) para el intercambio de datos entre todas las aplicaciones. También un buen middleware debe de realizar servicios tales como el manejo y tratado de errores, el almacenamiento de datos temporales y demás servicios a nivel sistema operativo de una manera transparente para el usuario.

El middleware debe de ser capaz de permitir a los usuarios el mover una aplicación de un ambiente de red a otro sin necesidad de modificar el código fuente.

Una última consideración es que un buen sistema de middleware debe de ser capaz de soportar tanto sistemas grandes como pequeños y correr lo mismo en

computadoras poderosas que en no tan poderosas. De igual manera, es importante considerar las facilidades que preste el middleware para escalar sistemas distribuidos pequeños a sistemas distribuidos de mayor magnitud.

4.2.4 Futuro del Middleware

Como ya lo hemos venido exponiendo el middleware, a pesar de todas las bondades que presenta, esta poco difundido o es muy poco comprendido. En gran parte esto lo presenta debido a que los ambientes de computación distribuida aún se encuentran en pleno crecimiento y desarrollo y mientras estos no se consoliden, el middleware, que fue creado para servir de soporte a estos últimos, no alcanzará la potencia de la cual es capaz.

Son dos las áreas en que el middleware está teniendo una gran influencia. En primer lugar, en el área de las redes, pues hace posible que sistemas y protocolos distintos trabajen conjuntamente. En segundo lugar, gracias al middleware, cualquier base de datos frontal es capaz de trabajar con cualquier base de datos de fondo (en los ambiente multitareas una base de datos frontal es aquella que tiene mayor prioridad, mientras que ésta es menor para las de fondo).

"Aunque el middleware para conectividad de bases de datos ha recibido mucha atención en toda la industria, los expertos consideran que el aprovechamiento del middleware en redes constituye un mercado que está a punto de explotar y

que tendrá grandes repercusiones en la computación cliente-servidor. El middleware para redes está diseñado para simplificar mucho la vida de los desarrolladores de aplicaciones."[5]

Supongamos que un programador desea escribir un programa para el procesamiento de transacciones en línea para una compañía que tiene tres tipos distintos de redes, por ejemplo, SNA, TCP/IP e IPX. El desarrollador tendría que escribir tres programas diferentes. Sin embargo, gracias al middleware, basta con que el desarrollador escriba para la capa de middleware -que da soporte a los tres protocolos- para que la implementación corra en todas las redes.

Respecto al Middleware Frank Dzubeck, presidente de la firma consultora Communications Networks Architects, comenta: "*El middleware representa un cambio completo de paradigma respecto a la forma como se escribirán las aplicaciones distribuidas. El mercado apenas se está desarrollando, pero en los años próximos habrá muchísima acción*"[6]

De acuerdo a la firma Tucker/Deboever Technologies, para 1996 el mercado para el middleware se extenderá hasta alcanzar la cifra de 12 000 millones de dólares[7], distribuidos de la siguiente manera:

- Paquetería de aplicación-> 6000 millones

[5] Moeller, Mike; Personal Computing México; Agosto 1994; Pag.43-45.
[6] Boletín Semanal Communication Networks Architects, semana dos, Febrero de 1995.
[7] Fuente: Tucker/Deboever Technologies, Computer Systems Series, Diciembre de 1994.

- Bases de datos frontales -> 3000 millones
- Servidores -> 2000 millones
- Herramientas -> 1000 millones

Cifra que según la misma firma crecerá a razón de 1200 millones por año, esta cantidad nos da una idea de la importancia que tiene el middleware, importancia que se irá haciendo más grande a medida que se implementen sistemas de computación distribuida tales como los de cliente-servidor.

En lo que se refiere a las bases de datos, los gerentes MIS (Manager Information Systems), tienen los ojos puestos en el middleware. Por primera vez, los gerentes podrán comprar una base de datos frontal, como Paradox o FoxPro, sin tener que preocuparse por su conectividad con bases de datos de fondo como Oracle, Sybase o Informix. Los dos programas middleware más importantes que se están desarrollando por el momento son ODBC (Open Data Connectivity), de Microsoft, e IDAPI, desarrollado por un consorcio que integran Novell, IBM, WordPerfect y Borland. Ambos productos se ajustan a los estándares que desarrollaron ANSI y X/Open. Tanto ODBC como IDAPI suministran una interfaz para accesar datos en un ambiente heterogéneo de sistemas para el manejo de bases de datos relaciónales y no relaciónales. Las dos capas de middleware están basadas en la especificación Call Level Interface de SQL Access Group y ofrecen un método para accesar datos que es independiente de cualquier proveedor. Los datos pueden estar almacenados en

diversas bases de datos propietarias para PC, minicomputadoras y mainframes.

Una diferencia importante entre ODBC e IDAPI es que esta última se conecta tanto con bases de datos navegacionales como de conjunto, mientras que ODBC trabaja únicamente con las de conjunto. Las bases de datos de conjunto se diferencian de las navegacionales en la forma como el usuario busca la información. En una base de datos de conjunto como es Access, los usuarios elaboran una lista de detalles y hechos que hay que localizar a la hora de consultar una base en busca de un ítem. En una base de datos navegacional como Paradox, el usuario nada más tiene acceso a una base y le "echa un vistazo" a los datos hasta que encuentra el ítem. Otra diferencia entre ambas es que IDAPI permite que los usuarios consulten dos bases de datos al mismo tiempo, mientras que los usuarios de ODBC nada más pueden consultar una.

De esta manera el middleware viene a simplificar la vida tanto de desarrolladores como de usuarios; si tomamos en cuenta que el futuro de la computación es constituirse en sistemas distribuidos como el de cliente-servidor, entonces el middleware es una tecnología que aún dará mucho de que hablar.

Capitulo 5

Implementación y Perspectivas de los Sistemas de Computación Distribuida.

5.1 Implementación.

5.1.1 Consideraciones para la implementación de un sistema de computación distribuida.

Ninguna discusión acerca de la computación distribuida (bases de datos distribuidas, procesos cooperativos o tecnología cliente-servidor) estará completa sin considerar sus facilidades de implementación. Examinemos ahora las facilidades que encontramos en una estación de trabajo en las implementaciones de los sistemas distribuidos; para nuestra discusión una estación de trabajo o workstation puede ser cualquier máquina tipo PC debidamente adaptada.

Las workstations vienen provistas de tal manera que son ideales para usarlas en aplicaciones tipo monousuario (stand-alone) ya que cuentan con disco para realizar almacenamiento local. Recientemente también han sido empleadas en aplicaciones del tipo cliente/servidor basadas en su facilidad para realizar acceso remoto, compartición de archivo, capacidades gráficas y su velocidad de procesamiento.

Tres son las áreas que han hecho de las computadoras tipo PC hayan alcanzado alto desarrollo en aplicaciones distribuidas: la habilidad para tener múltiples aplicaciones ejecutándose concurrentemente con acceso a recursos comunes tales como archivos o bases de datos; la habilidad para trabajar bajo sistemas operativos diferentes; y una interface gráfica de usuario (GUI/ Graphic User Interface).

El poder de procesamiento es factor muy importante para obtener un desempeño satisfactorio en la computación distribuida. En la última década el poder de procesamiento se ha venido aumentando hasta en un 300 por ciento, al igual las aplicaciones han pasado de ser de 8 bits a 32 bits, lo que ha aumentado considerablemente el poder de la máquina. Con el advenimiento de la quinta generación de computadoras y por la entrada al mercado de la tecnología RISC (Reduced Instruccion Set Computer/ Conjunto Reducido de Instrucciones de Computadora), las estaciones de trabajo se han vuelto capaces de ejecutar rangos de instrucciones a razones de 100 MIPS (Millions of Instruccion Per Second/ Millones de Instrucciones Por Segundo) por cada pastilla de procesador. Si a lo anterior le aunamos las facilidades del procesamiento en paralelo estaremos hablando de sistemas distribuidos con velocidad de procesamiento casi instantáneo. Así el poder de procesamiento ha ayudado a que los sistemas distribuidos hayan tomado cada día más fuerza ya que no existen limitantes de velocidad.

Otro factor importante a tomar en cuenta en los sistemas distribuidos es la memoria. En tales sistemas se emplean memorias que tienen tiempos de acceso alrededor de los nanosegundos ya que de lo contrario se podría alentar drásticamente al sistema. También la capacidad de almacenamiento primario es factor importante, actualmente los sistemas han pasado de tener 64 k bytes de almacenamiento a contar con memoria extendida y expandida que han logrado proporcionar hasta 64 MB de capacidad de almacenamiento primario direccionable. Las técnicas del empleo de caches, como lo es la de cache de disco también han permitido mejorar el rendimiento en cuanto a las capacidades de almacenamiento primario. Los sistemas RISC, debido a que ejecutan un número muy pequeño de instrucciones, requieren sólo un número mínimo de ciclos de máquina para ejecutarlas; lo cual significa que más instrucciones puedan ser ejecutadas por segundo en cada ciclo de reloj, por lo que también menores ciclos de reloj son necesitados y la memoria permanece con mayor espacio direccionable disponible.

Las estaciones de trabajo son extremadamente flexibles en lo que se refiere a su capacidad gráfica ya que pueden estar provistas con capacidades de alta resolución y soportar una amplia gama de colores. El poder de las estaciones de trabajo también incluye capacidades gráficas de gran poder y velocidad, las cuales son muy importantes en aplicaciones de índole compleja.

El último factor a tener en cuenta en la implementación de los sistemas distribuidos es la capacidad de almacenamiento secundario de las estaciones de

trabajo. Si una máquina de éste tipo se encuentra formando parte de un sistema cliente-servidor y constantemente recibe peticiones, ya sea de entrada o salida de información, necesitará tener un espacio de almacenamiento lo suficientemente grande para poder almacenar todos los datos que maneja. Al igual si se tienen aplicaciones que hacen uso intensivo de las operaciones de entrada/salida, entonces podremos mejorar el desempeño de éstas mediante el empleo de la RAM de disco.

La capacidad de almacenamiento que se requiera dependerá de los requerimientos de los usuarios. No se ha definido una interface estándar para los dispositivos de almacenamiento de las workstations, pero una interface IDE (Intelligent Drive Electronic/ Manejador de Disco Inteligente) o una SCSI (Small Computer System Interface/ Interface de Disco de Computadora Pequeña) pueden ser consideradas como los estándares ya que la mayoría de los sistemas distribuidos las emplean. La interface IDE es más usada cuando se requieren grandes cantidades de almacenamiento, mientras que la SCSI encuentra su mayor aplicación cuando se tiene acceso a otros periféricos de almacenamiento tales como unidades de CD-ROM o unidades de cinta.

5.1.2 ¿Porqué implementar sistemas distribuidos?

Hay muchas razones para la implementación de un sistema distribuido. Los recursos disponibles para una aplicación no están limitados a los de una sola máquina. Por ejemplo una aplicación de diseño asistido por computadora

(CAD/ Computer Aided Design) puede requerir excelentes capacidades gráficas, gran cantidad de espacio disponible para almacenar datos y un procesador lo suficientemente rápido como para realizar simulaciones intensivas. Adquirir todas estas capacidades en una sola computadora puede resultar muy costoso. Alternativamente se pueden comprar varías computadoras, de menor precio, y conectarlas a través de un sistema de red a fin de reducir los costos; una aplicación distribuida utilizará las computadoras sobre la red tal y como si se tratará de una sola computadora. Como el uso de las aplicaciones distribuidas crece cuando se incrementa el número de usuarios, el poder de computación también podrá incrementarse agregando nuevas estaciones a la red.

Las aplicaciones distribuidas tienen la ventaja de ser tolerantes a las fallas. En las redes comunes tales como las de área local (LAN), el compartimiento de recursos y el acceso a los mismos está limitado. Si se tiene una única impresora conectada a una computadora que se encuentra fuera de servicio, las demás computadoras estarán imposibilitadas de producir información impresa. En los sistemas distribuidos las aplicaciones son diseñadas para accesar los recursos sobre otras computadoras de manera independiente a las computadoras sobre las que "cuelgan" tales recursos.

5.1.3 Los ambientes bajo sistemas de computación distribuida.

Los sistemas distribuidos están provistos por un número específico de servidores ya que debido a que se requiere que las solicitudes de aplicaciones

locales sean totalmente transparentes para el usuario no es posible construir modelos de aplicación muy específicos.

Los mainframes son las máquinas predilectas en cuanto a servidores se refiere; IBM ha dominado el mercado de los servidores durante largo tiempo, el problema principal que se llega a presentar entre las máquinas IBM y las workstations es que las primeras manejan código EBCDIC y las segundas código ASCII, problema que aparentemente ha sido superados gracias a las rutinas de conversión de archivos que últimamente han fructificado.

La aplicación más común que se le da a los mainframes es en los procesos cooperativos con los que llega a formar lo que en inglés se le conoce como SQLserver. El SQL (Structure Query Lenguage/ Lenguaje de consulta estructurado) es la plataforma común para los lenguajes de cuarta generación y es provisto por muchos de los principales fabricantes tales como IBM, Unisys o NCR.

Otra aplicación común para los mainframes es que funjan como servidores. En éste caso el mainframe cuenta con una base de datos o una serie de aplicaciones las cuales están destinadas a servir a un grupo de usuarios que el mainframe deberá atender. Un uso frecuente de esto lo encontramos en los correos electrónicos los cuales permiten la comunicación entre los empleados.

Aparte de las ya mencionadas entre los mainframes y las estaciones de trabajo encontramos otras diferencias como lo es el mayor número de canales de entrada/salida que presentan los mainframes con respecto a una workstation y

un mayor espacio de almacenamiento por parte de los mainframes, lo que los hace accesibles a grandes aplicaciones.

Una arquitectura distribuida generalmente podrá ofrecer los siguientes servicios:

Servicios de ejecución distribuida.- es la habilidad para ejecutar módulos de programa sobre sistemas remotos como parte de la ejecución de un programa local.

Servicios distribuidos de interface de usuario.- es la habilidad para dar una interface consistente a un operador independientemente de si el dispositivo de acceso está siendo utilizado.

Servicios de nombre distribuido.- es la facilidad que se tiene para localizar un servicio que es requerido por un usuario, a través de su nombre, a fin de satisfacer sus necesidades.

Servicios de sincronización de tiempo distribuido.- es la capacidad para sincronizar todas las aplicaciones manteniendo la entera consistencia.

Servicios de archivos distribuidos.- Es la facilidad para dar a los programas de aplicación acceso a recursos y bases de datos remotos, de una manera tal, como si se tratara de un acceso local.

Servicios de seguridad.- es la habilidad para controlar las operaciones y permitir el acceso a determinados servicios dependiendo del tipo de usuario.

5.1.4 Ejemplos de Computación Distribuida en arquitecturas de aplicación.

Hoy en día encontramos que la gran mayoría de las compañías fabricantes de hardware están proveyendo a sus equipos con la arquitectura necesaria para la implementación de sistemas distribuidos. IBM por ejemplo provee la arquitectura SAA (Systems Application Architecture/ Arquitectura de Aplicación de Sistemas), Digital ha desarrollado su arquitectura NAS (Network Application Support/ Soporte para Aplicaciones en Red) y NCR tiene la arquitectura OCCA (Open Cooperative Computing Architecture/ Arquitectura de Computación Abierta Cooperativa). Estas arquitecturas generalmente permiten a los usuarios de las workstations programables tener acceso a servicios de sistemas de computación remota.

Existen dos estándares que tratan de normalizar a los ambientes que operan bajo computación distribuida. El primero de ello es ATLAS creado por UNIX International, el segundo y el más utilizado, es el de Open Software Foundation conocido como Distributed Computing Environment (DCE), el cual soporta los siguientes ambientes:

- HP/Apollo NCS, llamadas a procesos remotos para ejecución distribuida.
- Sistema Digital's DECdns y X.500
- Sistema MIS's KERBEROS para sistemas de seguridad
- Sistemas Digital's CMA (Concert Multithread Architecture) para sincronización de procesos distribuidos.
- Sistemas Digital's DECdts para servicios de tiempo.
- Sistemas LAN Manager/X para integración de computadoras personales.

El ambiente DCE en un conjunto integrado de servicios que soporta el desarrollo, uso y mantenimiento de aplicaciones distribuidas. La disponibilidad de un conjunto uniforme de servicios, en cualquier parte de la red, dan la posibilidad a las aplicaciones, de aprovechar aspectos inusuales en muchas de las redes de computadoras. DCE es también un sistema operativo, el cual provee compatibilidad con todos los ambientes existentes. OSF se ha preocupado por dar a los ambientes que trabajen bajo DCE una apertura total (Open sistems), lo cual significa que en estos sistemas se pretende alcanzar una compatibilidad completa, es decir, son sistemas que pueden estar formados tanto por software como por hardware de distintos fabricantes sin ningún problema para los usuarios de los mismos. El ambiente DCE incluye poderosas herramientas para el desarrollo de aplicaciones distribuidas, entre estas podemos mencionar: llamadas a procesos remotos (RPC), servicios de directorio, servicios de tiempo y servicios de seguridad.

Los *servicios de directorio distribuidos* permiten a los usuarios identificar, mediante un nombre, los recursos tales como servidores, archivos, discos o impresoras y tener acceso a estos sin tener que conocer en que partes de la red se localizan. Como resultado, el usuario puede continuar refiriéndose al recurso por su nombre, aún cuando su dirección dentro de la red cambie.

Los *servicios de tiempo*, por su parte, permiten que diferentes componentes de las aplicaciones distribuidas puedan obtener el tiempo de los relojes situados en computadoras diferentes. Un servicio distribuido de tiempo regula los ciclos de reloj de cada computadora de la red, a fin de que todos los elementos del

sistema trabajen de la manera más sincronizada posible. El servicio de tiempo distribuido de DCE, es un software que sincroniza el reloj de cada computadora a un tiempo común. Para sincronizar los relojes de computadoras situadas en localidades muy distantes se utiliza el protocolo NTP (Network Time Protocol/ Protocolo de Tiempo en Red), el cual es un conjunto de especificaciones sobre como regular el tiempo según el uso horario de la localidad donde este ubicada la computadora. En los ambientes de red más convencionales, el sistema operativo verifica la identidad del usuario y le autoriza el acceso a los recursos; sin embargo, en un ambiente de computación distribuida, donde las actividades se realizan bajo la vigilancia de múltiples sistemas operativos, se requiere que las actividades de verificación y autorización sean tratadas de manera independiente al sistema operativo a fin de que puedan ser ejecutadas sobre diferentes host. El *servicio de seguridad* de DCE proporciona tales facilidades. Una de las ventajas de los sistemas DCE es que permiten que bajo una misma red de computadoras se tengan conectadas PCs y Macintosh, máquinas que toda persona que haya trabajado con ellas sabrá que por naturaleza son incompatibles. DCE también soporta los estándares del modelo OSI y de Internet, tales como el protocolo de transporte TCP/IP y de tiempo NTP. Este ambiente de computación distribuida fue escrito utilizando C estándar y puede correr bajo los principales sistemas operativos para redes: OSF/1, AIX, DOMAIN OS, ULTRIX, HP-UX, SINIX, SunOS y UNIX. Además, DCE también tolera trabajar con otros ambientes que ofrecen servicios similares, pero que manejan diferentes interfaces, tal como es el caso de VMS, OS/2 y otros sistemas operativos.

5.1.5 El mercado de las aplicaciones distribuidas.

Las industrias líderes en el sector de la computación distribuida son las que elaboran software para bases de datos. Entre ellas se encuentran Oracle, Informix Software Inc., Progress Software, Ingres, Cyborg, SAP y Sybase. Se pueden dividir en dos grupos los programas para elaborar bases de datos: uno de ellos organiza datos corporativos en los servidores, y el otro corre en PC para permitir que los programadores adapten el software y que los empleados hagan preguntas sobre la base de datos. La base de datos desempeña un papel muy importante en la computación con cliente-servidor debido a que es donde se encuentra almacenada la información corporativa, por lo que muchos usuarios tienen acceso a ella en forma expedita.

Novell, Microsoft y Banyan desempeñan un papel muy importante en el mercado de aplicaciones distribuidas bajo cliente-servidor debido a que sus productos actúan como ladrillos en las redes con este ambiente. El software de sistemas operativos para redes, como es el caso de NetWare de Novell y Vines de Banyan, corre en servidores. Se encarga de controlar el movimiento de los documentos electrónicos y de distribuir aplicaciones a los clientes.

Muchos criterios deben ser tomados en cuenta para decidirse por elegir uno de los productos de los fabricantes anteriores ya que la mejor opción dependerá en gran medida de los objetivos que se hayan trazado. Respecto a este asunto Joe Armitage, integrador de sistemas de la compañía Montare International

menciona "Primero hay que escoger el software, porque es lo que determina los requerimientos del hardware"[1]. Armitage menciona que, para muchos usuarios pequeños y medianos, las compañías adquieren la base de datos SQL Server de Microsoft con Microsoft Acces como procesador frontal en un ambiente con Windows NT. Acces hace que los usuarios se sientan más cómodos, en vista de que tiene la conocida GUI de Microsoft Excel. Armitage asegura que, de acuerdo a su experiencia, las compañías menores a los 500 empleados eligen los recursos computacionales con base en el precio. Es decir, las personas que toman las decisiones deben buscar el mayor rendimiento y confiabilidad dentro de las limitaciones presupuestales que tengan. Además, Armitage recomienda hardware como Compaq ProLiant, IBM PC Server o los servidores NEC ExpressRISC. Los anteriores son máquinas recomendadas para compañías medias y que tengan planes de expansión, pero también se ajustan a pequeñas empresas que se encuentren en crecimiento. Si hay muchas restricciones de presupuesto, debe dársele prioridad al servidor por encima del cliente, pues aquél debe ser la parte más poderosa de la red. No debe haber ningún problema con Windows NT, pues Microsoft tiene una lista de compatibilidad de Hardware. Si estamos hablando de una empresa muy pequeña que se encuentra creciendo y que desee implantar un sistema basado en cliente-servidor, se recomienda como sistema operativo a Microsoft Windows para trabajo en grupo ya que es muy conveniente como parte de una vía migratoria para la compañía en crecimiento.

[1] Soluciones Cliente-Servidor; Personal Computing México; Agosto 1994, Año 6, No. 74; pag. 38-40.

La figura 5.1 muestra el posicionamiento en el mercado de las seis compañías que liderean el mercado para cliente servidor en el área de administración de recursos humanos, área que es considerada como una de las más fuertes dentro de las compañías que trabajan en el campo de la computación distribuida.

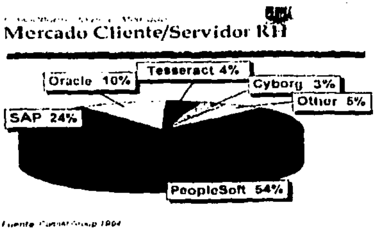


Fig. 5.1 Las seis compañías líderes en aplicaciones cliente-servidor de RH

Por otro lado la tabla que 5.1 muestra a las principales compañías que en nuestro país se han destacado por trabajar ya con sistemas distribuidos.

Clientes

AT&T	ALCOA	Astra/Merck
Banamex	BP Chemicals	Black & Decker
Cannon Computers	Cervecería Polar	Chemical Bank
Ciba-Geigy	Citicorp Latino	CompuServe
Cray Research	Data General	Deutsche Bank
Domino's Pizza	Eastman Kodak	Eli Lilly
Ernst & Young	Eveready Battery	Ford
Gerber	The Gillette Co	Gov. Fed. Canada
Hewlett-Packard	Fondo Monetario Int	Kmart Corp
Levi Strauss	Lotus Development	Malaysia Airlines
Metropolitan Life	Microsoft	NY Stock Exchange
Novell	PepsiCola Co	Pizza Hut
[illegible]	Ricoh Corp	Sony Corp
Sequent	Shell Oil	Warner Bros
Sybase	Tandem Computers	The Coca-Cola Co
United Airlines	Unisys	Whirlpool

Tabla 5.1. Los principales clientes de sistemas distribuidos en México.

FALLA DE ORIGEN

5.1.6 El Software para ambientes distribuidos.

Muchísimos proveedores participan o intentan participar en el sector de la computación distribuida, atraídos por su rápido crecimiento. Este dinamismo se debe a que las compañías hacen esfuerzos desesperados por modernizarse y competir en los mercados internacionales. En nuestro país podemos disponer de casi todos estos productos que nos ofrecen las diversas instituciones que se dedican al desarrollo de software y la adquisición y modernidad de éste sólo estará sujeta a los límites presupuestales que tenga cada compañía.

En el mercado han aparecido muchísimas aplicaciones para ambientes distribuidos, predominando aquellas para cliente-servidor. Las compañías que se dedican a la fabricación de éste son cada ves más numerosas y su número sigue en aumento. De acuerdo con la firma Internacional Data Corporation (IDC/ Corporación Internacional de Información), el propio dinamismo del mercado impulsará el crecimiento de nuevas compañías. Según cifras proporcionadas por IDC, en 1993 el mercado de mantenimiento y el de software financiero tan sólo para cliente-servidor tenía un valor aproximado de $270 millones de dólares. La misma firma pronostica que en los próximos cinco años este sector tendrá un crecimiento del 44% y llegará a los $1,650 millones de dólares.

Entre los desarrolladores de software para los ambientes distribuidos podemos mencionar a *PeopleSoft* que fue la primera compañía que lanzo al mercado productos para Cliente-Servidor, y lo sigue haciendo. Entre la gama de productos que ha elaborado se encuentra PeopleSoft Distribution, un grupo de

cuatro aplicaciones para cliente-servidor totalmente integradas. Este producto esta enfocado a la administración y distribución de materiales, y cuenta con módulos de compras, inventario, procesamiento de ordenes y facturación. Esta línea de productos es una extensión de PeopleSoft Financials, y funciona en forma autómata, en combinación con otras aplicaciones de PeopleSoft (People Soft Human Recourse Management System, Financials, Tools, Distribution) o bien se puede enlazar con otros sistemas que no son de PeopleSoft. El software corre en Hardware de DEC, HP, IBM, NCR, Sequent y Sun Systems.

Aunque la compañía PeopleSoft sigue ocupando actualmente uno de los primeros lugares, en cuanto a producción de software de cliente-servidor, este sector es encabezado, según especialistas en el área, por Oracle y SAP América Inc. Tales compañías son las que más a menudo han logrado introducir, con éxito, software cliente-servidor al mercado.

Mención especial merece FlexiWare Corp., la cual fue la primera empresa que desarrolló y sigue desarrollando software basado en la tecnología de objetos. Sin embargo, IDC considera que tendrán que pasar unos cuantos años más antes de que se note la influencia del desarrollo orientado a objetos, pues los usuarios todavía no comprenden cabalmente los beneficios de esta tecnología.
La compañía Lawson Software suministraba software para mainframes, pero ahora ofrece muchos productos para ambientes distribuidos, como algunos de contabilidad , recursos humanos, manejo de la distribución y de materiales.

Los productos corren en las plataformas, sistemas operativos y bases de datos más importantes, incluyendo Windows NT.

Otras compañías que ofrecen software para sistemas de computo distribuido son: Computer Associates, Attachmate Corp., Cincom Systems, Cross Access, Cyborb Systems, Encore, FourGEn Software, PowerSoft, IBM, KnowledgeWare, Progress y Ross Systems.

Las aplicaciones distribuidas empezaron por sistemas operativos y después abarcaron el campo de las bases de datos del tipo relacional. Lo último en software para sistemas distribuidos del tipo cliente-servidor, son los productos para la elaboración y manejo de presupuestos. Este software representa una alternativa a los antiguos métodos de hojas de cálculo de cálculo como Excel o Lotus 1-2-3.

Los usuarios coinciden en que la mayoría de las compañías han venido utilizando las hojas de cálculo para el manejo de sus presupuestos. Asimismo, han tenido que hacer programación de macros para pasar de los presupuestos departamentales a los divisionales y de éstos a los corporativos. Compañías como Pillar Corp. y Comshare admiten que esta tendencia es demasiado complicada y han decidido adoptar un enfoque diferente. En base a esto la compañía Pillar ha lanzado al mercado el paquete FYPlan el cual sirve para llevar las finanzas en toda empresa y que se ejecuta en ambiente cliente servidor de Windows O Macintosh.

La figura 2.2 muestra a las principales compañías que se han destacado en la producción de software cliente-servidor, clasificadas tanto por amplitud de visión como por poder de ejecución.

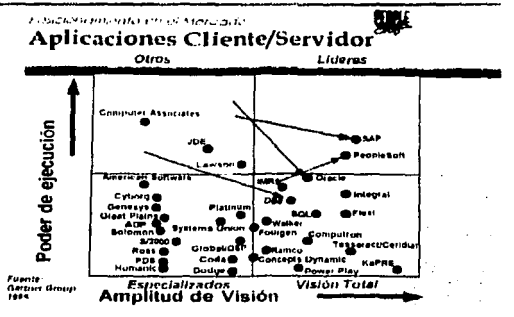


Fig. 5.2 Las principales compañías en la producción de aplicaciones cliente-servidor.

En la figura 5.3 observamos a las seis compañías que más aplicaciones han lanzado al mercado de cliente-servidor.

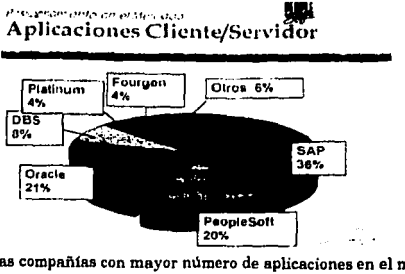


Fig. 5.3 Las compañías con mayor número de aplicaciones en el mercado.

5.1.7 El Hardware para ambientes distribuidos.

Algunas de las empresas más importantes que participan en el mercado del hardware para los ambientes de cómputo distribuido son Compaq Computer, Hewler Packar, DEC, Tricord, AST, IBM, NEC y NetFRAME.

La compañía NetFRAME Systems Inc. de Milpitas, California, lanzó el servidor NF250ES Network Mainframe, a un precio de alrededor de los $14,000 dólares. Este servidor esta diseñado para LAN de departamentos medianos o de sucursales, cuyas consideraciones fundamentales sean la confiabilidad y el crecimiento.

El sistema básico de NF250ES incluye una CPU Intel 80486DX a 50 MHZ como procesador central y un Intel 80C188 como procesador de servicio. La máquina presenta memoria con código para la corrección de errores, verificación de paridad en todas las vías para datos y Channel-Start, que protege a los usuarios automáticamente de los errores intermitentes que haya en el sistema. En caso de que falle un proceso de entrada o salida debido a un error intermitente, Channel-Start reinicia automáticamente el proceso fallido sin que tenga que intervenir el usuario y sin que se produzca interrupción alguna.

Los servidores de aplicaciones son módulos para servidores de aplicaciones virtuales que se pueden agregar a los recursos especializados para labores de gran intensidad, como las bases de datos. Cada uno de los procesadores de aplicaciones tiene su CPU y memoria, y ejecuta su propia copia de un sistema

operativo (Unix SVR4.2, NetWAre o Windows NT). Las máquinas de NetFrame ejecutan diferentes sistemas operativos en el mismo servidor y al mismo tiempo. El servidor de aplicaciones 486 incluye un Intel 486 de 50MHZ con memoria ECC de entre 16 MB y 128 MB, y memoria cache de 128 KB.

NEC Technologies Inc. (NECT) es otro de los proveedores de hardware que dan apoyo al sistema operativo Windows NT con sus sistemas Image RISCstation y EXpress RISCserver.

La familia de plataformas para servidores y estaciones de trabajo es de alto rendimiento y esta basada en el procesador VR4400 MIPS RISC. Está diseñada para Windows NT y aprovecha totalmente la funcionalidad inherente a este sistema. Incluye recorrido múltiple de entrada y salida, y multiprocesamiento directo, según la propia compañía.

Entre las compañías de software que ya desarrollaron aplicaciones para el ambiente NT y se ejecutan en Image RISCstation y Express RISCserver, se encuentran Microsoft, Executivo Sotfware, SQL Business System Inc. y Welcom Software Technology.

5.1.8 Ventajas y Desventajas de los ambientes distribuidos.

Las ventajas más importantes son:

-Permiten un mejor aprovechamiento de la potencia de computo de los equipos al poder descargar en diversas máquinas parte de la carga del trabajo que antes era realizado por un sólo CPU.

-Permite el acceso de una máquina cliente a uno o varios servidores en forma simultánea, lo que resulta cada vez más importante en los ambientes heterogéneos actuales.

-Reduce el tráfico en la red, ya que por ella sólo viajan los requerimientos y las atenciones a ellos

-La arquitectura cliente/servidor es particularmente adecuada para los sistemas OLTP (On-Line Transaction Processing/ Procesamiento de Transacciones en Línea) ya que procesa los archivos maestros tan pronto como los recibe el ordenador.

- Los procesos Cooperativos ayudan a simplificar aquellas operaciones, que por su índole muy compleja, requieren de mucho tiempo de procesamiento haciendo que la tarea sea dividida y procesada en paralelo con un correspondiente ahorro en tiempo y recursos.

- Se pueden compartir recursos tales como impresoras, bases de datos, monitores gráficos, etc. o acceder a sistemas más poderosos.

-Puede y, de hecho, tiene que operar bajo sistemas abiertos, lo que significa capacidad de cambio de plataformas con un mínimo de problemas y riesgos

-Permite el uso de interfaces gráficas de usuario muy versátiles y amigables en los clientes.

Como desventajas tenemos que las aplicaciones distribuidas suelen ser más complejas que las tradicionales en la forma de anfitrión/terminales, y que exigen más de la red. En ambos casos, las desventajas pueden ser superadas mediante las ventajas anotadas, y ciertamente, todo tiene su costo.

Los expertos en seguridad informática mantienen, que bajo ciertos tipos de aplicaciones distribuidas la seguridad es más sencilla de romper. Pero la respuesta a esta aseveración es que la seguridad está sujeta al diseño y medidas de seguridad que se incorporen al ambiente, y a las aplicaciones desarrolladas para operar bajo éste tipo de sistemas.

5.1.9 Desarrollo de aplicaciones distribuidas.

Para el desarrollo de aplicaciones distribuidas se puede emplear un rango de herramientas que van desde lenguajes de tercera generación, hasta ambientes de desarrollo, no siendo exclusivo de algún tipo de herramienta, el desarrollo de las aplicaciones distribuidas.

Lenguajes de tercera generación son los lenguajes tradicionales con extensiones para su interfaz con la red de cómputo y con las bases de datos remotas. Ejemplos de este caso son: los lenguajes C y COBOL, con las bibliotecas de APIs adecuadas.

Lenguajes de cuarta generación son aquellos lenguajes manejadores de bases de datos tales como Informix, Oracle, Progress y Sybase; son más fáciles de aprender que los de tercera generación aunque sus requerimientos (en cuanto a capacidades del equipo) son mayores. Estos tipos de lenguajes pueden emplearse para el desarrollo de bases de datos distribuidas; también hay algunos que pueden operar con una variedad de bases de datos con los controladores correspondientes, Accell cae en este caso.

Cuando hablamos de ambientes de desarrollo nos referimos a algo mucho mayor que un lenguaje de cuarta generación. Un ambiente de desarrollo es un entorno de trabajo completo. Además de incluir un 4GL y su editor, también incluye un repositorio (generalmente basado en objetos) de procedimientos, mensajes de error, mensajes de ayuda, atributos de las entidades, etc. Además incluye herramientas de control de versiones, herramientas de diseño, herramientas de desarrollo de prototipos, herramientas de usuario final y utilerías de referencia cruzadas, por mencionar tan solo los recursos ofrecidos con mayor frecuencia.

Por otro lado, muchas herramientas CASE tienen compuertas con lenguajes de tercera y cuarta generación, además, con ambientes de desarrollo. Esto permite la comunicación de y hacia estas herramientas.

La tabla 5.2 resume algunos de los aspectos mencionados anteriormente. En primer lugar se muestran, en orden de importancia, los principales manejadores de bases de datos relacionales (RDBMS/ Relational Data Base Manager System) empleados en los ambientes distribuidos; posteriormente tenemos a los sistemas operativos más populares y finalmente a los equipos que más predominan en compañías que ya han instalado algún tipo de sistema distribuido.

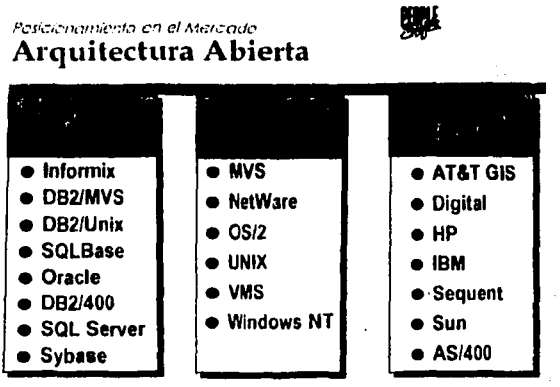


Tabla 5.2 Los líderes en la computación abierta

5.2 Perspectivas de los sistemas de computación distribuida.

5.2.1 La nueva Ola de la computación.

El impresionante crecimiento que han venido teniendo los sistemas de computación distribuida, hace pensar que en un futuro no muy lejano, la mayoría de las plataformas de cómputo existentes estarán conectadas entre sí formando todas ellas un gran sistema distribuido; en relación con este sistema de computación, en el que se tendrá una interoperabilidad total, ya algunos

autores han comenzado a referirse a él como la era de la **computación intergaláctica**.

Los equipos multiusuarios empezaron como un anfitrión con terminales tontas conectadas a ellos. En esta configuración, las terminales tontas actúan sólo como estaciones de entrada y salida y sin ninguna capacidad de cómputo asociada, dejando al anfitrión toda la carga de proceso y control. Los equipos maestro/esclavo constituyeron una etapa posterior en que la computadora maestra coordinaba el trabajo de los equipos esclavos, los que a su vez se encargaban de los equipos periféricos. La tercera etapa en los ambientes de cómputo es propiamente la de la computación distribuida, en la que el trabajo se divide entre los elementos que componen a un sistema. La interconectividad entre los elementos de esta tercera etapa es lo que esta dando origen a la era intergaláctica. La figura 5.1 muestra aproximadamente la fecha de inicio de cada etapa. En las primeras aplicaciones se utilizaron servidores de archivos que son aquellos que permiten compartir archivos entre un grupo de trabajo. Cuando un cliente solicita un archivo el servidor envía una copia completa al cliente, incluyendo los índices del archivo. Al suceder esto, se bloquea el archivo hasta que el cliente lo libera. Los servidores de datos y de computo pertenecen a una etapa posterior, el servidor de datos se encarga de proporcionar los datos que el servidor de computo le solicita, uno valida únicamente los datos y el otro ejecuta la aplicación. Por último tenemos a los servidores de bases de datos y de comunicaciones. Los sistemas cliente-servidor hacen uso de los servidores de bases de datos en los que se efectúan funciones

de validación, actualización, consulta, administración de recursos, etc. Los servidores de comunicaciones son comunes en los procesos cooperativos, sus funciones principales son el control de acceso, enrutamiento y puenteo de tráfico.

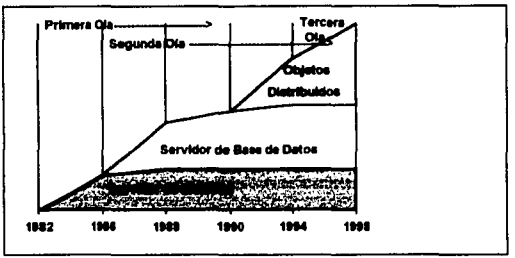


Fig. 5.1 Evolución de los sistemas multiusuarios, basándose en arquitectura cliente servidor.

5.2.2 Los nuevos Sistemas Operativos.

En la mayor parte de las plataformas de cómputo actuales encontramos que los sistemas operativos para estaciones cliente administran el escritorio, mientras que los sistemas operativos del servidor administran los recursos compartidos. Sin embargo, para la nueva era se requerirán sistemas operativos híbridos multitareas que desempeñen bien ambos trabajos. Tales sistemas deberán correr aplicaciones de 32 bits y se deberá contar con interfaces de usuario tal como la interface OOUI (Object Oriented User Interface/ Interfaz de Usuario

Orientada a Objetos), la cual es un sitio en el que se integran cosas múltiples que se ejecutan concurrentemente. Estas cosas son objetos en pantalla que representan a sus contrapartes del mundo real. Las tecnologías OLE2 y OpenDoc son ejemplos de interface OOUI, permitiendo a los usuarios ensamblar, escribir, guardar y vincular sus aplicaciones junto con las cosas que contengan.

Por otro lado, las estaciones cliente deberán de ser capaces de ejecutar miles de aplicaciones existentes (incluyendo aplicaciones DOS, OS/2, Windows y Macintosh), así como todos aquellos controladores de dispositivos que los usuarios hayan adquirido. También se deberá contar con tecnología Plug and Play (Conectar y Usar) por lo que los sistemas operativos deberán integrar el software medio necesario.

Los sistemas operativos de red siempre han tenido la tarea de ocultar la ubicación de los recursos a las aplicaciones. Sin embargo, en la era intergaláctica de la computación, esta tarea deberá ser el crear, ante el usuario, la ilusión de una imagen de un solo sistema en potencialmente, millones de máquinas híbridas conectadas a lo largo de todo el sistema. Para lograr tal ilusión los sistemas operativos deberán contar con las siguientes características:

Transparencia de ubicación. Debido a que los usuarios, los servicios y los recursos se conectan y se desconectan constantemente de la red no quedando nunca atados a ubicaciones fijas, se requiere que los sistemas proporcionen

siempre los mismos servicios sin importar la localización del que hace la petición. Para ello se requiere que el directorio global del NOS agrupe a las personas, los programas y los objetos para llevar a cabo una tarea. El directorio global es una base de datos de objeto distribuida y replicada. Distribución, significa que pueden existir los dominios administrativos autónomos. La réplica mejora la disponibilidad y el desempeño. La orientación a objetos permite al directorio desarrollarse en forma orgánica, como las estructuras del mundo real que representa.

Transparencia del nombre de lugar. Todo lo que contiene la red global debe tener la apariencia de pertenecer al mismo nombre de lugar. Es decir, se debe dar la apariencia que los elementos siempre están ubicados en la misma locación, aunque esto no sea así.

Transparencia administrativa. El NOS debe dar la apariencia de integrarse con los servicios de administración del sistema operativo local y proporcionar transparencia de réplica. Por ejemplo, si un directorio de nombramiento (aquel que contiene las direcciones físicas de los componentes de la red) se oculta en muchos sistemas, depende del NOS el sincronizar las actualizaciones. El NOS debe además proteger a los usuarios contra fallas de la red, administrar en forma transparente los reintentos y los reinicios de sesión, además de sincronizar los relojes en máquinas geográficamente dispersas.

Transparencia de acceso asegurado. Los usuarios deben ser capaces de acceder a cualquier recurso del servidor desde cualquier posible ubicación, aún desde lugares tales como habitaciones de hotel, oficinas, hogares y teléfonos celulares, utilizando un mismo enlace. La seguridad debe ser aplicada en base a una desconfianza mutua. Las estaciones cliente deben probar a los servidores que son quienes dicen ser y viceversa, consultando a un tercero que sea de confianza. Kerberos de MIT, el sistema de seguridad para el Entorno de Computación Distribuida, trabaja de esta manera. Tras la comprobación, las aplicaciones de servidor deben utilizar varias Listas de Control de Acceso (ACL/ Access Control List) para regular el acceso de las estaciones cliente a las funciones y a los datos:

Transparencia en comunicaciones. Al igual que lo hacen los NOS modernos, los NOS de la era intergaláctica deberán ocultar las complejidades de los múltiples protocolos y de las representaciones de datos disimilares que se encuentran detrás de un conjunto de abstracciones para las comunicaciones de interproceso. Para ello se puede emplear alguna forma de Llamada de Procedimiento Remoto (RPC) a fin de hacer que todas las llamadas parezcan como si fueran echas localmente. En aplicaciones cliente-servidor se podrá emplear el middleware para hacer que las estaciones cliente y los servidores puedan tolerar el retraso de las comunicaciones.

Entre los sistemas operativos que se encuentran en el mercado, y que ya presentan algunas o todas de las características mencionadas anteriormente, están: los DCE de la Open Software Foundation, NetWare 4.x de Novell y ONC (Open Network Computing) de Sun. DCE es un NOS técnicamente superior que satisface la mayoría de los requerimientos; así mismo, es el NOS estratégico para la mayoría de los jugadores intergalácticos, incluyendo Digital Equipment, Hewlett-Packard, IBM, Microsoft y Tandem. Novell tiene la ventaja de que domina los entornos cliente-servidor de hoy en día, aunque la transición a NetWare4.x hasta ahora ha sido difícil. ONC, integrado a millones de nodos Unix, influyó en el desarrollo de la infraestructura Internet de comunicaciones.

5.2.3 Aplicaciones de los Sistemas de Computación Distribuida.

No esta lejos el día en que los sistemas distribuidos invadan nuestra vida diaria. Imaginemos por ejemplo un centro comercial electrónico a nivel planetario, con boutiques virtuales, tiendas departamentales, librerías, servicios de inversión, bancos, agencias de viajes, y muchas cosas más. Al igual que un Club Med, este centro comercial contará con su propia moneda electrónica para facilitar las compras y las transacciones entre comercios. Los agentes electrónicos rastrearán la red, buscando gangas y negociarán con otros agentes. Miles de millones de transacciones y océanos de información multimedia viajarán por la red todos los días.

Lo anterior no es un sueño, ya que si observamos un poco a nuestro alrededor nos percataremos de que esto ya ha comenzado. Mencionemos, por ejemplo, los cajeros automáticos, en los que podemos hacer transacciones financieras en diferentes localidades e incluso en cajeros de diferentes instituciones a la nuestra, lo cual es posible gracias a que estos se encuentran conectados a través de un sistema distribuido maestro.

Otro ejemplo de un sistema distribuido es la actual red Internet, la cual es una red internacional orientada a la investigación y que comprende más de 3000 redes gubernamentales y académicas en 40 países. Sin embargo los sistemas distribuidos del futuro no serán parecidos al Internet actual, en donde los usuarios navegan a través de telarañas de hipertexto de información con el Lenguaje de Marca de Hipertexto (HTML/ Hypertext Markup Language). Por lo mismo será necesario crear nuevas tecnologías, incluyendo:

Procesamiento rico en transacciones.

Se requerirá de transacciones anidadas que puedan atravesar redes, transacciones que se ejecuten durante largos periodos de tiempo mientras viajan de un servidor a otro, transacciones en cola para llevar a cabo negociaciones seguras entre comercios y sagas que puedan encadenar varias piezas de trabajo y deshacer selectivamente parte de los efectos de la transacción.

Agentes de navegación.

Los consumidores tendrán agentes personales que velarán por sus intereses. Los comercios emplearán agentes para vender sus bienes en la red. Los agentes rastreadores analizaran las tendencias y compilarán estadísticas. Las tecnologías de agentes incluyen mecanismos de escritura de plataforma cruzada, mecanismos de flujo de trabajo y una infraestructura que permite a los agentes residir en cualquier máquina de la red.

Administración rica en datos.

Será importante que desde cualquier sitio en la red, podamos crear, almacenar, visualizar y editar documentos compuestos con contenido multimedia. La mayoría de los nodos ofrecerá tecnología de documentos compuestos (por ejemplo, OLE u OpenDoc) para la administración local de documentos. Los superservidores proporcionarán depósitos para el almacenamiento y la distribución de cantidades masivas de documentos. Naturalmente, tampoco podemos olvidar la información estructurada (por ejemplo, bases de datos SQL).

Además de los casos mencionados anteriormente, los sistemas de computación distribuida se pueden manifestar de muchas otras formas; por ejemplo, en equipos Unix. Un tipo de sistema operativo de red que se esta volviendo popular es el NFS (Network File System/ Sistema de Archivos en Red). Bajo este sistema el disco duro de las máquinas más poderosas y con mayor capacidad de almacenamiento puede ser empleado como disco de las demás

máquinas del sistema, permitiendo guardar en ellos archivos de datos, programas y en general, cualquier cosa que se pudiera archivar en los discos locales de los clientes.

De esta manera, aún cuando la tecnología de los sistemas de cómputo distribuido no es nueva, sí es relativamente reciente. Podríamos argumentar que sus beneficios son grandes, pero la mejor manera de comprobarlo es utilizándola en una misma aplicación, desarrollada en dos versiones. Una de éstas a nivel de anfitrión y otra en alguna de sus variantes tal como cliente-servidor, la diferencia es evidente. No es pues de extrañar, que muchas redes de computadoras estén siendo entrelazadas entre sí a fin de obtener más poder de procesamiento, más facilidad en la obtención de recursos y bancos más grandes de información.

GLOSARIO

ALU (Arithmetic Logic Unit)
Unidad Aritmética y Lógica; circuito de alta velocidad en la CPU que realiza las comparaciones y los cálculos reales.

ANSI (American National Standards Institute)
Coordina el desarrollo de estándares voluntarios a nivel nacional, que incluyen lenguajes de programación, EDI, telecomunicaciones y propiedades de medios de disco y cinta. Es la entidad de los Estados Unidos miembro de ISO. New York.

ASCII (American Standard Code for Information Interchange)
Código estándar de los Estados Unidos para intercambio de información.
Código binario de datos que se usa en comunicaciones, en la mayor parte de los minicomputadores y en todos los computadores personales. Solo los primeros 128 caracteres (0-127) dentro de las 256 combinaciones de un byte constituyen el estándar ASCII. El resto se utiliza en forma diferente de acuerdo con el computador.

CLUD MED
Club comercial de carácter internacional que se caracteriza por realizar todas sus operaciones financieras a través de medios electrónicos.

COLISIONES DE RED
Dicese de los problemas que se llegan a presentar en un sistema de red de computadoras, como por ejemplo cuando dos computadoras intentan utilizar al mismo tiempo un mismo recurso, como podria ser una impresora.

DOMINIO
Área de un sistema sobre la cual tiene control un dispositivo, tanto lógico como fisico.

EBCDIC (Extended Binary Coded Decimal Interchange Code)
Código Binario ampliado de intercambio decimal codificado.
Son los códigos de datos utilizados en mainframe IBM y en la mayor parte de las computadoras de rango medio. Es un código de ocho bit (256 combinaciones) que almacena un carácter alfanumérico o dos dígitos decimales dentro de un byte.

EXPEDITA
Que no tiene estorbos o impedimentos para obrar.

HERRAMIENTAS CASE (Computer Aided Systems Engineering)
Ingeniería de sistemas asistida por computador.
Software que se utiliza en cualquiera o en todas las faces del desarrollo de un sistema de información. Este incluye análisis, diseño y programación. El principal objetivo de CASE es proveer un lenguaje para describir el sistema completo, que sea suficiente para generar todos los programas necesarios.

HIPERTEXTO
Vincular información relacionada. Por ejemplo, al seleccionar una palabra en una frase, se recupera información sobre esa palabra, si existe, o se encuentra la próxima vez que aparezca la palabra.

INTERPRETE
Traductor de lenguajes de programación de alto nivel que traduce y ejecuta el programa al mismo tiempo. Traduce una sentencia de programa a lenguaje de maquina, la ejecuta y luego pasa a la sentencia siguiente.

KERNEL
También conocido como núcleo es la parte fundamental de un programa, como un sistema operativo, que reside en la memoria principal todo el tiempo.

MICROTECNOLOGIA
Técnica que consiste en la miniaturización de circuitos electrónicos.

PLUG AND PLAY (Poner y Usar)
Técnica que tiene la filosofía de que basta con agregar nuevos componentes de hardware (tales como tarjetas de video o de sonido) sobre la placa principal quedando listo el nuevo servicio sin necesidad de su configuración lógica.

PROCESAMIENTO DE TRANSACCIONES EN LÍNEA
Procesar transacciones en el momento que las recibe el computador. Este procesamiento, también es llamado realtime (de tiempo real), actualiza los archivos maestros tan pronto como se introducen las transacciones en las terminales o llegan por las líneas de comunicación.

RAM DE DISCO
Unidad de disco simulada en la memoria. Para usarla, los archivos se copian del disco magnético al disco RAM. El procesamiento es mas rápido, porque no hay ningún accionamiento mecánico del disco, solo transferencias de memoria. Los archivos de datos actualizados deben volver a copiarse en el disco antes de cortar el suministro de energía, de lo contrario, se perderán las actualizaciones.

REPOSITORIO
Lugar ubicado en alguna parte de la memoria o bien, en algún archivo temporal, donde se guarda algo.

SIMBIÓTICOS
Que tienen comportamiento simbiótico. Asociación de organismos diferentes en la que éstos sacan provecho de la vida en común.

SINÉRGICOS
Se Dice de aquellos sistemas que se asocian para la producción de un trabajo.

SOFTWARE DE PRODUCTIVIDAD
Son todos aquellos programas cuyo propósito es auxiliar a los usuarios finales en la realización de sus actividades diarias. Entre estos podemos mencionar: EXCEL, WORD, POWERPOINT, etc..

SSCP (Sistem Service Control Point)
El punto central de control de un "dominio" de red.

STAND-ALONE
Modo en que un sólo usuario opera totalmente sobre el sistema. Se dice que se esta en modo stand-alone cuando se trabaja en una sola computadora, la cual no se encuentra conectada a ninguna red de dispositivos de computo.

TELEIMPRESORAS
Terminal similar a una maquina de escribir que tiene incorporados un teclado e impresora.

TRADUCTOR
Es el termino empleado para denotar a aquellos programas que transforman a un lenguaje de alto nivel en código de maquina para que pueda ser ejecutado. Se dividen en compiladores (primero traducen todo el código y luego lo ejecutan) y en intérpretes (revisan y ejecutan una por una cada sentencia del programa).

Bibliografia

A. D. BIRRELL Y B. J. NELSON, "Implementing Remote Procedure Calls"; ACM Trans. Computer Systems, Vol. 2, No. 1, Febrero de 1984, pp 39-59.

AGARWAL, A., y CHERIAN, M.: "Adaptive Backoff Synchronization Techniques" Proc. 16th Ann. Int'l Symp. on Computer Arch., ACM, pp. 396-406, 1989.

ALMASI, G.S.., y GOTTLIEB, A. : Highly Parallel Computing, Redwood City, C.A.: Benjamin/Cummings, 1989.

A. S. TANENBAUM AND R. VAN RENESSE. "Distributed Operating Systems". ACM Computing Surveys 17; pag's 419-470, Diciembre 1985.

BAL, H.E.: The shared Data-Object Model as a Paradigm for Distributed Programming, Ph. D. dissertation, Vrije Universiteit, 1989.

B. I. WITT, "Communicating Modules: A Software Design Model for Concurrent Distributed Systems"; Computer, Vol. 18, No. 1, Enero de 1985, pp. 67-77.

Byte, México; Rober Orfali y Dan Harkey, "Computación Cliente Servidor", Año 9 No. 89, Pag's 44-59, Junio de 1995.

CARPENTER PHILLIP, "Think Umbrella". Software Magazine; Septiembre 8, 1991.

CHOW, F.: "Minimizing Register Usage Penalty at Procedure Calls," Proc. ACM SIGPLAN 88 Symp. on Prog. Lang. Design and Impl., ACM, pp. 85-94, 1988.

COULORIS, G. F. Y DOLLIMORE J.,"Distributed Systems Concepts and Design", Addison-Wesley, New York, 1988.

Curso de Informatica Windows, "Super Comunicaciones", Año 1 No.11, pag's 341-350; Septiembre 1994.

Datapro: "Workstations and Servers", "Client/Server Computing" y "The Mainframe as a Server"

DeROSA, J.A. y LEVY, H. M.: "An Evaluation of Branch Arquitectures", proc. 15th Ann. Int'l Symp. on Computer Arch., ACM, pp. 10-16, 1988.
DESROCHERS, G.R.: Principles of parallel and Multiprocessing, New York: McGraw-Hill, 1987.

DIJKSTRA, E. W.: "Co-operating Sequential Processes," in Programming Languages, F. Genuys (ed..), New York: Academic Press, 1968b.

DONGARRA, J. J. (De.): Experimental Parallel Computing Architectures, Amsterdam: North Holland, 1987.
FLYNN, M. J.; "Directions and Issues in Architecture and Lenguage"; IEEE Computer Magazine, Vol. 13, Octubre de 1992, pp. 5-22.

FLYNN, M. J.: "Some Computer Organizations and their Effectiveness, " IEEE Trans. on Computer, vol. C-21, pp. 948-960. 1989.

FRED B. CHAMBERS, "Distributed Computing", Academic Press INC., Londres 1984.

FREEDMAN ALAN, "Diccionario de Computación"; McGraw-Hill, Santafé de Bogotá, 1994.

FRITZ, T. E., HEFNER, J. E. Y RALEIGH, T. M. "A Network of Computers Running the UNIX System" AT&T Bell Laboratories Technical Journal, vol. 63, no. 8, pp. 1877-1896. Oct. 1994.

FORRESTER RESEARCH; "Computing Strategy -Resizing Client/Server" (December 1991).

GARCÍA-PELAYO Y GROSS, "Pequeño Larousse en color"; Ediciones Larousse, Buenos Aires, Argentina, 1988.

GEORGE A. CHAMPINE, "Distributed Computer System"; North-Holland Publishing Company, Ansterdam 1992.

G. R . ANDREWS Y F. B. SCHNEIDER, "Concepts and Notations for Concurrent Programming"; ACM Computer Surveys, Vol. 15, Marzo de 1988, pp. 3-43.

HWANG. k., t BRIGGS, F.A.: "Computer Architecture and Parallel Processing." New York: McGraw-Hill, 1984.

HWU, W. W., CONTE, T. M., y CHANG, P.P.: "Comparing Software and Hardware Schemes for Reducing the cost of Branches", Proc. 16 th. Ann. Int'l Symp. on Computer Arch., ACM, pp. 224-233, 1989.

J.H. SALTZER AND K. T. POGRAN. "A Star-Shaped Ring Network with High Maintainability". Computer Networks No. 4; pag´s 239-244, 1990.
JOHN R. CORBIN, "The Art of Distributed Applications", Ediciones Springer-Verlag, Mountain View, CA., 1992.

JOHN SHIRLEY, WEI HU y DAVID MAGID, "Guide to writing DCE Applications", Ediciones Sebastopol, segunda edición, California E.U.A, 1994.

J. SVENTEK, W. GREIMAN, M. O´DELL, y A. JANSEN. "Token Ring Local Networks -A Comparison of Experimental and Theoretical Performance." Lawrence Berkeley Laboratory Report 16254, 1983.

KERNIGHAN, B. W. Y RITCHIE, D. M. "The C Programming Languaje", Segunda edición, Prentice Hall, Englewood Cliffs, N. J., 1988.

KNAJEWSKI, R. "Multiprocessing: An Overview", Byte, vol. 10, 171-181, May. 1985.

KNIGHTSON, K. G., KNOWLES, T., Y LARMOUNTH, J. "Standards for Open Systems Interconnection", McGraw-Hill, New York, 1988.

LYON B. "Sun Remote Procedure Call Specification", Sun Microsystems, Inc. Mountain View (Calif.), 1985.

M. D. Mills et al., "The Management of Software Engineering"; IBM Systems, Vol. 19, No.4, 1980, pp. 414-477.

Personal Computing Mexico, "El Rápido y Necesario Tránsito a Cliente-Servidor"; Año 6 No. 75; pag´s 38-48, Agosto de 1994.

R. C. LINGER, H. D. MILLS, AND B. I. WITT, "Structured Programming, Theory and Practice", Addison-Wesley, Reading, Mass., 1979.

ROCHKIND, M. J. "Advanced UNIX Programming", Prentice Hall, Englewood Cliffs, N. J. 1985.

R. VAN RENESSE, H. VAN STAVEREN, A. S. TANENBAUM. "Performance of the World´s Fastest Distributed Operating Systems". Operating Systems Review No. 22; pag´s 25-34, Octubre 1990.

SHAKU ATRE, "Distributed Databases, Cooperative Processing & Networking"; McGraw-Hill, Series Advisor, New York 1992.

Sun Microsystems, "RPC: Remote Procedure Call, Protocol Specification, Version 2", RFC 1057, pag´s 1-25, junio de 1989.

Y. PAKER Y J.P. VERJAS, "Distributed Computing Systems"; Academic Press INC., segunda edición, Londres 1988.

W. RICHARD STEVENS. "Unix Network Programming", Ed. Prentice Hall, New Jersey 1993.