PUBLICACIONES ICESI

INSTITUTO COLOMBIANO DE ESTUDIOS SUPERIORES DE INCOLDA

Publicaciones ICESI	Cali Colombia	Nº 54	P.P. 189	Ene.-Mar. 1995	ISSN 0120-6648

CONSEJO SUPERIOR

Germán Holguín Zamorano
PRESIDENTE

Adolfo Carvajal Quelquejeu
VIICEPRESIDENTE

Francisco J. Barberi Ospina
Jorge Enrique Botero Uribe
Francisco Castro Zawadski
Henry Eder Caicedo
Mauricio Cabrera Galvis
Isaacs Gilinski Sragovicz
Hugo Lora Camacho
Juan María Rendón Gutiérrez
Oscar Varela Villegas
Augusto Solano Mejía

JUNTA DIRECTIVA

Francisco J. Barberi Ospina
PRESIDENTE

Oscar Varela Villegas
VICEPRESIDENTE

Jaime Orozco Abad
William Barlow Murray
Augusto Solano Mejía
Gabriel Angel Botero
Esther Ventura de Rendón

DIRECTIVOS DEL ICESI

Alfonso Ocampo Londoño
Rector

María Cristina Navia Klemperer
Secretaria General

Héctor Ochoa Díaz
Decano de Postgrado

Francisco Velásquez Vásquez
Decano de Administración de Empresas

Mario Tamayo y Tamayo
Director de Investigaciones y Publicaciones

Carlos Fernando Cuevas Villegas
Director Administración de Empresas Nocturno

María Fernanda Barney
Directora de Admisiones y Registro

María Cristina Navia Klemperer
Directora de Relaciones Universitarias

Martha Cecilia Lora Garcés
Directora de la Biblioteca

Hipólito González Zamora
Vicerrector

Lucrecia C. de Arango
Directora Administrativa

Edgar Sarria Campo
Director de Planeación

Henry Arango Dueñas
Decano de Ingeniería de Sistemas

Rodrigo Varela V.
Director del Centro de Desarrollo del Espíritu Empresarial

Olga Ríos Restrepo
Directora del Centro de Cómputo

María Isabel Velasco de Lloreda
Directora de Relaciones Empresa - Universidad ICESI

Amparo Beltrán Hurtado
Directora de Promoción Académica


ICESI

CONSEJO EDITORIAL

Avenida 10 de Mayo cruce con Avenida Cañasgordas - Pance
Apartado Aéreo 25608, Unicentro
Teléfono: 5552334
CALI - COLOMBIA - SUDAMERICA

CONTENIDO

PRESENTACION

La Dirección de Investigaciones y la Facultad de Ingeniería de Sistemas e Informática del ICESI, presentan en su No. 54 los siguientes trabajos:

- La Seguridad en los Sistemas Operacionales.
- Modelos de Estimación de Costo y Esfuerzo en Proyectos de Software.
- Propuesta de Interconexión a Internet.
- Sistemas Operativos Distribuidos.

Elaborados por los alumnos del curso de Investigación de VIII Semestre de Ingeniería de Sistemas, bajo la dirección académica de los doctores Guillermo Londoño, Alvaro Pachón de la Cruz, Olga Ríos R. y Juan Manuel Madrid, con la coordinación y apoyo económico de la Dirección de la Oficina de Investigaciones.

Mario Tamayo y Tamayo
Director de Investigaciones

LA SEGURIDAD EN LOS SISTEMAS OPERACIONALES

MARÍA DE LOS ANGELES PELÁEZ, GLORIA PIEDAD PONCE,
LUIS FERNANDO COLMENARES, SILVIO BAZANTE
Alumnos del curso de Investigación de VIII Semestre de Ingeniería de Sistemas del ICESI.

INTRODUCCION

Con el pasar de los años y de las generaciones, en la historia de la humanidad, ésta se ha dado cuenta de que la persona que maneja el mundo y lo controla no es la que tiene más poder económico, sino la que maneja la información.

Esto se puede apreciar en esta época, cuando muchas personas con gran poder económico han sucumbido ante el poder de la información. Con los grandes avances tecnológicos de la era, las herramientas desarrolladas anteriormente se han venido perfeccionando y se les ha aumentado su radio de acción. Estas herramientas son los computadores.

El computador se ha vuelto un elemento inseparable en la vida, a tal punto que, a donde vayamos, siempre vamos a encontrar algo controlado por un computador; por ejemplo: estaciones de gasolina controladas por computador; en los bancos, se ven computadores por todos lados; también en las industrias, en los hospitales, etc. Muchas de las actividades que realizan hoy en día los computadores son de vital importancia, porque un error en cualquier gestión puede representar la vida de muchas personas.

A raíz de que los computadores pueden realizar tareas de mucha importancia, es necesario dotarlos de cierta seguridad para evitar que personas ajenas a dicha actividad puedan causar un gran daño, sin proponérselo.

Cuando se menciona la palabra "seguridad" no sólo nos estamos refiriendo a la seguridad física de los equipos, sino también a la seguridad interna del sistema operativo. En esta última es donde más fallan las compañías, ya que creen que al poseer una buena seguridad a nivel físico, nadie va a poder usar el computador y tener acceso a la información. Hoy en día, con el avance de las comunicaciones, no es necesario que una persona tenga que burlar toda esa vigilancia física; simplemente, con un "modem" y un computador, puede conectarse a otro computador, y penetrar el sistema, sin mayor problema.

En el país, la seguridad a nivel del sistema operativo es muy incipiente;

apenas está en una etapa de internacionalización donde se tiene que proteger contra un mundo hostil. Quien maneja la información es quien domina el mundo.

En los países industrializados no se escatiman recursos en el desarrollo de nuevos sistemas, con un mayor grado de seguridad, para que estos sean más confiables y seguros.

1. LA SEGURIDAD EN LOS SISTEMAS OPERACIONALES

En este capítulo se explicará qué es y en qué consiste la seguridad de los sistemas operacionales; qué técnicas se han desarrollado para hacer un sistema más seguro.

También se presentarán varios casos de violación de sistemas operacionales y las consecuencias que trajeron. Además, se analizará cómo se castiga la violación de los sistemas operacionales en Colombia.

1.1. Qué es la seguridad en los sistemas operacionales

Cuando se escucha o se menciona la palabra "seguridad" se piensa inmediatamente en un tipo de seguridad física. Pues bien, cuando se habla de seguridad en el sistema operacional, no se está hablando únicamente de la seguridad física del equipo, sino también, de la seguridad de la información contenida en los equipos.

Cada sistema operacional tiene un nivel de seguridad específico, con el cual se puede manipular el flujo de la información, es decir, el usuario decide quién puede tener acceso a cierta información y quién no.

"El nivel de seguridad que debe proporcionarse a un sistema, depende del valor de los recursos por asegurar".[1]

El concepto de seguridad en los sistemas operacionales es relativamente nuevo. Solamente se vino a hablar de seguridad en los sistemas operacionales a mediados de los años sesenta y setenta. Este campo, al igual que otros relacionados con la tecnología de los computadores, se fue desarrollando a partir de los proyectos militares.

1.2. En qué consiste la seguridad en los computadores

La seguridad en los sistemas operativos no consiste en cualquier dispositivo físico, por fuera del equipo, sino en una serie de programas que se encargan de controlar el acceso de los usuarios a los recursos de un computador.

Básicamente, la seguridad del sistema se concentra en tres actividades que son:

* Auditorías.
* Protección del sistema.
* Protección de datos.

En 1985, el Departamento de Defensa de los Estados Unidos publicó el libro *The Trusted Computer System Evaluation Criteria*. Este libro es conocido como *The Orange Book* (el Libro Naranja). Su importancia se basó en que determinó un estándar en la seguridad de los sistemas operacionales.

1.2.1. *Auditorías*

Las auditorías en los sistemas operativos son el equivalente de las auditorías en los negocios. El objetivo de esta estrategia es determinar si ha ocurrido una violación en el sistema, o si hay alguien que se está apropiando de muchos recursos, cuando no puede hacerlo; o verificar lo que están haciendo los usuarios y poder identificar quiénes son.

Las auditorías se deben adelantar periódicamente para llevar un buen control de la seguridad. Es recomendable realizarla dos a tres veces por mes, pero esto es determinado por el administrador del sistema. También es bueno llevar a cabo auditorías-sorpresa, para que los usuarios no tengan tiempo de cubrir todo tipo de huellas, si han infringido la integridad del sistema.

En algunos sistemas operacionales, realizar las auditorías es un trabajo muy complicado, ya que el administrador del sistema debe vigilar todas las etapas de la auditoría, mientras que en otros es tan fácil que viene siendo una tarea que el mismo sistema realiza, sin que el administrador esté, todo el tiempo, vigilando las etapas del proceso.

1.2.2. *Protección del sistema*

La protección del sistema consiste en prevenir que alguien se adueñe de los recursos del sistema y, por consiguiente, no permita usarlos, o use recursos inhabilitados para los usuarios.

1.2.3. *Protección de datos*

La protección de datos consiste en un control sobre todas las operaciones de los datos, en lo que se refiere a lecturas, escrituras, modificaciones, borrado, etc.; para que los usuarios solamente puedan realizar las operaciones a que tienen derecho.

1.2.4. *El Libro Naranja*

En este libro no se menciona un diseño específico para la construcción de un sistema que sea seguro; sino que se clasifican los sistemas operativos, de acuerdo con la manera como manejan la seguridad.

Para calificar la seguridad de un sistema, el libro utiliza las letras A,B,C,D, en donde las letras B y C tienen subdivisiones. En total, se manejan siete niveles de seguridad. La letra "A" quiere decir que el nivel de seguridad del sistema es muy elevado y que prácticamente es imposible de violar, y así sucesivamente va disminuyendo hasta llegar a la letra "D", la cual quiere decir que el sistema es muy fácil de penetrar.

1.2.4.1 *Categorías definidas por el Libro Naranja*

Las categorías definidas por el Libro Naranja, son las siguientes:

D: Protección mínima, en donde la seguridad solamente se limita a la seguridad física. Un ejemplo de sistemas de categoría D es el famoso DOS.

C1: Protección de seguridad discrecional. Esto significa que la protección y el control de acceso están definidos por el administrador de la máquina; la gran mayoría de los sistemas operacionales UNIX caen dentro de esta categoría.

C2: Protección de acceso controlado. Esto quiere decir que, además de actuar como el nivel C1, lleva registros de auditoría. El ejemplo de sistemas operacionales que cumplen este nivel, es el mismo ejemplo del nivel anterior.

B1: Protección mandataria. El administrador de la máquina no puede definir si existe o no una clave. No pueden existir cuentas sin clave; un ejemplo de sistemas operativos que cumplan con este nivel, son los AS400 de IBM.

B2: Protección estructurada. Existen políticas formales de seguridad y se separan las funciones de administración y operación.

B3: Dominio de la seguridad. Existe la posibilidad de realizar especificaciones, definir permisos por usuarios y por grupos y tener control sobre los periféricos.

A1: Protección verificable. Se define el modelo de protección; existe protección a nivel de la fuente y del código ejecutable, etc. Muy pocos sistemas se pueden acercar a esta categoría; en la actualidad el único sistema que ha sido clasificado como A1, es el sistema operacional SCOMP de Honeywell. Para que un sistema sea clasificado comoA1,

1. *Introducción a los Sistemas Operativos*, pág. 447.

se requiere por lo menos un período de estudio de tres años y pasar todas las pruebas a que es sometido el sistema.

1.3. Técnicas de seguridad

Un sistema de computación contiene muchos objetos que necesitan protegerse. Estos objetos pueden ser elementos de "hardware", como unidades centrales de procesamiento, segmentos de la memoria, terminales, unidades de disco, impresoras; o bien pueden ser elementos de "software", como archivos, bases de datos, etc.

Cada objeto tiene un nombre único, por el cual se refiere, y un conjunto de operaciones que se pueden ejecutar con el Read y el Write, las cuales son operaciones adecuadas para un archivo; UP y DOWN tienen sentido con un semáforo. Los objetos son el equivalente del sistema operativo, lo que, en lenguajes de programación, se conoce como tipos de datos abstractos.

Está claro que se necesita contar con alguna manera de prohibir que los procesos den entrada a aquellos objetos para los que no se tiene acceso autorizado. Además, este mecanismo también debe hacer posible limitar los procesos a un subconjunto de las operaciones legales, cuando se necesite. Por ejemplo, el proceso "A" puede tener derecho a leer, pero no a escribir el archivo "F". Para ofrecer una manera de analizar diferentes mecanismos de protección, conviene presentar el concepto de dominio.

1.3.1. *Dominio de protección*

Un dominio es un conjunto de parejas (objetos, derechos). Cada pareja especifica un objeto y algún subconjunto de las operaciones que se pueden efectuar con él. Un derecho, en este contexto, significa autorización para ejecutar una de las operaciones. Es posible que el mismo objeto esté en múltiples dominios, con diferentes derechos en cada uno.

En cada momento, cada proceso se ejecuta en algún dominio de protección. En otras palabras, existe algún conjunto de objetos que pueden acceder y, por cada objeto, se tiene algún conjunto de derechos. Los procesos también pueden correrse de un dominio a otro, durante la ejecución. Las reglas para el cambio de dominios dependen en gran medida del sistema.

Para llevar un control sobre cuál objeto pertenece a cuál dominio, el sistema utiliza una matriz grande (Figura 1) donde los renglones son los dominios y las columnas los objetos. Cada caja lista los derechos, si hay alguno, que el dominio contiene para el objeto.

OBJETO

Arch	Arch	Arch	Arch	Arch	Arch	Impre	Grafi
R	R,W						
		R	R,W,X	R,W		W	
					R,W,X	W	W

Figura 1. *Matriz de protección*

Dados esta matriz y el número de dominio corriente, el sistema siempre puede indicar, si se permite el intento de acceso, a un objeto específico, de manera particular, a partir de un dominio especificado.

La Figura 2 muestra la matriz de la Figura 1, una vez más, sólo que ahora con los tres dominios como objetos. Los procesos del dominio 1 pueden correrse al dominio 2, pero una vez ahí ya no pueden regresar.

OBJETO

A.1	A.2	A.3	A.4	A.5	A.6	Imp	Gra	D.1	D.2	D.3
R	RW								Int	
		R	RWX	RW		W				
					RWX	W	W			

Figura 2. *Matriz de protección*

1.3.2. *Listas con control de acceso*

En la práctica, el almacenamiento real de la matriz de protección rara vez se hace porque es grande y dispersa. La mayoría de los dominios no tienen acceso en absoluto a la mayoría de los objetos, de manera que el almacenamiento de una matriz grande, vacía, se traduce en una pérdida de espacio, en el disco. Sin embargo, dos métodos prácticos son el almacenamiento de la matriz por renglones o por columnas y el almacenamiento sólo de los elementos no vacíos. Estos métodos son sorprendentemente distintos.

La primera técnica consiste en asociar con cada objeto una lista (ordenada) que contenga todos los dominios a que puede acceder el objeto y que indique cómo hacerlo. A esta lista se le llama "Lista de Control de Acceso" o "ACL".

1.3.3. *Capacidades*

La otra manera de dividir la matriz de protección es por renglones. Cuando se emplea este método en asociación con cada proceso, hay una lista de objetos que pueden acceder, junto con una indicación de qué operaciones se permiten con cada uno, en otras palabras, su dominio. Esta lista se llama "Lista de Capacidades" y los elementos individuales contenidos en ella se denominan "capacidades".

Una lista común de "capacidades" se muestra en la Figura 3. Cada capacidad tiene un "campo de tipo", el cual indica

	Tipo	Derechos	Objeto
0	Archivo	R - -	Apunt al Arch 3
1	Archivo	R W X	Apunt al Arch 4
2	Archivo	R W -	Apunt al Arch 3
3	Archivo	- W -	Apunt a la Impr.

Figura 3. *Lista de capacidades de un objeto.*

de qué tipo de objeto se trata; un "campo de derechos", que es un mapa de bits que indica cuáles de las operaciones legales con este tipo de objeto están permitidas y un "campo de objeto", el cual es un apuntador del objeto mismo.

Como es evidente, las listas de capacidades deben protegerse de la alteración por parte del usuario. Para esto se han propuesto tres métodos:

A) El primer método es un diseño de "hardware" que requiere una arquitectura etiquetada, en el cual cada palabra de la memoria tiene un bit (o etiqueta) extra, el que indica si la palabra tiene capacidad o no.

B) El segundo método consiste en conservar la lista de capacidades, dentro del sistema operativo, y simplemente hacer que los procesos se refieran a las capacidades,por su número de ranura.

C) El tercer método consiste en conservar la lista de capacidades en el espacio del usuario, sólo que poniendo en clave cada capacidad, con una clave secreta, desconocida para el usuario.

Además de los derechos específicos que dependen del objeto, las capacidades suelen tener derecho a genéricos, los cuales son aplicables a todos los objetos. Ejemplos de derechos genéricos son:

- Capacidad de copiado: Crear una nueva capacidad para el mismo objeto.
- Copiar el objeto: Crear un duplicado del objeto con una nueva capacidad.
- Capacidad de eliminación: Suprimir la captación de la lista C, sin afectar el objeto.
- Destruir el objeto: Eliminar en forma permanente el objeto y la capacidad.

1.3.3.1. *Revocación de las capacidades*

La revocación del acceso a un objeto es muy difícil, ya que resulta complicado para el sistema hallar todas las capacidades restantes de cualquier objeto, para devolverlas, ya que éstas pueden estar almacenadas en listas C, por todo el disco. Como solución a este problema, existen dos métodos:

A) El primero consiste en hacer que cada capacidad apunte a un objeto indirecto, en vez de al objeto mismo; al hacer esto, el sistema siempre puede romper esa conexión, con lo cual se invalidan las capacidades.

B) El segundo método consiste en que cada objeto contenga un número largo elegido al azar, el cual también está presente en la capacidad. Cuando una capacidad se presenta para su uso, las dos se comparan. Sólo si concuerdan se permite la operación. El propietario de un objeto puede solicitar que se cambie el número del objeto elegido al azar, con lo cual se invalidan las capacidades existentes.

Ninguno de estos dos métodos permite la revocación selectiva, es decir, retirar, por ejemplo, la autorización de John, pero la de nadie más.

1.3.4. ***Canales de conversión***

Se puede comprobar que aun en un sistema que ha sido rigurosamente probado y se ha concluido que es absolutamente seguro, la fuga de información entre procesos, que en teoría no se pueden comunicar en absoluto, es relativamente directa. Estas ideas se deben a Lampson (1973).

En el modelo de Lampson intervienen tres procesos y se aplica principalmente a sistemas grandes de tiempo compartido. El primer proceso es el cliente, el que desea que el segundo, el servidor, realice algún trabajo. El cliente y el servidor no confían, por completo, el uno en el otro.

El tercer proceso es el colaborador, el cual conspira con el servidor para, en realidad, sustraer datos confidenciales del cliente. El colaborador y el servidor son comúnmente, propiedad de la misma persona. Estos tres procesos se muestran en la Figura 4a.

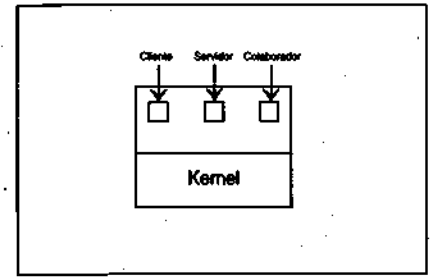


Figura 4(a)

Desde el punto de vista del diseñador del sistema, el objetivo consiste en encapsular al servidor de tal forma que no pueda pasar la información al colaborador. Con una matriz de protección, esto se puede evitar, pero por desgracia, pueden estar disponibles más canales de comunicación sutiles. Por ejemplo, el servidor puede intentar comunicar un flujo de bits binarios, de la manera siguiente. Para enviar un bit uno (1), éste hace lo que sea por determinar un intervalo fijo de tiempo. Para enviar un bit cero (0), éste se bloquea durante la misma longitud de tiempo.

El colaborador puede intentar detectar el flujo de bits, monitoreando con mucho cuidado su tiempo de respuesta. En términos generales, obtendrá una mejor respuesta cuando el servidor envíe un cero (0) que cuando el servidor envíe un uno (1). Este canal se conoce como de conversión y se ilustra en la Figura 4b.

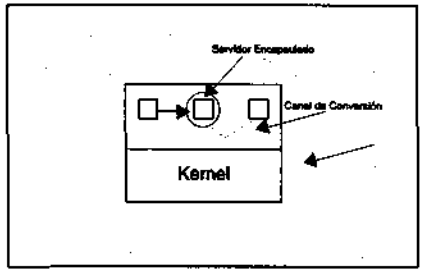


Figura 4 (b)

Desde luego que el canal de conversión es un canal ruidoso, pero la información se puede enviar, con toda confianza, por un canal ruidoso, mediante un código de corrección de errores. El uso de un código de corrección de errores reduce, aún más, el tamaño de la banda del canal de conversión, pero aún sigue siendo lo suficientemente ancho para dejar escapar la información sustancial.

Hallar todos los canales de conversión (sin hablar de su bloqueo) es en extremo difícil. En la práctica, poco se puede hacer al respecto.

1.3.5. ***La criptografía***

En el siguiente tema se hablará de la criptografía, de sus orígenes y de cómo pasó de ser una estrategia militar a una forma de seguridad en los sistemas operacionales.

La criptografía es el arte de comunicarse mediante mensajes secretos; su práctica es tan vieja como lo es la escritura. Su nombre se deriva de las palabras griegas *Kriptós* (escondido) y *Logos* (palabra).

El origen de la criptografía y de su evolución es un misterio, pero se tienen evidencias de que a mediados del siglo 400 A.C. usaban métodos de escritura secreta. En tiempos de Julio César se encontraron evidencias de un método de criptografía que consistía en reemplazar las letras, por la tercera letra que si-

gue en el alfabeto, es decir, la letra A se reemplaza por la letra D y así sucesivamente; a este método se le llama "Julius Caesar Cipher".

En la actualidad, la criptografía está muy desarrollada, y cada vez se recurre a fórmulas matemáticas complejas para codificar los mensajes. Pero esto también es violable y, basta con descubrir cuál es la fórmula utilizada para la codificación para poder descifrar el mensaje. Obviamente, esto no lo puede hacer cualquier persona.

En los sistemas operacionales se utilizan métodos de criptografía para proteger las claves de acceso, o para proteger documentos de vital importancia. Este tema también se ha llevado al cine. Recordemos la película *Héroes al azar*, que consistía en el desarrollo de una fórmula matemática, la cual descifraba cualquier sistema de criptografía utilizado en un sistema operacional, y con él se pretendía entrar en los sistemas de las organizaciones contra el crimen, para borrar todo tipo de evidencias contra los jefes de la mafia de los Estados Unidos.

1.4. Casos de violación a los sistemas operacionales

Se podría llegar a pensar que violar la seguridad de un sistema es difícil y que, por lo tanto, no hay tantas violaciones de los sistemas computacionales.

El problema es de tal magnitud, que sólo en los Estados Unidos las estafas que se originan a raíz de estas violaciones cuestan anualmente entre 500 y 600 millones de dólares, aproximadamente.

La seguridad en los sistemas es la "oveja negra" de la computación. A menudo no recibe respeto, particularmente de la alta gerencia, que está inclinada a invertir los recursos que controla en producción y mercadeo. Estas áreas generan utilidades de una manera más palpable. Pero nosotros, como futuros profesionales, tendremos la misión de disuadir de tal posición a dichas personas, con hechos y cifras. (Por ejemplo, combinar números aterradores con relatos aterradores, y así obtener mejores posibilidades de hacer ver la importancia de la seguridad en los sistemas).

A continuación, se presentarán algunos de los más escandalosos y multimillonarios delitos que han sucedido en los últimos quince años.

Marvin Maki:
Pionero en la violación de sistemas.

Este fue el primer delincuente por computador, enjuiciado por la Fiscalía del Distrito de Los Angeles (Estados Unidos, California). Utilizó el computador de la MDSI, que estaba acondicionado para producir una cinta perforada, la cual controlaba la maquinaria de manufactura de la compañía W&R Tool, y procesaba las cuentas por cobrar de esta empresa. También había hallado los códigos de acceso, utilizados por MDSI en sus oficinas francesas e inglesas, y los había utilizado para conseguir acceso a su sistema. La empresa había cambiado sus códigos locales, después que Maki dejó la compañía, pero no había tomado las medidas adicionales para cambiar los códigos en el extranjero.

Su delito consistió en no haber respetado algunos derechos de propiedad que no estaban claramente protegidos, (la toma del tiempo del computador sin autorización; entrar y tomar un poco de tiempo extra), con la única diferencia de que dicho tiempo de trabajo generaba utilidades no para la compañía sino para Marvin Maki, hecho éste que pasaba inadvertido.

La información que se ha podido obtener, acerca de este caso es muy poca, pero se ha incluido en el trabajo para dar una viva muestra de los fracasos que pueden surgir cuando no se cuenta con un apropiado sistema de seguridad, que en este caso debió haber sido más rígido e invulnerable, en cuanto a la asignación de tiempos de trabajo y prioridades, tema éste tan discutido y tratado en el curso de sistemas operativos.

Harold Rossfields Smith:
El gran robo del Wells Fargo

Hace nueve años, Harold R. Smith fue condenado a prisión por la Corte Federal de Los Angeles, por haber robado 21.3 millones de dólares, en representación de dos organizaciones que le pertenecían.

El robo consistió en haber transferido fondos de una cuenta a otra, violando el sistema de liquidación de una de las sucursales del banco Wells Fargo. Comenzó cuando un empleado de la sucursal Beverly Drive giró un cheque de caja para cubrir un sobregiro de la cuenta de Smith y envió, tanto el cheque como el formato de liquidación del débito parcial de la sucursal, por medio del sistema de liquidación de ésta, como si estuviera transfiriendo fondos de una sucursal a otra (de la Beverly Drice a la Miracle Mile).

Este sistema de liquidación era básicamente de contabilidad o teneduría de libros, y el banco lo utilizaba para transferir dinero de una sucursal a otra.

El banco empleaba un computador, localizado en San Francisco, para llevar el registro de todas esas transferencias diarias entre sucursales; para hacerlas, se utilizaba un formato estándar (Formato de liquidación de la sucursal) para las entradas ( ) y se le daban instrucciones al computador sobre qué hacer.

El primero paso de la artimaña consistió en que el hombre dentro del banco, (la sucursal Miracle Mile), pagó un dinero al acusado (Smith), por medio de sus compañías.

Como paso siguiente, el débito del formato de liquidación lo entregó al computador, como si fuera una transacción normal. A los diez días, en vez de transmitir la mitad del crédito a Miracle Mile, la reportó al computador, violando el principio de que en una sucursal nunca se deben procesar ambas mitades del tiquete.

Entonces, para engañar al computador, simuló que la mitad del formato de crédito procedía del Miracle Mile, y cambió el código de la mitad del crédito. Puso el código especial de la Miracle Mile en números magnéticos al final de la mitad, del crédito, para evitar violar el principio de que una sucursal nunca usará el número de código de la otra sucursal. Con todo esto, impidió que se produjeran inconsistencias, porque el computador había recibido ambos pagos parciales del formato dentro de los diez días. Y tuvo éxito al engañar al computador para que "pensara" que las dos mitades del formato habían llegado de sucursales diferentes, porque la mitad del débito tenía el código de Beverly Drive y la mitad del crédito, el de Miracle Mile.

El problema surgió en el momento del balance, cuando se observó esta transacción, donde había dos créditos y tan sólo un débito. La razón para que faltara un débito radicaba en que no existía dinero real, entrando al sistema.

Así que, para suplir el débito faltante, el empleado de la Miracle Mile tomó otro tiquete de liquidación de la sucursal y la mitad del débito de este nuevo tiquete y, al tiempo que introducía la mitad del crédito del primer tiquete, incluyó un nuevo débito; y todo quedó balanceado. El computador había recibido las dos mitades de este tiquete dentro de los diez días, con dos números de códigos diferentes, y le dio otros diez días más para conseguir la mitad del crédito del segundo tiquete introducido.

Toda esta maniobra se convirtió en un ciclo vicioso del cual era muy difícil escapar, si no se reponía el dinero (cuya cantidad era imposible de saldar).

La única manera de descubrir el robo era un descuido por parte del empleado de Miracle Mile; hasta que olvidó enviar un tiquete débito y permitió reflejar, en los informes de auditoría, las inconsistencias en las cuentas.

En este caso, se puede apreciar no solamente la importancia de la seguridad en los sistemas, sino también, la necesidad de contar con personas de absoluta confianza para permitirles el acceso a las cuentas de las compañías. Con los computadores y la consecuente dispersión en el acceso a las cuentas de las empresas, el número de estafadores se incrementa dramáticamente.

Hubo deficiencias en los métodos de seguridad interna del sistema; una de las tantas posibles soluciones hubiera podido ser que los parámetros de control se hubieran cambiado ocasionalmente, en vez de investigar siempre los balances.

Jan Hanasz: Mezcla de alta tecnología.

En 1990, en Torun, Polonia, uno de los más sobresalientes científicos del espacio, Jan Hanasz, y tres colegas, fueron llevados a juicio "acusados de haber interrumpido una transmisión estatal por televisión... para instar a los votantes a que boicotearan las elecciones...".

Se afirma que estas personas utilizaron un computador casero, un circuito sincronizador y un transmisor para hacer aparecer mensajes en las pantallas de televisión.

Lo habían hecho, llevando a la práctica el equivalente electrónico de viajar, a dedo, en las ondas de transmisión de la red de televisión polaca. (Crearon un sistema que podía producir señales de televisión, sincronizarlas con las señales de televisión estatal polaca y dirigirlas en la misma ruta. El microcomputador regulaba la emisión de señales para que se sincronizaran con las de la estación del gobierno).

Esto es otro clásico delito por computador, mezclado con la tecnología electrónica y una causa justa. Cada vez más, se realizan grandes esfuerzos en todo el mundo para evitar que se violen los sistemas de comunicaciones por televisión. (De todas maneras, así se utilice cualquier método o procedimiento, si se logra traspasar la barrera de lo permitido, es porque hay fallas y vulnerabilidad en el sistema que protege la ejecución de los procesos).

Casos de violación en sistemas hay muchos y el trabajo se extendería demasiado si se pretendiera seguir exponiendo algunos más. No se busca convertir este trabajo en una recopilación de casos de violación de sistemas; lo que se desea es crear y formar una conciencia de la necesidad de proteger cada cosa que se haga y conocer cómo estas personas logran sus objetivos.

1.5. ¿Cómo se castiga en Colombia?

En Colombia, el delito llevado a cabo a través del computador no está tipificado; esto quiere decir que no hay ninguna ley que pueda actuar contra los delincuentes que usen un computador como medio para llevar a cabo un delito.

Esto no es de extrañarse, ya que Colombia posee un código penal ineficiente y atrasado; y esto es uno de los factores que hace que nuestra justicia sea mala.

Cuando se trató de averiguar por qué no existía una legislación para esto, nos encontramos con la siguiente respuesta: En Colombia, los computadores no están realizando tareas de importancia; además, no es de interés entrar a legislar sobre este campo, ya que no existen las bases para poder hacerlo.

Después de esta respuesta, se formuló el siguiente interrogante: Se supone que alguien redacta un proyecto de ley para tipificar esto como un delito, ¿cuál es el camino que debe seguir para que sea aprobado como ley? La respuesta que se encontró fue más increíble todavía: "ese proyecto de ley tiene que pasar por tanta burocracia, que si no tiene ningún problema se estaría aprobando como ley a los cuatro años". (Esto siguiendo los conductos normales).

Como se ve, antes de elaborar un proyecto de ley, para que se tipifique como delito, la violación de un sistema operacional, es mejor hacer una reforma al Código Penal para actualizarlo y volverlo eficiente.

2. RED

En este capítulo y en el siguiente, se llevará a cabo el caso de estudio. El caso consiste en analizar una red, descubrir las debilidades que tiene y atacar el sistema por esas debilidades. Para su desarrollo se contó con la ayuda del doctor José Hernando Bahamón, del Ingeniero Alvaro Pachón y del administrador de la red, Juan Manuel Madrid.

2.1. Red por atacar

La red por atacar, fue UNIX del ICESI. Se la escogió por las ventajas que se tendrían en cuanto al apoyo logístico; además porque los conocimientos adquiridos servirían para el próximo semestre.

2.2. Configuración de la red

La red en cuestión es una ETHERNET, con topología bus, que permite hacer multipunto. En esta red se encuentran dos servidores, un Sun y un Compaq. Ambos tienen conectadas varias terminales brutas.

El Sun y el Compaq se pueden comunicar entre ellos nada más, a través del protocolo TPC/IP. Las terminales brutas se comunican a través del protocolo CSMA/CD (Carrier Sense Method Acces/Carrier Detect).

El dispositivo físico utilizado para conectar las estaciones y las terminales es un cable coaxial delgado.

2.3. Configuración del sistema

En el tema anterior se mencionó la palabra "Multipunto"; esto quiere decir que el sistema es un sistema multiusuario donde las terminales se pueden comunicar entre sí, con la característica de que cuando habla una terminal todas las demás escuchan.

El criterio conque se maneja el derecho de quién debe usar el cable, se basa en la ley del más fuerte. Es decir, el primero en llegar es quien se adueña del medio de comunicación.

El servidor Sun usa el sistema operativo SUNOS 5.3 y el servidor Compaq usa el UNIX 3.2 de Compaq.

3. VULNERABILIDAD

3.1. Debilidades identificadas

Las siguientes son las debilidades que encontramos en el sistema:

* La primera dificultad que se pensó se iba a tener era: ¿Cómo se iba a entrar al sistema si no se poseía una cuenta? Pero, ¡cuál fue nuestra sorpresa! Se encontró una cuenta que es pública, y lo mejor, sin necesidad de dar clave, y se pudo entrar al sistema.
* El sistema no está protegido contra el procedimiento del *Caballo de*

Troya. Cuando se menciona al *Caballo de Troya,* no nos estamos refiriendo al virus de los microcomputadores, sino a los programas que utilizan el principio empleado en la guerra de Troya.

* No se realizan auditorías.

3.2. **Ataque en las debilidades**

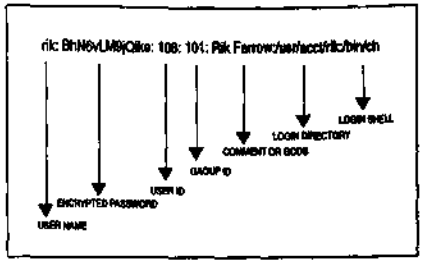
En UNIX, para poder realizar una tarea, se necesita que el grupo de trabajo o la persona tengan una cuenta asignada por el administrador de la red; sin esto no se puede hacer nada, ya que no habría forma de entrar al sistema y utilizar sus recursos.

Al caer en la cuenta de esto, se pensó que no se iba a poder realizar algo, y que se tendría que recurrir a que se nos asignara una cuenta por parte del administrador de la red, y tratar de saltarnos las limitaciones impuestas a nuestra cuenta. Pero, ¡cuál fue nuestra sorpresa! Se pudo entrar al sistema por medio de una cuenta pública de "Reserva", la cual no pedía "password" de entrada.

Si la meta era la de lograr entrar al sistema sin poseer una cuenta, prácticamente se había logrado sin mayor problema y se podría considerar que se había violado la seguridad del sistema.

Al estar en el sistema, se planteó el objetivo de tratar de colocarnos como superusuarios del sistema. Aprovechando una de las cualidades del sistema UNIX, navegamos por todos los directorios hasta que encontramos la lista de las cuentas activas y a quiénes pertenecían. Para una persona que conozca bien UNIX, esta información le es de mucha importancia, ya que puede identificar cuáles son los atributos de una cuenta en especial y tratar de adivinar el password de la cuenta. (Ver Figura 5).

Figura 5.
Estructura del archivo de claves



Para adivinar el "password" de una cuenta, se podría recurrir al método del ensayo y el error, también se podría identificar quién es la persona y, por medio de la sicología, ir tratando de dar con el password, ya que la mayoría de las personas no saben escoger un password adecuado. O por último, el más elaborado, la utilización del programa denominado *Caballo de Troya,* el cual consistía en hacerle creer a la persona que su "password" lo digitó mal la primera vez y que tiene que volver a digitarlo. Este programa lo que hace es que cuando se da la primera vez el "password", él lo toma y lo guarda, y así la persona que hizo el programa nada más tiene que consultar el programa para averiguar cuál es el "password" de esa persona.

Obviamente, realizar este método está fuera del alcance de nuestros conocimientos; sin embargo, el sistema no está protegido contra este tipo de programas y se hubiera podido lograr el robo de un "password" válido.

Otra forma de tratar de descubrir cuáles son los "password" válidos, es hacer un programa que "criptoanalice", ya que los "passwords" están "encriptados". Sobre criptografía se ha hablado mucho, ya que se han desarrollado métodos para descifrar, sin ningún problema, textos "encriptados" en los cuales se consideraba que se había utilizado un buen método. Hay que recordar que el realizar criptografía, a nivel del "software", es complicado; por lo tanto, en sistemas que están diseñados con un nivel medio de seguridad, dicho programa no es muy potente y, por tanto, fácil de descifrar.

Lamentablemente, ninguno de nuestros intentos dio resultado y no se pudo lograr la meta trazada, que era la de colocarnos como super usuarios. Se esperaba que los intentos llevados a cabo fueran detectados por la auditoría, la cual debe ser realizada por el administrador de la red; pero se encontró que no hay auditoría, ya que es muy difícil de llevar a cabo. Se nos informó que se recurría al registro de claves, cuando se tenía la sospecha de que había alguien intentando violar el sistema.

4. **CONCLUSIONES**

* La seguridad en los sistemas operacionales no es cualquier cosa que se pueda pasar por alto.
* Hasta el momento, no se puede afirmar que un sistema es 100% seguro, ya que siempre tendrá un lado débil, por el cual va a fallar, tarde o temprano.
* Sería bueno que desde la misma universidad se empezara a enseñar, a los futuros ingenieros de sistemas, la importancia de la seguridad en los sistemas operacionales.
* Empezar a presionar a la clase política y a los magistrados para que modernicen el Código Penal, para que se pueda hacer una buena legislación en cuanto se refiere al delito por medio del computador.
* Hablar de seguridad en los sistemas operacionales implica también hablar de la seguridad a nivel del "software" y no únicamente de la seguridad física.
* Siempre existirá alguien que trate de violar la seguridad de un sistema, ya sea como un reto, o por hacer una maldad.

BIBLIOGRAFIA

DEITEL M. Harvey, *Introducción a los Sistemas Operativos.*

HUNKA H. Bruce, *Administration and Managen Handbook.*

UNIX *System V relax 4 Administration.*

Grolier Inc. *The New Grolier Multimedia.* Encyclopedia Release 6.

Grandes Estafas por Computador.

Principios Avanzados de Seguridad para UNIX.

GLOSARIO DE TERMINOS

Arch, arc = Archivo.

R = Lectura.

W = Escritura.

X = Ejecución.

Impre = Impresora.

Grafic = Graficadora.

MODELOS DE ESTIMACION DE COSTO Y ESFUERZO EN PROYECTOS DE SOFTWARE

RAFAEL ANTONIO GORDILLO
ALBERTO NOE GIRALDO
LUIS BALDOMERO DAVILA

Alumnos del curso de Investigación de VIII Semestre de Ingeniería de Sistemas del ICESI.

INTRODUCCION

En todo proyecto de desarrollo de software están incluidos factores que afectan, de una u otra forma, el costo final del producto, así como también los requerimientos de personal. El factor tiempo también ha de tenerse en cuenta a la hora de evaluar las características finales del producto. Es necesario, por lo tanto, contar con herramientas y técnicas que permitan estimar el costo, el esfuerzo y el tiempo inherentes al desarrollo de un producto de software.

Este documento presenta una descripción de los principales modelos utilizados en la estimación de costo, esfuerzo y tiempo de desarrollo, así como los parámetros y variables en que se basan estos modelos.

El principal modelo que se trata en este documento es el Modelo COCOMO, propuesto por Barry Boehm, puesto que, en la actualidad, es el más utilizado y referenciado. Este modelo no sólo aporta un conjunto de ecuaciones sino que, además, permite presentar razonadamente el porqué de sus resultados, comprobar las diferentes valoraciones que aportan sus tablas e identificar claramente los factores de mayor influencia en el proceso de desarrollo.

Ya que los modelos basan sus estimaciones en la determinación previa del tamaño del producto, se tratará la metodología de los Puntos de Función (PF) desarrollada por A. Albrecht, que permite valorar el tamaño de una aplicación, en las etapas previas de su desarrollo.

OBJETIVO GENERAL

Elaborar un documento sobre los diferentes modelos y herramientas que existen para la estimación de costo, esfuerzo y tiempo en el desarrollo de software, el cual sirva de guía, para su aplicación, a quienes desarrollan productos de software.

OBJETIVOS ESPECIFICOS

1. Identificar los diferentes modelos existentes para la estimación de costos y esfuerzos para el desarrollo de software.
2. Identificar los principales parámetros y variables utilizados por los modelos de estimación.

3. Determinar las condiciones básicas que permitan realizar estimaciones más confiables.
4. Definir en qué etapas del desarrollo de un proyecto de software son aplicables las estimaciones de los modelos.

1. ESTIMACION EN LOS PROYECTOS DE SOFTWARE

Aunque la estimación es más un arte que una ciencia, es una actividad importante que no debe llevarse a cabo de forma descuidada. Existen técnicas útiles para la estimación de costos y tiempos. Y dado que la estimación es la base de todas las demás actividades de la planificación y que la planificación del proyecto sirve como guía para una buena ingeniería del software, no es en absoluto aconsejable embarcarse sin ella.

Un proyecto de software generalmente comienza con la producción de un plan de desarrollo, donde se identifica el trabajo por ser realizado, el presupuesto y el itinerario. La efectividad de la planeación depende de una estimación correcta.

Una importante faceta en el desarrollo de software es la capacidad de estimar el costo asociado con las primeras etapas del ciclo de vida. La estimación de recursos, costos e itinerarios requiere experiencia y acceso a una buena información histórica. Además, esta estimación conlleva un riesgo inherente, y los factores que lo aumentan son:

- **La complejidad del proyecto:** es una medida relativa que se ve afectada por la familiaridad con anteriores esfuerzos.
- **El tamaño del proyecto:** es el factor más importante que afecta la precisión y la eficacia de las estimaciones. A medida que crece el tamaño, la interdependencia entre los distintos elementos del software crece rápidamente. Sin embargo, estimar el tamaño del software es un problema complicado que requiere conocimiento específico de las funciones del sistema en términos del entorno, complejidad e interacciones. Las medidas más frecuentes utilizadas son Líneas de Código (LDC) y Análisis de puntos de función (PF).
- **El grado de estructuración del proyecto:** en este contexto, la estructuración se refiere a la facilidad con que las funciones pueden ser modularizadas y a la naturaleza jerárquica de la información que debe ser procesada. A medida que aumenta el grado de estructuración, la posibilidad de estimar con precisión mejora y el riesgo disminuye.

La disponibilidad de información histórica también determina el riesgo de la estimación. Cuando se dispone de una amplia métrica del software de proyectos pasados se pueden hacer las estimaciones con gran seguridad, se pueden establecer métodos para evitar anteriores dificultades y se puede reducir el riesgo global.

1.1. Recursos humanos

Es necesario evaluar el entorno de desarrollo y seleccionar las habilidades técnicas que se requieren para llevar a cabo el desarrollo. Hay que especificar tanto la posición dentro de la organización como la especialidad.

El número de personas requerido para un proyecto de software sólo puede ser determinado después de hacer una estimación del esfuerzo del desarrollo (por ejemplo, personas-mes o personas-año).

1.2. Categorías del hardware

Durante la planificación del proyecto de software se deben considerar tres categorías de hardware: el **sistema de desarrollo,** que está compuesto por la computadora que se utilizará durante la fase de desarrollo del software y sus periféricos asociados; la **máquina objetivo,** que es el equipo donde se ejecutará el software desarrollado, y los demás elementos de hardware del nuevo sistema.

1.3. Recursos del software

Al igual que utilizamos el hardware como herramienta para construir nuevo hardware, utilizamos el software como ayuda en el desarrollo de nuevo software. La primera aplicación que se le dio al software en el desarrollo de software fue lo que se denominaba **reconstrucción.** Se comenzó por escribir un primitivo traductor de lenguaje ensamblador a lenguaje máquina, y se usó para desarrollar un ensamblador más sofisticado. Aumentando las posibilidades de cada versión previa, los equipos de desarrollo fueron reconstruyendo eventualmente el software, hasta llegar a construir compiladores de lenguajes de alto nivel y otras herramientas.

1.4. Software reutilizable

Cualquier estudio sobre recursos de software no estará completo sino se considera la **reusabilidad**, esto es, la creación y reutilización de módulos constructivos de software. Estos módulos constructivos deben ser catalogados para una fácil referencia, estandarizados para una fácil aplicación y validados para una fácil integración.

La mayoría de los observadores de la industria del software están de acuerdo en que la mejora de la productividad del desarrollo del software y de la calidad de los productos minimizará el nivel de mantenimiento. En un mundo así, abundaría el software reutilizable. Los módulos constructivos de software estarían disponibles para la construcción de grandes programas, con un mínimo esfuerzo de desarrollo, partiendo "de la nada".

1.5. Formas de estimación

La estimación del costo y del esfuerzo del software nunca será una ciencia exacta. Son demasiadas las variables (humanas, técnicas, de entorno, políticas) que pueden afectar el costo final del software y el esfuerzo aplicado para desarrollarlo. Sin embargo, la estimación del proyecto de software puede dejar de ser un oscuro arte para convertirse en una serie de pasos sistemáticos que proporcionen estimaciones con un grado de riesgo aceptable.

Para realizar estimaciones seguras de costos y esfuerzos tenemos varias opciones posibles:

1. Dejar la estimación para más adelante (obviamente, podemos realizar una estimación del 100% fiable, tras haber terminado el proyecto).
2. Utilizar técnicas de descomposición relativamente sencillas, para generar las estimaciones de costo y esfuerzo del proyecto.
3. Desarrollar un modelo empírico, para el cálculo de costos y esfuerzos del software.
4. Adquirir una o varias herramientas automáticas de estimación.

Infortunadamente la primera opción, aunque atractiva, no es práctica. Las estimaciones de costos han de ser proporcionadas de antemano. Sin embargo, hay que reconocer que cuanto más tiempo esperemos más cosas sabremos, y cuanto más sepamos menor será la probabilidad de cometer serios errores en nuestras estimaciones.

Las técnicas de descomposición utilizan un enfoque de "divide y vencerás", para la estimación del proyecto de software. Mediante la descomposición del proyecto en sus funciones principales y en las tareas de ingeniería de software que le corresponden, la estimación del costo y del esfuerzo pueden realizarse de una forma escalonada e idónea.

Igualmente, se pueden utilizar los modelos empíricos de estimación como complemento de las técnicas de descomposición. Cada modelo se basa en la experiencia (datos históricos) y toma como base: **d=f(v$_i$) donde:**

d: es uno de los valores estimados (por ejemplo: esfuerzo, costo, duración del proyecto).

v$_i$: son determinados parámetros independientes (por ejemplo: Líneas de código o Puntos de función estimados).

Las herramientas automáticas de estimación implementan una o varias técnicas de descomposición o modelos empíricos. Cuando se combinan con una interfaz atractiva hombre-máquina, las herramientas automáticas resultan muy atractivas para la estimación.

Cada una de las opciones viables para la estimación de costos y esfuerzo del software sólo será buena si los datos históricos que se utilizan como base de la estimación son buenos. Si no existen datos históricos, la estimación del costo descansará sobre una base muy inestable.

1.6. Parámetros y variables utilizados en la estimación

Las estimaciones que suelen realizarse para el software involucran variables y constantes, además de datos históricos. Se suele utilizar el supuesto volumen del software (líneas de código-LDC), con otras entradas que caracterizan los factores de riesgos principales en el desarrollo.

Debido a que las estimaciones de costo y esfuerzo para un proyecto de software son demasiado complejas si se consideran como un solo problema, se hace necesario descomponerlo en problemas más pequeños. Las técnicas de estimación, las líneas de código LDC y los puntos de función PF difieren en el nivel de detalle que requiere la descomposición del problema. Los datos de LDC y PF se emplean de dos formas en el proceso de estimación: como **variables de estimación** y como **métricas de base,** recogidas de anteriores proyectos.

Debe tenerse en cuenta que mientras el LDC se estima directamente, los PF se determinan indirectamente, mediante la estimación del número de entradas, salidas, archivos de datos, peticiones e interfaces externas.

1.6.1. *Líneas de código (LDC)*

La métrica de tamaño tradicional para estimar tamaño, esfuerzo y para medir productividad ha sido el de las líneas de código. Un gran número de modelos de estimación de costo ha sido propuesto, muchos de los cuales están fundamentados en las líneas de código (o miles de líneas de código - KLDC). Generalmente, los modelos de estimación de esfuerzo se componen de dos partes; la primera ofrece una base de estimación en función del tamaño del software y es de la forma:

$$E=A+B\ (KLDC)^C$$

Donde E es el esfuerzo estimado en hombres-mes; **A, B** y **C** son constantes y **KLDC** es el número estimado en miles de líneas de código en el sistema final.

En la segunda parte, se ajusta la base de estimación para responder a la influencia de factores ambientales. Entre estos factores ambientales se incluyen el uso de técnicas, tales como el código estructurado, el diseño ***Top-Down***, la revisión estructurada y los equipos de programadores especializados; la capacidad de personal; los requerimientos de hardware. Por ejemplo, el Modelo COCOMO usa líneas de código elevadas a una potencia entre 1.05 y 1.20, para determinar la base de estimación.

El exponente específicamente depende de la simplicidad promedio o de la complejidad del proyecto.

Los siguientes son ejemplos de modelos típicos:

$E=5.2(KLDC)^{0.91}$	Modelo Wlaston-Félix
$E=5.5+0.73(KLDC)^{1.16}$	Modelo Bailey-Basili
$E=3.2(KLDC)^{1.05}$	Modelo COCOMO Básico
$E=3.0(KLDC)^{1.12}$	Modelo COCOMO Intermedio
$E=2.8(KLDC)^{1.2}$	Modelo COCOMO Avanzado
$E=5.288(KLDC)^{1.044}$	Modelo Doty.

La definición de KLDC es importante a la hora de comparar estos modelos. Algunos modelos incluyen líneas de comentarios y otros no. La definición de qué tipo de esfuerzo se ha estimado es igualmente importante, ya que el esfuerzo se puede asumir como la sola codificación o como el análisis total (el diseño, la codificación y la fase de pruebas conjuntamente). Esto nos indica que la comparación de modelos que usan criterios distintos, en cuanto a los mismos parámetros, es relativamente difícil.

Hay numerosos problemas con el uso de líneas de código como unidad de medida para el tamaño del software. El primero consiste en establecer una definición exacta de lo que es una línea de código y, a la vez, que esta definición sea aceptada universalmente. Otra dificultad con las líneas de código es su dependencia respecto del lenguaje utilizado. No es posible comparar directamente proyectos desarrollados en lenguajes diferentes. Por ejemplo, el tiempo por línea para un lenguaje de alto nivel puede ser mayor que para uno de bajo nivel. Puede que no sea posible lograr un mínimo de líneas de código en un lenguaje de alto nivel para una función de un lenguaje de bajo nivel.

Todavía se cuenta con un problema más y es el de estimar el número de líneas requeridas para desarrollar un sistema, contando sólo con la información proveniente de la fase de requerimientos o diseño. Si los modelos para costos, basados en el tamaño, son utilizados, es necesario tener la capacidad de predecir el tamaño del producto final tan pronto como sea posible. Es aquí donde entra en juego la recolección de datos. Infortunadamente la estimación del tamaño, usando las métricas LDC, depende de experiencias previas con proyectos similares, más específicamente de datos históricos.

Una forma de estimar el tamaño del proyecto, en líneas de código LDC, consiste en descomponer el proyecto en sus funciones principales y se estiman las LDC para cada una de éstas. Las estimaciones de las funciones se combinan para producir una estimación global del proyecto. A partir de los datos históricos, se estima tanto el valor optimista, más probable,como el pesimista del LDC. Si no se cuenta con datos históricos no queda más remedio que recurrir a la intuición para los valores antes mencionados. Luego, se calcula el **valor agregado** del LDC, el cual será una media ponderada de las estimaciones pesimistas (a), más probable (m) y optimista (b), así:

$$E=(a+4m+b)/6$$

Por ejemplo, consideremos un paquete de software por desarrollar, para una aplicación de diseño asistido por computador (CAD). Supongamos que las funciones principales para este proyecto son:

- Interfaz de usuario y facilidad de control.
- Análisis geométrico bidimensional.
- Análisis geométrico tridimensional.

- Control de periféricos.
- Módulos de análisis de diseño.

El paso siguiente consiste en establecer los valores optimista, probable y pesimista para cada una de las funciones anteriores. Tomemos por ejemplo el control de periféricos, para el cual se tiene un valor optimista de 2.000 LDC, un valor probable de 2.100 LDC y un valor pesimista de 2.450 LDC. Aplicamos entonces la función del valor esperado y obtenemos el siguiente resultado:

Función	Optimista	Probable	Pesimista	Esperado
Control de periféricos	2.000	2.100	2.450	2.140

Realizamos el mismo proceso con las demás funciones y construimos una tabla como la siguiente:

Función	Optimista	Probable	Pesimista	Esperado
Interfaz de usuario y facilidad de control	1.800	2.400	2.650	2.340
Análisis geométrico bidimensional	4.100	5.200	7.400	5.380
Análisis geométrico tridimensional	4.600	6.900	8.600	6.800
Control de periféricos	2.000	2.100	2.450	2.140
Módulos de análisis de diseño	6.600	8.500	9.800	8.400
Total				25.060

Sumando verticalmente, en la columna del valor esperado, se establece una estimación de 25.060 LDC.

1.6.2. *Análisis del punto de función*

El analisis del punto de función es un método de cuantificación del tamaño y complejidad del software, en términos de las funciones que el sistema entrega al usuario. La función entregada no está relacionada con el lenguaje o herramientas utilizadas en el desarrollo del proyecto. Este análisis está diseñado para aplicaciones comerciales; no es apropiado para aplicaciones de tipo técnico o científico, ya que éstas tratan con algoritmos complejos, que los puntos de función no están diseñados para manejar.

En este método se contabilizan los varios tipos de función de usuarios, los cuales se dividen en datos y transacciones. Los tipos de función de datos corresponden a Archivos lógicos internos y Archivos externos de interfase. Los tipos de función de transacciones corresponden a entradas, salidas y consultas externas. Todos estos términos serán tratados más adelante.

El enfoque de puntos de función (PF) tiene características que se sobreponen a las deficiencias en el uso de las líneas de código, mencionadas anteriormente. Primero, los puntos de función son independientes de la metodología, las herramientas y los lenguajes utilizados en el desarrollo del proyecto. Segundo, los puntos de función pueden estimarse con base en la especificación de los requerimientos o en el diseño; además, hacen posible la estimación del esfuerzo para el proyecto en las primeras fases del desarrollo. Ya que los puntos de función están directamente relacionados con la especificación de requerimientos, cualquier cambio en éstos puede ser seguido fácilmente por una nueva estimación. Tercero, los usuarios no especializados en el sistema tienen una mejor comprensión de qué están midiendo los puntos de función.

1.7. Estimación del costo del software

Básicamente, la estimación del costo se hace multiplicando las unidades del tamaño del software para factores apropiados de productividad. Muchas ecuaciones del costo tienen la forma:

$E=pL^c$ donde:

E=esfuerzo

p=ajuste de productividad

L=tamaño en líneas de código

c>1, una constante.

El ajuste de productividad puede ser un factor simple o compuesto de factores, tal como aparece en el Método COCOMO.

Los principales factores para determinar la productividad son, en su orden de importancia, la capacidad del personal, la complejidad de la aplicación, el software reutilizable y la tecnología aplicada.

La productividad se ve muy influenciada por el tipo de personal escogido para realizar el trabajo; luego está el tipo de trabajo ejecutado. Por ejemplo, COCOMO prefiere el ajuste de la estimación del tamaño (DSI) para contabilizar la reutilización antes de aplicar cualquier factor de productividad.

1.8 Estimación del esfuerzo

La estimación del esfuerzo es la técnica más utilizada para calcular el costo de un proyecto de ingeniería del software. En la resolución de cada tarea del proyecto se aplica un número determinado de personas/día, personas/mes o personas/año. Se asocia un costo a cada unidad de esfuerzo y se deriva el costo total estimado.

Al igual que las técnicas LDC y PF, la estimación del esfuerzo comienza con una delimitación de las funciones del software, obtenidas del ámbito del software. Para cada función debe realizarse una serie de tareas de ingeniería del software, tales como análisis de requisitos, diseño, implementación y pruebas. Se aplican las tarifas laborales (costo/unidad de esfuerzo) a cada una de las tareas de ingeniería del software.

En el último paso se calculan los costos y el esfuerzo para cada función y se deriva el costo total del proyecto.

1.9. Modelos empíricos de estimación

Un modelo de estimación utiliza fórmulas derivadas empíricamente para predecir los datos que se requieren, en el paso de planificación del proyecto de software. Los datos empíricos que soportan la mayoría de los modelos se obtienen de una muestra de proyectos limitada. Se identifican cuatro clases de modelos: **modelos univariables estáticos, modelos multivariables estáticos, modelos multivariables dinámicos y modelos teóricos.**

Un modelo univariable estático toma la forma:

Recurso $=c_1$ x(característica estimada)c_2

donde el **recurso** podría ser el esfuerzo, la duración del proyecto, la cantidad de personal o las líneas requeridas de documentación del software. Las constantes c_1 y c_2 se derivan de los datos recopilados de los anteriores proyectos. La *característica estimada* puede ser la cantidad de líneas del código fuente, el esfuerzo (si ya está estimado) u otra

característica del software. Un ejemplo de este tipo de modelo es la versión básica del modelo de costo constructivo o COCOMO.

Los **modelos multivariables estáticos** emplean los datos históricos para obtener relaciones empíricas. Un modelo de esta categoría toma la forma:

$$\text{Recurso} = c_1 e_1 + c_2 e_2 + \ldots$$

donde e_i es la característica i-ésima del software y $c_i 1 + c_i 2$ con constantes obtenidas para e_i.

Los **modelos multivariables dinámicos** proyectan los requisitos de recursos como una función del tiempo. Los recursos se definen en una serie de pasos consecutivos en el tiempo, que asignan cierto porcentaje de esfuerzo (o de otro recurso) a cada etapa del proceso de ingeniería del software. Este modelo incluye una "curva continua de utilización del recurso" como hipótesis, y a partir de ella obtiene ecuaciones que modelizan el comportamiento del recurso.

El **modelo teórico de recurso** examina el software desde un punto de vista microscópico; es decir, las características del código-fuente (por ejemplo: el número de operadores y operandos).

2. MODELO COCOMO

Es el modelo más completo existente hasta el momento y ha sido propuesto por Barry Boehm en su libro *Software Engineering Economics.* En este libro, Boehm trata el ciclo de vida del software desde una perspectiva económica, en términos del tiempo y el esfuerzo requeridos para completar cada fase del ciclo de vida del software. Establece una relación matemática que permite estimar el esfuerzo y el tiempo requerido para desarrollar un proyecto, en función del número de instrucciones-fuente desarrolladas.

El modelo inicial, que tenía un único modo de desarrollo, se ajustó utilizando los datos correspondientes a doce proyectos completos. El modelo obtenido fue evaluado contrastando sus resultados frente a los obtenidos en otros 36 proyectos.

Sin embargo, la aplicación del modelo a otros proyectos, en una amplia variedad de entornos, indicaron que un único modo de desarrollo no era suficiente para explicar las variaciones observadas, lo que indujo a introducir en el modelo actual tres modos de desarrollo, de forma que sus predicciones pudiesen aplicarse, con un buen grado de precisión, en cualquier entorno de desarrollo.

Boehm propuso entonces una jerarquía de modelos de estimación para el software denominada COCOMO, por Constructive COst MOdel, o Modelo Constructivo de Costo; dicha jerarquía de modelos está constituida así:

Modelo básico

Modelo intermedio

Modelo avanzado o detallado.

Para poder aplicar adecuadamente estos modelos se deben tener en cuenta las siguientes consideraciones:

1. El factor principal sobre el que se basan las estimaciones de costos es el tamaño del producto, es decir, el número de instrucciones-fuente entregadas DSI *(Delivered Source Instructions).*

   Este término incluye todas las instrucciones creadas por el personal asignado al proyecto, y procesadas en código máquina, excluyendo las líneas de comentarios.

   Igualmente, deben considerarse las sentencias de Lenguaje de Control de Trabajos (JCL), las sentencias de formatos y las declaraciones de datos.

   Las instrucciones se definen como líneas de código y, por tanto, toda línea que contenga dos o tres sentencias se considerará como una sola instrucción.

2. COCOMO considera sólo las fases del período de desarrollo, comprendidas desde el comienzo de la fase de diseño del producto hasta el final de la fase de integración y prueba.

   Los costos, esfuerzo y tiempo de las otras fases (planificación, especificaciones, mantenimiento, etc.) deben estimarse por separado.

3. Los costos estimados mediante este modelo no incluyen los correspondientes a las actividades de formación del usuario, la instalación y los trabajos de conversión, aunque estas actividades se realicen durante la etapa de desarrollo. Se cubren todos los costos directos del proyecto; es decir, la gestión del proyecto, los trabajos de documentación y librería, pero se excluyen los correspondientes al personal del centro de cómputo, las secretarias, los subalternos y los altos directivos que no estén directamente involucrados en el desarrollo del proyecto.

4. La unidad de esfuerzo HM (Hombre-Mes) supone un total de 152 horas de trabajo por persona, valor tomado basándose en la experiencia práctica, que considera aspectos tales como vacaciones, permisos, festivos, licencias, etc.

Para convertir las unidades HM a otras unidades empleamos las siguientes equivalencias:

1 HM = 152 MH (MH: hombre-hora)
1 HM = 19 MD (MD: hombre-día)
1 HM = 1 MY/12 (MY-Hombre-año).

5. Se supone que después de la fase de especificación de requerimientos, éstas no cambiarán substancialmente, aunque evidentemente esto será inevitable. En el caso que se produzcan modificaciones significativas, se estaría obligado a revisar las estimaciones anteriores.

6. El modelo avanzado del COCOMO asume que los factores que influyen sobre los costos de desarrollo dependen de la fase en que aparecen. El COCOMO básico y el intermedio no consideran los factores que afectan el costo, excepto para distinguir entre desarrollo y mantenimiento.

7. En los costos por fase se incluyen todos aquellos que se produzcan durante ese espacio de tiempo. Es decir que los costos relativos a los trabajos de actualización del plan de integración y prueba, la terminación del plan de pruebas de aceptación, la actualización de la documentación, etc., se incluyen dentro de los costos de la fase de diseño detallado.

8. Para convertir las estimaciones obtenidas de hombres-mes (HM) a unidades monetarias, la mejor forma es aplicar un valor medio de unidades monetarias, por unidad de esfuerzo, para cada una de las fases en el desarrollo.

2.1. Modos de desarrollo

Como se mencionó anteriormente, COCOMO comprende tres modos de desarrollo que de algún modo reflejan las diferentes condiciones o entornos de desarrollo, y aunque matemáticamente pueden expresarse en forma similar, conducen a estimaciones diferentes; estos modos son los siguientes:

2.1.1. *Modo orgánico*

Este modo abarca sólo los proyectos relativamente pequeños y sencillos; en éstos trabajan pequeños equipos, con buena experiencia en otros proyectos relacionados con la misma organi-

zación y con pleno conocimiento de cómo el sistema en desarrollo contribuirá con los objetivos de la organización.

Esto significa que la mayoría de las personas pueden contribuir de forma efectiva a la terminación puntual de cada una de las etapas, sin necesidad de mucha comunicación, para determinar con precisión las tareas que cada uno debe desarrollar en el proyecto. Existe, por lo tanto, facilidad para establecer los requisitos y las especificaciones de cada una de las fases del proyecto.

Otros factores característicos de este modo son:

- Un ambiente generalmente estable de desarrollo, con muy poca ocurrencia de nuevo hardware y procedimientos operacionales.
- Mínima necesidad para innovar arquitectura de procesamiento de datos o algoritmos.
- Muy pocos proyectos, en este modo, han desarrollado productos con más de cincuenta KLDC.

2.1.2. *Modo semilibre*

Representa un estado intermedio entre el modo orgánico y el modo restringido. Son proyectos intermedios en tamaño y complejidad en los que equipos de trabajo, con variados niveles de experiencia, deben satisfacer requisitos poco o medio rígidos.

Con respecto a la experiencia del equipo de trabajo, se consideran las siguientes situaciones:

- Todos los miembros del equipo tienen un nivel intermedio de experiencia en sistemas relacionados con el proyecto.
- El equipo de desarrollo está formado por una mezcla de gente experta e inexperta.
- Los miembros del equipo tienen experiencia en algunos de los aspectos del sistema que se pretende desarrollar, pero no en todos.

Un típico proyecto, en modo semilibre, puede ser un sistema de proceso de transacciones con pocas interfaces rigurosas (por ejemplo con la terminal hardware) y algunas otras interfaces muy flexibles (naturaleza y formatos de presentación de mensajes). El término semilibre hace referencia a la flexibilidad parcial que presenta un proyecto de este tipo, el cual generalmente sobrepasa las 300 KLDC.

2.1.3. *Modo restringido*

La principal característica de un proyecto, en este modo, es la necesidad de operar dentro de fuertes restricciones. El término incorporado se refiere básicamente al hecho que el producto debe operar dentro de un complejo altamente interconectado de hardware, software, reglas y procedimientos operacionales, como es el caso, por ejemplo, de un sistema de control de tráfico aéreo.

En condiciones de modo de desarrollo restringido, el costo de modificar parte del complejo sistema es tan alto, que sus características se podrían considerar inmodificables. Como resultado, este modo de proyecto no siempre tiene la opción de realizar fácilmente cambios de software, por más sencillos que sean, debido a la modificación de los requerimientos y la especificación de las interfaces. Por lo tanto, el proyecto debe emplear más esfuerzo en realizar y adecuar los cambios y correcciones, y en asegurar que el software actualmente satisfaga las especificaciones. También se debe emplear más esfuerzo en verificar que los cambios se realicen correctamente.

Los proyectos, en este modo, generalmente abarcan áreas más amplias y menos conocidas que en los casos anteriores.

Los modelos COCOMO calculan esencialmente el costo a la entrega, que puede ser una pequeña proporción del costo total de la vida del software.

2.2. Modelo COCOMO básico

Es un modelo univariable, estático, que calcula el costo y el esfuerzo de un proyecto de software, exclusivamente en función de su tamaño, expresado en líneas de código (KLDC) estimadas. Este modelo es apropiado para una pronta y rápida estimación del costo del software, pero su precisión es necesariamente limitada, porque carece de factores que involucren los aspectos del hardware, la calidad del personal, la experiencia, el uso de modernas herramientas y técnicas y otros tributos del proyecto que tienen una significativa influencia en los costos del software.

La siguiente tabla presenta las ecuaciones para los tres modos de desarrollo del modelo COCOMO básico:

Modo	Esfuerzo	Tiempo estimado
Orgánico	$E=2.4(KLDC)^{1.05}$	$T=2.5(E)^{0.38}$
Semilibre	$E=3.0(KLDC)^{1.12}$	$T=2.5(E)^{0.35}$
Restringido	$E=3.6(KLDC)^{1.20}$	$T=2.5(E)^{0.32}$

La ecuación **$E=2.4(KLDC)^{1.05}$** es utilizada para la estimación de hombres-mes (HM) requeridos para desarrollar el tipo más común de productos de software, en términos de miles de instrucciones-fuente entregadas KLDC.

Se obtiene también que **$T=2.5(HM)^{0.38}$** es una ecuación para estimar el tiempo de desarrollo, en meses.

Estas ecuaciones son aplicables a la mayoría de los proyectos de software, de tamaño mediano/pequeño, desarrollados en un ambiente de software familiar. Ejemplo:

Du Bridge Chemical Inc. es una importante compañía de productos químicos, que planea desarrollar una aplicación para mantener la información sobre materias primas. Se desarrollará una aplicación por un equipo propio de programadores y analistas, quienes han desarrollado aplicaciones similares durante varios años. Además, es un buen ejemplo de un software de modo orgánico. Un estudio inicial ha determinado que el tamaño del programa será aproximadamente de 32.000 líneas de código (32 KLDC). De las ecuaciones básicas obtenemos las características del proyecto, que son:

esfuerzo:	$E=2.4(32)^{1.05}$ =91 hombres-mes
productividad:	32.000 LDC/91HM=352 LDC/HM
tiempo de desarrollo:	$T=2.5\ (91)^{0.38}$= 14 meses,

y el valor medio del número de personas que, de tiempo completo, serían necesarias para desarrollar el proyecto es:

P_E= 91 HM14 meses = 65 FSP

FSP: *Full-time-equivalent Personnel:* personal de tiempo completo para el software.

E=91 HM es el esfuerzo total en hombres-mes necesario en el proyecto.

T=14 meses es el tiempo de desarrollo, expresado en meses.

El Modelo COCOMO ofrece perfiles obtenidos de aplicar las anteriores ecuaciones a proyectos de tamaño estándar. Los tamaños estándares, utilizados por el modelo, son los siguientes:

Pequeño 2.000 líneas de código

Intermedio 8.000 líneas de código

Medio 32.000 líneas de código

Grande 128.000 líneas de código.

En la siguiente tabla puede observarse que un proyecto pequeño es esencialmente un trabajo de una persona,

mientras que un proyecto considerado grande requiere un nivel medio de 16 personas. Cabe anotar que para un proyecto pequeño el esfuerzo estimado de 5 HM hace referencia al desarrollo de 2.000 instrucciones del producto, incluyendo, por lo tanto, el esfuerzo de documentación, las pruebas, las correcciones, etc.

Tamaño	Esfuerzo (HM)	Productividad (línea/HM)	Tiempo(meses)	P_E
Pequeño 2 KLDC	5.0	400	4.6	1.1
Intermedio 8 KLDC	21.3	376	8.0	2.7
Medio 32 KLDC	91.0	352	14.0	6.5
Grande 132 KLDC	392.0	327	24.0	16.0

Obviamente, el tiempo para desarrollar un programa, por ejemplo de 2000 instrucciones para uso personal, será menor, ya que éste no requerirá de muchas exigencias, mientras que si el mismo se desarrolla como un producto profesional, entonces requerirá un poco más de tiempo y esfuerzo.

2.2.1. *Distribución del esfuerzo y tiempos por fases*

Es de esperar que las estimaciones para cada una de las fases del proyecto difieran considerablemente de un modo de desarrollo a otro. La distribución por fases varía en función del tamaño; los proyectos de gran tamaño precisan mayor tiempo y esfuerzo para desarrollar actividades, tales como la integración y la prueba, mientras que pueden reducir el tiempo durante la fase de programación, distribuyendo esta actividad entre un número mayor de programadores. Los pequeños proyectos presentan una distribución más uniforme y tienen que dedicar, relativamente, más recursos a las fases de diseño y programación que a las de prueba e integración.

Las siguientes tres tablas presentan los porcentajes de distribución del esfuerzo de desarrollo, entre las distintas fases, en los tres modos de desarrollo.

Distribución del esfuerzo estimado (%). Modo orgánico

FASE	TAMAÑO (KLDC)				
	Pequeño (2)	Interm. (8)	Medio (32)	Grande (128)	Muy grande (512)
Planificación y especificaciones	6	6	6	6	–
Diseño del producto	16	16	16	16	–
Programación	68	65	62	59	–
Diseño detallado	26	25	24	23	–
Codificación y prueba de unidades	42	40	38	36	–
Integración y pruebas	16	19	22	25	–

Distribución del esfuerzo estimado (%). Modo semilibre

FASE	TAMAÑO (KLDC)				
	Pequeño (2)	Interm. (8)	Medio (32)	Grande (128)	Muy grande (512)
Planificación y especificaciones	7	7	7	7	7
Diseño del producto	17	17	17	17	17
Programación	64	61	58	55	52
Diseño detallado	27	26	25	24	23
Codificación y pruebas de unidades	37	35	33	31	29
Integración y pruebas	19	22	25	28	31

Distribución del esfuerzo estimado (%). Modo restringido

Fase	Tamaño (KLDC)				
	Pequeño (2)	Interm. (8)	Medio (32)	Grande (128)	Muy grande (512)
Planificación y especificaciones	8	8	8	8	8
Diseño del producto	18	18	18	18	18
Programación	60	57	54	51	48
Diseño detallado	28	27	26	25	24
Codificación y prueba de unidades	32	30	28	26	24
Integración y pruebas	22	25	28	31	34

Las siguientes tres tablas presentan los porcentajes de tiempo de desarrollo, para cada una de las fases, en los tres modos de desarrollo.

Distribución del tiempo (%). Modo orgánico

Fase	Tamaño (KLDC)				
	Pequeño (2)	Interm. (8)	Medio (32)	Grande (128)	Muy grande (512)
Planificación y especificaciones	10	11	12	13	–
Diseño del producto	19	19	19	19	–
Programación	63	59	55	51	–
Integración y pruebas	18	22	26	30	–

Distribución del tiempo (%). Modo semilibre

Fase	Tamaño (KLDC)				
	Pequeño (2)	Interm. (8)	Medio (32)	Grande (128)	Muy grande (512)
Planificación y especificaciones	16	18	20	22	24
Diseño del producto	24	25	26	27	28
Programación	56	52	48	44	40
Integración y pruebas	20	23	26	29	32

Distribución del tiempo (%). Modo restringido

Fase	Tamaño (KLDC)				
	Pequeño (2)	Interm. (8)	Medio (32)	Grande (128)	Muy grande (512)
Planificación y especificaciones	24	28	32	36	40
Diseño del producto	30	32	34	36	38
Programación	48	44	40	36	32
Integración y pruebas	22	24	26	28	30

Pero ¿cómo usar estas tablas? Retomando el ejemplo anterior, vamos a calcular el esfuerzo y el tiempo durante la fase de programación. Recordemos que en este proyecto se consideran 32 KLDC estimadas; por lo tanto, de acuerdo con la tabla para el esfuerzo en el modo orgánico, tenemos que el porcentaje para el esfuerzo estimado, en programación para 32 KLDC, es del 62%; entonces:

$$Eprog=(0.62)(E)$$

$$Eprog=(0.62)(91\ HM)=56\ HM$$

De la misma manera, el porcentaje de tiempo destinado a la fase de programación, para 32 KLDC, en el modo orgánico, es del 55%; por lo tanto:

$$Tprog=(0.55)(T)$$

$$Tprog=(0.55)(14\ meses)=7.7\ meses$$

El nivel medio de personal equivalente sería de:

$$P_E=(56\ HM)/(7/7\ meses)$$
$$=7.3\ hombres.$$

Un proceso similar se puede realizar con las otras tablas, para los otros modos de desarrollo, de acuerdo con el tamaño estimado del producto cuando éste se ajusta a los tamaños estándares.

La siguiente tabla muestra los valores nominales obtenidos de aplicar los porcentajes de las tablas de esfuerzo y tiempo de desarrollo, en el modo orgánico, a los proyectos de tamaño estándar. Además, se muestran los valores relativos a la productividad del proyecto y el personal medio equivalente necesario en cada fase.

Conceptos	Fases	Pequeño (2 KLDC)	Intermedio (8 KLDC)	Medio (32 KLDC)	Grande (128 KLDC)
Esfuerzo (HM)	Planificación y requisitos	0.3	1.3	5	24
	Diseño del producto	0.8	3.4	15	63
	Programación	3.4	13.8	56	231
	Diseño detallado	1.3	5.3	22	90
	Codificación y prueba de unidades	2.1	8.5	34	141
	Integración y pruebas	0.8	4.1	20	98
	Esfuerzo total	5.0	21.3	91	392
Tiempo (Meses)	(Planificación y requisitos	0.5	0.9	1.7	3.1
	Diseño del producto	0.9	1.5	2.7	4.6
	Programación	2.9	4.7	7.7	12.2
	Integración y pruebas	0.8	1.8	3.6	7.2
	Tiempo total estimado	4.6	8.0	14	24
Personal 8.0 Equiv. (P_E)	Planificación y requisitos	0.6	1.4	2.9	
	Diseño del producto	0.9	2.3	5.6	14
	Programación	1.2	2.9	7.3	19
	Integración y pruebas	1.0	2.3	5.6	14
Productividad (LDC/HM)		400	376	352	327

De igual manera, aplicando los porcentajes y ecuaciones adecuados para los modos semilibre y restringidos, obtenemos tablas similares a la anterior para proyectos de tamaño estándar.

A continuación se presenta una tabla donde se totalizan los valores de esfuerzo, tiempo, productividad y personal equivalente, para todos los modos de desarrollo, en proyectos de tamaño estándar. Por lo tanto, no se discrimina por fases.

Esfuerzo (HM)	Pequeño (2 KLDC)	Intermedio (8 KLDC)	Medio (32 KLDC)	Grande (128 KLDC)	Muy grande (512 KLDC)
Orgánico	5.0	21.3	91	392	
Semilibre	6.5	31	146	687	3.250
Restringido	8.3	44	230	1.216	6.420
Productividad (KLDC/HM)	**Pequeño (2 KLDC)**	**Intermedio (8 KLDC)**	**Medio (32 KLDC)**	**Grande (128 KLDC)**	**Muy grande (512 KLDC)**
Orgánico	400	376	352	327	
Semilibre	308	258	219	186	158
Restringido	241	182	139	105	80

Tiempo (meses)	Pequeño (2 KLDC)	Intermedio (8 KLDC)	Medio (32 KLDC)	Grande (128 KLDC)	Muy grande (512 KLDC)
Orgánico	4.6	8.0	14	24	
Semilibre	4.8	8.3	14	24	42
Restringido	4.9	8.4	14	24	41
Personal equivalente –P_E (Hombres)	**Pequeño (2 KLDC)**	**Intermedio (8 KLDC)**	**Medio (32 KLDC)**	**Grande (128 KLDC)**	**Muy grande (512 KLDC)**
Orgánico	1.1	2.7	6.5	16	
Semilibre	1.4	3.7	10	29	77
Restringido	1.7	5.2	16	51	157

2.2.2. *Distribución del esfuerzo por actividades*

Por último, una vez determinada la distribución del esfuerzo entre las fases principales del ciclo de vida de un producto software, es necesario conocer cómo se distribuye entre las diferentes actividades, de forma que sirva de referencia para:

- preparar planes de distribución de los recursos y
- adecuar la estructura de la organización al tamaño de los equipos dedicados a cada una de las actividades que, en proyectos de gran tamaño, será necesario modificar a medida que avanza el proyecto, en función de los cambios de actividades y según la fase en que se encuentre el proyecto.

Las siguientes tablas muestran la distribución por actividades, dentro de cada una de las fases principales.

Distribución del esfuerzo por actividades. Modo restringido

	FASE																					
	Planificación y requisitos					Diseño del producto					Programación					Integración y prueba					Desarrollo				
Tamaño	P	I	M	G	MG	P	I	M	G	MG	P	I	M	G	MG	P	I	M	G	MG	P	I	M	G	MG
(%) fase completa	8	8	8	8	8	18	18	18	18	18	60	57	54	51	48	22	25	28	31	34					
ACTIVIDAD																									
Análisis de requisitos	50	48	46	44	42	10	10	10	10	10	3	3	3	3	3	2	2	2	2	2	4	4	4	4	4
Diseño del producto	12	13	14	15	16	42	42	42	42	42	6	6	6	6	6	4	4	4	4	4	12	12	12	12	12
Programación	2	4	6	8	10	10	11	12	13	14	55	55	55	55	55	32	36	40	44	48	42	43	43	44	45
Plan de pruebas	2	3	4	5	6	4	5	6	7	8	4	5	6	7	8	3	3	4	4	5	4	4	5	6	7
Verificación y validación	6	7	8	9	10	6	7	8	9	10	8	9	10	11	12	30	28	25	23	20	12	13	14	14	14
Oficina de proyectos	16	14	12	10	8	15	13	11	9	7	9	8	7	6	5	10	9	8	7	6	10	9	8	7	6
GC/CC	5	4	4	4	3	4	3	3	3	2	8	7	7	7	6	10	9	9	9	8	8	7	7	7	6
Manuales	7	7	6	5	5	9	9	8	5	5	7	7	6	5	5	9	9	8	7	7	8	8	7	7	6

Distribución del esfuerzo por actividades. Modo orgánico

	FASE																			
	Planificación y requisitos				Diseño del producto				Programación				Integración y prueba				Desarrollo			
Tamaño	P	I	M	G	P	I	M	G	P	I	M	G	P	I	M	G	P	I	M	G
(%) Fase completa				6				16	69	65	62	25	16	19	22	25				
ACTIVIDAD																				
Análisis de requisitos			46				15				5				3				6	
Diseño del producto			20				40				10				6				14	
Programación			3				14				58				34				48 47 46 45	
Plan de pruebas			3				5				4				2				4	
Verificación y validación			6				6				6				34				10 11 12 13	
Oficina de proyectos			15				11				6				7				7	
GC/CC			2				2				6				7				7	
Manuales			5				7				5				7				7	

Distribución del esfuerzo por actividades. Modo semilibre

	FASE																			
	Planificación y requisitos					Diseño del producto					Programación									
Tamaño	P	I	M	G	MG	P	I	M	G	MG	P	I	M	G	MG					
(%) Fase completa	7	7	7	7	7	17	17	17	17	17	64	61	58	51	52					
ACTIVIDAD																				
Análisis de requisitos	48	47	46	45	44	12.5	12.5	12.5	12.5	12.5	4	4	4	4	4					
Diseño del producto	16	16.5	17	17.5	18	41	41	41	41	41	8	8	8	8	8					
Programación	2.5	3.5	4.5	5.5	6.5	12	12.5	13	13.5	14	56.5	56.5	56.5	56.5	56.5					
Plan de pruebas	2.5	3	3.5	4	4.5	4.5	5	5.5	6	6.5	4	4.5	5	5.5	6					
Verificación y validación	6	6.5	7	7.5	8	6	6.5	7	7.5	8	7	7.5	8	8.5	9					
Oficina de proyectos	15.5	14.5	13.5	12.5	11.5	13	12	11	10	9	7.5	7	6.5	6	5.5					
GC/CC	3.5	3	3	3	2.5	3	2.5	2.5	2.5	2	7	6.5	6.5	6.5	6					
Manuales	6	6	5.5	5	5	8	8	7.5	7	7	6	6	5.5	5	5					

Distribución del esfuerzo por actividades. Modo semilibre
(Continuación)

	FASE									
	Integración y pruebas					Desarrollo				
Tamaño	P	I	M	G	GM	P	I	M	G	MG
(%) Fase completa	19	22	25	28	31					
ACTIVIDAD										
Análisis de requisitos	2.5	2.5	2.5	2.5	2.5	5	5	5	5	5
Diseño del producto	5	5	5	5	5	13	13	13	13	13
Programación	33	33	37	39	41	45	45	44.5	44.5	44.5
Plan de pruebas	2.5	2.5	3	3	3.5	4	4	4.5	5	5.5
Verificación y validación	32	31	29.5	28.5	27	11	12	13	13.5	14
Oficina de proyectos	8.5	8	7.5	7	6.5	8.5	8	7.5	7	6.5
GC/CC	8.5	8	8	8	7.5	6.5	6	6	6	5.5
Manuales	8	8	7.5	7	7	7	7	6.6	6	6

2.2.3. *Proyectos de tamaño no estándar (Interpolación)*

En el caso que el tamaño estimado del producto no se ajuste a los tamaños estándares, podemos obtener el perfil del proyecto por interpolación de los datos de los proyectos estándares de tamaño inferior y superior.

Por ejemplo, supongamos que deseamos obtener el esfuerzo de desarrollo en la fase de programación de un producto de software, cuyo tamaño fijamos en 12.800 líneas fuente en el modo orgánico.

Aplicando las ecuaciones de estimación del esfuerzo total y el tiempo de desarrollo, tenemos:

Esfuerzo total $E=2.4(12.8)^{1.05}=35$ HM

Tiempo de desarrollo $T=2.5(35)^{0.38}=9.7$ meses

Para obtener el esfuerzo en la fase de programación, tenemos que referirnos a los proyectos de 8 KLDC y 32 KLDC. El porcentaje sobre el esfuerzo total de nuestro proyecto estará comprendido entre el 65% del proyecto de 8 KLDC y el 62% del proyecto de 32 KLDC.

El porcentaje buscado, empleando interpolación lineal, será:

$$y=yo+\left(\frac{y-xo}{x1-xo}\right)(y1-yo)$$

donde: **y** es el porcentaje que se desea calcular

x es el tamaño del proyecto en KLDC

yo y **y1** corresponden al 65% y 62%, respectivamente.

xo y **x1** corresponden a 8 KLDC y 32 KLDC, respectivamente.

$$y=65+\frac{12.8-8}{32-8}(62-65)=64.4$$

El porcentaje de esfuerzo, para el tamaño no estándar, es del 64.4% y, por lo tanto, el esfuerzo dedicado, en la fase de programación, será:

$$Eprog=(0.644)(E)$$
$$Eprog=(0.644)(35)=22.5 \text{ HM}$$

De la misma forma podemos obtener el porcentaje de tiempo de programación, interpolando los porcentajes de tiempo, del 59% para 8 KLDC y del 55% para 32 KLDC.

$$y=59+\frac{12.8-8}{32-8}(55-59)=58.2$$

Entonces, el tiempo en la fase de programación corresponde al 58.2% del tiempo total, es decir:

$$Tprog=(0.582)(9.7 \text{ meses})=5.6 \text{ meses}$$

El valor medio de personal, equivalente en la fase de programación, será de:

$$P_E=22.5\text{HM}/5.6 \text{ meses}=4.01 \text{ hombres}$$

2.3. **Modelo COCOMO intermedio**

En este modelo se calcula el esfuerzo de desarrollo de software en función del tamaño del programa y de un conjunto de atributos conductores del costo, que incluyen la evaluación subjetiva del producto, del hardware, del personal y de los atributos del proyecto.

El COCOMO intermedio es una versión ampliada del COCOMO básico, pues presenta un mayor nivel de detalle y de seguridad, pero conservando la misma sencillez.

Los atributos conductores de costos son 15 factores, compuestos por cuatro categorías y que no se tienen en cuenta en el modelo básico. Dichos atributos son:

- *Atributos del producto:*
  - Confiabilidad requerida del software.
  - Tamaño de la base de datos.
  - Complejidad del producto.
- *Atributos del hardware:*
  - Restricciones de tiempo de ejecución.
  - Restricciones de memoria.
  - Volatilidad de máquina virtual
  - Tiempo de respuesta del equipo
- *Atributos del personal:*
  - Capacidad de análisis
  - Experiencia en aplicaciones
  - Capacidad de los programadores
  - Experiencia en el sistema operativo utilizado
  - Experiencia con el lenguaje de programación
- *Atributos del proyecto:*
  - Aplicación de métodos de ingeniería del software
  - Utilización de herramientas de software
  - Restricción del tiempo de desarrollo.

Cada uno de estos atributos tiene asociado un factor multiplicativo, el cual estima el efecto del atributo en el esfuerzo del desarrollo del software; este factor se denomina **multiplicador de esfuerzo.** Estos multiplicadores se aplican a un estimativo-base del esfuerzo, para obtener un estimativo refinado (ajustado) del esfuerzo del desarrollo. Este modelo inicia la estimación, con la generación de un estimativo nominal o básico del esfuerzo, usando funciones escalares similares a las del modelo básico. Este estimativo nominal es, entonces, ajustado, aplicando los multiplicadores de esfuerzo, derivados de la clasificación del proyecto respecto de los 15 atributos del costo.

La siguiente tabla presenta las ecuaciones del esfuerzo nominal, para los tres modos de desarrollo, utilizados en el COCOMO, intermedio.

Modo de desarrollo	Ecuación nominal de esfuerzo
Orgánico	$E_{nom}=3.2(KLDC)^{1.05}$
Semilibre	$E_{nom}=3.0(KLDC)^{1.12}$
Restringido	$E_{nom}=2.8(KSDI)^{1.2}$

El modelo intermedio presenta una tabla de multiplicadores de esfuerzo, donde cada atributo conductor de costo tiene asociado un conjunto de multiplicadores. Estos, a su vez, están relacionados con un conjunto de clasificaciones de proyectos.

Atributos directores del costo	Muy bajo	Bajo	Nominal	Alto	Muy alto	Extra alto
Restricciones fiabilidad del software	Efecto: Inconvenientes de escasa consideración	Bajo, pérdidas fácilmente recuperables	Moderado, pérdidas recuperables	Pérdidas financieras altas	Riesgo de vidas humanas	
Tamaño de la base de datos		DB Bytes/prog D/S<10	10<=D/P<100	100<=D/P<1.000	D/P>=1.000	
Restricciones de tiempo de ejecución			<=50% de ejecución disponible	70%	85%	95%
Restricciones de memoria principal			<=50% de la memoria existente	70%	85%	95%
Volatilidad máquina virtual		Cambios significativos: cada 12 meses poco significativos: 1 mes	Significativos: 6 meses Poco signif.: 2 semanas	Significativos: 2 meses Poco signif.: 1 semana	Significativos: 2 semanas Poco signif.: 2 días	
Tiempo de respuesta		Interactivo	Tiempo medio de retorno<4Horas	4-12 horas	>12 horas	
Capacidad de los analistas	15%	35%	55%	75%	90%	
Experiencia en la aplicación	<=meses de exper.	1 año	3 años	6 años	12 años	
Capacidad de los programadores	15%	35%	55%	75%	90%	
Experiencia en el S.O. utilizado	<=1 mes	4 meses	1 año	3 años		
Empleo de técnicas utilizadas en programación	No se usan	Comienzan a emplearse	Alguna utilización	Uso general	Uso rutinario	
Utilización de herramientas software	Herramientas básicas de micros	Herramientas básicas de minis	Grandes computadores	Herramientas avanzadas de Programación y verificación	Incluyendo herramientas de diseño, gestión y documentación	
Restricciones de tiempo de desarrollo	75% del nominal	85%	100%	130%	160%	

Atributos	VALOR					
	Muy bajo	Bajo	Nominal	Alto	Muy alto	Extra alto
Atributos del producto						
Confiabilidad del software requerida	0.75	0.88	1.0	1.15	1.40	–
Tamaño de la base de datos	–	0.94	1.0	1.08	1.16	–
Complejidad del producto	0.70	0.85	1.0	1.15	1.30	1.65
Atributos del hardware						
Restricciones de tiempo de ejecución	–	–	1.0	1.11	1.30	1.66
Restricciones de memoria	–	–	1.0	1.06	1.21	1.56
Volatilidad de la máquina virtual	–	0.87	1.0	1.15	1.30	–
Tiempo de respuesta del equipo	–	0.87	1.0	1.07	1.15	–
Atributos del personal						
Capacidad de análisis	1.46	1.19	1.0	0.86	0.71	–
Experiencia en aplicaciones	1.29	1.13	1.0	0.91	0.82	–
Capacidad de programadores	1.42	1.17	1.0	0.86	0.70	–
Experiencia en el S.O. utilizado	1.21	1.10	1.0	0.90	–	–
Experiencia con el lenguaje	1.14	1.07	1.0	0.95	—	—
Atributos del proyecto						
Aplicación de la ingeniería del software	1.24	1.10	1.0	0.91	0.82	–
Utilización de herramientas del software	1.24	1.10	1.0	0.91	0.83	–
Restricción del tiempo de desarrollo	1.23	1.08	1.0	1.04	1.10	–

Como ayuda para seleccionar el factor aplicable en cada caso, la siguiente tabla establece los criterios que sirven para elegir el factor adecuado.

Por ejemplo, si consideramos el desarrollo de una aplicación del modo semilibre, con 32 KLDC, que presenta diferentes requerimientos de confiabilidad debido a la naturaleza de su ambiente de trabajo, el esfuerzo nominal se calculará así:

$E_{nom}=3.0(32)^{1.12}=146$ hombres–mes.

Indica un esfuerzo total de 146 hombres-mes que son requeridos para desarrollar un producto de 32 KLDC, en modo semilibre, independiente de cualquier consideración de los requerimientos de confiabilidad.

A partir de aquí, los ajustes en el estimativo final deben hacerse de acuerdo con el tipo de producto final. Por ejemplo, si el sistema por desarrollar es un modelo de predicción del clima, podría tener un relativo nivel bajo de calidad, puesto que su impacto operacional es generalmente a largo plazo y muchas fallas podrían ocasionar pérdidas para los usuarios, fácilmente recuperables. Estas características le dan un peso, en la tabla mencionada, de 0.88 (bajo) para un ajuste de:

$E=(E_{nom})$(Factor multiplicativo)
E=(146 HM)(0.88)-128 hombres-mes

Pero si el sistema por desarrollar es, por ejemplo, un sistema de control de reactor nuclear, podría tener un muy alto requerimiento de confiabilidad. Aquí el efecto de fallas del software podría provocar la pérdida de vidas humanas; por lo tanto, tiene un factor multiplicativo mayor que el anterior. La estimación del esfuerzo será, de acuerdo con el nivel de confiabilidad 1.40 (muy alto), la siguiente:

$E=(E_{nom})$(Factor multiplicativo)
E=(146 HM)(1.40)=204 hombres-mes

Por lo tanto, para ajustar la estimación del esfuerzo usamos la siguiente ecuación, empleando los factores multiplicativos:

$$E=(KLDC)^{S}\ \Pi^{15}_{j=1} F_J$$

Donde S es el exponente correspondiente al modo de desarrollo y F el factor multiplicativo.

Veamos ahora un ejemplo sobre cómo aplicar los factores multiplicativos en el modelo de COCOMO intermedio. Supongamos que estamos negociando con la compañía Megabit Comunicaciones el precio de desarrollar un producto de software para el procesamiento de funciones de comunicación, en un microprocesador comercial, cuyo tamaño se estima en 10 KLDC y se desarrolla en modo restringido. De acuerdo con la ecuación nominal del esfuerzo para este modo, obtenemos el siguiente esfuerzo nominal:

$$2.8(10)^{1.20}=44 \text{ HM}$$

Deseamos ahora determinar el efecto de varias características del proyecto, en el esfuerzo y costo de desarrollo. Por ejemplo, el software para el procesamiento de comunicaciones generalmente tiene una valoración muy alta, en la escala de complejidad, pero se planea usar personal analista y programador con alta capacidad, lo cual balancea la tendencia a incrementar los costos, debido a la complejidad. El costo del personal será de $6.000 por mes.

En la siguiente tabla tenemos las características presentes para el desarrollo del producto:

Atributo	Situación	Clasificación	Multiplicador de esfuerzo
Confiabilidad del software requerida	Uso local del sistema No hay problemas serios de recuperación	Nominal	1.00
Tamaño de la base de datos	20.000 bytes	Baja	0.94
Complejidad del producto	Procesamiento de comunicaciones	Muy alta	1.30
Restricciones de tiempo de ejecución	Se usará el 70% del tiempo disponible	Alta	1.11
Restricciones de memoria	45K de 64K disponible	Alta	1.06
Volatilidad de la máquina virtual	Basado en microprocesador comercial	Nominal	1.00
Tiempo de respuesta del equipo	2 horas promedio	Nominal	1.00
Capacidad de análisis	Analistas con experiencia	Alto	0.86
Experiencia en aplicaciones	3 años	Nominal	1.00

Atributo	Situación	Clasificación	Multiplicador de esfuerzo
Capacidad de programadores	Programadores con experiencia	Alta	0.86
Experiencia en el S.O. utilizado	6 meses	Baja	1.10
Experiencia con el lenguaje	12 meses	Nominal	1.00
Aplicación de la ingeniería del software	Muchas técnicas por más de un año	Alta	0.91
Utilización de herramientas del sotfware	Herramientas	Baja	1.10
Restricción del tiempo de desarrollo	9 meses	Nominal	1.00
Factor de esfuerzo ajustado (producto de multiplicadores de esfuerzo)			1.17

Nótese que el factor de ajuste se incrementa en 1.30, debido a la complejidad muy alta, pero más adelante se reduce el valor a 0.86, debido a la alta capacidad de analistas y programadores. El factor de ajuste final, de 1.17, representa un incremento del 17% en el esfuerzo nominal; entonces, el costo y el esfuerzo estimados finalmente son:

Esfuerzo: (44 HM)(1.17)=51 HM

Costo: (51 HM)($6.000/HM)=$306.000

2.3.1. *Análisis de sensibilidad*

El modelo COCOMO intermedio permite realizar un análisis de sensibilidad con respecto a los atributos directores del costo, logrando así estimar el efecto sobre el costo de desarrollo, debido a los cambios en los niveles de clasificación de los atributos.

Por ejemplo, supongamos que tenemos la opción de realizar el proyecto del ejercicio anterior con personal de menor capacidad y menos costoso. Para este caso, el costo por persona-mes podría ser de $5.000 en lugar de los $6.000 anteriores, y la capacidad de análisis y programación la podríamos clasificar como Nominal. De acuerdo con la tabla anterior, este nivel de capacidad del personal resulta en un factor multiplicador de 1.00, en lugar del 0.86 obtenido anteriormente. Esto significa que el estimativo de COCOMO debe ser ajustado de acuerdo con las consideraciones anteriores:

Nuevo factor de ajuste de esfuerzo =1.58

Esfuerzo=(44 HM)(1.58)=70 HM

Costo: (70 HM)($5.000/HM)=$350.000

Según lo anterior podemos notar que resulta más ventajoso usar personal de mayor capacidad, según el análisis realizado con el factor de ajuste del esfuerzo, porque el estimativo de costo es menor.

Veamos otro ejemplo para el factor de ajuste de restricción del almacenamiento. Supongamos que por $10.000, Megabit podría comprar 96K palabras de memoria para el microprocesador por usar, en lugar de usar 64K. Esto podría cambiar las restricciones del almacenamiento principal del 70% al 47%. El resultado en el multiplicador del esfuerzo sería de 1.00, que corresponde a una clasificación Nominal, en lugar de 1.06 (alta), que era el anterior caso. Con lo anterior se tiene que:

Factor de ajuste del esfuerzo=1.10

Esfuerzo=(44 HM)(1.10)=48HM

Costo=(48 MM)($6.000/HM)=$288.000

Lo anterior nos indica que se logra un ahorro de $18.000, lo cual compensa la inversión de $10.000 en costos de hardware. Además, al negociar con la compañía Megabit, se puede proponer esta opción como forma para reducir el costo del software.

2.3.2. *Estimación de los efectos de software reutilizable*

Hasta ahora todas las estimaciones se habían basado en el supuesto que la totalidad del producto estaba siendo especificado, diseñado y desarrollado por completo, sin considerar que alguno de sus componentes hubieran sido desarrollados para otros proyectos previamente y que pudiesen utilizarse directamente o por adaptación en el nuevo producto.

Obviamente, las líneas-fuente de un programa adaptado no pueden tener la misma consideración que la del nuevo software, para efectos de la estimación. Por ejemplo, supongamos que utilizamos, como parte de nuestro nuevo producto, una rutina simple de entrada/salida, que ya forma parte de la librería de utilidades de otros proyectos, con lo cual se incrementa el número de líneas-fuente de nuestro nuevo producto, pero a la vez no incrementa la cantidad de esfuerzo requerido para el desarrollo del nuevo producto.

Por otra parte, en muchas ocasiones no se necesita adaptar completamente un paquete, sino que sólo se precisa realizar un esfuerzo adicional, para rediseñar el software adaptado, para ajustarlo a los objetivos del nuevo producto, el cual consiste en rehacer algunas porciones para acomodarlas a los cambios del entorno del nuevo producto.

Los efectos de la adaptación de productos existentes son tratados por el Modelo COCOMO, a través del cálculo del número equivalente de instrucciones fuente-IE, que se emplean en sustitución de las líneas de código-fuente LDC.

El número de líneas IE de un producto adaptado se calcula a partir de la estimación de las siguientes cantidades:

- Número de líneas-fuente adaptadas al nuevo producto, IA.
- Porcentaje de rediseño del software MD. Es el porcentaje de diseño del software que ha sido modificado para adaptarlo al nuevo entorno. Necesariamente ésta es una estimación subjetiva.
- Porcentaje de código modificado. Es aquél que es modificado con el fin de adaptarlo al nuevo entorno.
- Porcentaje del esfuerzo de integración del software adaptado MI. Es el porcentaje de esfuerzo requerido para integrar el software adaptado al nuevo producto, el cual se obtiene por comparación con el esfuerzo de integración necesario para un producto de tamaño similar.

El total de líneas equivalentes se calcula a través de un factor de adaptación definido como FA:

FA=0.40 MD+0.30MC+0.30MI

A partir del cual el número de líneas equivalentes vendría dado por:

IE=IA*FA Líneas equivalentes.

Este valor puede añadirse a las estimaciones de nuestro producto y tratarlas como hemos estado haciendo hasta ahora, de forma que el tamaño final del producto será:

LDC_{total}=LDC+IE.

Como ejemplo, supongamos que deseamos convertir un programa de análisis de circuitos, escrito en FORTRAN, de 50.000 líneas, desarrollado en modo orgánico desde un entorno hardware-software determinado en otro diferente.

Para esta situación podríamos considerar que:

MD=0 (es decir, no hay cambios en el diseño del programa).

MC=15 (posiblemente el 15% de las líneas de código deban cambiarse para adaptarlo a las características propias del nuevo sistema operativo, del lenguaje de control, etc.).

MI=5 (se precisa una pequeña cantidad de esfuerzo para integrar estos cambios).

Los resultados de la adaptación serían:

FA=0.40(0)+0.30(15)+0.30(5)=6

y por lo tanto, el número de líneas equivalentes sería:

IE=50.000 LDC*0.06=3.000 líneas equivalentes.

Utilizando las estimaciones del modelo básico, se obtiene un esfuerzo de adaptación de:

$E=2.4(3)^{1.05}$=7.6 HM

Los coeficientes de la ecuación del factor de adaptación se determinan a partir del valor medio de los porcentajes dedicados a las tareas de diseño, codificación e integración y pruebas dadas por el modelo:

Diseño:	40%
Codificación:	30%
Integración y prueba:	30%

Para estimar el tiempo de desarrollo se emplean las mismas ecuaciones utilizadas en el Modelo COCOMO básico.

En cuanto a la distribución del esfuerzo por fase, el modelo intermedio utiliza las mismas tablas empleadas en el modelo básico.

Los multiplicadores de esfuerzo, en este modelo, pueden aplicarse a la fase de mantenimiento, del mismo modo que se realizan en la fase de desarrollo.

2.4. **Modelo COCOMO detallado**

Este modelo incorpora todas las características del modelo intermedio y lleva a cabo una evaluación del impacto de los atributos conductores del costo, en cada fase del proceso de la ingeniería del software.

El modelo intermedio es altamente efectivo para muchos propósitos de estimación del software. Sin embargo, tiene dos limitantes principales que pueden ser significativas en estimaciones de costo más detalladas para proyectos extensos:

- La estimación distribuida de esfuerzo, por fases, puede ser inexacta.
- Puede ser muy engorroso para usarlo en productos con muchos componentes.

El modelo detallado provee dos principales facultades que consignan las limitantes del modelo intermedio.

- Multiplicadores de esfuerzo de fase sensitiva: en la práctica, factores tales como confiabilidad, experiencia y desarrollo interactivo afectan a algunas fases mucho más que otras. El modelo detallado ofrece un con-

junto de multiplicadores de esfuerzo, de fase sensitiva, para cada atributo del costo. Estos son usados para determinar la cantidad de esfuerzo requerido para completar cada fase.

Los multiplicadores se usan de acuerdo con la precisión que refleje el efecto de los conductores de costo, en la distribución de fase del esfuerzo. Por ejemplo, un nivel bajo en experiencia de aplicaciones puede significar esfuerzo adicional en las primeras fases. En las últimas fases el equipo podrá familiarizarse con la aplicación y, así, no se requerirá mucho esfuerzo adicional.

De otra parte, un tiempo de respuesta alto en el computador tendrá muy poco efecto en las fases de diseño y requerimientos, pero ciertamente consumirá más tiempo del equipo durante la codificación y las pruebas.

- Jerarquía de producto de tres niveles: en el modelo intermedio la clasificación separada de los conductores de costo puede ser suministrada por diferentes componentes del producto. Este proceso puede ser tedioso e innecesariamente repetitivo si un número de componentes se agrupa en un subsistema, con prácticamente todas las mismas clasificaciones. El modelo detallado excluye este problema, ofreciendo la jerarquía de producto de tres niveles, para lo cual:
- Algunos efectos, que tienden a variar con cada módulo de nivel interno, son tratados en el nivel de módulo.
- Algunos efectos, que varían menos frecuentemente, son tratados en el nivel de subsistema.
- Algunos efectos, tales como el efecto de tamaño total del producto, son tratados en el nivel del sistema.

La jerarquía módulo-subsistema-sistema se da por la descomposición jerárquica del producto, del cual se va a estimar el costo.

El nivel más bajo, el nivel de módulo, se describe por el número de líneas de código-fuente en el módulo, y por aquellos atributos de costo que tienden a variar al más mínimo nivel. Por ejemplo, la complejidad del módulo y su adaptación al software existente; la capacidad de los programadores y la experiencia en el manejo del lenguaje en el que se desarrolla el software.

El segundo nivel, nivel de subsistema, se caracteriza por el resto de los conductores de costo (tiempo, condiciones de almacenamiento, capacidad de analistas, herramientas, etc.), los cuales tienden a variar de un subsistema a otro, por lo cual tiende a ser lo mismo para todos los módulos del subsistema.

El nivel más alto, el nivel del sistema, se usa para aplicar las principales relaciones del proyecto, tales como las ecuaciones de esfuerzo y tiempo, y para aplicar, durante las interrupciones, en el esfuerzo y los tiempos del proyecto.

3. METODOS DE LOS PUNTOS DE FUNCION

La métrica de los puntos de función permite cuantificar una aplicación, basándose en dos áreas de evaluación: La primera consiste en el cálculo de puntos de función simples. La segunda área consiste en ajustar el modelo, en función de la complejidad y del ambiente de desarrollo de la aplicación.

La finalidad de esta metodología es, en primer lugar, estimar el tamaño de un producto de software en las etapas previas de su desarrollo y, por otra parte, estimar el esfuerzo de desarrollo, expresado en este caso en horas trabajadas, por el punto de función desarrollado. Un punto de función es una métrica que describe una unidad del producto de trabajo, tales como el esfuerzo de desarrollo y/o mantenimiento del software. Una componente o función es un proceso único o requerimiento de datos de la aplicación, por ejemplo, una pantalla donde se adicionan datos o un reporte impreso.

En este método se evalúa la funcionabilidad de una aplicación en términos de **qué** se le entrega al usuario en la aplicación y no cómo se le entrega. Unicamente los componentes visibles y los requerimientos del usuario son contabilizados. Estos componentes son tratados como Tipos de Función, los cuales están divididos en Datos y Transacciones:

Tipos de Función Datos: Archivos Lógicos Internos (ALI) y Archivos Externos de Interface (AEI).

Tipos de Función Transacciones: Funciones de Entrada (Entradas Externas), Funciones de Salida (Salidas Externas), Consultas Externas (CE).

Los objetivos principales de esta metodología son:

Medir lo que el usuario requiere y lo que se le entrega.

Medir la aplicación, independientemente de la tecnología usada para la implementación.

Proveer una métrica de tamaño que soporte análisis de calidad y productividad.

Proveer un vehículo para la estimación del software.

Este último aspecto es el que se trata en este documento.

3.1. Tipos de función

Los puntos de función sin ajuste tratan exclusivamente del conteo de las funciones del usuario y se realiza tras una previa clasificación de las funciones, basándose en aquellos componentes de una aplicación que son requeridos y son visibles al usuario.

3.1.1. ***Archivo Lógico Interno:*** Es un conjunto de datos del usuario que están lógicamente relacionados y que son mantenidos y utilizados, por la aplicación, dentro de sus propios límites. Mantener los datos se refiere a la capacidad de adicionar, modificar o borrar datos a través de procesos estandarizados de la aplicación. También entra en este grupo la información de control, es decir, los datos usados por la aplicación para asegurar que se cumplan los requerimientos de funcionamiento especificados por el usuario.

Para identificar estos archivos se identifican todos los datos o grupos de datos que:

Son almacenados dentro del alcance de la aplicación.

Son mantenidos a través de procesos estandarizados de la aplicación.

Son definidos como requerimientos de la aplicación por el usuario.

También se identifican estos archivos agrupando los datos lógicamente, desde el punto de vista del usuario:

Agrupar los datos hasta un nivel de detalle, en el cual el usuario pueda identificar los datos como satisfacción a sus requerimientos.

Tratar los datos lógicamente; no deben tomarse en cuenta los archivos físicos.

Los siguientes son los tipos de datos que pueden relacionarse con uno o más Archivos Lógicos Internos, dependiendo de la visión de usuario.

Datos de la aplicación (archivos maestros, tales como información de personal o información consolidada).

Datos de seguridad de la aplicación.

Datos de auditoría.

Mensajes de ayuda.

Mensajes de error.

Ejemplos de Archivos Lógicos Internos son:

Datos de respaldo (Backup), los cuales se cuentan únicamente si son específicamente requeridos por el usuario, ya sea por requerimientos legales o algo similar.

Archivos Lógicos Internos que sean utilizados y mantenidos por más de una aplicación, es decir, archivos compartidos.

Los siguientes archivos no se consideran Archivos Lógicos Internos:

Archivos temporales.

Archivos de trabajo.

Archivos de ordenamiento.

3.1.2. ***Archivos Externos de Interface.*** Son grupos de datos lógicamente relacionados o información de control utilizados por la aplicación, pero que no son mantenidos por ésta sino por otras aplicaciones. En este tipo se cuentan archivos transferidos o distribuidos entre aplicaciones.

Para identificar estos archivos es necesario:

- Identificar todos los datos que estén almacenados fuera del alcance de la aplicación.
- Que no sean mantenidos por la aplicación propia.
- Que estén dentro de los requerimientos del usuario.

También se identifican estos archivos agrupando los datos lógicamente, desde el punto de vista del usuario, para lo cual se debe:

- Agrupar los datos hasta un nivel de detalle, en el cual el usuario pueda identificar los datos como satisfacción a sus requerimientos.
- Tratar los datos lógicamente; no deben tomarse en cuenta los archivos físicos.

Los siguientes son tipos de datos que pueden relacionarse con uno o más Archivos Lógicos Internos, dependiendo de la visión del usuario:

- Datos de referencia (datos externos utilizados por la aplicación, pero que no hacen parte de sus Archivos Lógicos Internos).
- Mensajes de ayuda.
- Mensajes de error.

Los siguientes no se consideran como Archivos Externos de Interface:

- Datos que son mantenidos por la aplicación, pero a los que se tiene acceso y son utilizados por otra aplicación.
- Datos formateados y procesados, para ser utilizados por otra aplicación.

Los Archivos Externos de Interface no se atribuyen a la aplicación fuente, es decir, a la aplicación donde se originan estos archivos. Esto significa que se atribuyen estos archivos a las aplicaciones, básandose en la dirección del flujo y el uso de los datos, por la aplicación.

El tipo de archivo se determina básándose en cómo los utiliza la aplicación que recibe los datos. Si los datos son usados para actualizar los Archivos Lógicos Internos, se considera como una Entrada Externa o una Salida Externa, dependiendo del flujo. Si los datos no se usan para el mantenimiento de un Archivo Lógico Interno, se consideran como un Archivo Externo de Interface, de acuerdo con el flujo de datos.

El cálculo de los puntos de función debe ser actualizado si, después del cálculo, el acceso a un Archivo Lógico Interno se le permite a otra aplicación.

No siempre se puede determinar cómo otra aplicación está utilizando los datos y varios métodos no han involucrado esta situación, lo cual ha dado

como resultado cálculos inconsistentes. Para resolver este problema, únicamente la aplicación que recibe los datos puede tener Archivos Externos de Interface, es decir, que el cálculo se asume desde el punto de vista de la aplicación que se analiza. Como resultado, el cálculo de los puntos de función depende únicamente del sistema, como existe actualmente, y no de futuros eventos o de la utilización de los datos.

3.1.3. ***Funciones de Entrada (Entradas Externas - E.E).*** Se hace referencia a las funciones de entrada de datos. Se consideran entonces los datos únicos de entrada o de control que deben suministrarse a la aplicación con el fin de añadirlos a un Archivo Lógico Interno o de modificarlo. Se deben contar las introducidas directamente por el usuario y las obtenidas como resultado de una transacción, desde otra aplicación.

Una Función de Entrada (Entrada Externa) se considera única si tiene diferente formato o si, por su diseño interno, requiere un proceso lógico, diferente de otra función de entrada, con el mismo formato.

El formato puede ser un conjunto de datos únicos o un arreglo único de datos o un orden único de datos.

Para identificar las Entradas Externas se debe:

- Identificar todos los procesos que actualizan los Archivos Lógicos Internos.
- Por cada proceso identificado, considerar cada formato como un proceso separado, si los datos usados por el proceso pueden ser recibidos en más de un formato; asignar una Entrada Externa por cada una de las actividades de mantenimiento desempeñadas; esto es, adicionar, modificar y borrar.

Las siguientes son Entradas Externas, tomando en cuenta las anteriores consideraciones:

- Datos de transacciones: Datos externos usados para mantener los Archivos Lógicos Internos.
- Pantallas de entrada: Se cuenta una Entrada Externa para cada una de las funciones de mantenimiento. Si adiciona, modifica y borra, la pantalla debe contarse como tres Entradas Externas.
- Por cada proceso único que mantiene un Archivo Lógico Interno, contar una Entrada Externa por cada adición, modificación y borrado.

Dos o más archivos físicos pueden corresponder a una sola Entrada Externa, si el procesamiento lógico y el formato para cada uno de estos archivos es idéntico.

- Entradas Externas duplicadas: Se hace referencia a aquellos procesos que presentan la misma Entrada Externa, pero por diferentes medios. Por ejemplo, en un sistema bancario que acepta idénticas transacciones de depósito, a través de un proceso automático (cajero automático) y a través de procesos manuales. Por lo tanto, ambos procesos se cuentan cada uno como Entrada Externa.

Las siguientes no se consideran como Entradas Externas:

- Datos suministrados a la aplicación, por razones de tecnología empleada.
- Datos externos utilizados por la aplicación, pero no mantenidos en Archivos Lógicos Internos.
- Datos de entrada, usados para seleccionar la recuperación de datos.
- Pantallas de menú que únicamente permiten viajar por las pantallas de la aplicación y no contribuyen di-

rectamente en el mantenimiento de los Archivos Lógicos.

- Pantallas que facilitan la entrada a una aplicación; por ejemplo, pantallas de logon.
- Múltiples formas de invocar la misma entrada. Por ejemplo, digitar **A** o **Add** como un comando o como una tecla de función, para seleccionar la opción de adicionar; sólo se cuenta una Entrada Externa.
- Cualquier tipo de consulta no se cuenta, puesto que pertenece a un tipo de función diferente.

3.1.4. ***Funciones de Salida (Salidas Externas - SE).*** Se hace referencia a las funciones de salida de datos. Se consideran entonces datos únicos, de usuario o de control, los suministrados por la aplicación. Se incluyen dentro de este tipo, salidas tales como informes, mensajes al usuario, mensajes a otras aplicaciones, a través de archivos.

Una Función de Salida (Salida Externa) se considera única si tiene diferente formato o si, por su diseño interno, requiere un proceso lógico diferente de otra función de salida, con el mismo formato.

Para identificar las Entradas Externas es preciso:

- Identificar todos los procesos que suministran datos, fuera de los límites de la aplicación.
- Por cada proceso identificado, considerar cada formato como un proceso separado si los datos usados por el proceso pueden ser suministrados en más de un formato; asignar una Salida Externa por cada proceso identificado.

Las siguientes son Salidas Externas, tomando en cuenta las anteriores consideraciones:

- Datos transferidos hacia otras aplicaciones.
- Reportes. Cada reporte producido por la aplicación se cuenta como Salida Externa. Dos reportes idénticamente formateados se cuentan cada uno como Salida Externa si tienen procesamiento lógico diferente.
- Salidas en línea de datos que no se originan por una Entrada Externa.
- Reportes idénticos, pero producidos por medios diferentes; por ejemplo, un reporte originado por una impresora y por tarjetas perforadas.
- Una Salida Externa debe asignarse por cada Entrada Externa que origina mensajes de error o confirmación.

No se consideran Salidas Externas los siguientes tipos de datos:

- Múltiples formas de invocar la misma salida. Por ejemplo, para seleccionar un reporte, si se necesita digitar **R** o **Report** o una tecla de función, sólo se cuenta una Salida Externa.
- Mensajes de confirmación o error, originados por una consulta.

3.1.5. ***Consultas Externas - CE.*** Estas constituyen combinaciones únicas de Entrada/Salida, donde una entrada causa una salida inmediata. La consulta se considera única si tiene un formato diferente de otra función del mismo tipo (para la entrada y la salida), o si su diseño externo requiere un proceso lógico diferente.

Para identificar las Salidas Externas se debe tener en cuenta:

- Identificar todos los procesos en los que una entrada dispara una recuperación inmediata de datos.
- Por cada proceso identificado, verificar que cada combinación de Entrada/Salida es única y considerar cada combinación como un proceso diferente y acreditar una Consulta Externa por cada proceso.

Los siguientes son ejemplos de Consultas Externas:

- Pantallas de Modificación/Borrado, en las que se provoca la recuperación de datos, antes de ejecutar la función de Modificación/Borrado.
- Si las entradas y salidas de la consulta son idénticas en ambas funciones de Modificación y Borrado, se cuenta únicamente una Consulta Externa. Si se dispone de idénticas funciones de consulta en pantallas de Modificación/Borrado y en una pantalla aparte de consulta, se cuenta sólo una Consulta Externa.
- Pantallas de logon que proveen funciones de seguridad, se cuentan como Consultas Externas.

Tres categorías de ayudas son consideradas como Consultas Externas:

- Pantallas totales de ayuda, que son aquéllas que muestran el texto de ayuda relacionado con la pantalla que le llamó.
- Ayuda sensitiva del campo, que es aquélla dependiente de la posición del cursor o de algún método de identificación, que muestra la documentación de ayuda para ese campo. Se cuenta una Consulta Externa por cada pantalla.
- Subsistema de ayuda, que es una facilidad de ayuda a la que se puede acceder y que puede ser recorrida, independiente de la aplicación asociada. Si el texto recuperado es idéntico al de una ayuda total, no se cuenta.

3.2. Cálculo de los Puntos de Función (sin ajuste)

El cálculo de los Puntos de Función consta de tres pasos:

- Obtener Puntos de Función sin ajuste.
- Obtener el Factor de Ajuste.
- Ajustar los Puntos de Función, usando el Factor de Ajuste para obtener los Puntos de Función reales.

Para cada tipo de función se determina la complejidad de forma diferente, así:

Para los Tipos de Función de Datos, la complejidad funcional se identifica de acuerdo con el número de Elementos de Tipo Registro (ETR) y con el número de Elementos de Tipo Datos (ETD).

Los ETR son formatos únicos de registros, dentro de un ALI o un AEI.

Los ETD son ocurrencias únicas de datos, también tratados como elementos de datos, variables o campos.

Por cada ALI y AEI identificado, se le asigna una complejidad funcional, basándose en los ETD y los ETR.

Cómo identificar ETR's: Son subgrupos de los ALI's, vistos desde la perspectiva lógica que el usuario tenga de los datos, es decir, de acuerdo con sus requerimientos. ALI's que no puedan ser categorizados, se considera que tienen un solo ETR.

Cómo identificar ETD's: Son campos reconocibles por el usuario y no recursivos, que residen en un ALI.

Cada campo en un ALI debe identificarse como un ETD, tomando en cuenta las siguientes consideraciones:

- Campos que deberían ser vistos desde un nivel reconocible por el usuario. Por ejemplo, un número de cuenta o una fecha que es físicamente almacenada en múltiples campos se cuenta como un ETD.
- Campos que aparecen más de una vez en un ALI, debido a la tecnología o a las técnicas de implementación, deben contarse como un ETD, sólo una vez. Por ejemplo, si un ALI se compone de más de una tabla en una base de datos relacional, la clave usada para relacionar las tablas se cuenta una sola vez.

- Campos repetitivos que son idénticos en formato y existen para permitir múltiples ocurrencias de un valor de un dato, se cuentan una sola vez. Por ejemplo, un ALI que contiene 12 campos de presupuestos mensuales y un campo con un presupuesto anual (la suma de los mensuales) deberían contarse como dos DTE, uno para el campo mensual y otro para el campo anual.

Por cada ALI y AEI se identifican sus ETR y ETD; luego se le asigna un grado de complejidad, de acuerdo con la siguiente tabla:

		(1 a 19) ETD	(20 a 50) ETD	(51 o más) ETD
(1)	ETR	Baja	Baja	Media
(2 a 5)	ETR	Baja	Media	Alta
(6 o más)	ETR	Media	Alta	Alta

El peso para cada grado de complejidad es el siguiente:

	ALI	AEI
Baja	7	5
Media	10	7
Alta	15	10

Para calcular el total de Puntos de Función, para los ALI y AEI, se usa una tabla como la siguiente:

Tipo de función	Complejidad funcional			Complejidad total
ALI	—— Baja	x	7 =	——
	—— Media	x	10 =	——
	—— Alta	x	15 =	——

Por ejemplo, supongamos que se tienen tres ALI con complejidad baja, tres con complejidad media y tres con complejidad alta:

Tipo de función	Complejidad funcional			Complejidad total
ALI	3 Baja	x	7 =	21
	3 Media	x	10 =	30
	3 Alta	x	15 =	45
Total de puntos de función				96

Sumando la columna de complejidad total, tenemos un total de puntos de función sin ajuste de 96.

De igual forma, tomemos el caso de tres AEI con complejidad baja, tres con complejidad media y tres con complejidad alta:

Tipo de función	Complejidad funcional			Complejidad total
AEI	3 Baja	x	5 =	15
	3 Media	x	7 =	21
	3 Alta	x	10 =	30
Total de puntos de función				66

Sumando la columna de complejidad total, obtenemos un total de puntos de función sin ajuste de 66.

El siguiente paso consiste en calcular la complejidad funcional de las funciones de entrada EE (Entradas Externas), y SE (Salidas Externas) basándose en el número de Tipos de Archivo Referenciados TAR y el número de ETD.

Un Tipo de Archivo Referenciado se cuenta por cada ALI y por cada AEI referenciado, durante el procesamiento de una Entrada Externa, EE, o una SE, según el caso.

Para las EE, los ETD son tratados de igual forma que en los casos anteriores y, adicionalmente, un ETD se cuenta, para una EE, con las siguientes consideraciones:

- Líneas de comando o teclas de función que proveen la capacidad para especificar la acción que se tomará por la EE. Un ETD adicional por EE, no por tecla de comando.
- Campos que no son suministrados por el usuario, pero a través de EE son mantenidos en un ALI, deberían ser contados. Por ejemplo, un sistema de clave secuencial, mantenido en un ALI, pero no suministrado por el usuario, debería contarse como un ETD.

Para las SE se tratan inicialmente de igual forma los EDT, con las siguientes excepciones: No se incluyen literales como ETD.

No se cuentan las variables de páginas ó los sistemas generados por rótulos por el sistema.

La complejidad funcional, para las funciones de entrada EE, se basa en la tabla siguiente:

		(1 a 4) ETD	(5 a 15) ETD	(16 o más) ETD
(0 a 1)	FTR	Baja	Baja	Media
(2)	FTR	Baja	Media	Alta
(3 o más	FTR	Media	Alta	Alta

La complejidad funcional para las funciones de salida, SE, se basa en la tabla siguiente:

		(1 a 5) ETD	(6 a 19) ETD	(20 a más) ETD
(0 a 1)	FTR	Baja	Baja	Media
(2 a 3)	FTR	Baja	Media	Alta
(4 o más)	FTR	Media	Alta	Alta

El peso para cada grado de complejidad es el siguiente:

	EE	SE
Baja	3	4
Media	4	5
Alta	6	7

Para las funciones de consulta, Consultas Externas, se siguen los siguientes pasos para determinar el valor de los puntos de función sin ajuste:

Calcular la complejidad funcional para la parte de entrada de la Consulta Externa.

Calcular la complejidad funcional para la parte de salida de la Consulta Externa.

Seleccionar el valor más alto de las dos complejidades funcionales. Usando una tabla adecuada para los Puntos de Función sin Ajuste, se transcribe la clasificación de la complejidad a Puntos de Función sin Ajuste.

Se cuenta un Tipo de Archivo Referenciado por cada ALI y AEI, referenciados durante el proceso de la consulta.

Un ETD se cuenta por aquellos campos suministrados que especifican la Consulta por ejecutar o que especifican los criterios de selección de los datos.

Un ETD se cuenta por cada campo no recursivo que aparece en la parte de salida de la consulta.

La complejidad funcional para las Consultas Externas, CE, se basa en las tablas siguientes:

Complejidad de entrada				
		(1 a 4) ETD	(5 a 15) ETD	(16 o más) ETD
(0 a 1)	FTR	Baja	Baja	Media
(2)	FTR	Baja	Media	Alta
(3 o más)	FTR	Media	Alta	Alta

Complejidad de salida				
		(1 a 5) ETD	(6 a 19) ETD	(20 o más) ETD
(0 a 1)	FTR	Baja	Baja	Media
(2 a 3)	FTR	Baja	Media	Alta
(4 o más)	FTR	Media	Alta	Alta

El peso para cada grado de complejidad es el siguiente:

	CE
Baja	3
Baja	4
Alta	6

Una vez obtenidos los puntos de función sin ajuste para cada tipo de función, podremos obtener el total de puntos de función sin ajuste, mediante la siguiente tabla:

Tipo de función	Complejidad funcional	Complejidad total	Total de puntos de función
ALI	3 Baja x 7 =	21	
	3 Media x 10 =	30	
	3 Alta x 15 =	45	
			96
AEI	3 Baja x 5 =	15	
	3 Media x 7 =	21	
	3 Alta x 10 =	30	
			66
EE	3 Baja x 3 =	9	
	3 Media x 4 =	12	
	3 Alta x 6 =	18	
			39
SE	3 Baja x 4 =	12	
	3 Media x 5 =	15	
	3 Alta x 7 =	21	
		48	
CE	3 Baja x 3 =	9	
	3 Media x 4 =	12	
	3 Alta x 6 =	18	
		39	
Total de puntos de función sin ajuste			288

3.3. Características generales del sistema

La funcionalidad del modelo puede variar dependiendo del entorno de desarrollo. Por esta razón, el modelo introduce la valoración de 14 características que influyen sobre la complejidad del proceso.

Estas características se evalúan de conformidad con una escala de grado de influencia que toma valores enteros entre 0 y 5, para ajustar la complejidad del proceso.

Las características generales del sistema son:

Transmisión de datos.
Proceso distribuido.
Rendimiento, respuesta.
Configuración.
Indice de transacciones.
Entrada de los datos en línea.
Eficiencia del usuario.
Actualización en línea.
Complejidad del proceso.
Reutilización.
Facilidad de instalación.
Sencillez de operación.
Adaptabilidad.
Flexibilidad de cambio.

Los grados de influencia se clasifican así:

0: No incluye.
1: Influencia insignificante.
2: Influencia moderada.
3: Influencia media.
4: Influencia significativa.
5: Influencia fuerte.

3.3.1. *Transmisión de datos*

Los datos e información de control, usados en la aplicación, son enviados o recibidos por medio de facilidades de comunicación.

Puntaje:

0: La aplicación ocurre únicamente en el procesamiento por lotes o en un computador aislado.

1: La aplicación se hace por lotes, pero se tiene una entrada remota de datos o una salida remota hacia la impresora.

2: La aplicación se hace por lotes, pero se tiene una entrada remota de datos y una salida remota hacia la impresora remota.

3: Recolección de datos "on line" o por teleprocesamiento frontal a un proceso por lotes o un sistema de consulta.

4: Más de un proceso frontal, pero la aplicación soporta únicamente un protocolo de teleprocesamiento de comunicaciones.

5: Más de un proceso frontal, pero la aplicación soporta más de un protocolo de teleprocesamiento de comunicaciones.

3.3.2. ***Proceso distribuido***

Son características de la aplicación.

Puntaje:

0: La aplicación no se preocupa por la transferencia de los datos o el procesamiento entre los componentes del sistema.

1: La aplicación prepara datos para procesamiento del usuario final sobre otros componentes del sistema, tales como micros.

2: Los datos son preparados para ser transferidos y procesados en otros componentes del sistema (no son para el usuario final).

3: El procesamiento distribuido y la transferencia de datos están en línea y en una sola dirección.

4: El procesamiento distribuido está en línea y entre direcciones.

5: Las funciones de procesamiento son dinámicamente ejecutadas en más de un componente del sistema asociado.

3.3.3. ***Rendimiento, respuesta***

Los objetivos del desempeño de la aplicación son aprobados por el usuario, como respuesta o por la influencia del diseño, el desarrollo, la instalación o el soporte de la aplicación.

0: No hay requerimientos especiales de desempeño por parte del usuario.

1: Los requerimientos del desempeño y del diseño fueron establecidos y revisados sin ninguna acción requerida.

2: Tiempos de respuesta "on line" críticas durante las horas pico. Ningún diseño especial se ha requerido para el uso de la CPU.

3: Tiempos de respuesta críticos durante todas las horas de trabajo. Ningún diseño especial se ha requerido para el uso de la CPU.

4: Los requerimientos establecidos por el usuario, para el desempeño, son estrictos; lo suficiente para implementar análisis de desempeño en la fase de diseño.

5: Adicionalmente, las herramientas del análisis del desempeño han sido usadas en el diseño, desarrollo y/o implementación, para conocer los requerimientos del usuario.

3.3.4. ***Configuración***

Un uso pesado de la configuración operacional que requiere consideraciones especiales de diseño, es una característica de la aplicación (por ejemplo, el usuario quiere correr la aplicación sobre un equipo que será altamente usado).

Puntaje:

0: No están explícitas o implícitas las restricciones de la operación.

1: Existen restricciones en la operación, pero son menos restrictivas que en una aplicación típica. No se necesita esfuerzo especial para responder ante las restricciones.

2: Algunas consideraciones de seguridad o tiempo.

3: Los requerimientos específicos del procesador son necesarios para una parte de la aplicación.

4: Las limitaciones establecidas para la operación requieren restricciones especiales sobre la aplicación, en el procesador central o en un procesador dedicado.

5: Adicionalmente, hay restricciones especiales para la aplicación en los componentes distribuidos del sistema.

3.3.5. ***Indice de transacciones***

El índice o promedio de transacciones es alto y afecta el diseño, el desarrollo, la instalación y el soporte de la aplicación.

Puntaje:

0: No se adelantan los períodos críticos de la transacción.

1: Se anticipan los períodos críticos de transacción mensual.

2: Se anticipan los períodos críticos de transacción semanal.

3: Se anticipan los períodos críticos de transación diarla.

4: Altos promedios de transacción son planteados por el usuario en los requerimientos o acuerdos de la aplicación, y son lo suficientemente altos como para requerir tareas de análisis de desempeño en la fase de diseño.

5: Al igual que en el punto anterior, pero adicionalmente se requiere el uso de herramientas de análisis de desempeño en las fases de diseño, desarrollo y/o instalación.

3.3.6. ***Entrada de datos en línea***

La entrada de datos en línea (on-line) y las funciones de control son provistas en la aplicación.

Puntaje:

0: Todas las transacciones son procesadas en lotes.

1: Del 1% al 7% de las transacciones son interactivas.

2: Del 8% al 15% de las transacciones son interactivas.

3: Del 16% al 23% de las transacciones son interactivas.

4: Del 24% al 30% de las transacciones son interactivas.

5: Más del 30% de las transacciones son interactivas.

3.3.7 ***Eficiencia del usuario***

Las funciones en línea provistas enfatizan un diseño para la eficiencia del usuario final. Esto incluye:

- Menús.
- Documentación y ayudas en líneas.
- Movimiento del cursor automatizado.
- Desplazamiento de imágenes ("scrolling").
- Impresión remota.
- Teclas de función preasignadas.
- Selección de datos en pantalla por medio del cursor.
- Uso constante de video inverso, resaltamiento, delineado en color y otros indicadores.
- Copia permanente de la documentación del usuario sobre transacciones en línea.
- Interfases con el "mouse".
- Ventanas "Pop-Pup".
- Fácil navegación entre las pantallas.
- Soportar dos o más lenguajes.

Puntaje:

0: No se presenta nada de lo anterior.

1: De una a tres de las anteriores.

2: De cuatro a cinco de las anteriores.

3: Seis o más de las anteriores, pero no hay requerimientos específicos de los usuarios con respecto a la eficiencia.

4: Seis o más de las anteriores, pero se establecen requerimientos sobre la eficiencia de los usuarios, hasta un nivel en el que se requieren tareas de diseño para los factores humanos por incluirse.

5: Seis o más de los anteriores y se establecen requerimientos para la eficiencia del usuario, a un nivel en el que se requieren herramientas y procesos especiales para verificar que se cumplan los objetivos planteados.

3.3.8. ***Actualización en línea***

La aplicación provee actualización en línea para los archivos lógicos internos.

Puntaje:

0: Ninguna.

1: Actualización en línea de uno a tres archivos de control. La cantidad de actualización es baja y de fácil recuperación.

2: Actualización en línea de cuatro o más archivos de control. La cantidad de actualización es baja y de fácil recuperación.

3: Actualización en línea de los principales archivos lógicos internos.

4: Adicionalmente, la protección contra pérdida de datos es esencial y ha sido especialmente diseñada y programada en el sistema.

5: Adicionalmente, altas cantidades involucran consideraciones de costos en el proceso de recuperación.

Procedimientos altamente automatizados de recuperación, con un mínimo de intervención de operadores.

3.3.9. ***Complejidad del proceso***

La complejidad se categoriza así:

- El control sensible (por ejemplo, un procedimiento especial de auditoría) y/o un procesamiento específico de seguridad de una aplicación.
- El procesamiento lógico extenso.
- El procesamiento matemático extenso.
- Mucho procesamiento de excepción, que resulta en transacciones incompletas, las cuales deben ser procesadas de nuevo.
- Un procesamiento complejo para manipular múltiples posibilidades de entrada-salida; por ejemplo, multimedia, dispositivos independientes.

Puntaje:

0: Ninguno de los anteriores.

1: Alguno de los anteriores.

2: Dos cualquiera de los anteriores.

3: Tres cualquiera de los anteriores.

4: Cuatro cualquiera de los anteriores.

5: Cualquiera de los anteriores.

3.3.10. ***Reutilización***

Se refiere al grado de volver a utilizar la aplicación en otros proyectos.

Puntaje:

0: Sin código reutilizable.

1: El código reutilizable se usa dentro de la aplicación.

2: Menos del 10% de los módulos producidos consideran más de una necesidad del usuario.

3: El 10% o más de los módulos producidos consideran más de una de las necesidades de los usuarios.

4: La aplicación fue específicamente compactada y/o documentada, para fácil reutilización, y la aplicación está diseñada para los usuarios, a nivel del código-fuente.

5: La aplicación fue específicamente compactada y/o documentada, para fácil reutilización, y la aplicación está diseñada para usarse por medio de los parámetros de mantenimiento del usuario.

3.3.11. ***Facilidad de instalación***

Puntaje:

0: No se tuvieron consideraciones especiales, por parte del usuario, para la instalación.

1: No se tienen consideraciones especiales, por parte del usuario, pero sí se requiere configuración especial para la instalación.

2: Los requerimientos de instalación y conversión son requeridos por el usuario. El impacto de la conversión en el proyecto no es importante.

3: Los requerimientos de instalación y conversión son requeridos por el usuario. El impacto de la conversión, en el proyecto, se considera importante.

4: Adicionalmente al puntaje 2, las herramientas automatizadas de conversión e instalación fueron provistas y probadas.

5: Adicionalmente al puntaje 3, las herramientas automatizadas de conversión e instalación fueron provistas y probadas.

3.3.12. ***Sencillez de operación***

En esta característica se tiene en cuenta la efectividad del arranque, el respaldo y el procedimiento de recuperación, durante la fase de prueba. La aplicación minimiza la necesidad de actividades manuales, tales como montajes de cintas, manipulación del papel.

Puntaje:

0: No hay consideraciones especiales diferentes de los de procesos de "backup" normal.

- 1 - 4: Se seleccionan los siguientes ítems, de acuerdo con la aplicación. Cada ítem tiene un valor de un punto, si no se especifica lo contrario.
- Procesos efectivos de arranque, respaldo y recuperación, pero se requiere la intervención del operador.
- Igual que el ítem anterior, pero no se requiere la intervención del operador (2 ítems).
- La aplicación minimiza la necesidad del montaje de la cinta.
- La aplicación minimiza la necesidad de la manipulación del papel.

5: La aplicación no requiere la intervención directa del operador, salvo en los procesos de arranque y apagado.

3.3.13. ***Adaptabilidad***

Esta característica se refiere a la capacidad que tiene la aplicación de ser instalada en múltiples sitios, para múltiples organizaciones.

Puntaje:

0: No se consideran requerimientos necesarios para más de un sitio de instalación.

1: Se consideran múltiples sitios necesarios. La aplicación es diseñada para operar únicamente en idénticas condiciones de software y hardware.

2: La aplicación está diseñada para operar únicamente en condiciones similares de hardware y/o software.

3: La aplicación está diseñada para operar en condiciones diferentes de hardware y/o software.

4: La documentación y el plan de soporte son provistos y probados para que la aplicación soporte múltiples sitios. Tal como en los puntajes 1 y 2.

5: La documentación y el plan de soporte son provistos y probados para que la aplicación soporte múltiples sitios. Tal como en el puntaje 3.

3.3.14. ***Facilidad de cambio***

La aplicación ha sido específicamente diseñada, de forma tal que pueda soportar cambios fácilmente.

Puntaje:

0: No hay requerimientos especiales del usuario, con respecto al diseño de la aplicación, para minimizar o facilitar el cambio.

1-5: Seleccionar cualquiera de los siguientes ítems, de acuerdo con la aplicación.

- Flexibilidad y consultas sencillas (un ítem).
- Flexibilidad en las consultas, de forma que puedan manipular consultas simples o medianamente complejas (dos ítems).
- Flexibilidad en las consultas, de forma que se manejen consultas fáciles y complejas (tres ítems).
- El control de los datos se guarda en tablas que son mantenidas por el usuario con procesos interactivos, pero los cambios se efectúan únicamente pasado un tiempo.
- Igual que en el anterior, pero los cambios se efectúan inmediatamente (dos ítems).

3.4. Cálculo del valor del factor de ajuste

El factor de ajuste se calcula sobre la base de las 14 características generales del sistema. Este valor más adelante servirá para ajustar los puntos de función sin ajuste. El proceso para obtener este valor de ajuste es el siguiente:

Evaluar las 14 características generales del sistema, de acuerdo con el grado de influencia de cada uno.

Sumar los 14 grados de influencia para producir un grado de influencia total.

Introducir el grado de influencia, en la siguiente ecuación, para producir el factor de ajuste.

(TGI*0.01)+0.65=VFA

Donde TGI es el total de los 14 grados de influencia y VFA es el valor del factor de ajuste. Tomemos el siguiente ejemplo, en el cual se han evaluado las 14 características generales de un sistema, obteniendo, para cada una, su grado de influencia:

Características generales del sistema	Grados de influencia
Transmisión de datos	3
Procesamiento distribuido	4
Desempeño	4
Configuración	2
Indice de transacciones	3
Entrada de datos en línea	3
Eficiencia de usuario	4
Actualización en línea	3
Complejidad del proceso	3
Reutilización	2
Facilidad de instalación	3
Sencillez de operación	3
Adaptabilidad	2
Flexibilidad	3
Total de los grados de influencia (TGI)	42
Valor del Factor de Ajuste (VAF) (42 * 0.01)+0.65 =	1.07

Ahora estamos en capacidad de calcular los puntos de función, ajustados de acuerdo con la siguiente ecuación:

Puntos de Función sin Ajuste * VFA = Puntos de Función Ajustados

Supongamos ahora que el ejemplo de los puntos de función sin ajuste anterior, cuyo resultado fue de 288, corresponde a la misma aplicación a la que se le calculó el VAF anterior de 1.07. Aplicando la ecuación anterior, obtenemos los puntos de función, así:

288*1.07=308.16; es decir, obtenemos un valor aproximado de 308 puntos de función. Finalmente, la estimación del número de intrucciones-fuente se obtiene a partir de la relación siguiente:

LDC=66*PF.

Para el ejemplo anterior, entonces, tenemos 20.328 LDC.

Existen relaciones empíricas entre las líneas de código y los puntos de función para los lenguajes conocidos, así:

Assembler	320	LDC/PF
C	150	LDC/PF
Cobol	107	LDC/PF
Fortran	106	LDC/PF
ADA	71	LDC/PF
Lenguajes de bases de datos	40	LDC/PF
Generadores de lenguajes	32	LDC/PF
Orientado a objetos	29	LDC/PF
Lenguaje de consulta	25	LDC/PF
Pascal	85	LDC/PF
Hoja de cálculo	6	LDC/PF

De la tabla anterior se debe tener claro que, por ejemplo, para lenguaje C, 150 líneas de código equivalen a un punto de función.

CONCLUSIONES

1. Son muchos los modelos existentes para la estimación y que ofrecen diversas formas de obtener estimativos de costos y esfuerzo.
2. Cualquiera que sea el método escogido para la estimación del costo-esfuerzo, es importante que la organización que desee realizar estimaciones recolecte y mantenga un conjunto de datos históricos de estimaciones y métricas de proyectos anteriores, de forma que las estimaciones futuras sean mucho más confiables.
3. Todo proyecto de desarrollo de software tiene asociado un conjunto de atributos, ya sean tecnológicos, económicos y de mano de obra, los cuales afectan directamente el proceso de estimación.
4. Uno de los principales factores que influyen en el costo y esfuerzo de un proyecto de software es la complejidad misma del proyecto y el grado de confiabilidad requerida.
5. Es posible realizar estimaciones, en los proyectos de software, desde sus primeras etapas de vida. Para ello, quienes llevan a cabo la estimación,

deben conocer a fondo el sistema por desarrollar, la tecnología por emplear y la capacidad humana con que contará, puesto que con estos factores se logra un óptimo desarrollo del producto.

6. El estudio de los modelos de estimación requiere de buen tiempo de dedicación, ya que involucra muchos factores que son difíciles de evaluar, tales como tecnología, personal, herramientas de desarrollo, etc.

BIBLIOGRAFIA

- PRESSMAN, Rogger S. *Ingeniería del software: Un enfoque práctico.* Cap. 2 y 3. Segunda edición.
- BOEHM, Barry W. *Software Engineering Economics.* Prentice–Hall 1981.
- CARD, David N. with GLASS, R. *Measuring Software Design Quality.* Mc-Graw Hill.
- FRANCISCO, Gisbert Cantó. *Evaluación de Costes de Desarrollo - Modelos Algorítmicos.* Facultad de Informática. Universidad Politécnica de Madrid.
- IEEE. Transactions on software engineering. *Software development cost estimation using functions points.* Vol. 20 No. 24 April 1994, pág. 275.
- IEEE. Transactions on software engineering *Function point analysis: Dificulties and improvements.* Vol. 14 No. 1. January 1988.

PROPUESTA DE INTERCONEXION A INTERNET

JUAN CARLOS MACHADO
ANDRES FELIPE MILLA
JOHN FREDDY VALENCIA

Alumnos del curso de Investigación de VIII Semestre de Ingeniería de Sistemas del ICESI

INTRODUCCION

De las tres partes en las que ha sido dividido el proyecto de investigación (propuesta de Interconexión a **Internet**) presentamos la primera, que constituye la presentación teórica, la que permite ubicar un poco el contexto en el que se va a trabajar.

Con esto pretendemos empezar a tratar de una forma cronológica todo lo necesario para llegar a entender, de manera clara, lo que es el trabajo en **Internet**, lo cual permitirá, posteriormente, realizar un estudio de carácter técnico.

1. DEFINICION

La pregunta que más frecuentemente se hace es, "¿qué es **Internet?**".

La razón por la cual se hace tan a menudo esta pregunta es que no ha habido un acuerdo sobre una respuesta que ingeniosamente recapitule a **Internet.**

Internet puede ser considerada, a través de la relación con sus protocolos más comunes, como una colección física de enrutadores (routers) y circuitos, como un conjunto de recursos compartidos o simplemente como una actitud de interconexión e intercomunicación.

Algunas de las definiciones más comunes dadas en el pasado son:

- Una red de redes, basada en los protocolos del TCP/IP.
- Una comunidad de gente que usa y desarrolla las mencionadas redes.
- Una colección de recursos que pueden ser alcanzados, a través de estas redes.

Hoy en día, **Internet** es un recurso global que interconecta a millones de usuarios, los cuales empezaron como un experimento (veinte años atrás) del Departamento de Defensa de los Estados Unidos de América.

Mientras que las redes que "inventaron" a **Internet** están basadas en un conjunto estándar de protocolos (un acuerdo mutuo de métodos de comunicación entre los interesados), **Internet** cuenta con pasarelas (gateways), hacia redes y servicios que están basados en otros protocolos.

2. HISTORIA

Internet nació hace 21 años, al intentar conectarse entre sí la red del Departamento de Defensa Norteamericano (llamada ARPAnet) y otras redes radiales y satelitales.

ARPAnet fue una red experimental diseñada particularmente para dar soporte a los recursos militares que se invertían en la construcción de redes que pudieran soportar tanto situaciones peligrosas (tales como ataques con bombas) como funciones de rastreo.

En el modelo ARPAnet, las comunicaciones siempre ocurrían entre el computador-fuente y el computador-destino. La red en sí misma era asumida como de poca confianza; cualquier porción de la red podría desaparecer en cualquier momento (hoy en día el corte de cables es más catastrófico que las bombas). Además, la red estaba diseñada para requerir el mínimo de información de los clientes computacionales.

Para enviar un mensaje a través de la red, un computador sólo tenía que poner sus datos en un sobre, llamado paquete de Protocolo Internet (IP), y dirigir correctamente dicho paquete.

Los computadores de comunicaciones, y no la red en sí, eran los responsables de asegurar que la comunicación fuera llevada a cabo como se esperaba. La filosofía era que cada computador, en la red, pudiera hablar frente a frente con cualquier otro computador.

Estas decisiones podían sonar extrañas, y más con la suposición de que la red era "poco confiable", pero la historia nos dice que la mayoría de ellas probaron que eran razonablemente correctas.

Si bien la Organización para la Estandarización Internacional (ISO) empleó varios años en el diseño del estándar supremo de las redes computacionales, la gente no pudo esperar.

Los desarrolladores de **Internet** en los Estados Unidos, el Reino Unido y Escandinavia, para responder a las presiones del mercado, empezaron a colocar su Software IP en cada tipo de computador concebible. Esto llevó al desarrollo del único método práctico de comunicación entre los computadores de diferentes fabricantes.

Este hecho se volvió atractivo, tanto para el gobierno como para las universidades, los cuales no tenían políticas definidas acerca de que todos los computadores debían de comprarse al mismo vendedor. Cada cual compró el computador que más le gustaba, y esperó que los demás computadores pudieran trabajar con él, a través de la red.

Casi al mismo tiempo que **Internet** empezaba a ser, las redes Ethernet de Area Local (LANs) fueron desarrolladas. Esta tecnología fue madurada silenciosamente hasta que las estaciones de trabajo empezaron a estar disponibles, en 1983. La mayoría de estas estaciones de trabajo (WorkStations) venían con el UNIX de Berkeley, el cual incluía el software de red, IP.

Este hecho creó una nueva demanda: además de conectarse a un simple computador, para utilizar el tiempo compartido, las organizaciones necesitaban conectar toda su red local a ARPAnet. Esto permitiría a todos los computadores, en esa LAN, tener acceso a las facilidades de ARPAnet.

Casi por la misma época, otras organizaciones empezaron a construir sus propias redes, usando los mismos protocolos de comunicación de ARPAnet, denominados IPs, y sus parientes.

De esta manera resultó obvio que si estas redes podían hablar entre sí, otros usuarios en otras redes podrían comunicarse con éstas, y así obtener muchos beneficios.

Una de las redes nuevas más importantes fue NFSNET, comisionada por la Fundación Nacional de Ciencia (NSF: National Science Foundation), una agencia del gobierno norteamericano. A finales de los años 80, la NSF creó cinco centros de supercomputadores. Acerca de este punto, los computadores más rápidos del mundo estaban disponibles sólo para los desarrolladores de armas, y algunos otros recursos, sólo para corporaciones muy grandes.

Mediante la creación de los centros de supercomputadores, la NFS puso estos recursos al alcance de cualquier entidad estudiantil. Solamente cinco centros fueron creados, debido a su alto costo, a pesar de que debían ser compartidos por todos los que los requirieron.

Esto creó un problema de comunicación. Ellos necesitaban una manera de conectar sus centros entre sí y permitir que los clientes de estos centros pudieran acceder a ellos.

Al principio, NSF intentó usar a ARPAnet para sus comunicaciones, pero esta estrategia falló debido a la burocracia y a problemas de personal.

En respuesta a esto, NSF decidió construir su propia red, basada en la tecnología de ARPAnet. Esta conectó los centros a 56.000 bits por segundo (56k bps), mediante las redes telefónicas.

Sin embargo, era obvio que si ellos intentaban conectar directamente cada universidad con un centro supercomputacional, dicha red se podría romper. Por esta línea telefónica se pagaba por milla. Una línea, por campus (Ciudad Universitaria), conectada con un centro supercomputacional involucraba un montón de millas de líneas telefónicas. Por lo tanto, ellos decidieron crear redes regionales. En cada área del país, las escuelas debían estar conectadas a su vecino más cercano. Cada cadena estaba conectada a un centro supercomputacional en un punto, y cada centro estaba conectado a otros. Con esta configuración un computador podría eventualmente comunicarse con cualquier otro, remitiendo la comunicación entre vecinos.

Esta solución fue exitosa, y, como toda solución de este tipo, no se dejó esperar. El compartir super-computadores también permitió a los puntos conectados, compartir un montón de otras cosas no relacionadas con los centros.

De repente, estas escuelas tuvieron un mundo de datos y colaboradores a su alcance. El tráfico de la red se incrementó tanto, hasta eventualmente, que los computadores que controlaban la red y las líneas telefónicas que los conectaban se sobrecargaron.

En 1987 se firmó un contrato para manejar y actualizar la red, con la Merit Network Inc., la cual había puesto a correr la red educacional de Michigan, en asociación con IBM y MCI. La vieja red fue reemplazada con líneas telefónicas mucho más rápidas (por un factor de 20), y con computadores más veloces para controlarla.

Este proceso de mejoramiento continuo prosigue hoy en día, a diferencia de cambios a la autopista del sistema; sin embargo, la mayoría de estos cambios no son notados por la gente que trata de usar a **Internet** para hacer un trabajo real.

Usted no va a su oficina; hace "loggin" en su computador y encuentra un mensaje en el cual le dicen que **Internet** va a estar *inaccesible*,por los próximos seis meses, debido a mejoramientos. Quizás algo más importante: el temor a quedarse sin capacidad y el deseo de mejorar la red, han creado una tecnología que es extremadamente madura y práctica. Las ideas han sido probadas, los problemas han aparecido pero se han resuelto.

Para nuestros propósitos, el aspecto más importante del esfuerzo, en cuanto a redes, hecho por la NSF, radica en que permitió, a cualquiera, tener acceso a la red. Además, el acceso a **Internet** había estado disponible únicamente para los desarrolladores de las ciencias computacionales, los empleados del gobierno y los contratistas del mismo.

La NSF promueve el acceso educacional universal, mediante la creación de conexiones, en los campus, pero solamente si éstos tienen un plan para la extensión del acceso. De manera que cualquiera que vaya a tomar sus años de "college" podrá ser un usuario **Internet.**

La demanda continúa creciendo. Ahora, cuando la mayoría de los "colleges" llevan más de cuatro años conectados, la gente está tratando de que los colegios de primaria y secundaria se conecten.

La gente que se ha graduado en los "colleges" sabe para qué es bueno el **Internet,** y les hablan a sus empleados sobre cómo conectarse a sus corporaciones.

Toda esta actividad apunta a seguir creciendo; a ir resolviendo problemas, a través de la red; e involucrar tecnologías y a desarrollar un trabajo de seguridad para las personas de la red (networkers).

3. ADMINISTRACION

¿De qué consta **Internet?** Es una pregunta difícil de responder pues la respuesta cambia con el tiempo. Hace unos cinco años, la respuesta podría haber sido fácil: "Todas las redes, usando el protocolo IP, las cuales cooperan para formar una red sin costuras, para todos sus usuarios colectivos".

Esto podría incluir variadas redes federales, un conjunto de redes regionales, redes de campus y algunas redes extranjeras.

Recientemente, algunas redes nobasadas en IP vieron que **Internet** era algo muy bueno. Ellas necesitaban proveer servicios a sus clientes, de manera que idearon la manera de interconectar a estas "extrañas" redes (por ejemplo: Bitnet, DECnets, etc.) con **Internet.**

Al principio, estas conexiones, llamadas "gateways" (compuertas), meramente servían para transferir correo electrónico entre las dos redes.

Algunas, sin embargo, se desarrollaron para trasladar otros servicios, a través de las redes. ¿Hacían parte del **Internet**? Tal vez sí y tal vez no. Eso dependía de si en su corazón lo querían o no. Puede que suene extraño, pero así es.

De muchas maneras, **Internet** es como una iglesia: Esta tiene su Consejo de Mayores; cada miembro tiene su opinión, acerca de cómo deben funcionar las cosas y tú puedes tomar parte en esto o no hacerlo. Es tu decisión.

Internet no tiene Presidente, Oficial de Operación en Jefe, o Papa. Las redes constituyentes deben tener presidentes y CEO's, pero esto es un tema diferente; no hay una autoridad ni figura única para **Internet,** como un todo.

La autoridad suprema, por donde **Internet** es apoyada, es la Sociedad Internet o ISOC (Internet Society).

ISOC es una organización voluntaria de miembros cuyo propósito es promover el intercambio global de información, a través de la tecnología **Internet.**

Esto apunta a un Consejo de Mayores, quienes tienen la responsabilidad del manejo técnico y de la dirección de **Internet.**

El Consejo de Mayores es un grupo de voluntarios invitados, llamados la Directiva de la Arquitectura Internet, o IAB (Internet Architecture Board). La IAB se reúne regularmente para "bendecir" estándares y asignar los recursos, tales como direcciones. **Internet** trabaja debido a que existen maneras estándar para que los computadores y los software aplicativos puedan hablar los unos con los otros. Esto permite que computadores de diferentes fabricantes se puedan comunicar sin problemas. Esta no es sólo una red de IBM, de SUN's o de Macintosh.

La IAB es responsable por estos estándares, ella decide cuándo un estándar es necesario y qué debe hacer dicho estándar. Cuando un estándar se hace necesario, ella considera el problema, adopta el estándar y lo anuncia a través de la red.

La IAB también lleva el control sobre varios números (y otras cosas) que se deben mantener como únicos. Por ejemplo, cada computador de **Internet** tiene una única dirección de 32 bits, ningún otro computador tiene igual número.

Tal como en una iglesia, cada cual tiene opiniones acerca de cómo deben ocurrir las cosas. Los usuarios de **Internet** pueden expresar sus criterios a través de encuentros (reuniones) de la Fuerza de Tareas de Ingeniería de Internet, IETF (Internet Engineering Task Force). La IETF es otra organización voluntaria; ésta se reúne regularmente para discutir sobre los problemas operacionales y técnicos de la red.

Cuando ellos consideran que un problema tiene el suficiente mérito, la IETF organiza un "grupo de trabajo" para la investigación posterior. Cualquiera puede asistir a las reuniones de la IETF y participar de los grupos de trabajo; lo importante es que esa persona trabaje en la resolución del problema.

Estos grupos de trabajo usualmente producen un reporte. Dependiendo de la clase de recomendación, dicho reporte podría constituir una documentación para cualquiera que lo necesitara, o podría ser aceptado voluntariamente, como una buena idea que la gente podría seguir, o podría ser enviado a la IAB, para ser declarado como un estándar.

Si usted va a una iglesia y acepta sus enseñanzas y su filosofía, usted es admitido por ella, y recibe los beneficios. Si a usted no le gustan éstos, la puede abandonar.

Es el mismo caso para **Internet.** Si una red acepta las enseñanzas de **Internet,** es conectada a ésta y es considerada, ella misma, como parte de la primera, es decir, ya es una parte de **Internet.** Ella podría encontrar cosas que no le gusten y dirigirlas a través de IETF. Algunas de estas consideraciones podrían ser estimadas como válidas e **Internet** las cambiaría gustosamente. Algunos de estos cambios podrían correr en contra de esta religión y serían rechazados.

Además, si la red hace algo que cause daño a **Internet,** será incomunicada hasta cuando deje de hacer sus obras "diabólicas" (dañinas).

4. INTERNET y TCP/IP

Hoy, las comunicaciones tienen un papel importante en la computación; atrás quedaron los años en que las organizaciones tenían redes exclusivas para un solo grupo. En la actualidad los grupos establecen el intercambio de información mediante el uso del correo electrónico; los aficionados intercambian programas desde sus computadores caseros; los científicos procesan y evalúan la información, en supercomputadoras remotas, intercambiando estos resultados con otros colegas en el mundo.

La pregunta es: ¿cómo enfrentarnos a este desafío? Hace algunos años emergió una nueva alternativa para resolver dicho reto, con la posibilidad de interconectar redes físicamente dispersas, prestando todos estos servicios a

los usuarios de los computadores. A esta tecnología se le llama "internetwork o internet". Pero, ¿cómo es posible "internetwork"?

4.1 Internetwork con el TCP/IP

4.1.1. *¿Qué es el TCP/IP?*

Podemos definir el TCP/IP como un conjunto de estándares de red, desarrollados por DARPA (Departamento de Defensa de E.U.A.) que especifican las características de comunicación entre los computadores; además, como un conjunto de convenciones para la interconexión y el enrutamiento, en redes de computadores.

Las primeras investigaciones sobre los TCP/IP se iniciaron a mediados de los años 70, por parte de DARPA, pero fue sólo en 1983 cuando DARPA exigió a todos los computadores conectados a ARPANET ("la red de DARPA") el uso del TCP/IP, cuando esta familia de protocolos comenzó a comercializarse de manera extensa.

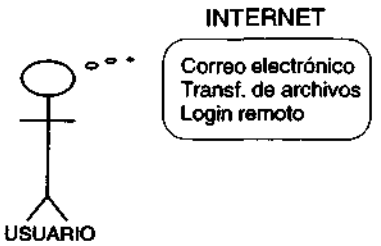
Ahora, la familia TCP/IP es usada por prestigiosas redes de agencias de los EE.UU., como la Fundación Nacional de la Ciencia (NSF), el Departamento de Energía, la NASA y muchas universidades importantes de ese país.

4.1.2. *¿Por qué Internetwork con el TCP/IP?*

La familia de protocolos del TCP/IP es apropiada para el trabajo de interconexión entre redes de computadores, por dos principales características:

* Separa la comunicación del concepto de tecnología.
* Oculta al usuario los detalles de bajo nivel.

Es decir, el TCP/IP permite hacer "transparente", al usuario, la tecnología usada, haciendo ver a la red más bien como una red de Servicios (Correo Electrónico, Transferencia de Archivos, Login Remoto) y clarificando y facilitando el uso de la red por parte del usuario final.



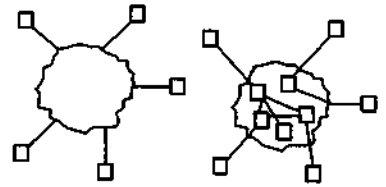
4.2 Principios básicos del TCP/IP

4.2.1. *Arquitectura del TCP/IP*

En muchas ocasiones, hemos tenido que resolver el problema de conectar máquinas en una red local. Pero ¿cómo interconectar las redes en la tecnología internetwork? Físicamente, podríamos conectar las dos redes a un computador que puede atender a ambas; éste debería permitir el envío de paquetes de información, de una red a otra. A estos computadores o dispositivos que enlazan dos tipos de redes, se les llama "gateways".

También es factible tener conexiones más complejas, como sería el caso de establecer una interconexión entre "gateways".

Esta arquitectura permite ver el trabajo de internetwork como una red virtual, donde están conectadas todas las máquinas, independientemente de su conexión física. La ventaja de proveer esta interconexión a nivel de red es que su función es mucho más clara para el usuario.



Como el usuario lo ve... Como es, en realidad, la estructura física...

4.2.2. *Direccionamiento del TCP/IP*

Una de las filosofías de **Internet** es ser una red de largas distancias con la posibilidad de interconectar a cualquier otra red. Pero, obviamente, cada uno de estos puntos puede soportar distintas configuraciones, tamaños, etc. impidiendo así que se referencie, mediante una dirección física, a algún punto de la red.

Es entonces cuando surge la necesidad de tener un servicio de identificación global, para dirigirse a cada "host", en **Internet.** A esta identificación se le llama número _ip, compuesto de 32 bits, que representan información en función de la cantidad de redes y de "hosts".

Existen tres tipos de dirección del TCP/IP:

* Tipo A: (Muchos "host" - pocas redes)
  A: OB1... B7 B8 ..... B31
  (red) (host)

* Tipo B: (intermedio)
  B: 10B1 ..... B15 ..... B31
  (red) (host)

* Tipo C: (pocos "host" - muchas redes)
  C: 11B1 ..... B23 .... B31
  (red) (host)

Para hacer uso de estos tipos de dirección, **Internet** se vale de la notación . , para representar así las direcciones IP, de los usuarios. Cada dirección puede contar con:

Notación ● :

	@		●		●		●	
Usuario		Máquina		Institución		Sector		País

Ejemplo:

Icesi	@	Mafalda	●	Univalle	●	EDU	●	CO

En donde cada palabra hace parte del Sistema de Dominios del Nombre. Dichos dominios se establecen según el lugar donde se encuentran tanto el usuario como el "host" con el que se desea comunicar.

Nivel 1 (Usuario y "host" dentro de la misma institución)

	@		●	
Usuario		Máquina		Institución

Nivel 2 (Usuario y "host" dentro del mismo país)

	@		●		●	
Usuario		Máquina		Institución		Sector

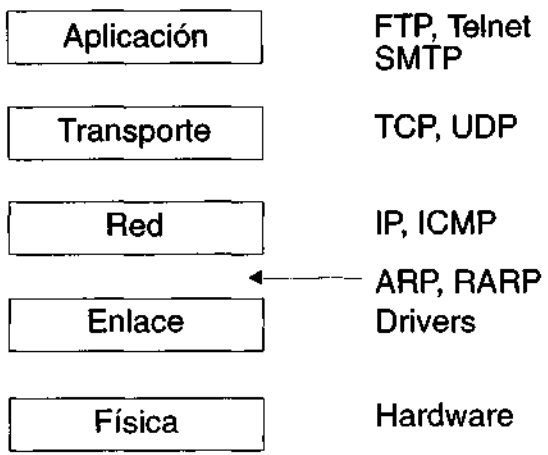
Nivel 3 (Se asume que el país difiere entre usuario y "host").

	@	•	•	•
Usuario	Máquina	Institución	Sector	País

De esta manera, según sea nuestro dominio de los Niveles 1, 2, ó 3, esta notación nos permite utilizar la dirección **Internet** que, en este caso, equivaldría a la Dirección de Conexión de la red.

Es importante tomar nota de los tipos de dominios más utilizados y que, comúnmente, se referencian en las direcciones IP:

Nombre del Dominio	Descripción
COM	Organización comercial
EDU	Institución educativa
GOV	Institución gubernamental
MIL	Grupos militares
NET	Centrales de soporte de redes grandes
ARPA	Dominio temporal del **Internet**
ORG	Cualquier otra organización
Código del país	Para países fuera de EE.UU.



Modelo TCP/IP

4.3. Protocolos de la familia del TCP/IP

4.3.1. *ARP (Address Resolution Protocol)*

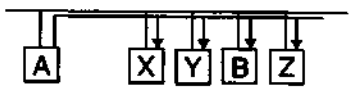
Cuando conocemos que en una red tipo Ethernet, las direcciones son de seis bits y éstas son propias de la tarjeta (adapter LAN), es posible que nos intrigue saber ¿cómo es posible que haya comunicación con la dirección lógica del TCP/IP (dirección **Internet**)?

En realidad, la comunicación se establece mediante direcciones físicas; por eso es necesario el protocolo ARP, cuya función es convertir las direcciones **Internet** en direcciones físicas para comunicarse.

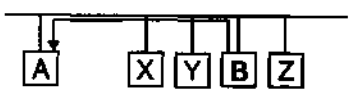
Dirección Internet >>>> ARP >>>> Dirección física

Las direcciones físicas se pueden encontrar mediante Broadcast (envío de un mensaje a todas las estaciones, para preguntar: ¿Quién tiene la dirección lógica de la estación, con la que deseo comunicarme?)

Ejemplo: Deseo comunicar A con B.



A, mediante "broadcast", hace una requisición acerca de la dirección lógica de B, en todas las máquinas.



B responde, trayendo la dirección física de B.

También es posible implementar el ARP mediante un "caché" que contenga las direcciones **Internet** y las direcciones físicas más utilizadas últimamente, disminuyendo así el costo del "broadcast" y limitándose, en muchas ocasiones, a una búsqueda rápida, en el caché de direcciones, de la máquina origen.

4.3.2. *RARP (Reserve Address Resolution Protocol)*

Existe el caso en que tenemos estaciones brutas, sin disco, conectadas a un servidor, donde lo único que ellas conocen es su dirección física. ¿Cómo puede saber ella cuál es su dirección lógica? La respuesta es: mediante un servidor RARP, es decir un servidor que almacena la relación entre las direcciones físicas de las estaciones y sus respectivas direcciones lógicas **Internet.** Así, una estación bruta puede solicitar, a un servidor RARP, que le devuelva la dirección lógica mediante el envío previo de su dirección física.

En pocas palabras:

Dirección física >>>> RARP >>>> Dirección Internet

4.3.3. *IP (Internet Protocol)*

Como ya se analizó, uno de los servicios que presta el TCP/IP es poder enrutar paquetes desde una estación X a una estación Y, sin importarle al usuario cómo se realiza dicha comunicación. El encargado de realizar esta labor, en la familia TCP/IP, es el protocolo IP, no orientado hacia la conexión, el cual provee las funciones de red necesarias para transportar Datagramas de una máquina-origen a una máquina-destino. ¿Qué es un Datagrama? Es un paquete de información compuesto por un área de cabecera o Header y un Area de Datos. En el Header se encuentra la dirección-origen y el destino de la comunicación.

Datagrama Internet

Además de caracterizarse por trabajar en las redes de conmutación de datagramas, IP permite fragmentar los datagramas, de forma que pueda cumplir con su responsabilidad, al enrutar la información desde el origen hasta el destino. Pero claro, IP no se encarga de todas las labores de la familia de protocolo, tales como:

* Confiabilidad de extremo a extremo
* Control de la secuencia
* Reconocimiento
* Control de flujo

En resumen, IP enruta los datagramas entre "gateways", de manera deliberada, sin tener rutas preestablecidas en la red. Su misión, como ya lo mencionamos, es "colocar los paquetes de información en la máquina-destino".

4.3.4. *ICMP (Error and Control Messages Protocol)*

Pero si IP sólo se encarga de enrutar los datagramas, sin preocuparse por la confiabilidad de la transferencia, ¿cómo sabemos cuándo ocurrió un error en el enrutamiento? ICMP permite reportar los errores en el procesamiento de datagramas; por ejemplo: Datagrama sin destino, Gateway sin almacenamiento, Tiempo excedido para datagrama, Problemas de parámetros en un datagrama, etc.

4.3.5. *UDP (Use Datagram Protocol)*

Ya hemos conocido la capacidad del TCP/IP para transferir datagramas IP a "hosts", pero, hasta el momento, no hemos reconocido algún protocolo dentro de la familia que nos permita identificar al usuario o la aplicación especifica que va a recibir el datagrama, es decir, necesitamos saber "qué hay dentro del host-destino". La solución a este interrogante es UDP.

UDP provee un mecanismo que distingue los distintos recipientes de datagramas en una máquina específica, además de contener el número del puerto-destino y el número del puerto-fuente. En síntesis, UDP ofrece el servicio de IP para transportar los mensajes a las máquinas y, adicionalmente, tiene la habilidad de identificar el destino último de "éstos".

UDP = Identificación Ultimo Destino + IP

4.3.6. *TCP (Transmisión Control Protocol)*

Cuando estudiamos el protocolo IP, mencionamos que éste transfería los datos de una manera poco confiable, lo que podría incluir duplicidad, pérdida o entrega fuera de secuencia, de los paquetes de información. Para resolver dicho inconveniente, que surge cuando existe la exigencia de transferir grandes volúmenes de información, se hace necesario un protocolo de software, de carácter general, y que provea una entrega confiable de información. Este protocolo es el TCP.

	Servicio		
Aplicación >>	Entrega confiable (TCP)	+	Servicio Entrega poco confiable (IP) >> TX

La interfase de transporte que permite el TCP, se caracteriza por constituir, en un chorro de bits, los paquetes de información, aunque estos se encuentren virtualmente segmentados. Además, provee un circuito transparente, entre el origen y el destino, rasgo característico de la familia TCP/IP. Para realizar éstas y muchas otras funciones, puede utilizar reconocimientos o el sistema de ventanas deslizantes (métodos reconocidos también en otros protocolos).

5. CONCEPTOS BASICOS DE INTERNET

Nosotros estamos realmente en una sociedad de la información y ahora más que nunca, cuando el mover grandes cantidades de información, de manera rápida, a través de grandes distancias, es una de nuestras más preciadas necesidades.

Desde el esfuerzo del más pequeño empresario al de la más grande de las corporaciones, más y más gente profesional ha descubierto que la única forma de tener éxito en los años 90, y ser superior, es realizar esa tecnología, con paso cuidadoso, para no ir a perder la cabeza y de alguna manera, cuidarse.

Por otro lado, desarrolladores de todos los rincones del planeta están encontrando que su trabajo está progresando, en los ambientes de red. El acceso inmediato al trabajo de otros colegas y una biblioteca "virtual" de millones de volúmenes y de miles de documentos variados, obligan al usuario a incorporarse a este conocimiento, que antes era inimaginable. Los grupos de trabajo pueden ahora conducir conferencias interactivas, con cada uno de los integrantes de éstos, sin cobrar por cada ubicación física, por lo cual las probabilidades son infinitas.

Tenemos la posibilidad de hablar, en "tiempo real", con cualquier persona, en el Japón; enviar una corta historia de 2.000 palabras a un grupo de gente que estaría gustosa de hacer una buena crítica; ver si un Macintosh, localizado en un laboratorio en Canadá, está prendido y buscar si hay alguien sentado frente a su computador (logged on) en Australia, todo esto en treinta minutos. ¡No hay ninguna aerolínea que nos pueda satisfacer este itinerario!

El mayor problema que tiene la gente que usa por primera vez una red es comprender todo lo que está disponible (la potencia de la red). Es normal ver la sorpresa de un usuario cuando descubre un nuevo servicio o un mecanismo que nunca imaginó que pudiera existir.

5.1. Dominios

Encontrar dónde es que deseamos ir puede resultar uno de los más difíciles aspectos del uso de las redes. La variedad de formas en que los lugares son nombrados podrá, probablemente, dejar una mirada de dolor en su cara la primera vez. No se inquiete; hay un método para esta aparente locura.

Si alguien ha querido averiguar por la dirección de la residencia de alguien, probablemente ha esperado una calle, un número de la casa, una ciudad y un departamento. Esta es toda la información que la oficina del correo necesita para enviarlo de manera razonablemente rápida. Así, las direcciones de los computadores tienen una estructura parecida.

La forma general es:

Una dirección de correo electrónico, sobre una persona, en un computador:

usuario@AlgúnLugar dominio

Un nombre de computador:
AlgúnLugar.dominio

Nota: email(Electronic Mail = Correo Electrónico).

La porción del usuario es comúnmente el nombre de la cuenta de la persona en el sistema, pero no quiere decir que así tiene que ser.

AlgúnLugar.dominio nos dice el nombre del sistema o locación y qué tipo de organización es ésta.

El dominio del que estamos hablando es a menudo el siguiente:

com: Usualmente una compañía u otra institución comercial u organización, tal como Samsung Computers ('samsung.com').

edu: Una institución educativa; ej: Universidad ICESI, nombrada como 'icesi.edu'.

gov: Un sitio gubernamental; por ejemplo, NASA es "nasa.gov'.

mil: Un sitio militar, tal como la Fuerza Aérea ('ffaa.mil').

net: Gateways y otros hosts administrativos de una red (esto no significa todos los hots de la red). Un gateway podría ser 'cerca.net'.

org: Este es un dominio reservado para las organizaciones privadas, las cuales no se acomodan muy bien en las otras clases de dominios. Un ejemplo puede ser el Comité de Estudiantes de Ingeniería de Sistemas, nombrado como 'ceis.org'.

Cada país tiene, a su vez, su propio dominio. Por ejemplo están:

au: Australia

ca: Canadá

fr: France

uk: Reino Unido

co: Colombia

La apropiada terminología para el nombre del dominio del lugar (el AlgúnLugar. dominio de arriba) se dice que es su Fully Qualified Domain Name (FQDN). Es seleccionado, usualmente, para dar una clara indicación del lugar de la organización o agente patrocinador.

Por ejemplo, el FQDN del *Massachusetts Institute of Technology's* es 'mit.edu'. Es normal que usualmente se usen nombres obvios, pero hay casos como 'vt.edu' donde se pensaría que 'vt' es por Vermont, y en realidad viene por Virginia Tech.

5.2 Números Internet

Cada máquina, en **Internet,** tiene por lo menos una dirección (posiblemente dos y hasta tres), llamada su número **Internet** o Dirección IP. Actualmente es un número de 32 bits, pero es representado más comúnmente como cuatro números unidos por puntos ('.'); por ejemplo, 147.31.254.130.

Esta dirección es llamada algunas veces una cuádrupla punteada (dotted quad); hay, literalmente, miles de diferentes "dotted quads".

ARPAnet (la mamá de la actual **Internet**) originalmente tenía solamente la capacidad de soportar 256 sistemas sobre ella, debido a la manera en que cada sistema había sido dirigido.

Al principio de los años 80 empezó a hacerse claro que las cosas iban a crecer más rápido de los límites que se habían pensado; nació la dirección de 32 bits y se liberaron miles de números para los diversos "hosts".

Cada pieza de una dirección **Internet** (tal como 192) es llamada un Octete, representando uno de los cuatro conjuntos de 8 bits.

Las primeras 2 ó 3 piezas (ej: 192.55.239) representan la red en la que está el sistema, llamada subred (subnet).

Por ejemplo, todos los computadores en la Universidad de Wesleyan están en la subnet 129.33. Ellos pueden tener números como 129.33.10.10, 129.33.230.19, y más de 65.000 posibles combinaciones (posibles computadores).

Las direcciones IP y los nombres de los dominios no son asignados arbitrariamente. Los candidatos se pueden inscribir en la Network Information Center (NIC), ya sea electrónicamente (a Hostmaster@nic.ddn.mil) o a través del correo regular.

5.3. Resolución de nombres y números

Bueno, los computadores pueden ser referidos por su FQDN o por su dirección **Internet,** pero ¿es posible que un usuario los recuerde todos?

¡Seguro que no! **Internet** está diseñada para que uno pueda usar el método que quiera. Hasta ahora, los humanos han encontrado que es más amigable referirse a ellos con palabras que con números; de todos modos el FQDN para cada host es mapeado a su Número Internet.

Cada dominio es servido por un computador, con ese dominio, el cual provee toda la información necesaria para ir de un nombre de dominio a una dirección IP y viceversa.

Por ejemplo, cuando alguien se refiera a foosun.bar.com, el "resolucionador" sabe que debe preguntar al sistema foovax.bar.com acerca de los sistemas en bar.com. Este le pregunta qué dirección **Internet** tiene foosun-.bar.com; si el nombre foosun.bar.com realmente existe, foovax retornará su número. Todo esto es transparente (invisible) al usuario.

5.4. La conexión física

Las conexiones actuales entre las variadas redes tienen diversas formas. Las más prevalentes conexiones para **Internet** son las líneas arrendadas de 56k (líneas telefónicas dedicadas, con conexiones de 56 kilobits por segundo) y las conexiones T1 (líneas telefónicas especiales, con conexiones de 1 Mbps).

También hay instaladas conexiones T3, actuando como "backbones" (espinas dorsales) entre las principales estaciones manejando cargas de tráfico masivo a 45 Mbps.

Estas conexiones son pagadas por cada institución a un carrier local. (Por ejemplo, Bell Atlantic es la dueña de PrepNet, la principal proveedora de Pennsylvania).

También están disponibles las conexiones SLIP, las cuales manejan tráfico **Internet** (paquetes) sobre modems de alta velocidad.

6. TIPOS DE USUARIOS QUE USAN Y PUEDEN USAR INTERNET

Los cambios presentes que se han venido dando en el mundo, tales como la internacionalización de los mercados, han hecho que las empresas empiecen a desarrollar estrategias que les permitan ser más competitivas. De todo esto tenemos ejemplos, como la Reingeniería, el Outsourcing, la Horizontalización de las compañías, que nos reafirman la tendencia al cambio de la que estamos hablando.

Acabamos de ver algunas estrategias que se están utilizando, pero eso no es todo, debido a que el flujo de información que requiere la compañía no ha sido tratado con mucha claridad por estos modelos. Para todos es claro que el manejo de la información es cada vez más importante en áreas como:

- *El mercadeo.*
  (Conocer los precios de la competencia, su comportamiento nacional e internacional, etc.)
- *La producción.*
  (Conocer sobre nuevos equipos que están saliendo, nuevas técnicas para disminuir el desperdicio, métodos de planificación de la producción, etc.)
- *Las finanzas.*
  (Saber qué está sucediendo en el mercado nacional, conocer el comportamiento de las monedas, tanto las extranjeras, como la nacional; las posibilidades de inversión, los peligros, etc.)
- *La alta gerencia.*
  (Intercambio de información con casas matrices, con sucursales, con

socios en otras ciudades, en otros países, etc.)

- *El gobierno.*
  (Intercambio de información con otros países, embajadas, etc.)
- *Las universidades.*
  (Comunicación entre universidades, intercambios de información en investigaciones, entre docentes, etc.)

Si al leer estas referencias usted deja volar un poco la imaginación, se dará cuenta de que existe un innumerable conjunto de hechos que en la vida diaria generan un movimiento de información entre los diferentes estamentos de un país.

Habiendo hecho el ejercicio anterior, usted debería preguntarse: ¿qué tendría que hacer para poder realizar el intercambio de información del que le estamos hablando?

La duda que lo debe estar embargando ahora es la misma que en su momento se les presentó a muchas organizaciones y que desde luego se manifestará en los demás, en un futuro muy cercano.

Veamos ahora lo que han hecho los usuarios para resolver su "problema".

Cuando cada organización estudió los costos que tendría que pagar para solucionar sus problemas en forma individual, comprendió que era casi imposible hacerlo en forma separada, debido a las inversiones tecnológicas requeridas. Por esto surgió **Internet,** como una solución que daba a todos la oportunidad de utilizar una misma tecnología y adaptarla a lo que ya existía en cada compañía, lo que le permitiría al usuario realizar todas las transacciones indispensables para satisfacer cada una de sus necesidades.

Si después de leer los párrafos anteriores piensa que hemos sido un poco emotivos, no se quede con la duda; acuda a un nodo **Internet** y explore. Le anticipamos que terminará reconociendo que hemos sido modestos en nuestras apreciaciones.

Tipos de organizaciones:

Empresas de producción.

Empresas de servicios.

Empresas gubernamentales.

Universidades.

Institutos educativos.

Institutos de investigación.

Organizaciones de salud.

Dese cuenta que estos tipos de organizaciones trabajan con la información que mencionamos hace un momento.

Si quisiéramos hacer un resumen de este aspecto, tendríamos unas pocas palabras con las cuales responderíamos la pregunta sobre los tipos de usuarios que pueden utilizar **Internet.**

Deténgase un poco en la respuesta que daremos a continuación.

No hay organización, competitivamente activa, que no pueda hacer uso de Internet.

7. TIPO DE INFORMACION MANEJADA Y QUE PUEDE MANEJAR INTERNET

En el parágrafo anterior hemos dado algunas pautas sobre el tipo de información utilizado por las empresas.

Con el fin de ir orientando un poco el estudio que estamos planteando, hacemos énfasis, en este punto, sobre el manejo que se le puede dar a **Internet** en las universidades; éstas son, quizás, las que más utilizan los servicios de **Internet,** debido simplemente a que son los centros de recopilación de información, con la cual se llevan a cabo las investigaciones que actualmente son el punto de partida del desarrollo mundial.

Veamos algunos usos, que incluso se llevan a cabo en nuestro país:

A: Ingreso a consultas en bibliotecas.

B: Manejo de directorios internos, en la universidad.

C: Presentación de informes generales sobre la universidad.

D: Información a la comunidad universitaria sobre eventos nacionales e internacionales.

E: Información, al exterior, sobre actividades programadas por la universidad.

F: Intercambio de información entre las universidades, sobre seminarios, congresos, para realizar actividades en forma conjunta que beneficien a la comunidad universitaria.

G: Intercambio de información entre los diferentes comités de las universidades, para compartir vivencias y poder hacer más sólida la participación, como institución, en otras regiones.

H: Intercambio de información entre los docentes de todas las áreas, con el fin de actualizarse y transmitir conocimientos a otras universidades.

I: Intercambio de conocimientos en proyectos de investigación, que permitan el trabajo a nivel interinstitucional.

J: Tomar información de bases de datos internacionales, que permitan mantener actualizada a toda la universidad, en cada una de las áreas en que esté especializada.

K: Conectarse con institutos de investigación para estar al tanto de las últimas innovaciones o tendencias, que permitan mantener a la universidad informada en tecnología de punta.

L: Poder tener contacto directo con las compañías de la región, con el fin de conocer los cambios que viven, para así orientar a sus estudiantes a que sean más competitivos.

M: Poder participar en conferencias sobre dónde obtener información para irrigarla en el medio ambiente.

N: Ejercer la labor de docencia institucional, al implantar una red de preguntas y respuestas donde el conocimiento de las universidades sea vertido al medio ambiente.

O: Participar en grupos de interés, donde se traten temas de toda índole. Cabe recordar que de estas charlas han salido ideas que revolucionaron al mundo.

Encontramos hasta el momento unas buenas razones por las cuales es tan importante la participación de las universidades en la red de redes **Internet.**

Aunque pueden ser bastantes las opciones presentadas, pensemos con detenimiento en las posibilidades que hay para una universidad y, en general, para cualquier organización. Realmente, el límite lo pone nuestra imaginación, ya que existe una fuente sumamente rica en información y en posibilidades inmensas para su intercambio.

Pensemos en que ahora estamos en una red de redes: ¿alcanzamos a imaginarnos en una autopista de autopistas de información?

Lo que existe y las posibilidades, hablan por sí mismas. No olvide que la lista anterior era sólo para las universidades.

8. QUIENES OFRECEN SERVICIOS DE ACCESO A INTERNET

A continuación veremos algunas de las empresas que permiten tener acceso a **Internet.** Estas son de las que tenemos más referencias comerciales, hasta el momento: Orbinet (Sprint), Saitel (Telecom), Cetcol

8.1. Orbinet (Sprint)

Esta compañía presta entre sus múltiples servicios (telecomunicaciones,

transmisión de datos, consulta a bases de datos internacionales, a través de la red Sprintnet International), el acceso a **Internet.**

Todos los servicios anteriores los presta gracias a que Orbinet opera como nodo y centro de acceso a la red Sprint, desde Colombia.

Un usuario afiliado a esta red tiene acceso desde Bogotá, a la red mundial Sprint, mediante una llamada telefónica local o una de larga distancia, para los que estén fuera de Bogotá.

Conectarse a **Internet,** con esta empresa, es muy sencillo, ya que al afiliarse se le asignará su vía de acceso, la cual podrá utilizar a través de la línea telefónica convencional.

8.2. Saitel (Telecom)

Saitel es un servicio que está prestando Telecom para tener acceso a Internet.

Existen tres formas de acceso a Internet, con Saitel:

A través de Coldapaq.

RTPC-Red Telefónica Pública Conmutada.

Canal Dedicado.

Al igual que Orbinet, Saitel presta el servicio de acceso a **Internet,** de tal forma que el usuario final sólo tendrá que preocuparse por mantener al día su Modem y su teléfono, para poder conectarse.

Las dos empresas trabajan con el sistema de suscripción, tarifas por consumo o tarifas mensuales.

8.3. Cetcol

Cetcol surge de un convenio suscrito entre Colciencias e ICFES, en el que pretenden aunar esfuerzos para gestionar la implantación y el funcionamiento de una infraestructura de comunicaciones de datos, que sea común al Sistema Nacional de Información Científica y Tecnológica, al Sistema Nacional de Información de la Educación Superior y a otros sistemas nacionales de información que puedan utilizar.

Mientras se organiza la corporación de usuarios que manejará la red, las corporaciones que han suscrito el convenio están financiando y controlando la conexión de la red.

Al estudiar las conexiones posibles con **Internet** se verán con más detalle las posibilidades de conexión con Cetcol.

9. EJEMPLOS DE TIPOS DE USUARIOS Y TIPO DE INFORMACION

Si no está del todo convencido de lo que hemos hablado hasta el momento, le presentamos a continuación algunas impresiones de pantallas que fueron hechas, mientras disfrutábamos de un pequeño viaje, sin movernos de la oficina del profesor Alvaro Pachón y sin un costo mayor del que se puede tener al hacer una llamada telefónica, desde el ICESI hasta Univalle.

Lo que verá enseguida es una muestra minúscula de lo que podrá encontrar en **Internet;** por eso nuestra propuesta es: ¡Convénzase usted mismo; venga y vea!

COMO SE DIO INICIO A ESTA SESION

```
atx0
OK
atdp0w3301249
CONNECT

 _______...
| |...
|-------      /\
.\\  /.\ /\ /  \Contacto: ahoyer@univalle.edu.co
...--\/.. V V   \        +57 2 330 15 76
.....V...        \
                  \

-Red de UniValle-Farallones
Farallones>

Farallones>telnet mafalda
Translating "MAFALDA" ...domain server (157.253.103.1) [OK]
Trying MAFALDA.UNIVALLE.EDU.CO (157.253.103.1)... Open

SunOS UNIX
(mafalda.univalle.edu.co) (ttypd)

login: varela
Password:
Last login: Wed Oct 5 21:35:58 from cs-tel.univalle.
SunOS Release 4.1.3. (X25-IP-TFTP-MCAST)#1: Sun Aug 7 19:53:21 EDT 1994
You have mail.
Terminal recognized as vt100(ANSI/VT100)

Printer usage for varela (Oct94):
Printer pages_printed quota
laser      0 10

Terminal set to 'vt100'.
[41]mafalda-{home2/pino/varela}%
```

MENU DE RECEPCION EN UNIVALLE

Internet Gopher Information Client v2.0.16

Home Gopher server:gopher.univalle.edu.co

1. Búsqueda dentro del Gopher <?>
2. Mensaje de bienvenida
3. → Universidad del Valle - general/
4. Directorio Telefónico <CSO>
5. Información sobre el Gopher/
6. La red de UniValle/
7. Gophers de UniValle/
8. Gophers nacionales/
9. Gophers internacionales/
10. Bibliotecas/
11. Direcciones electrónicas de interés general/
12. Internet/
13. Preguntas más frecuentes sobre.../
14. Pruebas/

MIRANDO LA INFORMACION GENERAL

Internet Gopher Information Client v2.0.16

Home Gopher server: gopher.univalle.edu.co

1. Búsqueda dentro del Gopher <?>
2. Mensaje de bienvenida
3. → **Universidad del Valle - general/**
4. Directorio Telefónico <CSO>
5. Información sobre el Gopher/
6. La red de UniValle/
7. Gophers de UniValle/
8. Gophers nacionales/
9. Gophers internacionales/
10. Bibliotecas/
11. Direcciones electrónicas de interés general/
12. Internet/
13. Preguntas más frecuentes sobre.../
14. Pruebas/

Internet Gopher Information Client v2.0.16

Universidad del Valle - general

1. → Servicios de información/
2. Trucos para PC
3. Eventos, conferencias, cursos .../
4. 3er. Encuentro Nacional de Estudiantes de Ing. Eléctrica y Electro.../
5. Publicaciones/

SOBRE LA RED DE UNIVALLE

Internet Gopher Information Client v2.0.16

Home Gopher server: gopher.univalle.edu.co

1. Búsqueda dentro del Gopher <?>
2. Mensaje de bienvenida
3. Universidad del Valle - general/
4. Directorio Telefónico <CSO>
5. Información sobre el Gopher/
→ **6. La red de UniValle/**
7. Gophers de UniValle/
8. Gophers nacionales/
9. Gophers internacionales/
10. Bibliotecas/
11. Direcciones electrónicas de interés general/
12. Internet/
13. Preguntas más frecuentes sobre.../
14. Pruebas/

Internet Gopher Information Client v2.0.16

La red de UniValle

→ 1. Bienvenido a la Red Institucional de Transmisión de Datos
2. Descripción de los grupos de NEWS de Colombia
3. Grupos de NEWS de Internet
4. Reglas de uso de las cuentas en la red
5. Especificaciones/
6. Modems/
7. NOTIRED/
8. Lista de máquinas de UniValle
9. Lista de usuarios de Mafalda
10. Usuarios de UniValle conectados

Internet Gopher Information Client v2.0.16

La red de UniValle

1. Bienvenido a la Red Institucional de Transmisión de Datos
2. Descripción de los grupos de NEWS de Colombia
3. Grupos de NEWS de Internet
4. Reglas de uso de las cuentas en la red
→ **5. Especificaciones/**
6. Modems/
7. NOTIRED/
8. Lista de máquinas de UniValle
9. Lista de usuarios de Mafalda
10. Usuarios de UniValle conectados

Internet Gopher Information Client v2.0.16

Especificaciones

→ 1. Cableado.txt
2. Enrutadores
3. Fibra óptica-especif.txt
4. Fibra Optica.ps
5. Memo UTP Facomec.txt
6. Memo UTP.ps
7. Tarjetas

SOBRE INTERNET

Internet Gopher Information Client v2.0.16

Home Gopher server: gopher.univalle.edu.co

1. Búsqueda dentro del Gopher <?>
2. Mensaje de bienvenida
3. Universidad del Valle - general/
4. Directorio Telefónico <CSO>
5. Información sobre el Gopher/
6. La red de UniValle/
7. Gophers de UniValle/
8. Gophers nacionales/
9. Gophers internacionales/
10. Bibliotecas/
11. Direcciones electrónicas de interés general/
12. → Internet/
13. Preguntas más frecuentes sobre.../
14. Pruebas/

Internet Gopher Information Client v2.0.16

Internet

1. → Introducción/
2. Correo/
3. Cultura de Red/
4. Especificaciones/
5. Internet List
6. Resources Guides/
7. Sitios FTP/
8. Dominios/
9. Guide to Internet
10. IRC List
11. Internet Special Issue on Newsstands

SALIMOS DE UNIVALLE AL EXTERIOR

Internet Gopher Information Client v2.0.16

Home Gopher server: gopher.univalle.edu.co

1. Búsqueda dentro del Gopher <?>
2. Mensaje de bienvenida
3. Universidad del Valle - general/
4. Directorio Telefónico <CSO>
5. Información sobre el Gopher/
6. La red de UniValle/
7. Gophers de UniValle/
8. Gophers nacionales/
9. → **Gophers internacionales/**
10. Bibliotecas/
11. Direcciones electrónicas de interés general/
12. Internet/
13. Preguntas más frecuentes sobre.../
14. Pruebas/

Internet Gopher Information Client v.2.0.16

Servidores de Gopher

1. → International Organizations/
2. Africa/
3. América Latina/
4. América del Norte/
5. Asia/
6. Europa/
7. Medio Este/
8. Pacífico del Sur/
9. University of Minnesota (Gopher Home)/
10. Todos los Gophers del mundo/

HACIA AMERICA LATINA

Internet Gopher Information Client v.2.0.16

Servidores de Gopher

1. International Organizations/
2. Africa/
3. **América Latina/**
4. América del Norte/
5. Asia/
6. Europa/
7. Medio Este/
8. Pacífico del Sur/
9. University of Minnesota (Gopher Home)/
10. Todos los Gophers del mundo/

Internet Gopher Information Client v2.0.16

América Latina

1. Argentina/
2. Argentina's top-level domain/
3. BBRC - Brazilian Bioinformatics Resource Center/
4. Base de Datos Tropical (Tropical Data Base), Campinas, Brasil/
5. CHILE - Red Universitaria Nacional (REUNA)/
6. CONICYT - CHILE/
7. Cuyonet Gopher Server/
8. ECUADOR - EcuaNet/
9. ESPOL - Escuela Politécnica del Litoral, Guayaquil -Ecuador/
10. Ecuador Gophers/
11. Ecuador Ministerio de Relaciones Exteriores/
12. Escuela Politécnica Nacional (Ecuador-South America)/
13. Facultad de Ciencias Astronómicas y Geofísicas/
14. Gopher CONICYT-VENEZUELA/
15. Gopher RED CIENTIFICA PERUANA-PERU/
16. Gopher Server Federico Santa María University, Chile/
17. Gopher de Universidad Nacional de Ingeniería./
18. Gopher de la Universidad de Carabobo (Venezuela)

Internet Gopher Information Client v2.0.16

América Latina

19. Instituto Venezolano de Investigaciones Científicas (IVIC)/
20. Municipality of Recife, Pernambuco, Brazil/
21. Pontificia Universidad Católica de Chile/
22. Pontificia Universidad Católica del Ecuador/
23. Pontificia Universidad Católica del Perú/
24. Pontificia Universidad Católica do Rio
25. RNP - Rede Nacional de Pesquisa, Brazil/
26. Red Académica Uruguaya (RAU) - Uruguay/
27. Universidad Interamericana en Ponce Gopher/
28. Universidad San Francisco de Quito/
29. Universidad San Francisco de Quito (Ecuador-Sud América)/
30. Universidad Simón Bolívar Laboratorio de Computación/
31. Universidad Técnica Federico Santa María/
32. Universidad de Buenos Aires (Argentina)/
33. Universidad de Chile (Departamento de Ciencias de la Computación)/
34. Universidad de Concepción Gopher Server (Chile)/
35. Universidad de Lima - Perú/
36. Universidad de los Andes

EN LA CATOLICA DE CHILE

Internet Gopher Information Client v2.0.16
América Latina

19. Instituto Venezolano de Investigaciones Cientificas (IVIC)/
20. Municipality of Recife, Pernambuco, Brazil/
→ **21. Pontificia Universidad Católica de Chile/**
22. Pontificia Universidad Católica del Ecuador/
23. Pontificia Universidad Católica del Perú/
24. Pontificia Universidad Católica do Rio
25. RNP - Rede Nacional de Pesquisa, Brazil/
26. Red Académica Uruguaya (RAU) - Uruguay/
27. Universidad Interamericana en Ponce Gopher/
28. Universidad San Francisco de Quito/
29. Universidad San Francisco de Quito (Ecuador-Sud América)/
30. Universidad Simón Bolívar Laboratorio de Computación/
31. Universidad Técnica Federico Santa María/
32. Universidad de Buenos Aires (Argentina)/
33. Universidad de Chile (Departamento de Ciencias de la Computación)/
34. Universidad de Concepción Gopher Server (Chile)/
35. Universidad de Lima - Perú/
36. Universidad de Los Andes

Internet Gopher Information Client v2.0.16
Pontificia Universidad Católica de Chile

→ 1. Bienvenidos a la Pontificia Universidad Católica de Chile (PUC)
2. Informaciones de Gopher - About Gopher/
3. PUC - Admisión/
4. PUC - Carreras de Postgrado
5. PUC - Carreras de Postítulo
6. PUC - Misceláneos/
7. PUC - Pontificia Universidad Católica de Chile/
8. PUC - Publicaciones de la Universidad/
9. PUC - Servicios Internet/
10. FTP Anónimo/
11. News <tolten>/
12. Otros servicios Internet/
13. Otros servidores de Gopher en el mundo/
14. Requests for Comments (RFC)/
15. Verónica (búsqueda de menús en el Gopher-espacio)/
16. ENCUESTA GOPHER PUC <??>

FTP ANONIMO DESDE LA CATOLICA DE CHILE

Internet Gopher Information Client v2.0.16
Pontificia Universidad Católica de Chile

1. Bienvenidos a la Pontificia Universidad Católica de Chile (PUC)
2. Informaciones de Gopher - About Gopher/
3. PUC - Admisión/
4. PUC - Carreras de Postgrado
5. PUC - Carreras de Postítulo
6. PUC - Misceláneos/
7. PUC - Pontificia Universidad Católica de Chile/
8. PUC - Publicaciones de la Universidad/
9. PUC - Servicios Internet/
10. **FTP Anónimo/**
11. News <tolten>/
12. Otros servicios Internet/
13. Otros servidores de Gopher en el mundo/
14. Requests for Comments (RFC)/
15. Verónica (búsqueda de menús en el Gopher-espacio)/
16. Encuesta Gopher PUC <??>

Internet Gopher Information Client v2.0.16
FTP Anónimo

→ 1. Búsqueda de archivos <Archie>/
2. Servidor de FTP, anónimo a consultar <?>
3. Servidores PUC/
4. Servidores Populares/

SERVIDORES PUC EN LA CATOLICA

Internet Gopher Information Client v2.0.16

FTP Anónimo

1. Búsqueda de archivos <Archie>/
2. Servidor de FTP, anónimo a consultar <?>
3. → **Servidores PUC/**
4. Servidores Populares/

Internet Gopher Information Client v2.0.16

Servidores PUC

1. → Departamento de Matemáticas PUC/
2. Servidor de Archivos SECICO/
3. Servidor de Archivos de la Escuela de Ingeniería/

VEAMOS ALGO MAS

Internet Gopher Information Client v2.0.16

Servidores PUC

1. Departamento de Matemáticas PUC/
2. → **Servidor de Archivos SECICO/**
3. Servidor de Archivos de la Escuela de Ingeniería/

Internet Gopher Information Client v2.0.16
Servidor de Archivos SECICO

1. aux/
2. bbs/
3. dos/
4. gopher/
5. mac/
6. security/
7. src/
8. tmp/
9. → **unix/**
10. www/

Internet Gopher Information Client v2.0.16

unix

1. bind/
2. games/
3. gzip-1.1.1.tar.Z. <Bin>
4. → **net/**
5. patch.tar.gz.
6. patch2/

Internet Gopher Information Client v2.0.16

net

1. archie-1.3.2.tar.z <Bin>
2. → gated/
3. inn.tar.gz
4. ncftp 164.tar.gz
5. ntp3.tar.gz
6. tcpdump-2-2.1.tar.gz.
7. tfpd.tar.gz.
8. tin-aix.gz.
9. wu-ftpd-2.1a.tar.gz

VOLVEMOS AL PAIS

Internet Gopher Information Client v2.0.16

Home Gopher server: gopher.univalle.edu.co

1. Búsqueda dentro del Gopher <?>
2. Mensaje de bienvenida
3. Universidad del Valle - general/
4. Directorio Telefónico <CSO>
5. Información sobre el Gopher/
6. La red de UniValle/
7. Gophers de UniValle/
→ **8. Gophers nacionales/**
9. Gophers internacionales/
10. Bibliotecas/
11. Direcciones electrónicas de interés general/
12. Internet/
13. Preguntas más frecuentes sobre.../
14. Pruebas/

Internet Gopher Information Client v2.0.16

Gophers Nacionales

→ 1. Universidad de los Andes/
2. EAFIT/
3. Colciencias/
4. Universidad del Cauca/

AHORA EN UNIANDES

Internet Gopher Information Client v.2.0.16

Gophers Nacionales

→ **1. Universidad de los Andes/**
2. EAFIT/
3. Colciencias/
4. Universidad del Cauca/

Internet Gopher Information Client v.2.0.16

Universidad de los Andes

→ 1. Bienvenido al Gopher de la Universidad de los Andes
2. Información general sobre la Universidad de los Andes/
3. Directorio Telefónico de la Universidad <CSO>
4. Eventos programados en la universidad/
5. Noticias y Anuncios/
6. _____
7. RDUA - Red de Datos de la Universidad de los Andes/
8. _____
9. Información sobre Colombia/
10. NOTICOL: Noticias de Colombia/
11. _____
12. Información sobre Internet/
13. Información sobre CETCOL/
14. _____
15. Acceso a otros servicios nacionales/
16. Acceso a otros servicios internacionales/
17 _____
18. Varios/

VEAMOS LA BIENVENIDA

Internet Gopher Information Client v.2.0.16

Universidad de los Andes

→ 1. Bienvenido al Gopher de la Universidad de los Andes
2. Información general sobre la Universidad de los Andes/
3. Directorio Telefónico de la Universidad <CSO>
4. Eventos programados en la universidad/
5. Noticias y Anuncios/
6.
7. RDUA - Red de Datos de la Universidad de los Andes/
8.
9. Información sobre Colombia/
10. NOTICOL: Noticias de Colombia/
11.
12. Información sobre Internet/
13. Información sobre CETCOL/
14.
15. Acceso a otros servicios nacionales/
16. Acceso a otros servicios internacionales/
17.
18. Varios/

Bienvenido al Gopher de la Universidad de los Andes (OK) 95%

+ ——————————————————————— +

Bienvenido al Servidor de GOPHER
de la Universidad de los Andes

Estamos haciendo un esfuerzo permanente para colocar a su disposición información que permita difundir las características de nuestra institución, sus desarrollos y sus logros.

Adicionalmente, de manera paulatina pero continuada, se coloca información diversa, actualizada y de interés general, respecto a nuestro maravilloso país.

A través de este medio, también deseamos colocar a su alcance otros servicios de la Red de Datos de la Universidad de los Andes (RDUA), servicios nacionales y servicios internacionales.

Le invitamos a que continúe explorando, mediante la selección de cualquiera de las opciones que se presentan en este espacio.

EAFIT ES LA SIGUIENTE

Internet Gopher Information Client v2.0.16

Gophers Nacionales

1. Universidad de los Andes/
→ 2. EAFIT/
3. Colciencias/
4. Universidad del Cauca/

Internet Gopher Information Client v2.0.16
EAFIT

→ 1. Mensaje de bienvenida Universidad EAFIT
2. Bibliotecas/
3. Conexión otras máquinas Universidad EAFIT/
4. Directorio Telefónico Universidad EAFIT <CSO>
5. Documentos Varios/
6. Información Académica Estudiantes/
7. Información Otras Dependencias/
8. Información Universidad EAFIT/
9. Pruebas/
10. Servicios Externos/

EL MENU DE ACCESO A LA BIBLIOTECA DE EAFIT

```
+================================================== +
UNIVERSIDAD SISTEMA DE INFORMACION BIBLIOGRAFICA SINBAD
EAFIT                                        sbf_60000_01
                    MENU CONSULTAS
+==================================================

          (1)   Por tema
          (2)   Por Autor
          (3)   Por título
          (4)   Tesauro

          Su Opción: (1)



                    <F2>: TERMINAR.
DIGITE SU OPCION Y PRESIONE <ENTER> PARA CONTINUAR.
```


EN COLCIENCIAS

Internet Gopher Information Client v2.0.16

Gophers Nacionales

1. Universidad de los Andes/
2. EAFIT/
3. → Colciencias/
4. Universidad del Cauca/

Internet Gopher Information Client v2.0.16

Colciencias

1. Búsqueda dentro del Gopher <?>
2. A. Mensaje de bienvenida
3. B. Guía Institucional de Colciencias/
4. → C. Qué es el Sistema Nacional de Ciencia y Tecnología/
5. E. Actividades de ciencia y tecnología financiadas en los Programa../
6. I. Becas otorgadas en las convocatorias de formación/
7. J. Pasantías y cursos financiados/
8. K. Actividades impulsadas a través de la Red Caldas/
9. Q. Convocatorias a la comunidad científica/
10. R. Cartas de Colciencias/
11. S. Comunicados emitidos por Colciencias/
12. T. Sugerencias para presentar propuestas/
13. U. Acerca de la Red Internet/
14. V. Biblioteca electrónica/
15. X. Preguntas más frecuentes sobre.../
16. Z. Otros Gophers/

PASAMOS A LA UNIVERSIDAD DEL CAUCA

Internet Gopher Information Client v2.0.16

Gophers Nacionales

1. Universidad de los Andes/
2. EAFIT/
3. Colciencias/
→ 4. Universidad del Cauca/

Internet Gopher Information Client v2.0.16

Universidad del Cauca

→ 1. Bienvenido al Gopher de UniCauca
2. Información general de la Universidad del Cauca/
3. Red de Datos UniCauca
4. Red CETCOL
5. Información sobre el Gopher/
6. Directorio Telefónico de UniCauca <CSO>
7. Gophers de UniCauca/
8. Gophers nacionales/
9. Gophers internacionales/
10. Bibliotecas/
11. Información sobre Bitnet/
12. Información sobre Internet/
13. Información sobre telecomunicaciones/
14. Archivo histórico
15. Revista Pulsos FIET
16. Información sobre vías y transporte
17. Preguntas/

VEAMOS LAS ORGANIZACIONES INTERNACIONALES

Internet Gopher Information Client v2.0.16

Servidores de Gopher

→ 1. International Organizations/
2. Africa/
3. América Latina/
4. América del Norte/
5. Asia/
6. Europa/
7. Medio Este/
8. Pacífico del Sur/
9. University of Minnesota (Gopher Home)/
10. Todos los Gophers del Mundo/

Internet Gopher Information Client v2.0.16

International Organizations

19. International Telecommunication Union (ITU)/
20. Internet Engineering Tasks Force (IETF)/
21. Internet Society (ISOC)/
22. Internet Users Group International Gopher/
23. NORDINFO - The Nordic Council for Scientific Information/
24. North Atlantic Treaty Organization (NATO)/
25. Open Source Solutions, Inc. --OSSgopher/
26. PanAmerican Society for Pigment Cell Research (PASPCR)/
27. SHARE, Inc./
28. Schlumberger/
29. Share International/
30. The Global Cycling Network (VeloNet)/
31. Together Foundation Gopher/
→ 32. United Nations/
33. World Bank/
34. World Health Organization (WHO)/

ORGANIZACIONES INTERNACIONALES

Internet Gopher Information Client v2.0.16

International Organizations

19. International Telecommunication Union (ITU)/
20. Internet Engineering Task Force (IETF)/
21. Internet Society (ISOC)/
22. Internet Users Group International Gopher/
23. NORDINFO - The Nordic Council for Scientific Information/
24. North Atlantic Treaty Organization (NATO)/
25. Open Source Solutions, Inc. -- OSSgopher/
26. PanAmerican Society for Pigment Cell Research (PASPCR)/
27. SHARE, Inc./
28. Schlumberger/
29. Share International/
30. The Global Cycling Network (VeloNet)/
31. Together Foundation Gopher/
→ 32. United Nations/
33. World Bank/
34. World Health Organization (WHO)/

Internet Gopher Information Client v.2.0.16

United Nations

1. The United Nations, what it is and what it does/
2. United Nations Current Information (Highlights, Press Releases, et../
→ 3. United Nations Documents (General Assembly, Sec. Council)/
4. United Nations Conferences/
5. United Nations Economic and Social Council (ECOSOC)/
6. United Nations System Directories/
7. United Nations System Telecommunications Catalogue/
8. United Nations Development Programme (UNDP) Documents/
9. Other United Nations & related Gophers/
10. Environment Related Information/
11. Other Gopher & Information Servers/
12. Access to External Public Databases/
13. United Nations 50th Anniversary/
14. Announcements - What's new/

VEAMOS LOS DOCUMENTOS

Internet Gopher Information Client v.2.0.16

United Nations

1. The United Nations, what it is and what it does/
2. United Nations Current Information (Highlights, Press Releases, et../
→ 3. United Nations Documents (General Assembly, Sec. Council)/
4. United Nations Conferences/
5. United Nations Economic and Social Council (ECOSOC)/
6. United Nations System Directories/
7. United Nations System Telecommunications Catalogue/
8. United Nations Development Programme (UNDP) Documents/
9. Other United Nations & related Gophers/
10. Environment Related Information/
11. Other Gopher & Information Servers/
12. Access to External Public Databases/
13. United Nations 50th Anniversary/
14. Announcements - What's new/

Internet Gopher Information Client v2.0.16

United Nations Documents (General Assembly, Sec. Council)

→ 1. A: General Assembly Documents/
2. E: ECOSOC Documents/
3. Economic and Social Council (ECOSOC) Resolutions/
4. General Assembly - Conventions/
5. General Assembly Resolutions/
6. General Assembly Special Sessions Resolutions/
7. Security Council Resolutions/

DE DONDE VAMOS A TOMAR UN DOCUMENTO

Internet Gopher Information Client v2.0.16

A: General Assembly Documents

1. C.5: Fifth Committee/
2. CONF.164/
3. CONF. 166: World Summit for Social Development/
4. CONF. 167: Global Conf. on the Sustainable Dev. of Small Island/
5. CONF. 171: International Conference on Population and Development/
6. CONF. 172: World Conference on Natural Disaster Reduction/
7. General Assembly, 48th session/
8. General Assembly, 49th session/
9. INF/

Internet Gopher Information Client v2.0.16

General Assembly, 49th session

→ 1. 94-07: July 94/
2. 94-08: August 94/
3. 94-09: September 94/
4. 94-10: October 94/

Economic and Social Council at its substantive session of 1994,

Internet Gopher Information Client v2.0.16

94-10: October 94

→ 1. 345:A

VEAMOS ALGO SOBRE LA ULTIMA RESOLUCION DE LAS NACIONES UNIDAS.
(APARTES TABLA DE CONTENIDO)

45: A (46k) 0%

+----------+

United
Nations

A General Assembly

Distr.
General
A/49/345
29 August 1994
Original: English

Forty-ninth session
Item 101 of the provisional agenda*

INTERNATIONAL DRUG CONTROL

Implementation of the Global Programme of Action Adopted by the General Assembly at its seventeenth special session

Report of the Secretary-General

CONTENTS

* A/49/150.

94-34597 (E)
280994

/...

CONTENS (continued)

ALGO SOBRE LA INTRODUCCIÓN

1. The General Assembly, at its seventeenth special session, adopted a Global Programme of Action (resolution S-17/2, annex) on international cooperation against illicit production, supply, demand, trafficking and distribution of narcotic drugs and psychotropic substances, setting out a comprehensive list of measures and activities to be undertaken by States and United Nations, entities, collectively and simultaneously, in the fight against all aspects of drug abuse and illicit traffic. A United Nations Decade against Drug Abuse, covering the period from 1991 to 2000, is devoted to effective and sustained national, regional and international action to promote the implementation of the Global Programme of Action.

2. In paragraph 97 of the Global Programme of Action, it is stated that the Commission on Narcotic Drugs and the United Nations drug control bodies should continuously monitor progress on the implementation of the Global Programme of Action, and that the Secretary-General should report annually to the General Assembly on all activities relating to the Global Programme of Action and the efforts of Governments.

3. The General Assembly, in various resolutions since 1990, has called upon States to take all possible stepts to promote and implement the mandates contained in the Global Programme of Action, and requested the Commission on Narcotic Drugs and the United Nations International Drug Control Programme (UNDCP) to promote and continuously monitor the progress on its implementation. The Assembly has also requested the Secretary-General to report annually to it on all activities relating to the Global Programme of Action, including those of Governments. The previous report of the Secretary-General (A/48/286) related to activities through the first quarter of 1993; the present report analyses implementation during the last three quarters of 1993 and the first six months of 1994.

4. The Commission on Narcotic Drugs, at its thirty-seventh session, examined the implementation of the Global Programme of Action and adopted resolution 4 (XXXVII) of 20 April 1994 on monitoring of the implementation of the Global Programe of Action. In that resolution, the Commission requested the Secretary -General to include the following in future reports: (a) an introductory section containing an evaluation of progress in implementating the Global Programme of Action: (b) a summary of activities undertaken by States, the competent organs of the United Nations system and the specialized agencies in promoting and implementing the Global Programme of Action; and (c) an identification of the specific aspects of each section of the Global Programme of Action deemed by the Secretary-General to require greater attention by States, with a view to promoting their implementation. By that resolution, the Commission also authorized the use of a simplified questionnaire to be sent to Governments at the beginning of each year requesting information on activities undertaken. Accordingly, the report reflecting the new outline, will be before the General Assembly at its fiftieth session.

ALGO DE MEDICINA EN USA

Internet Gopher Information Client v2.0.16

Medicine/Medical Science Information and Resources

1. Medical Science Related Gopher Sites/
2. Medical Science Related Discussion Groups/
3. Medical Science Related Databases/
4. Credits
5. Directory of Scholarly E-conferences (ACADLIST)/
6. Health Sciences Resources
7. List of Academic Mailing Lists/
8. Medical Resources on the Internet
9. Morbidity and Mortality Weekly Report
10. Ontario Prevention Clearinghouse
11. U.S. Center for Biomedical Informatics
12. U.S. Library of Congress
13. U.S. National Institute of Health EDNET

→ 14. U.S. National Library of Medicine

U.S. National Library of Medicine (Ok) 100%

+ ---------- +

National Library of Medicine Educational Technology Network

A host of forums concerning medical education and the use of multimedia in medical education.

Telnet: etnet, nlm,nih.gov(130.14.1.27)

Login: etnet

CONCLUSIONES

Hemos observado, primero, que es conveniente mantener un orden lógico que permita ubicar a los lectores en el contexto que queremos manejar.

Al realizar este trabajo, podemos decir que nosotros mismos nos hemos llegado a convencer, con criterios, sobre las ventajas que ofrece el trabajar con Internet en las organizaciones actuales y en especial en nuestra universidad.

BIBLIOGRAFIA

- *Internetworking with TCP/IP Principles, Protocols and Architecture.* Douglas Comer. Prentice Hall. New Jersey, 1988.
- *Servicios Orientados a Conexion TCP/IP Bajo X 25.* Edgar García S. Minga Informática ACUC. Santafé de Bogotá, 1994.
- *Zen and the Art of the Internet.* Brendan P. Kehoe. January, 1992.
- *The Internet Worm.* American Scientist. March - April, 1989.
- *FYI on Introducing the Internet.* A short Bibliography of Introductory Internetworking. Reading for Network Navice.
- *Recent Internet Books.* Quaterman, J. 1993.
- *Internet Access for Everyone Isn't it time?* Thomas J. Cozzolino.
- *An Education's Guide to E-mail lists.* Prescott Smith.

SISTEMAS OPERATIVOS DISTRIBUIDOS

JOSE FERNANDO BASTIDAS C.
HERBERT CARRILLO G.
DAVID EDUARDO CIFUENTES C.
MARIA FERNANDA SERRANO B.
ANDRES VARGAS CABAS

Alumnos del curso de Investigación de VIII Semestre de Ingeniería de Sistemas del ICESI

OBJETIVOS

- Dar un concepto claro de lo que es un Sistema Distribuido, tocando algunos temas técnicos importantes, pero explicándolos de manera que lleguen fácilmente a las personas.
- Dar a conocer la importancia de apoyar la investigación en este campo, que apenas está surgiendo, pero que promete mucho para el desarrollo tecnológico mundial.
- Conocer la manera como cooperan y se sincronizan los procesos entre sí, en un Sistema Distribuido, así como la forma de implantar regiones críticas o asignar recursos.
- Conocer las características y componentes de un Sistema Distribuido de archivos.
- Exponer los puntos fundamentales en la implantación de un Sistema Distribuido de archivos.
- Lograr captar el interés de todas aquellas personas interesadas en consultar este libro, específicamente de quienes se encuentran de una u otra forma vinculados con el análisis y desarrollo del software, para que empiecen a interesarse en la creación de aplicaciones enfocadas hacia la distribución, las cuales deben conducir a que, en un futuro no muy lejano, podamos hablar de comunicaciones a nivel mundial, realmente confiables, transparentes y eficientes.

INTRODUCCION

Desde hace algunos años el uso de los computadores ha cambiado enormemente, debido a la transición, desde los sistemas centralizados que constan de un único CPU, sus periféricos de memoria y algunas terminales, hacia sistemas de cómputo compuestos por un gran número de CPUs, conectados mediante una red de alta velocidad, conocidos como *Sistemas Distribuidos.* Aunque los sistemas operativos necesarios

para estos *Sistemas Distribuidos* están apenas en etapa de surgimiento, ya es bastante lo que conocemos acerca de ellos.

Nuestro objetivo es presentar a ustedes, de manera clara y sencilla, los conceptos básicos acerca de los *Sistemas Distribuidos,* sus principales características, las ventajas y desventajas, el manejo de la comunicación y la sincronización dentro de estos sistemas, los sistemas distribuidos de archivos y un estudio de los procesos y procesadores de los *Sistemas Distribuidos.*

Esperamos que el lector comprenda la filosofía de los *Sistemas Distribuidos,* después de haber leído nuestra obra, y pueda llenar, si no todas, muchas de las expectativas que tenga acerca de este novedoso campo de los sistemas de computación.

1. INTRODUCCION A LOS SISTEMAS DISTRIBUIDOS

1.1. Historia

El uso de los computadores está a punto de sufrir una revolución. Desde 1945, cuando comenzó la era del computador moderno, hasta 1985, los computadores eran grandes y caros, incluyendo los minicomputadores. Como resultado, las organizaciones tenían un grupo pequeño de computadores que operaban independientemente, por carecer de una forma para conectarlos.

A partir de 1985 comenzó a cambiar esta situación. Primero, con el desarrollo de poderosos microprocesadores, se disponía de máquinas de 8 bits, pero pronto se volvieron comunes las CPU de 16, 32 e incluso 64 bits, muchas de las cuales tenían el poder de cómputo de un "mainframe" respetable; y segundo, con la invención de redes locales de alta velocidad (LAN), las cuales permitieron conectar docenas e incluso cientos de máquinas, de tal forma que se pudiesen transferir pequeñas cantidades de información, entre ellas, durante un milisegundo o un tiempo parecido. La mayor cantidad de datos se puede desplazar, entre las máquinas, a razón de 10 millones de bits/segundo y aun más.

Hoy en día es fácil reunir sistemas de cómputo, compuestos por un gran número de CPUs, conectados mediante una red de alta velocidad, los cuales reciben el nombre genérico de *Sistemas Distribuidos,* en contraste con los Sistemas Centralizados que constan de un único CPU, sus periféricos de memoria y algunas terminales.

Los Sistemas Distribuidos necesitan un software radicalmente distinto del de los sistemas centralizados. En particular, aunque los sistemas operativos, necesarios para estos Sistemas Distribuidos, están apenas en una etapa de surgimiento, ya se sabe bastante acerca de ellos.

1.2. Ventajas y desventajas de los Sistemas Distribuidos

1.2.1. *Ventajas de los Sistemas Distribuidos respecto de los Centralizados*

La fuerza real, detrás de la tendencia hacia la descentralización, es la economía. Por unos cuantos cientos de dólares es posible comprar un CHIP de CPU, que puede ejecutar más instrucciones, por segundo, de las que realizaba uno de los más grandes "mainframes" de los años ochenta. Si uno está dispuesto a pagar el doble, se obtiene el mismo CPU, sólo que con una velocidad un poco mayor. Como resultado, la solución más eficaz, en cuanto a costos, es limitarse a un gran número de CPUs baratos, reunidos en un mismo sistema. Así, la razón número uno de la tendencia hacia los Sistemas Distribuidos es que éstos tienen en potencia una proporción precio/desempeño mucho mejor que la de un único sistema centralizado.

Una ligera variación, con respeto a este tema, es la observación de que una

colección de microprocesadores no sólo facilita una mejor proporción precio/desempeño que un único "mainframe", sino que puede producir un mejor rendimiento del que podría proporcionar cualquier "mainframe" a cualquier precio. Por ejemplo, con la tecnología actual es posible construir un sistema a partir de los 1.000 chips de un computador moderno, cada uno de los cuales tiene una ejecución de 20 MIPS (millones de instrucciones por segundo), para un rendimiento total de 20.000 MIPS. Para que un único procesador (es decir, CPU) logre esto, tendría que ejecutar una instrucción en 0,05 nanosegundos (50 picosegundos). Ninguna máquina existente llega a acercarse a esta cifra, además de que consideraciones teóricas y de ingeniería lo consideran improbable durante algún tiempo. Así, si el objetivo es un rendimiento normal a bajo costo, o un alto rendimiento con un mayor costo, los Sistemas Distribuidos tienen mucho que ofrecer.

Otra razón para la construcción de un Sistema Distribuido es que ciertas aplicaciones son dispuestas en forma inherente. En un sistema de automatización de una fábrica, que controle los robots y las máquinas, a lo largo de una línea de ensamblaje, con frecuencia tiene sentido darle a cada robot o máquina su propio computador para que lo controle. Al conectarse éstos, se tiene un Sistema Distribuido Industrial.

Otra ventaja potencial de un Sistema Distribuido sobre uno Centralizado es una mayor confiabilidad. Al distribuir la carga de trabajo en muchas máquinas, la falla de un chip descompondrá, a lo sumo, una máquina, pero el resto seguirá intacto. Para el caso de aplicaciones críticas, como el control de los reactores nucleares o de los aviones, el uso de un sistema distribuido, para lograr una mayor confiabilidad, puede ser el factor dominante.

Por último, el crecimiento por incrementos también es una ventaja potencial. Con frecuencia ocurre que una compañía compra un "mainframe", con la intención de hacer todo su trabajo en él. Si la compañía prospera y la carga de trabajo aumenta, el "mainframe" no será adecuado en cierto momento. Las ideas de reemplazar el "mainframe" por otro más grande (si existe) o añadir un segundo "mainframe", pueden dar un tremendo castigo a las operaciones de la compañía. Por el contrario, con un sistema distribuido, podrían añadirse simplemente más procesadores al sistema, lo que permitiría un desarrollo gradual, conforme surjan las necesidades.

ECONOMIA	**Los microprocesadores ofrecen una mejor proporción precio-rendimiento que los mainframes.**
VELOCIDAD	**Un sistema distribuido puede tener un mayor poder de cómputo que un mainframe.**

DISTRIBUCION INHERENTE	**Algunas aplicaciones utilizan máquinas que están separadas una cierta distancia.**
CONFIABILIDAD	**Si una máquina se descompone, sobrevive el sistema como un todo.**
CRECIMIENTO POR INCREMENTOS	**Se puede añadir poder de cómputo en pequeños incrementos.**

Ventajas de los Sistemas Distribuidos con respecto de los Centralizados.

1.2.2. *Ventajas de los Sistemas Distribuidos respecto de los PC independientes.*

Muchos usuarios necesitan compartir ciertos datos. Por ejemplo, los empleados de reservaciones, en las líneas aéreas, necesitan tener acceso a la base maestra de datos de los vuelos y reservaciones existentes. Si se le diera a cada empleado una copia particular de toda la base de datos, eso no funcionaría, puesto que nadie conocería los asientos vendidos por los demás empleados. Los datos compartidos son absolutamente esenciales para ésta y otras aplicaciones, de modo que las máquinas deben estar conectadas entre sí. Los datos no son los únicos elementos que se pueden compartir; otros candidatos son también los periféricos caros, como las impresoras láser de color, los equipos de fotocomposición y los dispositivos de almacenamiento masivo (por ejemplo, las cajas ópticas).

Una tercera razón, para la conexión de un grupo de computadores aislados en un sistema distribuido, es lograr una mayor comunicación entre las personas, en donde, según la gente, el correo electrónico tiene numerosos atractivos respecto del correo con cartas, el teléfono o el fax.

Por último, un sistema distribuido tiene una mayor flexibilidad potencial que el hecho de darle a cada usuario un computador personal aislado. No sólo existe el método de conectar un grupo de computadores personales, mediante una LAN, sino también tener una mezcla de computadores personales y compartidos, tal vez con distintos tamaños, y dejar que los trabajos se ejecuten de la forma más adecuada, en vez de ejecutarlos siempre en el computador del propietario. De esta manera, la carga de trabajo se puede difundir entre los computadores, de forma más eficaz, y la pérdida de unas cuantas máquinas se puede compensar, si se permite a las personas que ejecuten sus trabajos en otra parte.

DATOS COMPARTIDOS	**Permite que varios usuarios tengan acceso a una base de datos común.**
DISPOSITIVOS COMPARTIDOS	**Permiten que varios usuarios compartan periféricos caros, como las impresoras de color.**
COMUNICACION	**Facilita la comunicación de persona a persona; por ejemplo, mediante el correo electrónico.**
FLEXIBILIDAD	**Difunde la carga de trabajo entre las máquinas disponibles, de la forma más eficaz, en cuanto a los costos.**

Ventajas de los Sistemas Distribuidos respecto de los PC

1.2.3. *Desventajas de los Sistemas Distribuidos*

El peor de los problemas de los sistemas distribuidos es el software, debido a que no se tiene mucha experiencia en el diseño, implantación y uso del software distribuido.

Un segundo problema potencial es el ocasionado por las redes de comunicación, ya que se pueden perder mensajes, por lo cual se requiere de un software especial para su manejo y éste puede verse sobrecargado. Al saturarse la red, ésta debe reemplazarse o añadirse una segunda. En ambos casos, hay que tender cables en uno o más edificios, con un gran costo; o bien, hay que reemplazar las tarjetas de interfaz de la red. Una vez que el sistema llega a depender de la red, la pérdida o saturación de ésta puede desvirtuar algunas de las ventajas que el sistema distribuido tenía.

Por último, la seguridad es, con frecuencia, un problema, debido a que si las personas pueden tener acceso a los datos en todo el sistema, entonces también pueden tener acceso a datos con los que no tienen que ver; por lo tanto, es preferible tener un computador personal aislado, sin conexiones de red con las demás máquinas, para que los datos se mantengan en secreto.

A pesar de estos problemas potenciales, se puede ver que las ventajas tienen mayor peso que las desventajas y, de hecho, es probable que en unos cuantos años muchas organizaciones conecten la mayoría de sus computadores a grandes sistemas distribuidos, para proporcionar un servicio mejor, más barato y más conveniente para sus usuarios.

SOFTWARE	**Existe poco software para los sistemas distribuidos en la actualidad.**
REDES	**La red se puede saturar o causar otros problemas.**
SEGURIDAD	**Un acceso sencillo también se aplica a datos secretos.**

Desventajas de los Sistemas Distribuidos.

1.3. Conceptos de hardware

Aunque todos los sistemas distribuidos constan de varios CPUs, existen diversas formas de organizar el hardware; en particular, en la forma de interconectarse y comunicarse entre sí. Se han propuesto diversos esquemas, pero ninguno de ellos ha tenido un éxito completo.

Es probable que la taxonomía más citada sea la de Flynn (1972), aunque es algo rudimentaria. Flynn eligió dos características, consideradas por él como esenciales: el número de flujos de instrucciones y el número de flujos de datos.

Un computador, con un solo flujo de instrucciones y un solo flujo de datos, se llama SISD ("Single Instruction Single Data"). Todos los computadores de un solo procesador caen dentro de esta categoría.

La siguiente categoría es SIMD ("Single Instruction Multiple Data"), con un flujo de instrucciones y varios flujos de datos. En este tipo, se ordenan los procesadores con una unidad que busca una instrucción y, después, ordena a varias unidades de datos para que la lleven a cabo, en paralelo, cada una con sus propios datos. Ciertos supercomputadores son SIMD.

La siguiente categoría es MISD (Multiple Instruction Single Data), con flujo de varias instrucciones y un solo flujo de datos. Ninguno de los computadores conocidos se ajusta a este modelo.

Por último viene MIMD (Multiple Instruction Multiple Data), que significa un grupo de computadores independiente, cada uno con su propio contador del programa, programas y datos. Todos los sistemas distribuidos son MIMD.

Los computadores MIMD se pueden dividir en dos grupos: aquellos que tienen memoria compartida, que, por lo general, se llaman ***Multiprocesadores*** y aquellos que no, que a veces reciben el nombre de ***Multicomputadores.*** La diferencia esencial es que un Multiprocesador tiene un solo espacio de direcciones virtuales, compartido por todos los CPU; y en un Multicomputador, cada máquina tiene su propia memoria particular.

Estas dos categorías se pueden subdividir, con base en la arquitectura de la red de interconexión, en ***Bus y con Conmutador.*** En la primera queremos indicar que existe una sola red, plano de base, bus, cable u otro medio que conecta todas las máquinas, como es el caso de la televisión comercial por cable; los sistemas con Conmutador no tienen una sola columna vertebral como la televisión por cable, sino que tienen cables individuales de una máquina a otra y utilizan varios patrones diferentes de cableado. Los mensajes se mueven a través de los cables y se hace una decisión explícita de conmutación en cada etapa, para dirigir el mensaje a lo largo de uno de los cables de salida.

En ciertos sistemas las máquinas están ***Fuertemente Acopladas*** y en otras están ***DébilmenteAcopladas;*** en un sistema Fuertemente Acoplado el retraso que se experimenta, al enviar un mensaje de un computador a otro, es corto y la tasa de transmisión de datos, es decir, el número de bits por segundo que se puede transferir es alto; en un sistema Débilmente Acoplado ocurre lo contrario.

Los sistemas fuertemente acoplados tienden a utilizarse más como sistemas paralelos (para trabajar con un solo problema) y los débilmente acoplados tienden a utilizarse como sistemas distribuidos (para trabajar con varios problemas no relacionados entre sí), aunque esto no siempre es cierto.

En general, los multiprocesadores tienden a estar más fuertemente acoplados que los multicomputadores, puesto que pueden intercambiar datos a la velocidad de sus memorias, pero algunos multicomputadores, basados en fibras ópticas, pueden funcionar también con velocidad de memoria.

Aunque el estudio de multiprocesadores y multicomputadores, con base en bus y con conmutador, no afecta directamente el estudio de los sistemas operativos distribuidos, es importante tener un concepto claro acerca de cada uno de ellos.

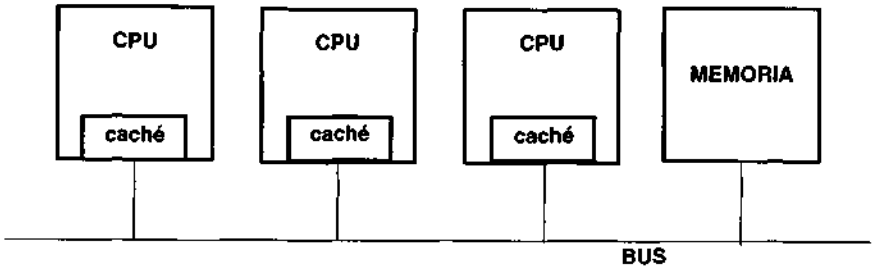
1.3.1. *Multiprocesadores con base en buses*

Los multiprocesadores con base en buses constan de cierto número de CPUs, conectados a un bus común, junto con un módulo de memoria. Una configuración sencilla consta de un plano de base ("backplane") de alta velocidad o tarjeta madre, en la cual se pueden insertar varias tarjetas de memoria y el CPU.

Para leer una palabra de memoria, un CPU coloca la dirección de la palabra deseada en las líneas de dirección del bus y una señal en las líneas adecuadas de control, para indicar que desea leer. La memoria responde y coloca el valor de la palabra en las líneas de datos, para permitir la lectura de ésta por parte del CPU solicitante. La escritura funciona de manera similar.

El problema con este esquema es que si sólo se dispone de 4 ó 5 CPUs; el bus estará, por lo general, sobrecargado y el rendimiento disminuirá en forma drástica. La solución es añadir una ***Memoria Caché*** de alta velocidad entre el CPU y el bus, donde se guardan las palabras de acceso reciente y no hay necesidad de utilizar el bus para ir a buscar, en la memoria, una palabra que

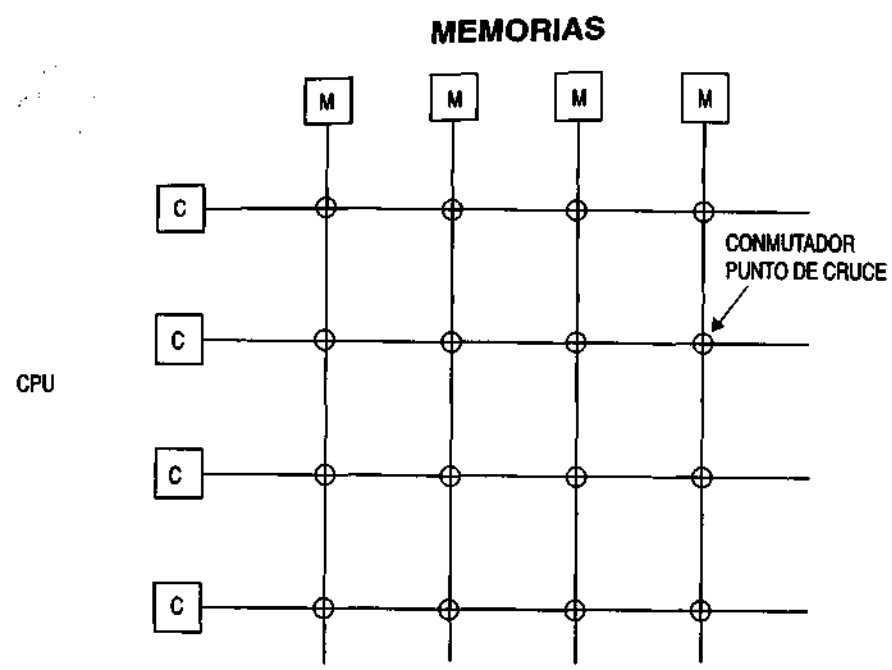
se encuentra en memoria caché. Pero también se presenta un problema de memoria incoherente, en el caso de que dos CPUs lean la misma palabra en sus respectivos cachés y uno de ellos la modifique y el otro lea el valor viejo de la palabra. Los investigadores han solucionado esto con el diseño de un caché que cuando una palabra sea escrita en él, también sea escrita en la memoria; y además, este caché mantiene un monitoreo permanente del bus, para saber cuándo una palabra que está en él ha sido modificada y, por lo tanto, también modificarla en él. Mediante el uso de este caché es posible colocar de 32 a 64 CPUs en el mismo bus.



1.3.2. *Multiprocesadores con conmutador*

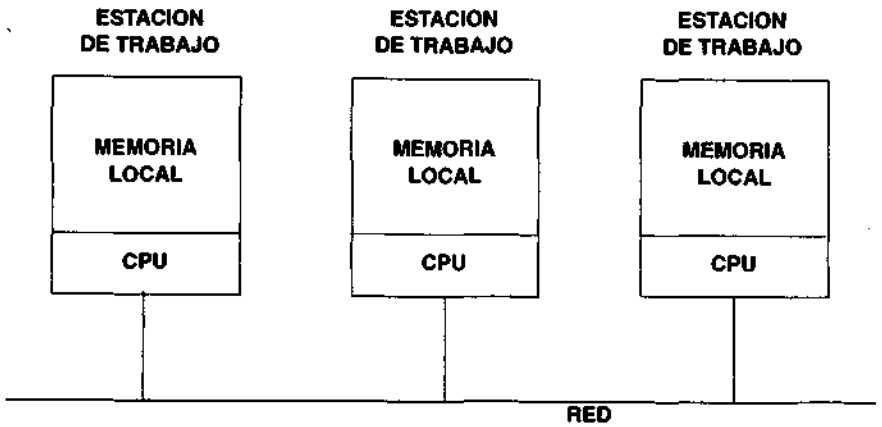
Para construir un multiprocesador con más de 64 procesadores, es necesario un método distinto para conectar cada CPU con la memoria. Una posibilidad es dividir la memoria en módulos y conectarlos a la CPU con un conmutador. Cada CPU y cada memoria tienen una conexión que sale de él. En cada intersección está un delgado ***Conmutador del punto de cruce*** electrónico, que el hardware puede abrir y cerrar. Cuando un CPU desea tener acceso a una memoria particular, el conmutador del punto de cruce que los conecta se cierra de manera momentánea, para que tenga lugar dicho acceso. Esta descripción es del ***Conmutador de Cruceta,*** pero los investigadores han dado a conocer otros tipos de conmutación, como es la ***Red Omega,*** que utiliza mucho menos conmutadores que el conmutador de cruceta y hace la misma función, pero tiene un problema de retraso, pues es muy lenta. Los investigadores diseñaron NUMA (Acceso no uniforme a la memoria), la cual tiene un rápido acceso a su memoria, pero un lento acceso a la memoria de los demás.

En resumen, los multiprocesadores basados en buses, incluso con cachés monitores, quedan limitados como máximo a 64 CPUs, por la capacidad del bus. Para rebasar estos límites es necesario una red de conmutador, como un conmutador de cruceta, una red omega o algo similar. Los grandes conmutadores de cruceta son muy caros y las grandes redes omega son caras y lentas. Las máquinas NUMA necesitan complejos algoritmos para un buen software de colocación. La conclusión es clara: La construcción de un multiprocesador grande, fuertemente acoplado y con memoria compartida, es difícil y cara.



1.3.3. *Multicomputadores con base en buses*

La construcción de un multicomputador es fácil. Cada CPU tiene una conexión directa con su propia memoria local. El único problema consiste en definir la forma en que los CPUs se comunican entre sí. Es claro que aquí también se necesita cierto esquema de interconexión, pero sólo es para la comunicación entre un CPU y otro. El volumen del tráfico será menor, varios órdenes, si se utiliza la red de interconexión para el tráfico CPU-Memoria; el sistema de interconexión puede tener una LAN de menor velocidad, en comparación con un "backplane".



1.3.4. *Multicomputadores con conmutador*

Se han propuesto y construido varias redes de interconexión, pero todas tienen la propiedad de que cada CPU tiene acceso directo y exclusivo a su propia memoria particular. Existen dos topologías populares, una ***Retícula*** y un ***Hipercubo.*** Las retículas son fáciles de comprender y se basan en las tarjetas de circuitos impresos. Se adecuan mejor a problemas, con una naturaleza bidimensional inherente, como la teoría de gráficos o la visión.

Un hipercubo es un cubo-n-dimensional. Cada vértice es un CPU. Cada arista es una conexión entre dos CPUs. Cada CPU tiene "n" conexiones con otros CPUs. Así, la complejidad del cableado aumenta en proporción logarítmica con el tamaño. Puesto que sólo se conectan los vecinos más cercanos, muchos mensajes deben realizar varios saltos, antes de llegar a su destino. Los hipercubos con 1.024 CPUs están disponibles en el mercado desde hace varios años y ya comienzan a estar disponibles los hipercubos, con hasta 16.384 CPUs.

1.4. **Conceptos de software**

Aunque el hardware es importante, el software lo es aún más. La imagen que un sistema presenta a sus usuarios y la forma de pensar de éstos, acerca del sistema, queda determinada, en gran medida, por el software del sistema operativo, no por el hardware.

Los sistemas operativos no se pueden colocar fácilmente dentro de unos estándares como el hardware. Por su propia naturaleza, el software es vago y amorfo. Aún así, es más o menos posible distinguir dos tipos de sistemas operativos para los sistemas de varios CPUs: ***Los Débilmente Acoplados y Los Fuertemente Acoplados.***

El software débilmente acoplado permite que las máquinas y usuarios de un sistema distribuido sean independientes entre sí, en lo fundamental, pero que interactúen, en cierto grado, cuando sea necesario. Consideremos un grupo de computadores personales, cada uno de los cuales tiene su propio CPU, su propia memoria, su propio disco duro y su propio sistema operativo, pero que comparten ciertos recursos, tales como las impresoras láser y las bases de datos, en una LAN. Este sistema está débilmente acoplado, puesto que las máquinas individuales se distinguen con claridad, cada una de las cuales tiene su propio trabajo por realizar. Si la red falla por alguna razón, las máquinas individuales continúan su ejecución en cierto grado considerable, aunque se puede perder cierta funcionalidad.

En el otro extremo, podríamos tener el caso de un multiprocesador dedicado a la ejecución de un único programa de ajedrez, en paralelo. A cada CPU se le asigna un tablero para su evaluación y éste ocupa su tiempo, en la evaluación de este tablero y de los tableros que se pueden generar a partir de él. Al terminar la evaluación, el CPU informa de sus resultados y se le proporciona un nuevo tablero para trabajar con él. El software para este sistema, es decir, el programa de aplicación y el sistema operativo necesario para soportarlo, están más fuertemente acoplados que en el caso anterior.

Con lo que hemos tratado hasta ahora, podemos ver que existen cuatro tipos de hardware distribuido y dos tipos de software distribuido. En teoría, deberían existir ocho combinaciones de hardware y software. Pero sólo existen cuatro, puesto que para el usuario la interconexión de la tecnología no es visible. Un multiprocesador es un multiprocesador, sin importar si utiliza un bus con cachés monitores o una red omega.

1.4.1. *Sistemas Operativos de Redes y NFS*

La combinación más común, en muchas organizaciones, es un software débilmente acoplado en un hardware igual. Un ejemplo típico es una red de estaciones de trabajo de ingeniería, conectadas mediante una LAN. En este modelo, cada usuario tiene una estación de trabajo para su uso exclusivo, lo mismo que su propio sistema operativo, en donde lo normal es que todos los comandos se ejecuten en forma local, precisamente en la estación de trabajo.

Sin embargo, a veces es posible que un usuario se conecte de manera remota con otra estación de trabajo, mediante un comando.

El efecto de este comando es convertir la propia estación de trabajo del usuario en una terminal remota, enlazada con la máquina remota. Los comandos escritos en el teclado se envían a la máquina remota y la salida de ésta se exhibe en la pantalla. En cualquier instante, sólo se puede utilizar una máquina y la selección de ésta se realiza en forma manual.

Las redes de las estaciones de trabajo también tienen un comando de copiado remoto, para copiar archivos de una máquina a otra, pero el usuario debe estar consciente de la posición de los archivos.

Aunque es mejor que nada, esta forma de comunicación es primitiva en extremo. Un método más conveniente de comunicación y distribución de la información consiste en proporcionar un sistema de archivos global compartido, accesible desde todas las estaciones de trabajo. Una o varias máquinas, llamadas ***Servidores de Archivos,*** soportan el sistema de archivos. Los servidores aceptan solicitudes de los programas de usuarios, los cuales se ejecutan en las otras máquinas (No servidoras), llamadas ***Clientes,*** para la lectura y escritura de archivos. Cada una de las solicitudes que llegue se examina, se ejecuta y la respuesta se envía de regreso.

Los servidores de archivos tienen, por lo general, un sistema jerárquico de archivos, cada uno de los cuales tiene un directorio raíz, con sus directorios y archivos. Las estaciones de trabajo pueden importar o montar estos sistemas de archivos, lo que aumenta sus sistemas locales de archivos con aquellos localizados en los servidores.

El sistema operativo por utilizar, en este tipo de ambiente, debe controlar tanto las estaciones de trabajo, en lo individual, como a los servidores de archivos, y también debe encargarse de la comunicación entre ellos. Es posible que todas las máquinas ejecuten el mismo sistema operativo, pero esto no es necesario. Si los clientes y los servidores ejecutan diversos sistemas, entonces, como mínimo, deben coincidir en el formato y significado de todos los mensajes que podrían intercambiar. En una situación como ésta, en la que cada máquina tiene un alto grado de autonomía y existen pocos requisitos a lo largo de todo el sistema, las personas se refieren a ella como un ***Sistema Operativo de Red.***

Uno de los mejores sistemas operativos de red es el ***Network File System,*** de *Sun Microsystems,* conocida en general como *NFS.* Es un sistema con amplio uso. *Sun Microsystems* diseñó e implantó *NFS,* en un principio para su uso en estaciones de trabajo con base en UNIX. Ahora lo soportan otros fabricantes tanto para Unix como para otros sistemas operativos (como MS-DOS). NFS soporta sistemas heterogéneos, por ejemplo, clientes de MS-DOS que hacían uso de servidores Unix. Ni siquiera se pide que las máquinas utilicen el mismo hardware. Es común encontrarse con clientes de MS-DOS que utilizan CPU de Intel 386 y obtienen servicios de servidores de archivo en Unix, con CPU de Motorola 68030 o SunSparc.

Existen tres aspectos importantes de NFS: la Arquitectura, el Protocolo y la Implantación.

- ***La Arquitectura de NFS:*** La idea fundamental de NFS es permitir que una colección arbitraria de clientes y servidores compartan un sistema de archivos común. En la mayoría de los casos, todos los clientes y servidores están en la misma LAN, pero esto no es necesario. Es posible ejecutar el NFS en una WAN. Tomamos a los clientes y servidores como máquinas distintas, pero NFS también permite que cada máquina sea un cliente y un servidor al mismo tiempo.

- La característica básica de la arquitectura de NFS es que los servidores exportan directorios y los clientes los montan de manera remota. Si dos o más clientes montan el mismo directorio al mismo tiempo, ellos se pueden comunicar y compartir archivos en sus directorios comunes. Un programa para un cliente puede crear un archivo y un programa para otro cliente distinto, puede leer dicho archivo. Una vez llevados a cabo los montajes, no hay que hacer nada especial para lograr compartir los archivos. Los archivos compartidos están ahí, en la jerarquía de directorios de varias máquinas, y se puede leer o escribir en ellos de la manera usual. Esta sencillez es uno de los grandes atractivos de NFS.

- ***Protocolos de NFS:*** Puesto que uno de los objetivos de NFS es soportar un sistema heterogéneo, en donde los clientes y servidores podrían ejecutar distintos sistemas operativos en hardware diverso, es esencial que la interfaz entre los clientes y servidores esté bien definida. NFS logra este objetivo mediante la definición de dos protocolos cliente-servidor. Un ***Protocolo*** es un conjunto de solicitudes que envían los clientes a los servidores, junto con las respuestas correspondientes, enviadas por los servidores de regreso a los clientes. Mientras un servidor reconozca y pueda manejar todas las solicitudes en los protocolos, no necesita saber algo de sus clientes. En forma análoga, los clientes consideran a los servidores como "cajas negras". El primer protocolo de NFS maneja el montaje. Un cliente puede enviar el nombre de una ruta de acceso a un servidor y solicitar el permiso para montar ese directorio en alguna parte de su jerarquía de directorios. El segundo protocolo de NFS es para el acceso a los directorios y archivos. Los clientes pueden enviar mensajes a los servidores para el manejo de los directorios y la lectura o escritura de archivos. Además, también pueden tener acceso a los atributos de archivo, tales como su modo, tamaño y fecha de la última modificación.

- **Implantación de NFS:** La implantación del código del cliente y del servidor es independiente de los protocolos. Consta de tres capas: la capa superior es la capa de llamadas al sistema; ésta maneja las llamadas del tipo OPEN, READ y CLOSE. Después de analizar la llamada y verificar los parámetros, llama a la segunda capa, la capa del sistema virtual de archivos (VFS). La tarea de la capa VFS es mantener una tabla, con una entrada por cada archivo abierto, análoga a la tabla de ínodos, para los archivos abiertos en Unix. Después de descubrir el nombre de la máquina donde se localiza el directorio por montar, se relaciona esa máquina buscando un "file handle" para el directorio remoto. Si el directorio existe y está disponible para su montaje remoto, el servidor regresa, entonces, un "file handle" para el directorio. Por último, se transfiere el "file handle" al núcleo.

1.4.2. *Sistemas realmente distribuidos*

El siguiente paso, en la evolución, es el de un software fuertemente acoplado en un hardware débilmente acoplado (es decir, multicomputadores). El objetivo de un sistema de este tipo es crear la ilusión, en las mentes de los usuarios, de que toda la red de computadores es un solo sistema de tiempo compartido, en vez de una colección de máquinas diversas. Esto nos lleva a nuestra definición de sistema distribuido: *un Sistema Distribuido es aquel que se ejecuta en una colección de máquinas sin memoria compartida, pero que aparece ante sus usuarios como un solo computador.*

Para que un sistema sea realmente distribuido, debe existir un mecanismo de comunicación global entre los procesos, de forma que cualquier proceso puede hablar con cualquier otro; además, no tiene que haber distintos mecanismos en distintas máquinas o distintos mecanismos para la comunicación local o la comunicación remota; también debe existir un esquema global de protección.

1.4.3. *Sistemas de multiprocesador con tiempo compartido*

La última combinación que queremos analizar es la de un software fuertemente acoplado en un hardware igual. Aunque existen varias máquinas de propósito especial en esta categoría (como las máquinas dedicadas a las bases de datos), los ejemplos más comunes de propósito general son los multiprocesadores, como los fabricados por Sequent y Encore, que operan como un sistema de tiempo compartido de Unix, sólo que con varios CPU en lugar de uno solo.

La característica clave de este tipo de sistema es la existencia de una sola cola para la ejecución: una lista de todos los procesos, en el sistema, que no están bloqueados en forma lógica y listos para su ejecución. La cola de ejecución es una estructura de datos contenida en la memoria compartida.

1.5 *Aspectos del diseño*

1.5.1. Transparencia

Un sistema es transparente cuando hace que las personas piensen que la colección de máquinas es tan sólo un sistema de tiempo compartido de un solo procesador, es decir, cuando se le oculta la distribución del sistema y piensan que un solo computador es el que hace todo el trabajo. Existen cinco tipos de transparencia:

TRANSPARENCIA DE LOCALIZACION.	**Los usuarios no pueden indicar la localización de los recursos.**
TRANSPARENCIA DE MIGRACION	**Los recursos se pueden mover a voluntad, sin cambiar sus nombres.**

TRANSPARENCIA DE REPLICA.	**Los usuarios no pueden indicar el número de copias existentes.**
TRANSPARENCIA DE CONCURRENCIA.	**Varios usuarios pueden compartir recursos de manera automática.**
TRANSPARENCIA DE PARALELISMO.	**Las actividades pueden ocurrir en paralelo, sin el conocimiento de los usuarios.**

1.5.2. *Flexibilidad*

Es importante que el sistema sea flexible, ya que apenas estamos aprendiendo a construir sistemas distribuidos. Es probable que este proceso tenga muchas salidas falsas y una considerable retroalimentación. Las decisiones de diseño que ahora parezcan razonables podrían demostrar ser incorrectas posteriormente.

Existen dos escuelas de pensamiento en cuanto a la estructura de los sistemas distribuidos. Una escuela dice que cada máquina debe ejecutar un núcleo tradicional que proporcione la mayoría de los servicios. la otra sostiene que el núcleo debe proporcionar lo menos posible y que el grueso de los servicios del sistema operativo se obtenga a partir de los servidores al nivel usuario. La primera escuela, conocida como el núcleo monolítico, opera en el actual sistema operativo y centralizado básico, aumentado con las capacidades de la red y la integración de los servicios remotos. La segunda escuela, conocida como el micronúcleo, opera en los sistemas distribuidos diseñados actualmente.

1.5.3. *Confiabilidad*

Uno de los objetivos originales de la construcción de sistemas distribuidos fue el hacerlos más confiables que los sistemas con un único procesador. La idea es que si una máquina falla, alguna otra máquina se encargue del trabajo.

1.5.4. *Desempeño*

La construcción de un sistema distribuido transparente, flexible y confiable no hará que usted gane premios si es muy lento. En particular, cuando se ejecuta una aplicación en un sistema distribuido, no debe parecer peor que su ejecución en un único procesador.

Se pueden utilizar diversas métricas del desempeño. El tiempo de respuesta es uno, pero también lo son el rendimiento (número de trabajos por hora), el uso del sistema y la cantidad consumida de la capacidad de la red. El problema del desempeño se complica por el hecho que la comunicación, factor esencial en un sistema distribuido (y ausente en un sistema con un único procesador), es algo lenta por lo general. Así, para optimizar el desempeño, con frecuencia hay que minimizar el número de mensajes.

1.5.5. *Escalamiento - Extensión*

La mayoría de los sistemas distribuidos está diseñada para trabajar con unos cuantos cientos de CPUs. Es posible que los sistemas futuros tengan mayores órdenes de magnitud y que las soluciones que funcionen bien para 200 máquinas fallen de manera total para 200 millones; por esto, es importante que al construir los sistemas distribuidos se piense en el número de máquinas que en un futuro se le acondicionarán.

2. COMUNICACIONES EN LOS SISTEMAS DISTRIBUIDOS

La diferencia más importante entre un sistema distribuido y un sistema de único procesador es la comunicación entre los procesos. En un sistema con un solo procesador, la mayor parte de la comunicación entre procesos supone, de manera implícita, la existencia de la memoria compartida. Un ejemplo típico es el problema de los lectores escritores, en donde un proceso escribe en un "buffer" compartido y otro proceso lo lee. Incluso, en las formas más básicas de sincronización, como el semáforo, hay que compartir una palabra (La variable del semáforo). En un sistema distribuido no existe tal memoria compartida, por lo que toda la naturaleza de la comunicación entre procesos debe replantearse a partir de cero.

Comenzaremos con el análisis de las reglas a las que se deben apegar los procesos respecto de la comunicación

conocida como protocolo. Para los sistemas distribuidos, estos protocolos toman con frecuencia la forma de varias capas, cada una con sus propias metas y reglas; luego analizaremos con más detalle el modelo cliente-servidor, del cual tanto se ha hablado. Finalmente, veremos la forma en que se intercambian los mensajes y las muchas opciones disponibles para los diseñadores del sistema.

2.1. **Redes y Comunicaciones**

Como vimos en la primera parte, el disponer de una red es uno de los requisitos para poder hablar de un sistema operativo completamente distribuido; por ello, se hace importante estudiar las características de las comunicaciones, en los diferentes tipos de redes, bien sea en las redes del área local LAN o en las redes de área amplia WAN. Estas últimas permitirán, en un futuro, ofrecer la posibllidad de conexión en el mundo, entre computadores y usuarios; por ello se deben acordar protocolos o reglas de comunicación estándar entre los usuarios, e incluso entre los diferentes tipos de redes.

En el exterior existen diferentes organismos, los cuales han tratado de establecer protocolos comunes para seguir en todas las comunicaciones que se puedan llevar a cabo en los diferentes tipos de redes. Entre los organismos más destacados vale la pena mencionar la Organización Internacional para la Internacionalización (ISO), con su modelo de referencia para la interconexión de sistemas abiertos OSI; el Comité Consultivo Internacional para la Telegrafía y las Telecomunicaciones (CCITT) y el Instituto Nacional Americano de Normas (ANSI).

Entre los conjuntos de normas más importantes podemos citar el OSI, que abarca una arquitectura de servicios y protocolos y es comúnmente usado entre las redes de área amplia, y el TCP/IP, utilizado entre los computadores del Departamento de Defensa de los Estados Unidos.

2.2. **Protocolos con capas**

Debido a la ausencia de memoria compartida, todas las comunicaciones en los sistemas distribuidos se basan en la transferencia de mensajes. Cuando el proceso A quiere comunicarse con el proceso B, construye primero un mensaje en su propio espacio de direcciones. Entonces, ejecuta una llamada al sistema para que éste busque el mensaje y lo envíe, a través de la red, hacia B. Como vemos, esto parece sencillo, pero, para evitar confusiones, A y B deben coincidir en el significado de los bits que se envíen. Por ejemplo, si A envía un mensaje, en código ASCII, y B espera un mensaje en el código EBCDIC, la comunicación no será óptima.

Como vemos, se necesitan muchos puntos de común acuerdo para establecer una buena comunicación. ¿Cuántos voltios hay que utilizar para la señal correspondiente a un bit cero y cuántos para un bit uno? ¿Cómo sabe el receptor cuál es el último bit del mensaje? ¿Cómo puede detectar si un mensaje ha sido dañado o perdido, y qué debe hacer si lo descubre? ¿Qué longitud tienen los números, cadenas y otros elementos de datos y cuál es la forma en que están representados? De este ejemplo se puede deducir que se necesitan varios puntos de acuerdo con una amplia gama de niveles, desde los detalles de bajo nivel de los bits hasta los detalles de alto nivel, acerca de la forma como debe expresarse la información.

2.2.1. ***Modelo de referencia para la interconexión de sistemas abiertos OSI***

Este modelo está diseñado para permitir la comunicación entre los sistemas abiertos. Un sistema abierto es aquel preparado para comunicarse con cualquier otro sistema abierto, mediante reglas estándar que gobiernan el formato, el contenido y el significado de los mensajes recibidos. Estas reglas se formalizan en lo que se llaman *protocolos*. En términos básicos, un protocolo es un acuerdo de la forma en que debe desarrollarse la comunicación. Cuando se presentan una mujer y un hombre, ella puede optar por extender la mano. El, a su vez, puede decidir si estrechársela o besársela según si, por ejemplo, ella es una abogada en una junta de negocios o una princesa europea en un baile formal. La violación del protocolo hará que la comunicación sea más difícil o, tal vez, imposible.

El modelo OSI distingue entre dos tipos generales de protocolos: los orientados hacia las conexiones y los que no tienen ninguna conexión; los primeros, antes de intercambiar los datos entre el emisor y el receptor, negocian el protocolo por utilizar, mientras que los segundos no necesitan ninguna configuración previa para empezar a transmitir el mensaje. Un ejemplo de comunicación sin conexión podría ser el depositar una carta en un buzón, ya que el receptor no espera que le llegue la carta que en ese momento ha sido depositada y mucho menos ha podido establecer una conexión previa con el emisor para negociar las normas que se deberían utilizar en el envío de la carta.

En el modelo OSI, la comunicación se divide hasta en siete niveles o capas, como se muestra en la Figura 2.1. Cada capa se encarga de un aspecto específico de la comunicación; de esta forma, el problema se puede dividir en piezas manejables, cada una de las cuales se puede resolver en forma independiente de las demás. Cada capa proporciona una interfaz con la capa siguiente. La interfaz consiste en un conjunto de operaciones, para definir el servicio que la capa está preparada a ofrecer a sus usuarios.

2.2.2. ***Capas del modelo OSI***

El modelo OSI cuenta con siete capas, una serie de interfaces y siete protocolos, los que garantizarán que la comunicación entre el emisor y el receptor sea eficiente y confiable, como se muestra en la Figura 2.1. Las siete capas del modelo OSI y su utilidad se describirán a continuación:

- La capa física: Se preocupa por la transmisión de los ceros y los unos. El número de voltios por utilizar (0 y 1), el número de bits por segundo que se pueden enviar, el hecho que la transmisión se lleve a cabo en ambas direcciones, en forma simultánea, son todos aspectos claves en la capa física. Además, el tamaño y forma del conector en la red, así como el número de PINS y el significado de cada uno son temas de interés para dicha capa. El protocolo de la capa física controlará que, cuando una máquina envíe el bit 0, sea realmente recibido como un bit 0 y no como un bit 1.

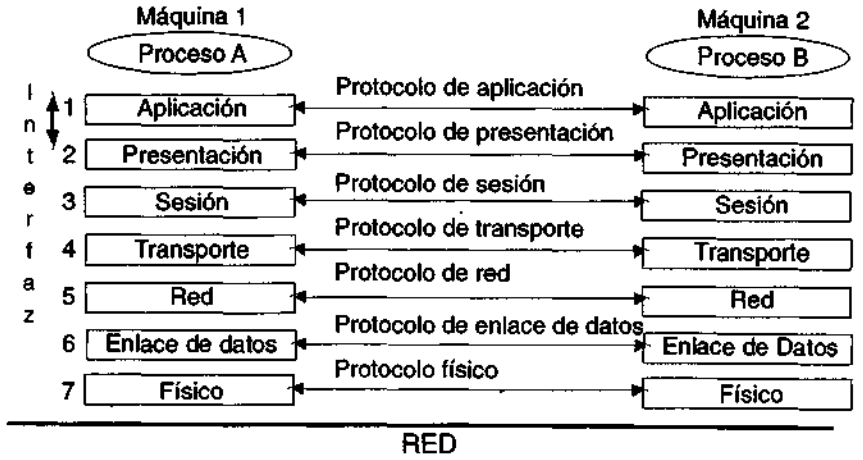


Figura 2.1. *Capas, interfaces y protocolos en el modelo OSI.*

- La capa de enlace de los datos: Esta capa no sólo detecta los errores sino que los corrige. Lo que hace es agrupar los bits en unidades, que a veces se llaman marcos, y revisar que cada marco se reciba en forma correcta; los marcos están conformados por tres campos: un bit al inicio, un bit al final y otro de verificación, conocido como CKECKSUM; este último contendrá la suma de todos los bits del marco, así, en el momento en que el mensaje llegue al receptor, éste sólo tendrá que volver a calcular la suma de los bits del marco y compararla con el contenido del CKECKSUM, para determinar si la información llegó correctamente; en caso contrario, el receptor pedirá al emisor la retransmisión de los datos.
- La capa de la red: Esta capa se encarga de enrutar los datos del emisor, para que lleguen por el camino más corto posible al receptor. Existen dos protocolos comúnmente usados en las capas de red: el orientado a las conexiones, denominado X.25, utilizado con frecuencia por las compañías telefónicas, y el orientado a la no conexión, llamado IP de protocolo internet, el cual es usado por el Departamento de Defensa de los Estados Unidos.
- La capa de transporte: Esta capa se encarga de recibir el mensaje proveniente de la capa de sesión, lo parte en pequeños pedazos, de forma que cada una se ajuste a un único paquete; le asigna a cada uno un número secuencial y después los envía al receptor. Existen diferentes protocolos de transporte, entre ellos están desde el TPO hasta el TP4, protocolos oficiales de transporte ISO; el TCP, protocolo de transporte del Departamento de Defensa de los Estados Unidos; la combinación TCP/IP, protocolo de transporte de la mayoría de los sistemas UNIX, y el UDP, utilizado por los programas de usuario que no necesitan las conexiones.
- La capa de sesión: Esta capa es una versión mejorada de la capa de transporte. Proporciona el control de diálogos, con el fin de mantener un registro de la parte que está hablando en cierto momento y proporciona facilidades en la sincronización. En la práctica, pocas aplicaciones están interesadas en la capa de sesión y rara vez se le soporta.
- La capa de presentación: Esta capa se encarga de dar un formato al mensaje del emisor, para hacerlo entendible por el receptor; dicho formato será una especie de registro, con campos de tamaño adecuado para albergar datos como nombres, direcciones, etc.
- La capa de aplicación: Esta capa se encarga de establecer protocolos para actividades comunes entre terminales remotas; los más conocidos son el protocolo de correo electrónico X.400 y el servidor de directorios X.500.

Para entender mejor el porqué de la existencia de las diferentes capas, en el modelo de comunicaciones OSI, veamos un ejemplo:

Consideremos la comunicación entre dos compañías, *Zyppy Airlines* y su proveedor, *Mushy Meals. Inc.* Cada mes, el jefe de servicios de pasajeros, en *Zippy,* le pide a su secretaria que haga contacto con la secretaria del gerente de ventas en *Mushy,* para ordenar cien mil cajas de pollo. Por lo general, las órdenes se solicitan por medio de la oficina postal. Sin embargo, como el servicio postal es malo, en cierto momento las dos secretarias deciden abandonarlo y comunicarse por fax. Lo pueden hacer sin molestar a sus jefes, puesto que su protocolo se refiere a la transmisión física de las órdenes y a su contenido. En forma análoga, el jefe de servicios de pasajeros decide no pedir el pollo y pedir el nuevo especial de *Mushy,* un filete, sin que esa decisión afecte a las secretarias. Lo que debemos observar es que aquí tenemos dos capas: Los jefes y las secretarias, y cada capa tiene su propio protocolo: temas de discusión y tecnología, que se pueden cambiar independientemente el uno del otro.

De la misma forma, cuando un proceso A de la máquina 1 desea comunicarse con el proceso B de la máquina 2, construye un mensaje y lo transfiere a la capa de aplicación en su máquina; el software de dicha capa agregará un encabezado y hará seguir el mensaje por medio de la interfaz hasta la capa de presentación, en donde se le agregará otro encabezado y así sucesivamente se repetirá el proceso hasta llegar a la capa de enlace de datos, como se muestra en la Figura 2.2. Una vez la máquina 1 tenga el mensaje, con los encabezados correspondientes, lo enviará a la máquina 2, la cual, realizando el proceso inverso, lo interpretará en la forma correcta.

EJEMPLO DE UN MENSAJE ENVIADO DE ACUERDO CON EL MODELO OSI

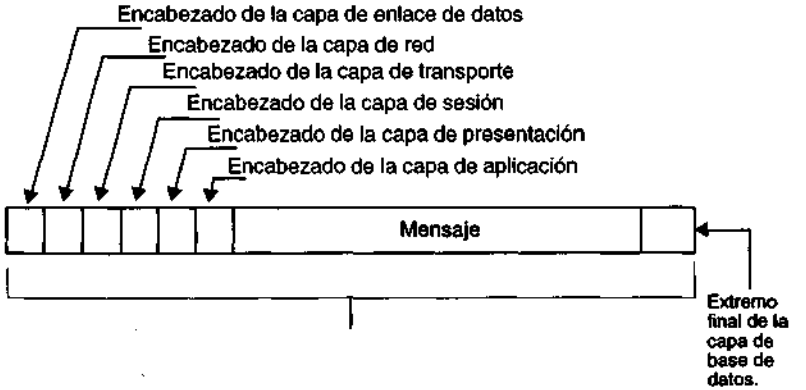


Figura 2.2. *Un mensaje típico, tal como aparece en la red.*

2.3. El modelo cliente-servidor

A primera vista, los protocolos con capas, a lo largo de las líneas OSI, se ven como una forma fina de organizar un sistema distribuido. En efecto, un emisor establece una conexión; es decir, una especie de entubamiento de bits con el receptor y, entonces, empieza a bombear los bits que llegan sin error, en orden, al receptor. Surge entonces la pregunta: ¿qué podría estar mal en este modelo?

A pesar de que el modelo OSI parece ser la forma más eficiente para controlar la comunicación entre el emisor y el receptor, esto sólo se puede afirmar si es utilizado en redes de tipo WAN, ya que, por tratarse de redes en las que los terminales se encuentran distanciados en muchas ocasiones por kilómetros, no importará tener que recurrir a un modelo tan complicado como el OSI, que a pesar de demandar gran cantidad de tiempo en la comunicación, nos brindará la certeza de contar con datos confiables, sin importar las diferencias que puedan existir, tanto en hardware como en el sistema operativo entre el emisor y el receptor, diferencias que podrán ser sorteadas fácilmente gracias a sus siete capas de control.

Para las redes de tipo LAND no es aconsejable utilizar este modelo, ya que por tratarse de comunicaciones entre terminales con características similares, las cuales no están separadas por grandes distancias, el utilizar OSI implicaría controles innecesarios y, por ende, desperdicio de tiempo de la CPU y un costo excesivo comparado con sus beneficios.

Por ello, se hace necesario estructurar el sistema operativo de una forma diferente, dando origen así al modelo cliente servidor, éste tendrá un sistema operativo estructurado por grupos de procesos en cooperación, llamados servidores, que ofrecerán servicios a los usuarios, llamados clientes. Las máquinas de los clientes y servidores ejecutan, por lo general, el mismo micronúcleo y ambos se ejecutan como procesos del usuario.

Para evitar un gasto excesivo en los protocolos orientados a la conexión, tales como OSI o TCP/IP, lo usual es que el modelo cliente-servidor se base en un protocolo solicitud/respuesta, sencillo y sin conexión. El cliente envía un mensaje al servidor, para pedir cierto servicio, por ejemplo, la lectura de un bloque de cierto archivo. El servidor hace el trabajo y regresa los datos solicitados o un código de error, para indicar la razón por la cual un trabajo no se pudo llevar a cabo, como se muestra en la Figura 2.3. La principal ventaja de dicho modelo es su sencillez, ya que el cliente envía un mensaje y obtiene una respuesta; por su parte, como se ve en la Figura 2.4, la pila de capas del modelo es mucho más corta que la de OSI, contando únicamente con tres capas: la física, la de enlace de datos y la de solicitud y respuesta.

TRANSFERENCIA DE MENSAJES EN EL MODELO CLIENTE SERVIDOR

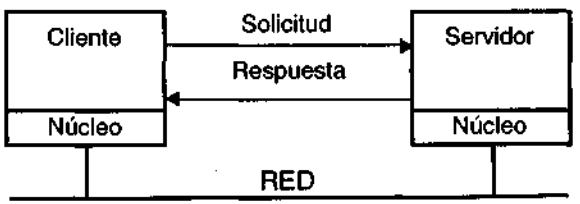


Figura 2.3.

TRANSFERENCIA DE MENSAJES EN EL MODELO CLIENTE SERVIDOR

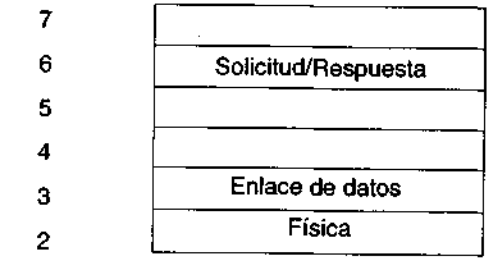


Figura 2.4.

Debido a esta estructura tan sencilla se pueden reducir los servicios de comunicación que presta el micronúcleo; por ejemplo, a dos llamadas al sistema, una para el envío de mensajes y otra para recibirlos. Estas llamadas al sistema se pueden pedir a través de procedimientos de librerías, como SEND (DEST, &MPTR) y RECEIVE (ADDR, & MPTR). La primera envía el mensaje al que apunta MPTR, en un proceso que se identifica como DEST, y provoca que quien hace la llamada se bloquee hasta cuando se envíe el mensaje. La segunda hace que quien hizo la llamada se bloquee hasta que reciba un mensaje. Cuando llega un mensaje, éste se copia en el "buffer" al que apunta MPTR y quien hizo la llamada se desbloquea. El parámetro ADDR determina la dirección en la cual escucha el receptor.

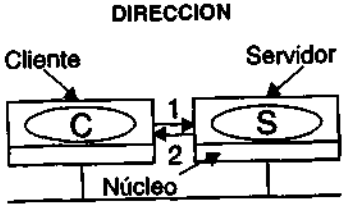
2.3.1. *Dirección*

Para que un cliente pueda enviar un mensaje a un servidor, debe conocer la dirección de éste. Si sólo existe un proceso en ejecución en la máquina-destino cuando se envía el mensaje, el núcleo sabrá qué hacer con el mensaje recibido (dárselo al único proceso en ejecución). Sin embargo, ¿qué ocurre si existen varios procesos en ejecución en la máquina-destino? ¿Cuál de ellos obtiene el mensaje? El núcleo no tiene forma de decidir. En consecuencia, un esquema que utilice las direcciones en la red para la identificación de los procesos indica que sólo se puede ejecutar un proceso en cada máquina. Aunque esta limitación no es fatal, a veces es una seria restricción, razón por la cual surgen tres tipos de métodos para la dirección de los procesos en un modelo cliente-servidor. El primero integra el "machine-number" al código del cliente; es decir, se envían mensajes a los procesos en lugar de enviárselos a las máquinas. Aunque este método elimina toda la ambigüedad acerca de quién es el verdadero receptor, presenta el problema de cómo identificar los procesos. Un esquema común consiste en utilizar nombres con dos partes, para especificar tanto la máquina como el proceso. Por ejemplo, el número 243.4 me indicará que estoy refiriéndome a la máquina 243 y al proceso 4, como se muestra en la Figura 2.5.

El segundo método será dejar que los procesos elijan sus direcciones al azar entre un espacio de direcciones grande y disperso, como lo es el de enteros binarios de 64 bits. La probabi-

lidad de que dos procesos elijan el mismo número es muy pequeña y el método puede utilizarse en sistemas más grandes. Sin embargo, aquí también existe un problema: ¿Cómo sabe el núcleo emisor a cuál máquina enviar el mensaje? La respuesta es simple; el núcleo emisor transmite un paquete especial de localización, con la dirección del proceso-destino, a toda la red; por su parte, los núcleos verifican si la dirección en circulación es la suya y en caso que lo sea, regresa un mensaje diciendo "aquí estoy", con su dirección en la red; es decir, el número de la máquina. En ese momento el núcleo emisor utiliza, entonces, esa dirección y la captura para evitar el envío de otra transmisión la próxima vez que necesite al servidor, como se muestra en la Figura 2.6.

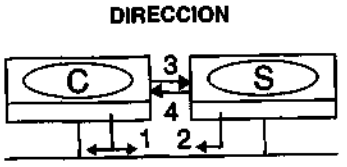
El tercer método, para el direccionamiento de procesos en un modelo-cliente servidor, consiste en utilizar una máquina adicional, llamada servidor de nombres, que me servirá para la asociación, en código ASCII, de los nombres de los servicios con las direcciones de las máquinas, como se muestra en la Figura 2.7; el servidor de nombres le suministra al cliente el número de la máquina donde se localiza, en ese momento, el servidor. Una vez obtenida la dirección, se puede enviar la solicitud de manera directa.



1: Solicitud a 243.0

2: Respuesta a 199.0

Figura 2.5. *Dirección máquina-proceso.*



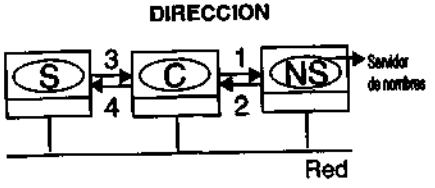
1: Transmisión

2: Aquí estoy

3: Solicitud

4: Respuesta

Figura 2.6. *Dirección de procesos con transmisión.*



1: Búsqueda

2: Respuesta a NS

3: Solicitud

4: Respuesta

Figura 2.7: *Búsqueda de dirección por medio de un servidor de nombres.*

2.3.2. *Primitivas de bloqueo vs. no bloqueo*

Las primitivas de transferencia de mensajes, descritas hasta el momento, reciben el nombre de primitivas de bloqueo, a veces llamadas primitivas síncronas. Cuando un proceso llama un SEND, especifica un destino y un "buffer" dónde enviar ese destino. Mientras se envía el mensaje, el proceso del emisor se bloquea. La instrucción que sigue a la llamada SEND se ejecuta cuando el mensaje se envía en su totalidad, como se muestra en la Figura 2.8. De manera análoga, una llamada a RECEIVE regresa el control cuando realmente se recibe un mensaje y éste se coloca en el "buffer" de mensajes a donde apunta el parámetro.

En RECEIVE, el proceso se suspende hasta cuando llega un mensaje, incluso, aunque tarde varias horas. En ciertos sistemas el receptor puede especificar de quiénes quiere recibir mensajes, en cuyo caso permanece bloqueado hasta que llegue un mensaje del emisor especificado.

Una alternativa a las primitivas con bloqueo son las primitivas sin bloqueo o asíncronas, como se muestra en la Figura 2.9. Si SEND no tiene bloqueo, regresa de inmediato el control a quien hizo la llamada, antes de enviar el mensaje. La ventaja de este esquema es que el proceso del emisor puede continuar su cómputo, en forma paralela con la transmisión del mensaje, en vez de tener inactivo al CPU.

La elección entre las primitivas con bloqueo o sin bloqueo la hacen, por lo general, los diseñadores del sistema, aunque en algunos cuantos sistemas se dispone de ambos y los usuarios pueden elegir su favorito. Sin embargo, la ventaja de desempeño que ofrecen las primitivas sin bloqueo se ve afectada por una seria desventaja: el emisor puede modificar el "buffer" de mensajes cuando se envíe el mensaje; en caso de que se modificara, se enviarían datos traslapados. Para solucionar dicho problema, podríamos optar por copiar el mensaje a un "buffer" interno del núcleo y, así, permitir al proceso continuar con el envío del mensaje y, a la vez, dejar el bloqueo.

Otra alternativa sería la primitiva sin bloqueo con interrupción, en la cual, una vez se ha terminado de enviar el mensaje, se interrumpe al emisor para avisarle que ya puede utilizar el "buffer".

Figura 2.8. *Una primitiva senda con bloqueo.*

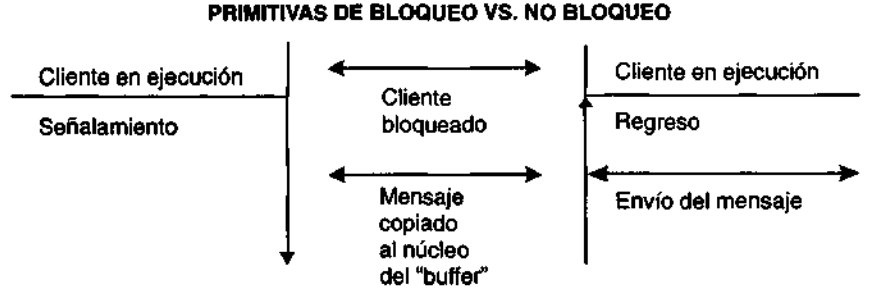


Figura 2.9. *Una primitiva senda sin bloqueo.*

2.3.3. Primitivas almacenadas en "buffer" vs. no almacenadas

Las primitivas descritas hasta ahora son esencialmente primitivas no almacenadas. Esto significa que una dirección se refiere a un proceso específico. Una llamada RECEIVE (ADDR, &M) le indica al núcleo de la máquina en dónde se ejecuta; además, que el proceso que hace la llamada escucha a la dirección ADDR y está preparada para recibir el mensaje enviado a esa dirección.

PRIMITIVAS ALMACENADAS VS. NO ALMACENADAS

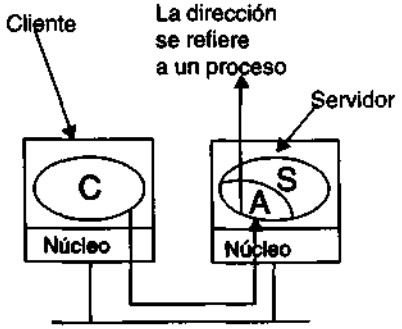


Figura 2.10. *Transferencia de mensajes sin almacenamiento en buffer.*

Se dispone de un único "buffer" de mensajes, al que apunta M, con el fin de capturar el mensaje por llegar. Cuando el mensaje llega el núcleo receptor, lo copia al "buffer" y elimina el bloqueo del proceso receptor, como se muestra en la Figura 2.10.

Un ejemplo de primitiva almacenada es aquél en el cual los mensajes que llegan al servidor son almacenados en un "buffer" llamado buzón, como se ilustra en la Figura 2.11.

PRIMITIVAS ALMACENADAS VS. NO ALMACENADAS

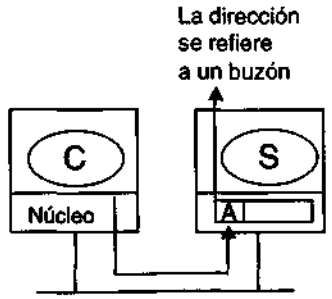


Figura 2.11. *Transferencia de mensajes con almacenamiento en buffer.*

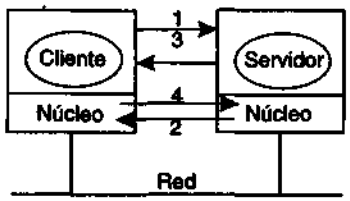
2.3.4. *Primitivas confiables vs. no confiables*

Hemos supuesto de una manera implícita que cuando un cliente envía un mensaje, el servidor lo recibirá. Los mensajes se pueden perder, lo cual afecta la semántica del modelo de transferencia de mensajes. Cuando se utilizan las primitivas por bloqueo y el cliente envía un mensaje, se le suspende hasta que el mensaje ha sido enviado. Sin embargo, cuando vuelve a iniciar, no existe garantía alguna de que el mensaje ha sido entregado. El mensaje podría haberse perdido. Existen tres distintos enfoques de este problema.

El primero consiste en volver a definir la semántica del SEND para hacerlo no confiable. El sistema no da garantía alguna acerca de la entrega de los mensajes. La implantación de una comunicación confiable se deja enteramente en manos de los usuarios. La oficina de correos funciona de esta manera. Cuando usted deposita una carta en un buzón, la oficina de correos hace lo mejor por entregarla, pero no promete nada.

El segundo método exige que el núcleo de la máquina receptora envíe un reconocimiento al núcleo de la máquina emisora. Sólo cuando se reciba este reconocimiento el núcleo emisor liberará el proceso hacia los usuarios (clientes). El reconocimiento va de un núcleo a otro; ni el cliente ni el servidor ven alguna vez un reconocimiento, de la misma forma que la solicitud de un cliente a un servidor es reconocida por el núcleo del servidor; la respuesta del servidor, de regreso al cliente, es reconocida por el núcleo del cliente. Así, una solicitud de respuesta consta de cuatro mensajes, como se muestra en la Figura 2.12.

PRIMITIVAS CONFIABLES VS. NO CONFIABLES

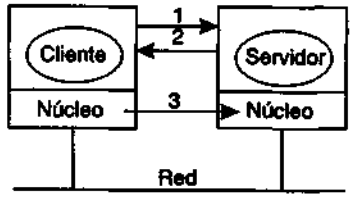


1. Solicitud (del cliente al servidor)
2. Reconocimiento (de núcleo a núcleo)
3. Respuesta del servidor al cliente
4. Reconocimiento (de núcleo a núcleo)

Figura 2.12. *Mensajes reconocidos en forma individual.*

El tercer método aprovecha el hecho que la comunicación cliente-servidor se estructura como una solicitud del cliente al servidor, seguida de una respuesta del servidor al cliente. En este método, el cliente se bloquea después de enviar un mensaje. El núcleo del servidor no envía de regreso un reconocimiento sino que la misma respuesta funciona como tal. Así, el emisor permanece bloqueado hasta que regresa la respuesta. Si tarda demasiado, el núcleo emisor puede volver a enviar la solicitud, para protegerse contra la posibilidad de una pérdida del mensaje. Este método se muestra en la Figura 2.13.

PRIMITIVAS CONFIABLES VS. NO CONFIABLES



1. Solicitud (del cliente al servidor)
2. Respuesta (del servidor al cliente)
3. Reconocimiento (de núcleo a núcleo)

Figura 2.13. *La respuesta se utiliza como reconocimiento de la solicitud.*

El problema con el modelo básico cliente-servidor es que, desde el punto de vista conceptual, la comunicación entre procesos se maneja como E/S, ya que dicha comunicación utiliza uno de los modelos más sencillos de protocolos (solicitud y respuesta). Ahora, cuando una colección de procesos, por ejemplo servidores duplicados de archivo, tiene que comunicarse con otra como grupo, se necesita algo más. Los sistemas de tipo ISIS proporcionan una solución a este problema: la comunicación en grupo. ISIS ofrece una gama de primitivas, de las cuales la más importante es CBCAST. CBCAST ofrece una semántica de comunicación débil con base en la causalidad y se implanta mediante la inclusión de vectores de números secuenciales en cada mensaje, para permitir al receptor que revise si el mensaje se debe entregar de manera inmediata o retrasarse hasta que lleguen algunos mensajes anteriores; éstos no serán tratados por ser casos particulares y más complicados.

De esta forma vemos que, gracias al modelo cliente-servidor, se puede entender un poco mejor cómo se desarrollan las comunicaciones entre procesos en una red de terminales inteligentes y, además, los diferentes aspectos que se

deberían tener en cuenta si se quisiera desarrollar un sistema de estas características; entre ellos aspectos de diseño, tales como si se utilizaran primitivas almacenadas o no almacenadas, confiables o no confiables, con bloqueo o sin bloqueo, etc.

3. SINCRONIZACION EN SISTEMAS DISTRIBUIDOS

Un aspecto muy importante es la forma en que los procesos cooperan y se sincronizan entre sí. Por ejemplo, la forma de implantar las regiones críticas o asignar los recursos en un sistema distribuido.

En los sistemas que sólo cuentan con una CPU, los problemas relativos a las regiones críticas, la exclusión mutua y la sincronización se resuelven, en general, mediante métodos tales como los semáforos y los monitores. Estos métodos no son adecuados para su uso en los sistemas distribuidos, puesto que siempre se basan (de manera implícita) en la existencia de la memoria compartida.

3.1. Sincronización de relojes

La sincronización es más compleja en los sistemas distribuidos que en los centralizados, puesto que los primeros deben utilizar algoritmos distribuidos.

Por lo general no es posible (o recomendable) reunir toda la información relativa al sistema en un solo lugar y después dejar que cierto proceso la examine y tome una decisión, como se hace en el caso centralizado.

En general los algoritmos distribuidos tienen las siguientes propiedades:

1. La información relevante se distribuye entre varias máquinas.
2. Los procesos toman las decisiones sólo con base en la información disponible en forma local.
3. Debe evitarse un único punto de fallo en el sistema.
4. No existe un reloj común o alguna otra fuente precisa del tiempo global.

Los primeros tres puntos indican que es inaceptable reunir toda la información en un solo lugar para su procesamiento; además, el hecho que sólo exista un solo punto de fallo, como éste, hace que el sistema no sea confiable. La idea es que un sistema distribuido debería ser más confiable que las máquinas individuales. Si alguna de ellas falla, el resto puede continuar su funcionamiento.

El último punto de la lista también es crucial. En un sistema centralizado, el tiempo no tiene ambigüedades. Cuando un proceso desea conocer la hora, llama al sistema y el núcleo se lo dice. En un sistema distribuido no es trivial poner de acuerdo todas las máquinas en la hora.

3.1.1. *Relojes lógicos*

Casi todas las computadoras tienen un circuito para el registro del tiempo denominado *cronómetro,* el cual es, por lo general, un cristal de cuarzo trabajado con precisión. Cuando se mantiene sujeto a tensión, un cristal de cuarzo oscila con una frecuencia bien definida, que depende del tipo de cristal, la forma en que se corte y la magnitud de la tensión. A cada cristal se le asocian dos registros, un *contador* y un *registro mantenedor (holding register).* Cada oscilación del cristal decrementa en uno al contador. Cuando el contador toma el valor cero, se genera una interrupción y el contador se vuelve a cargar mediante el registro mantenedor. De esta forma es posible programar un cronómetro, de modo que genere una interrupción con la frecuencia que se desee. Cada interrupción recibe el nombre de una *marca de reloj.*

Cuando se trabaja con varias máquinas, cada una con su propio reloj, la situación es muy distinta de la que se presenta con un solo sistema. Aunque la

frecuencia de un oscilador de cristal es muy estable, es imposible garantizar que los cristales de computadoras distintas oscilen con la misma frecuencia. En la práctica, cuando un sistema tiene *n* computadores, los *n* cristales correspondientes oscilarán a tasas un poco distintas, lo que provoca una pérdida de sincronía en los relojes (de software) y que, al leerlos, tengan valores distintos. La diferencia entre los valores del tiempo se llama *distorsión del reloj.* Como consecuencia de esta distorsión, podrían fallar los programas que esperan el tiempo correcto, asociado a un archivo, objeto, proceso o mensaje; esto es independiente del sitio donde haya sido generado (es decir, el reloj utilizado).

En un artículo clásico, Lamport (1978) demostró que la sincronización de relojes es posible y presentó un algoritmo para lograr esto. Lamport señaló que la sincronización de relojes no tiene que ser absoluta. Si dos procesos no interactúan, no es necesario que sus relojes estén sincronizados, puesto que la carencia de sincronización no será observable y, por tanto, no podría provocar problemas. Además, señaló que lo que importa, por lo general, no es que todos los procesos estén de acuerdo de manera exacta en la hora sino que coincidan en el orden en que ocurren los eventos.

Para la mayoría de los fines, basta que todas las máquinas coincidan en la misma hora. Para una cierta clase de algoritmos, lo que importa es la consistencia interna de los relojes, no su cercanía particular al tiempo real. Para estos algoritmos se conviene en hablar de los relojes como *relojes lógicos.* Para sincronizar los relojes lógicos, Lamport definió una relación llamada *ocurre antes de (happens-before).* La expresión $a \rightarrow b$ se lee: "*a* ocurre antes de *b*" e indica que todos los procesos coinciden en que primero ocurre el evento *a* y después el evento *b.* Esta relación se puede observar de manera directa en dos situaciones:

1. Si *a* y *b* son eventos en el mismo proceso y *a* ocurre antes de *b,* entonces $a \rightarrow b$ es verdadero.
2. Si *a* es el evento del envío de un mensaje por un proceso y *b* es el evento de la recepción del mensaje por otro proceso, entonces $a \rightarrow b$ también es verdadero. Un mensaje no se puede recibir antes de ser enviado o al mismo tiempo en que se envía, puesto que tarda en llegar una cantidad finita de tiempo.

Si dos eventos *x* y *y* están en procesos diferentes que no intercambian mensajes, entonces $x \rightarrow y$ no es verdadero, pero tampoco lo es $y \rightarrow x$. Se dice que estos eventos son *concurrentes,* lo que significa que nada se puede decir (o se necesita decir) acerca del momento en el que ocurren o cuál de ellos es el primero.

Lo que necesitamos es una forma de medir el tiempo, tal que a cada evento *a* le podamos asociar un valor del tiempo C (*a*) en el que todos los procesos estén de acuerdo. El tiempo del reloj C siempre debe ir hacia adelante (creciente) y nunca hacia atrás (decreciente). Se pueden hacer correcciones al tiempo, al sumar un valor positivo al reloj, pero nunca se le debe restar un valor positivo.

En la Figura 3.1 (a) vemos un ejemplo en donde existen tres procesos diferentes que se ejecutan en distintas máquinas, cada una con su propio reloj y velocidad. Cada reloj corre a una razón constante, sólo que las razones son distintas, debido a las diferencias en los cristales. Al tiempo 6, el proceso 0 envía el mensaje A al proceso 1; el reloj del proceso 1 lee 16 cuando el mensaje llega. Si el mensaje acarrea el tiempo de inicio, entonces el proceso 1 concluirá

que tardó 10 marcas de reloj en hacer el viaje. El mensaje B, de 1 a 2, tarde 16 marcas que, al igual que A, es un valor plausible.

El problema está en los mensajes C y D, puesto que van de un valor mayor a uno menor; estos valores son imposibles y ésta es la situación que hay que evitar.

En la solución propuesta por Lamport, para la asignación de tiempos a los eventos, Figura 3.1. (b), cada mensaje acarrea el tiempo de envío de acuerdo con el reloj del emisor. Cuando un mensaje llega y el reloj del receptor muestra un valor anterior al tiempo en que se envió el mensaje, rápidamente el receptor adelanta su reloj para que tenga una unidad más que la del tiempo de envío.

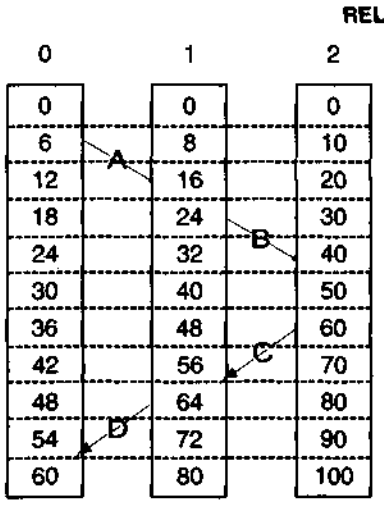


Figura 3.1 (a). *Tres procesos, cada uno con su propio reloj. Los relojes corren a diferentes velocidades.*

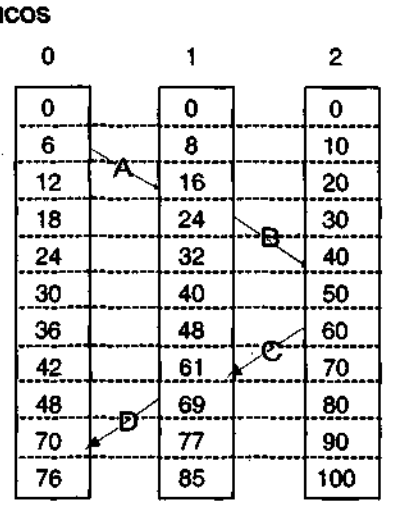


Figura 3.1 (b). *El algoritmo de Lamport corrige los errores.*

Este algoritmo satisface nuestras necesidades para el tiempo global, con una pequeña adición, que entre cualesquiera dos eventos, el reloj debe marcar al menos una vez. Si un proceso envía o recibe dos mensajes en serie muy rápidamente, debe avanzar su reloj en (al menos) una marca entre ellos.

En ciertas situaciones existe un requisito adicional: dos eventos no deben ocurrir exactamente al mismo tiempo. Para lograr esto podemos asociar el número del proceso en que ocurre el evento y el extremo inferior del tiempo, separados por un punto decimal. Así, si ocurren eventos en los procesos 1 y 2, ambos en el tiempo 40, entonces el primero se convierte en 40.1 y el segundo en 40.2.

Por medio de este método tenemos ahora una forma de asignar un tiempo a todos los eventos en un sistema distribuido, con las siguientes condiciones:

1. Si *a* ocurre antes de *b* en el mismo proceso, C (*a*) < C(*b*).
2. Si *a* y *b* son el envío y recepción de un mensaje, C (*a*) < C(*b*).
3. Para todos los eventos *a* y *b*, C(*a*) ≠ C(*b*).

3.1.2. *Relojes físicos*

Cuando existe la restricción adicional de que los relojes no sólo deben ser iguales sino que además no deben desviarse del tiempo real más allá de cierta magnitud, los relojes reciben el nombre de *relojes físicos*. Para estos sistemas se necesitan relojes físicos externos. Por razones de eficiencia y redundancia, por lo general, son recomendables varios relojes físicos, lo cual implica dos problemas:

1. ¿Cómo sincronizar los relojes con el mundo real?
2. ¿Cómo sincronizar los relojes entre sí?

Con la invención del reloj atómico, en 1948, fue posible medir el tiempo de manera mucho más exacta y en forma independiente de todo el ir y venir de la Tierra, al contar las transiciones del átomo de cesio 133. Los físicos retomaron de los astrónomos la tarea de medir el tiempo y definieron el segundo como el tiempo que tarda el átomo cesio 133 para hacer exactamente 9.192.631.770 transiciones.

Actualmente cerca de 50 laboratorios en el mundo tienen relojes de cesio 133. En forma periódica, cada laboratorio le indica a la *Oficina Internacional de la Hora en París (BIH)* el número de marcas de su reloj. La oficina hace un promedio de estos números para producir el *tiempo atómico internacional (TAI)*, que aunque es muy estable y está a disposición de todos los que quieran comprarse un reloj de cesio, existe un serio problema con él; 86.400 segundos TAI, a un tiempo cerca de tres milisegundos menor que el de un día solar medio, lo que significaría que, con el paso de los años, el medio día ocurriría cada vez más temprano.

La BIH resolvió el problema mediante la introducción de *segundos de salto*, siempre que la discrepancia entre el TAI y el tiempo solar creciera hasta 800 milisegundos constantes. Esta corrección da lugar a un sistema de tiempo basado en los segundos TAI, pero que permanece en fase con el movimiento aparente del sol; se le llama *tiempo coordenado universal* (UTC).

Para proporcionar UTC a las personas que necesitan un tiempo preciso, *el Instituto Nacional del Tiempo Estándar* (NIST) opera una estación de radio de onda corta, con las siglas WWV, desde Fort Collins, Colorado. WWV transmite un pulso corto al inicio de cada segundo UTC.

Varios satélites terrestres también ofrecen un servicio UTC, tal como el *Satélite de Ambiente Operacional Geoestacionario (GEOS)*.

3.1.3. *Algoritmos parar la Sincronización de Relojes*

Si una máquina tiene un receptor WWV, entonces el objetivo es hacer que todas las máquinas se sincronicen con ella. Si ninguna máquina tiene receptores WWV, entonces cada máquina lleva el registro de su propio tiempo y el objetivo es mantener el tiempo de todas las máquinas, tan cercano como sea posible.

El algoritmo de Berkeley

En este caso, el servidor de tiempo (en realidad, un demonio para el tiempo) está activo y realiza un muestreo periódico de todas las máquinas para preguntarles el tiempo. Con base en las respuestas, calcula un tiempo promedio y les indica a todas las demás máquinas que avancen su reloj a la nueva hora o que disminuyan la velocidad del mismo, hasta lograr cierta reducción específica. Este método es adecuado para un sistema donde no exista un receptor de WWV. La hora del demonio, para el tiempo, debe ser establecida en forma manual por el operador, de manera periódica.

En la Figura 3.2 (a), a las 3:00, el demonio para el tiempo indica, a las demás máquinas, su hora y les pregunta las suyas. En (b), las máquinas responden, con la diferencia de sus horas como la del demonio. Con estos números, el demonio calcula el tiempo promedio y le dice a cada máquina cómo ajustar su reloj (c).

EL ALGORITMO DE BERKELEY

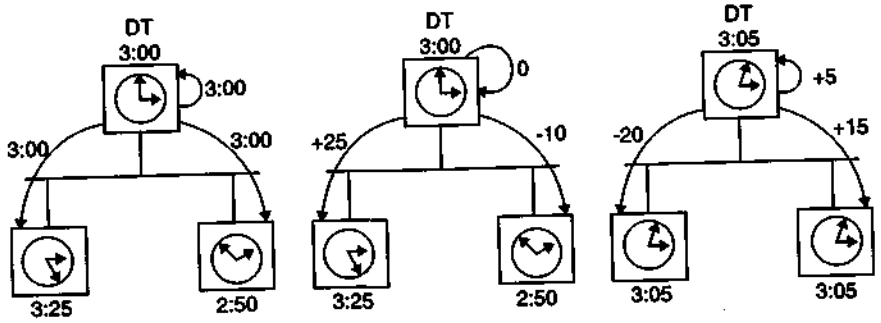


Figura 3.2 (a). *El demonio para el tiempo pregunta a todas las otras máquinas por el valor de sus relojes.* **(b).** *Las máquinas contestan.* **(c)** *El demonio para el tiempo les dice a todas la forma de ajustar sus relojes.*

Algoritmos con promedio

El método anterior es altamente centralizado. Una clase de algoritmos de reloj descentralizada trabaja para dividir el tiempo en intervalos de resincronización de longitud fija. El i-ésimo intervalo inicia en $T0 + iR$ y va hasta $T0 + (i + 1)R$ donde $T0$ es un momento ya acordado en el pasado y R es un parámetro del sistema. Al inicio de cada intervalo, cada máquina transmite el tiempo actual, según su reloj. Puesto que los relojes de las diversas máquinas no corren precisamente a la misma velocidad, estas transmisiones no ocurrirán exactamente en forma simultánea.

Después de que una máquina transmite su hora, inicializa un cronómetro local, para reunir las demás transmisiones que lleguen en cierto intervalo S. Cuando llegan todas las transmisiones, se ejecuta un algoritmo para calcular una nueva hora para ellos. El algoritmo más sencillo consiste en promediar los valores de todas las demás máquinas.

Una ligera variación de este tema consiste en descartar primero los m valores más grandes y los m valores más pequeños y promediar el resto. El hecho de descartar los valores extremos se puede considerar como autodefensa contra m relojes fallidos que envían mensajes sin sentido.

3.2. Exclusión mutua

Los sistemas con varios procesos se programan más fácilmente mediante las regiones críticas. Cuando un proceso debe leer o actualizar ciertas estructuras de datos compartidas, primero entra en una región crítica para lograr la exclusión mutua y garantizar que ningún otro proceso utilizará las estructuras de datos al mismo tiempo. Analizaremos algunos ejemplos de implantación de las regiones críticas y exclusión mutua en los sistemas distribuidos.

3.2.1. *Un algoritmo centralizado*

La forma más directa de lograr la exclusión mutua, en un sistema distribuido, es simular la forma en que se lleva a cabo en un sistema con un único procesador. Se elige un proceso como el coordinador; siempre que un proceso desea entrar en una región crítica envía un mensaje de solicitud al coordinador, donde se indica la región crítica en la que desea entrar y pide permiso. Si ningún otro proceso está por el momento en esa región crítica, el coordinador envía una respuesta para el permiso. Figura 3.3 (a).

Cuando llega la respuesta, el proceso solicitante entra en la región crítica. Supongamos ahora que otro proceso, 2, Figura 3.3 (b), pide permiso para entrar en la misma región crítica. El coordinador sabe que un proceso distinto ya se encuentra en esta región, por lo que no puede otorgar el permiso. El método exacto utilizado para negar el permiso depende de la máquina. En la Figura 3.3 (b) el coordinador sólo se abstiene de responder, con lo cual se bloquea el proceso 2, que espera una respuesta. Otra alternativa consiste en enviar una respuesta que diga "permiso denegado". De cualquier manera, forma en una cola la solicitud de 2, por el momento.

Cuando el proceso 1 sale de la región crítica, envía un mensaje al coordinador, para liberar su acceso exclusivo, Figura 3.3 (c). El coordinador extrae el primer elemento de la cola de solicitudes diferidas y envía a ese proceso un mensaje que otorga el permiso. Si el proceso estaba bloqueado (es decir, éste es el primer mensaje que se le envía), elimina el bloqueo y entra en la región crítica.

Es fácil ver que el algoritmo garantiza la exclusión mutua: el coordinador sólo deja que un proceso esté en cada región crítica a la vez. También es justo, puesto que las solicitudes se aprueban en el orden en que se reciben. Ningún proceso espera por siempre. Este esquema también es fácil de implantar y sólo requiere tres mensajes, por cada uso de una región crítica (solicitud, otorgamiento, liberación).

EXCLUSION MUTUA: Algoritmo centralizado

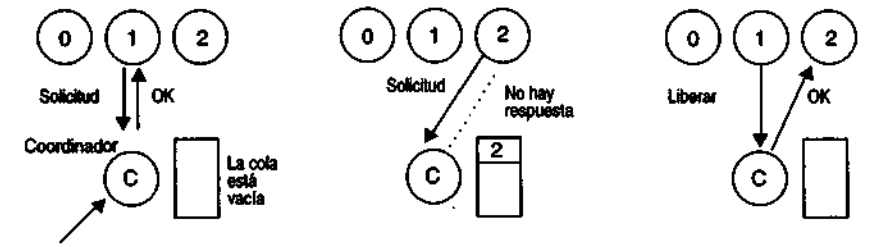


Figura 3.3 (a). *El proceso 1 pide permiso al coordinador para entrar en una región crítica. El permiso es concedido.* **(b).** *El proceso 2 pide, entonces, permiso para entrar en la misma región crítica.* **(c).** *Cuando el proceso 1 sale de la región crítica, se lo dice al coordinador, el cual responde, entonces, a 2.*

El método centralizado también tiene limitaciones. El coordinador es un único punto de fallo, por lo que, si se descompone, todo el sistema se puede venir abajo. Si los procesos se bloquean, por lo general después de hacer una solicitud, no pueden distinguir entre "un coordinador muerto" y un "permiso negado", puesto que en ambos casos no reciben una respuesta. Además, en un sistema de gran tamaño, un único coordinador puede convertirse en un cuello de botella para el desempeño.

3.2.2. *Un algoritmo distribuido*

Ricart y Agrawala (1981) hicieron más eficaz el artículo de Lamport (1978), relativo a la sincronización de los relojes, que fue el primer algoritmo distribuido.

El algoritmo de Ricart y Agrawala requiere la existencia de un orden total de todos los eventos en el sistema; debe ser claro cuál de ellos ocurrió primero.

Cuando un proceso desea entrar en una región crítica, construye un mensaje con el nombre de ésta, su número de proceso y la hora actual. Entonces, envía el mensaje a todos los demás procesos y, de manera conceptual, a él mismo.

Cuando un proceso envía un mensaje de solicitud de otro proceso, la acción que realice depende de su estado con respecto de la región crítica nombrada en el mensaje. Hay que distinguir tres casos:

1. Si el receptor no está en la región crítica y no desea entrar en ella, envía de regreso un mensaje *OK* al emisor.
2. Si el receptor ya está en la región crítica, no responde sino que forma la solicitud en una cola.
3. Si el receptor desea entrar en la región crítica pero no lo ha logrado todavía, compara la marca de tiempo en el mensaje recibido con la marca contenida en el mensaje que envió cada uno. La menor de las marcas gana. Si el mensaje recibido es menor, el receptor envía de regreso un mensaje *OK*. Si su propio mensaje tiene una marca menor, el receptor forma la solicitud en una cola y no envía nada.

Después de enviar las solicitudes que piden permiso para entrar en una región crítica, un proceso espera hasta que alguien más obtenga el permiso. Tan pronto llegan todos los permisos, pueden entrar en la región crítica. Cuando sale de ella, envía mensajes *OK* a todos los procesos en su cola y elimina a todos los elementos de la cola.

En el ejemplo de la Figura 3.4 (a) se muestra qué sucede si dos procesos intentan entrar en la misma región crítica de manera simultánea. El proceso 0 envía, a todos, una solicitud con la marca de tiempo 8, mientras que al mismo tiempo el proceso 2 envía, a todos, una solicitud con la marca de tiempo 12. El proceso 1 no se interesa en entrar en la región crítica, por lo que envía *OK* a ambos emisores. Los procesos 0 y 2 ven el conflicto y comparan las marcas. El proceso 2 pierde y envía un *OK* a 0. El proceso 0 entra en la región crítica, Figura 3.4 (b). Cuando termina, retira la solicitud de 2 de la cola y envía un *OK* a 2 y éste entra en la región crítica, Figura 3.4 (c).

Como en el caso del algoritmo centralizado, la exclusión mutua queda garantizada, sin bloqueo ni inanición. El único punto de fallo es reemplazado por *n* puntos de fallo. Si cualquier proceso falla, no podrá responder a las solicitudes. Este silencio será interpretado (incorrectamente) como una negación del permiso, con lo que se bloquearán los siguientes intentos de los demás procesos para entrar en todas las regiones críticas.

Es posible mejorar un poco este algoritmo. Por ejemplo, la obtención del permiso de todos para entrar en una región crítica es realmente redundante. Todo lo que se necesita es un método para evitar que dos procesos entren en la misma región crítica al mismo tiempo.

EXCLUSION MUTUA: Algoritmo distribuido

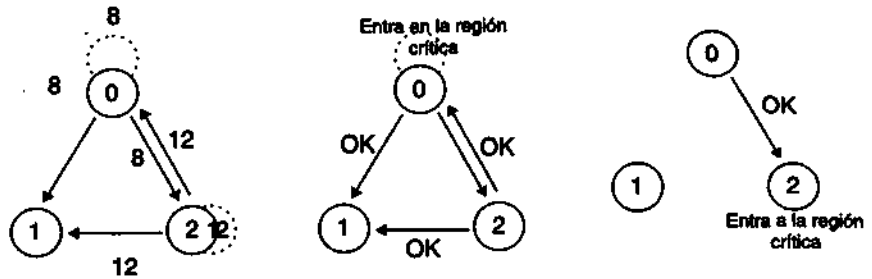


Figura 3.4(a). *Dos procesos desean entrar en la misma región crítica en el mismo momento.*

(b). *El proceso 0 tiene una marca de tiempo menor, por lo que gana.*

(c). *Cuando se termina el proceso 0, envía un OK, por lo que 2 puede ahora entrar en la región crítica.*

3.2.3. *Un algoritmo de anillo de fichas (Token Ring)*

Este es un método completamente distinto para lograr la exclusión mutua en un sistema distribuido. Se tiene por ejemplo una red basada en un bus, Figura 3.5 (a). En software se construye un anillo lógico y a cada proceso se le asigna una posición en el anillo, Figura 3.5 (b). Las posiciones en el anillo se pueden asignar en orden numérico de las direcciones de la red o mediante algún otro medio. Lo importante es que cada proceso sepa quién es el siguiente en la fila después de él.

Al iniciar el anillo, se le da al proceso 0 (cero) una *ficha*, la cual circula en todo el anillo. Se transfiere del proceso *k* al proceso *k +1* en mensajes puntuales. Cuando un proceso obtiene la ficha de su vecino, verifica si intenta entrar en una región crítica. En ese caso, el proceso entra en la región, hace todo el trabajo necesario y sale de la región. Después de salir, pasa la ficha a lo largo del anillo. No se permite entrar en una segunda región crítica con la misma ficha.

Si un proceso recibe la ficha de su vecino y no está interesado en entrar en una región crítica, sólo la vuelve a pasar. En consecuencia, cuando ninguno de los procesos desea entrar en una región crítica, las fichas sólo circulan a gran velocidad en el anillo.

EXCLUSION MUTUA: Anillo de fichas (*Token Ring*)

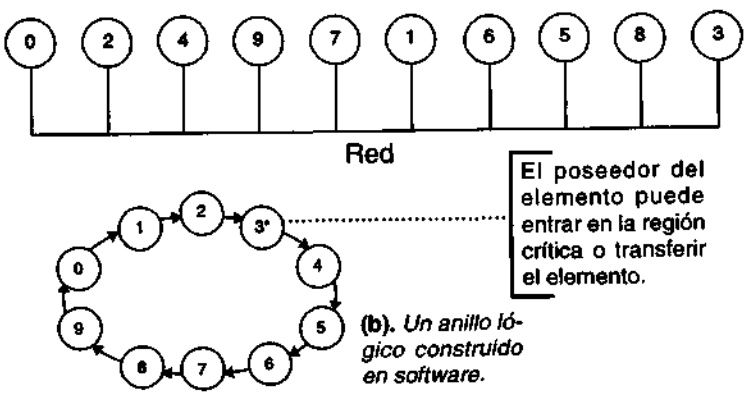


Figura 3.5
(a) *Un grupo no ordenado de procesos en una red.*

(b). *Un anillo lógico construido en software.*

También este algoritmo tiene problemas. Si la ficha llega a perderse, debe ser regenerada. De hecho es difícil detectar su pérdida, puesto que la cantidad de tiempo entre las apariciones sucesivas de la ficha en la red no está acotada. El hecho que la ficha no se haya observado durante una hora no significa su pérdida; tal vez alguien la está utilizando.

El algoritmo también tiene problemas si falla un proceso. Si pedimos un reconocimiento a cada proceso que reciba la ficha, entonces se detectará un proceso muerto si su vecino intenta darle la ficha y fracasa en el intento. En ese momento el proceso muerto se puede eliminar del grupo y el poseedor de la ficha puede enviar ésta, por encima de la cabeza del proceso muerto, al siguiente miembro, o al siguiente después de éste, en caso necesario.

3.3. Algoritmos de elección

Muchos de los algoritmos distribuidos necesitan que un proceso actúe como coordinador, iniciador, secuenciador o que desempeñe, en cierta forma, algún papel especial. Si todos los procesos son idénticos, no existe forma de elegir uno de ellos como especial. En consecuencia, supondremos que cada proceso tiene un número único; por ejemplo, su dirección en la red. En general los algoritmos de elección intentan localizar el proceso con el máximo número del proceso y designarlo como coordinador.

Además, también supondremos que cada proceso sabe el número del proceso de todos los demás; lo que sí desconoce es si los procesos están activos o inactivos. El objetivo de un algoritmo de elección es garantizar que al iniciar una elección, ésta concluya con el acuerdo de todos los procesos con respecto a la identidad del nuevo coordinador.

3.3.1. *El algoritmo del grandulón*

Diseñado por García-Molina (1982). Cuando un proceso observa que el coordinador ya no responde a las solicitudes, inicia una elección. Un proceso *P* realiza una elección de la siguiente manera:

1. *P* envía un mensaje *elección* a los demás procesos, con un número mayor.
2. Si nadie responde, *P* gana la elección y se convierte en el coordinador.
3. Si uno de los procesos, con un número mayor, responde, toma el control. El trabajo de *P* termina.

En cualquier momento un proceso puede recibir un mensaje *elección* de uno de sus compañeros, con un número menor. Cuando esto sucede, el receptor envía de regreso un mensaje *OK* al emisor para indicar que está vivo y que tomará el control. El receptor realiza entonces una elección, a menos que ya esté realizando alguna. En cierto momento, todos los procesos se rinden, menos uno, el cual se convierte en el nuevo coordinador. Anuncia su victoria al enviar un mensaje a todos los procesos para indicarles que, a partir de ese momento, es el nuevo coordinador.

Si un proceso inactivo se activa, realiza una elección. Si ocurre que es el proceso en ejecución, con el número máximo, ganará la elección y tomará para sí el trabajo del coordinador. Así, siempre gana el tipo más grande; de ahí el nombre.

En la Figura 3.6 vemos el funcionamiento de este algoritmo. El grupo consta de ocho procesos, numerados de 0 al 7. El proceso 7 era el coordinador pero acaba de fallar. El proceso 4 lo nota y envía mensajes *elección* a los procesos mayores que el 5, 6 y 7, Figura 3.6 (a). Los procesos 5 y 6 responden *OK* Figura 3.6 (b); el proceso 4 sabe que en este momento ha terminado su trabajo. En la Figura 3.6 (c), tanto 5 como 6 realizan elecciones enviando mensajes a procesos mayores que ellos. En la Figura 3.6 (d) el proceso 6 indica a 5 que tomará el control. Cuando está listo para asumir el control, el proceso 6 lo anuncia, para lo cual envía mensajes *coordinador* a todos los procesos en ejecución.

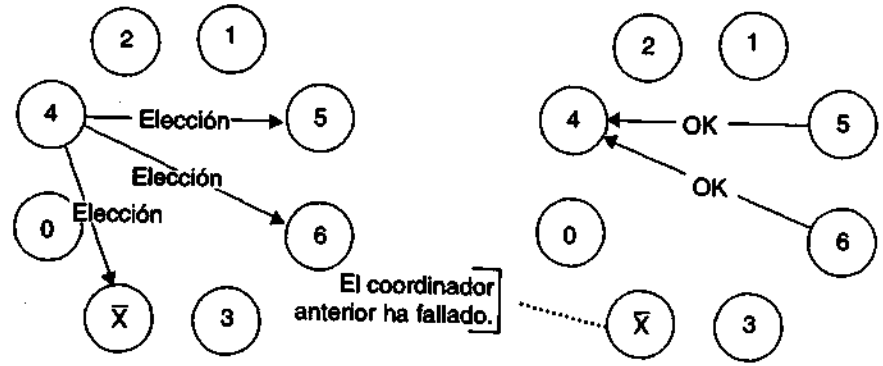
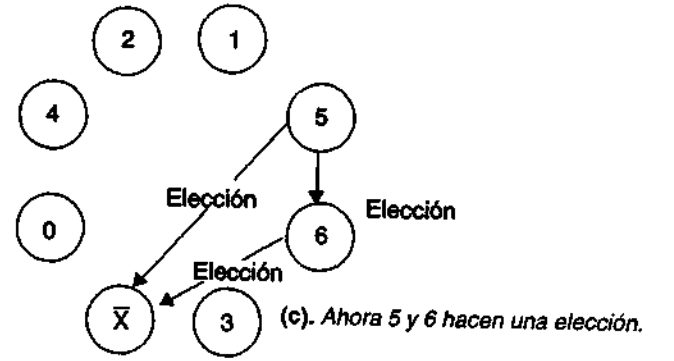
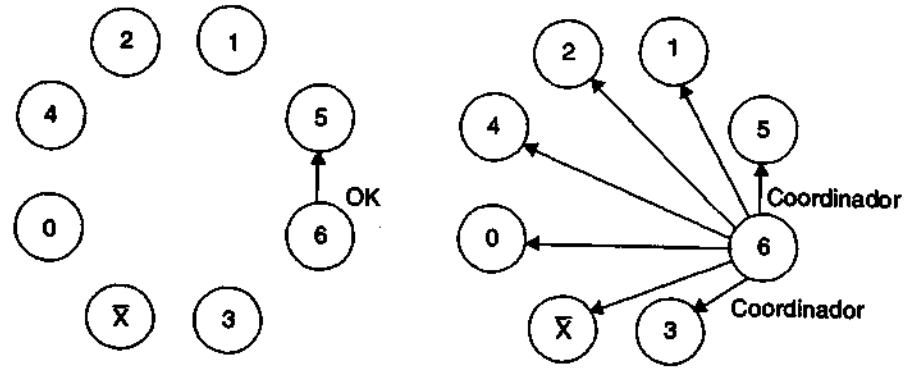


Figura 3.6 (a). *El proceso 4 hace una elección.* **(b).** *El proceso 5 y 6 responde e indica a 4 que se detenga.*



(c). *Ahora 5 y 6 hacen una elección.*



(d). *El proceso 6 indica a 5 que se detenga.* **(c).** *El proceso 6 gana y se lo informa a todos.*

3.3.2. *Un algoritmo de anillo*

Suponemos que los procesos tienen un orden, físico o lógico, de modo que cada proceso conoce a su sucesor. Cuando algún proceso observa que el coordinador no funciona, construye un mensaje *elección* con su propio número de proceso y envía el mensaje a su sucesor. Si éste está inactivo, el emisor pasa sobre el sucesor y va hacia el siguiente número del anillo o al siguiente de éste, hasta localizar un proceso en ejecución. En cada paso el emisor añade su propio número de proceso a la lista en el mensaje.

En cierto momento el mensaje regresa a los procesos que lo iniciaron. Ese proceso reconoce este evento cuando recibe un mensaje de entrada con su propio número de proceso. En este punto el mensaje escrito cambia a *coordinador* y circula de nuevo, esta vez para informar a todos los demás quién es el coordinador (el miembro de la lista con el número máximo) y quiénes son los miembros del nuevo anillo. Una vez ha circulado, el mensaje se elimina y todos se ponen a trabajar. En la Figura 3.7 vemos qué ocurre si dos procesos, 2 y 5, descubren simultáneamente que ha fallado el coordinador, proceso 7. Cada uno construye un mensaje *elección* y comienza a circularlo. En cierto momento ambos mensajes habrán dado la vuelta y ambos se convertirán en mensajes *coordinador*, con los mismos miembros y en el mismo orden. Cuando han dado una vuelta completa, ambos serán eliminados.

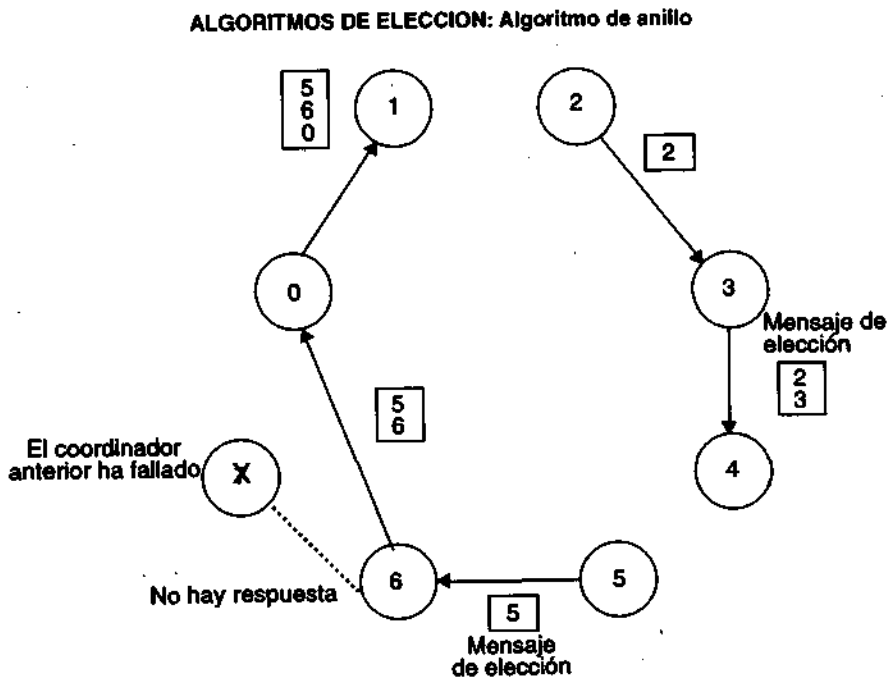


Figura 3.7(a) *Algoritmo de elección mediante un anillo.*

3.4. Bloqueos en sistemas distribuidos

Son más difíciles de evitar, prevenir e incluso detectar, además de ser más difíciles de curar cuando se les sigue la pista, puesto que toda la información relevante está dispersa en muchas máquinas.

Existen cuatro estrategias usuales para el manejo de los bloqueos:

1. El algoritmo del avestruz (ignorar el problema).
2. Detección (permitir que ocurran los bloqueos, detectarlos e intentar recuperarse de ellos).
3. Prevención (hacer que los bloqueos sean imposibles desde el punto de vista estructural).
4. Evitarlos (evitar los bloqueos mediante la asignación cuidadosa de los recursos).

Las cuatro estrategias son aplicables de manera potencial a los sistemas distribuidos. Nuestro análisis de los bloqueos de los sistemas distribuidos se centrará solamente en dos de las técnicas: detección y prevención.

3.4.1. *Detección distribuida de bloqueos*

Detección centralizada de bloqueos

Aunque cada máquina mantiene la gráfica de recursos de sus propios procesos y recursos, un coordinador central mantiene la gráfica de recursos de todo el sistema (la unión de todas las gráficas individuales). Cuando el coordinador detecta un ciclo, elimina uno de los procesos para romper el bloqueo.

A diferencia del caso centralizado, donde se dispone de toda la información de manera automática en el lugar correcto, en un sistema distribuido esta información se debe enviar de manera explícita.

Detección distribuida de bloqueos

Un algoritmo típico es el algoritmo de *Chandy-Misra-Haas*. Este permite que los procesos soliciten varios recursos (por ejemplo, cerraduras) al mismo tiempo, en vez de uno cada vez. Al permitir las solicitudes simultáneas de varios procesos, la fase de crecimiento de una transacción se puede realizar mucho más rápido. La consecuencia de este cambio al modelo es que un proceso puede esperar ahora a dos o más recursos en forma simultánea.

El algoritmo *Chandy-Misra-Haas* se utiliza cuando un proceso debe esperar cierto recurso.

Un algoritmo alternativo consiste en que cada proceso añada su identidad al final del mensaje de exploración, de modo que cuando regrese al emisor inicial se enliste todo el ciclo. El emisor puede ver, entonces, cuál de los procesos tiene el número más grande y eliminarlo, o bien enviar un mensaje que le pida que se elimine a sí mismo. De cualquier forma, si varios procesos descubren el mismo ciclo al mismo tiempo, todos elegirán la misma víctima.

3.4.2. *Prevención distribuida de bloqueos*

Consiste en el diseño cuidadoso del sistema, de modo que los bloqueos sean imposibles, desde el punto de vista estructural. Entre las distintas técnicas se incluye el permitir a los procesos que sólo conserven un recurso a la vez; exigir a los procesos que soliciten todos sus recursos desde un principio y hacer que todos los procesos liberen todos sus recursos cuando soliciten uno nuevo.

En un sistema distribuido con tiempo global y transacciones atómicas son posibles otros algoritmos prácticos. Ambos se basan en la idea de asociar,

a cada transacción, una marca de tiempo global al momento de su inicio.

La idea detrás de este algoritmo es que cuando un proceso está a punto de bloquearse, en espera de un recurso que está utilizando otro proceso, se verifica cuál de ellos tiene la marca de tiempo mayor (es decir, es más joven).

Podemos permitir, entonces, la espera sólo si el proceso en estado de espera tiene una marca inferior (más viejo) que el otro. Otra alternativa consiste en permitir la espera de procesos sólo si el proceso que espera tiene una marca mayor (es más joven) que el otro proceso, en cuyo caso las marcas aparecen en la cadena en forma descendente.

Aunque las dos opciones previenen los bloqueos, es más sabio dar prioridad a los procesos más viejos. Se han ejecutado durante más tiempo, por lo que el sistema ha invertido mucho en ellos y es probable que conserven más recursos.

En la Figura 3.8 (a), un proceso antiguo desea un recurso que mantiene un proceso joven; debemos permitir al proceso que continúe. En (b), un proceso joven desea un recurso que mantiene un proceso antiguo; en este caso, lo debemos eliminar. Este algoritmo se denomina *espera-muerte.*

BLOQUEOS: Prevención distribuida

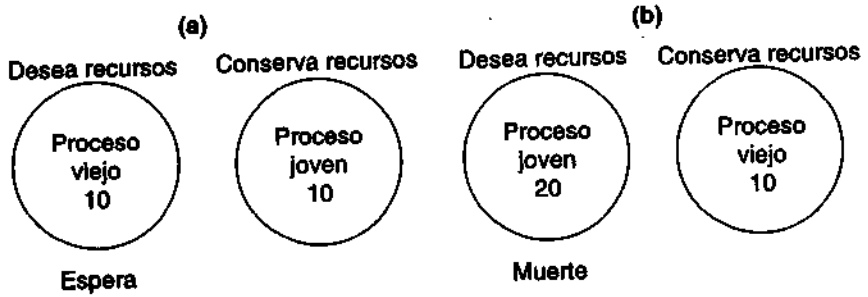


Figura 3.8. *Algoritmo espera-muerte para la prevención de bloqueos.*

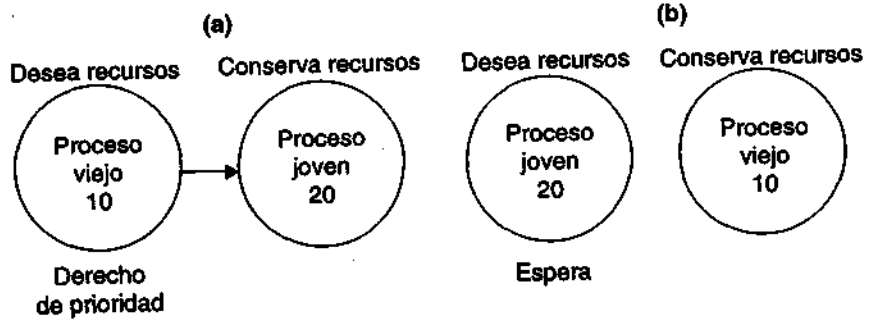


Figura 3.9. *Algoritmo herida-muerte para la prevención de bloqueos.*

La Figura 3.9 (a) recibe el nombre *Derecho de prioridad* y la (b) se denomina *espera.* Este algoritmo se llama *herida-espera.* Si un proceso antiguo desea un recurso mantenido por uno joven, el proceso antiguo ejerce su derecho de prioridad sobre el joven, Figura 3.9 (a). Es probable que el joven vuelva a iniciar su ejecución de manera inmediata, con lo que intentaría adquirir de nuevo el recurso; esto lleva a la situación de la Figura 3.9 (b).

4. PROCESOS Y PROCESADORES EN SISTEMAS DISTRIBUIDOS

Aunque los procesos son un concepto muy importante en los sistemas de un procesador, se tratará de hacer énfasis en los aspectos del manejo de procesos que frecuentemente no son estudiados en el contexto de los sistemas operativos clásicos.

En la mayoría de los sistemas distribuidos se presenta la posibilidad de tener muchos hilos ("threads") de control dentro de un proceso, lo cual llevará a tener ventajas significantes, pero también se presentarán varios problemas que serán estudiados más adelante.

4.1. Hilos

Los hilos se inventaron para permitir la combinación del paralelismo con la ejecución secuencial y el bloqueo de las llamadas al sistema. En la mayoría de los sistemas operativos tradicionales, cada proceso tiene un espacio de direcciones y un único hilo de control (puede ser ésta una aproximada definición de un proceso); a pesar de que frecuentemente existan situaciones en donde se desea tener varios hilos de control que compartan el mismo espacio de direcciones, pero que se ejecuten de manera cuasi-paralela, como si fuesen procesos independientes (siendo su única diferencia que comparten el mismo espacio de direcciones).

En la Figura 4.1, parte (a), se tiene una máquina con tres procesos. Cada uno de ellos posee su propio contador del programa, su propia pila, su propio conjunto de registros y su propio espacio de direcciones. Los procesos no tienen ninguna relación entre sí; la única manera posible de comunicación sería mediante las primitivas de comunicación entre procesos del sistema, como los semáforos, monitores o mensajes. En la parte (b) se tiene otra máquina con un solo proceso, con la única diferencia que éste tiene varios hilos de control, los cuales generalmente son llamados **hilos o procesos ligeros.** En muchas ocasiones se pueden considerar los hilos como unos miniprocesos. Cada hilo se ejecuta de manera estrictamente secuencial y tiene su propio contador del programa y una pila para llevar un registro de su posición.

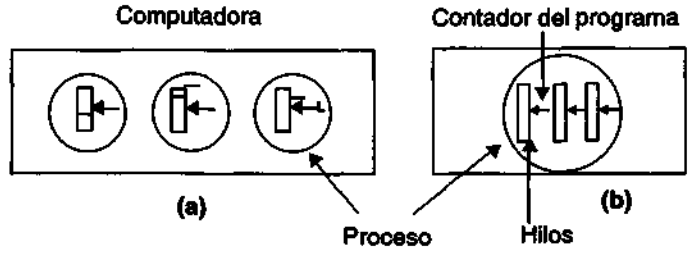


Figura 4.1 **(a).** *Tres procesos con un hilo cada uno.*
(b). *Un proceso con tres hilos.*

Los hilos comparten la CPU de la misma forma que lo realizan los procesos: primero se ejecuta un hilo y después otro (tiempo compartido).

Solamente teniendo un multiprocesador se podrían ejecutar realmente en paralelo.

Otra característica de los hilos es que éstos pueden crear hilos hijos y se pueden bloquear en espera de que se terminen los llamados al sistema, al igual que en los procesos regulares. Mientras un hilo se encuentre bloqueado se puede ejecutar otro hilo del mismo proceso (igualmente ocurre cuando se bloquea un proceso).

Claro está que los hilos no son tan independientes como ocurre en los procesos distintos; todos los hilos tienen el mismo espacio de direcciones, lo cual quiere decir que comparten las mismas variables globales. Cada hilo puede tener acceso a cada dirección virtual, y puede realizar las siguientes tareas: leer, escribir o limpiar de manera completa la pila de otro hilo. Como se puede observar, no existe ninguna protección entre los hilos, ya que podría llegar a ser casi imposible y, en muchos casos, hasta innecesario.

Además de poseer el mismo espacio de direcciones, todos los hilos comparten el mismo conjunto de archivos abiertos, procesos hijos, cronómetros, señales, etc., como se ilustra en la Figura 4.2.

Elementos por hilo
Contador del prog.
Pila
Conj. de registros
Hilos de los hijos
Estado

Elementos por proceso
Espacio de dirección
Variables globales
Archivos abiertos
Procesos fijos
Cronómetros
Señales
Semáforos
Información contable.

Figura 4.2 *Conceptos por hilo y por proceso*

En los procesos tradicionales (procesos con único hijo) los hilos pueden tener uno de los siguientes estados: en ejecución, bloqueado, listo o terminado. Un hilo en ejecución posee la CPU y se encuentra activo. Un hilo bloqueado espera que otro elimine el bloqueo (como ocurre en un semáforo). Un hilo listo está programado para su ejecución, la cual la realizará cuando llegue su debido turno. Y por último, un hilo terminado es aquél que ha hecho su salida pero que todavía no es recogido por su padre.

En la Figura 4.3 se ilustra un ejemplo de servidor de archivos, el cual muestra las diferentes formas de organizaciones de hilos en un proceso. En la parte (a) se muestra una primera organización en la que un hilo, el servidor, lee las solicitudes de trabajo en el buzón del sistema. Después de examinar la solicitud, él elige un hilo trabajador inactivo (bloqueado) y le envía la correspondiente solicitud, lo cual se realiza, con frecuencia, al escribir un apuntador el mensaje, en una palabra especial asociada a cada hilo, dando como resultado que el servidor despierte al trabajador dormido. Cuando el trabajador despierta, hace la correspondiente verificación por medio del bloque Caché compartido, el cual puede ser accesado por todos los hilos

y dirá si se puede satisfacer esa solicitud.

En el modelo de equipo, parte (b), todos los hilos son iguales y cada uno obtiene y procesa sus propias solicitudes. Como se puede observar en la figura, no hay servidor, y a veces llega un trabajo que un hilo no puede manejar, ya que cada hilo se especializa en manejar cierto tipo de trabajo. Para este caso, se puede manejar una cola de trabajo pendiente. Con este tipo de organización, un hilo debe verificar primero la cola de trabajo, antes de buscar en el buzón del sistema.

En la figura, parte (c), se ilustra el modelo de entubamiento, en el cual el primer hilo genera ciertos datos y los transfiere al siguiente, para su procesamiento. Los datos pasan de hilo en hilo y, en cada etapa, se lleva a cabo cierto tipo de procesamiento. Esto puede llegar a ser una buena solución en problemas como el de productores y consumidores, pero no sería adecuado para servidores de archivos. Los entubamientos se utilizan ampliamente en muchas áreas de los sistemas de cómputo, desde la estructura interna de la CPU RISC, hasta las líneas de comandos de UNIX.

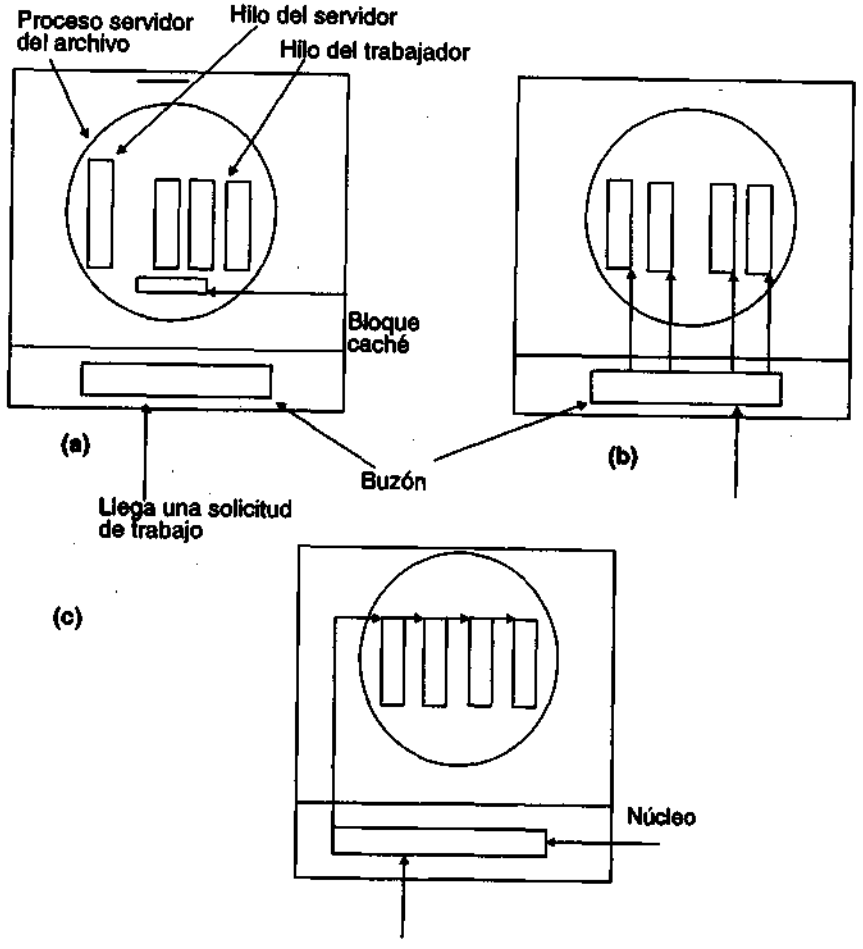


Figura 4-3. *Tres organizaciones de hilos en un proceso.*

4.2. Modelos de sistemas

Como ya es sabido, los procesos se ejecutan en procesadores; en un sistema tradicional sólo existe un único procesador, mientras que un sistema distribuido, dependiendo del aspecto del diseño, puede tener varios procesadores. Los procesadores de un sistema distribuido se pueden organizar de varias formas. En este parágrafo se tendrán en cuenta los principales, como son: el modelo de estación de trabajo y el modelo de pilas de procesadores. Estos dos modelos se basan en diferentes filosofías, en lo que concierne a un sistema distribuido.

4.2.1. *El modelo de estación de trabajo*

El modelo de estación de trabajo es un sistema que consta de estaciones de trabajo (computadoras personales de alta calidad), dispersas en un edificio o un lugar de trabajo y conectadas entre sí por medio de una LAN de alta velocidad, como se ilustra en la Figura 4.6.

SISTEMA BASADO EN EL MODELO DE LA PILA DE PROCESADORES

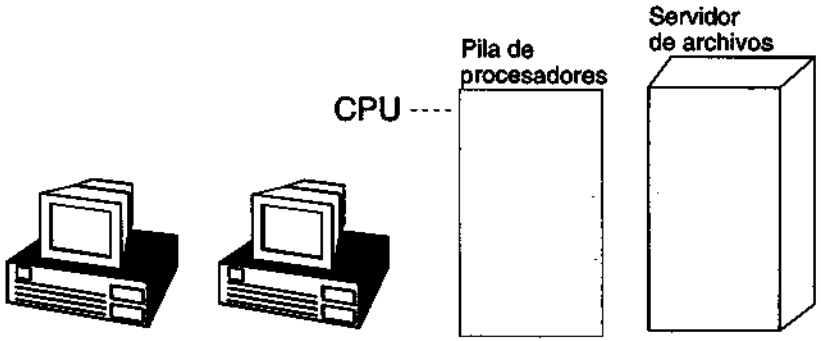


Figura 4.6

Las estaciones de trabajo pueden estar ubicadas en diferentes partes, como en las oficinas de trabajo, donde cada una de ellas se dedica a un único usuario; otras pueden estar en sitios públicos y tener distintos usuarios en el transcurso del día. Es posible que en algún instante de tiempo, ambas estaciones pueden tener un único usuario conectado a ella y tener, entonces, un "poseedor" o no tenerlo (estar inactiva).

Existen dos clases de estaciones de trabajo: estaciones de trabajo sin disco o estaciones de trabajo con disco. Si las estaciones carecen de disco, el sistema de archivos debe ser implantado mediante uno o varios servidores de archivos en la red. Las solicitudes de lectura o escritura serán enviadas a un servidor de archivos, el cual realizará el correspondiente trabajo y enviará de regreso las respuestas.

La popularidad de las estaciones de trabajo sin disco se debe a su precio, el cual es mucho más bajo que si se tuviese una estación equipada con un pequeño disco, razón por la cual se emplean uno o dos servidores de archivos con discos enormes y rápidos, a los cuales se tiene acceso mediante la LAN. Otras características que hacen populares a las estaciones de trabajo sin disco son: su fácil mantenimiento y su flexibilidad; por todo esto, la gran mayoría de estas estaciones se encuentra instalada en universidades y empresas.

Si la estación de trabajo tiene sus propios discos, podrán ser utilizados de las siguientes maneras:

1. Paginación y archivos temporales.

   En este modelo los discos locales se utilizan exclusivamente para la paginación y los archivos temporales no compartidos, que se puedan eliminar al final de la sesión.

2. Paginación, archivos temporales y binarios del sistema.

   Presenta las mismas características que el punto anterior, con la única variante que los discos locales también contienen los programas en binario (ejecutables), tales como compiladores, editores de texto y manejadores de correo electrónico. Cuando se llama a alguno de estos programas, se le busca en el disco local y no en el servidor de archivos, lo cual reduce la carga en la red.

3. Paginación, archivos temporales, binarios del sistema y ocultamiento de archivos.

   Este método consiste en utilizar, de forma explícita, los discos locales como "cachés" (además de utilizarlos para la paginación, los archivos temporales y los binarios). Los usuarios pueden cargar los archivos desde los servidores de archivos hasta sus propios discos; leerlos y escribir en ellos, de manera local y después regresar los archivos modificados, reduciendo la carga en la red, al mantener los archivos modificados al final de la sesión.

4. Un sistema local de archivos completo.

   En este caso cada máquina tiene su propio sistema de archivos autocontenido, con la gran posibilidad de acceder a los sistemas de archivos de otras máquinas, lo cual da como resultado reducir el contacto con el mundo exterior, generando así poca carga a la red, pero ocasionaría su principal desventaja, que será la de convertirse en un sistema operativo de red y no en un verdadero sistema distribuido.

Ventajas de las estaciones de trabajo

1. Los usuarios tienen una cantidad fija de poder de cómputo exclusivo, garantizando así su propio tiempo de respuesta.
2. Los programas gráficos sofisticados pueden ser muy rápidos, ya que pueden tener un rápido acceso a la pantalla.
3. Cada usuario tiene un alto grado de autonomía y puede asignar sus recursos correspondientes como lo juzgue necesario.
4. Los discos locales hacen posible que los trabajos continúen si el servidor de archivos llega a fallar.

4.2.2. *El modelo de la pila de procesadores*

El método de la pila de procesadores se ilustra en la Figura 4.7. Como se puede observar en el gráfico, en este modelo se les proporcionan terminales gráficas de alto rendimiento, denotadas en la figura con la letra X. Esta implementación se hace debido a que los usuarios desean tener una interface gráfica, de alta calidad y un buen desempeño.

La creación del modelo de la pila de procesadores surge para complementar la idea de las estaciones de trabajo sin disco. Como el sistema de archivos se concentra en un pequeño número de servidores, para economizar la escala, se tomó la idea de hacer lo mismo para los servidores de computadoras. Teniendo como base lo anterior, se colocan

todas las CPU en gabinetes de gran tamaño, dentro del cuarto de máquinas, reduciendo así los costos de energía y los de empaquetamiento, con lo cual se produce un mayor poder de cómputo por una misma cantidad de dinero, como también se permite el uso de terminales más baratas. Todo el poder de cómputo radica en poder tener "estaciones de trabajo inactivas", a las que se puede tener acceso de manera dinámica. Los usuarios podrán obtener las CPU que sean necesarias, durante períodos cortos, y después regresarlas para que otros usuarios puedan disponer de ellas.

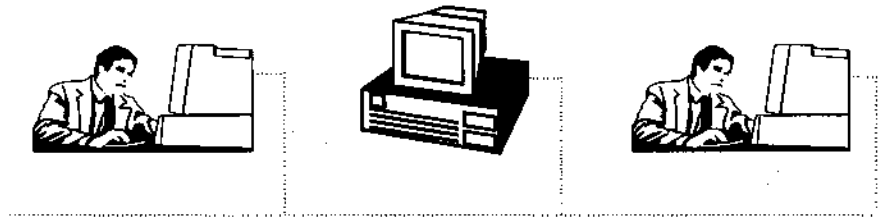


Figura 4.7 *Red de estaciones de trabajo personal.*

El poder de cómputo se ha centralizado como una pila de procesadores, debido a la teoría de colas, donde los usuarios generan, en forma aleatoria, solicitudes de trabajo a un servidor, y cuando éste se encuentre ocupado, los usuarios se forman para el servicio y se procesan de acuerdo con su turno, como se observa en la Figura 4.8.

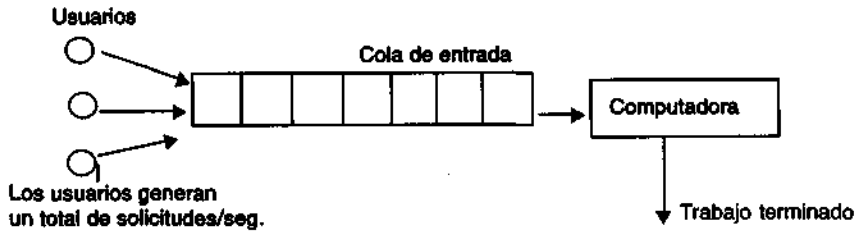


Figura 4.8. *Un sistema básico con colas.*

Una de las grandes aplicaciones de la pila de procesadores ocurre cuando varios usuarios se encuentran trabajando en un gran proyecto de software, los cuales llevan a cabo grandes simulaciones, ejecutan grandes programas de inteligencia artificial, obteniendo un fácil manejo y acceso al manejar una pila; mientras que el trabajar en estaciones de trabajo acarrearía enormes dificultades y no sería nada sencillo.

4.2.3. *Un modelo híbrido*

Este modelo es una combinación de los dos modelos anteriormente presentados, el cual combina las ventajas que posee cada uno, con la única diferencia de que resultaría mucho más costoso.

El modelo consiste en proporcionar a cada usuario una estación de trabajo personal y además tener una pila de procesadores. El trabajo interactivo se puede llevar a cabo en las estaciones de trabajo, garantizando su respuesta. Este modelo hace que las estaciones inactivas no sean utilizadas, lo cual hará más sencillo el sistema y además todos los procesos no interactivos se ejecutan en la pila de procesadores, así como todo el cómputo pesado en general.

La gran ventaja de este modelo es que proporciona una respuesta interactiva mucho más rápida, como también un uso eficiente en los recursos y presenta un diseño sencillo.

4.3. Asignación de procesadores

Como ya hemos mencionado anteriormente, un sistema distribuido consta de varios procesadores. Estos pueden organizarse como una colección de estaciones de trabajo personales, o como una pila pública de procesadores o de alguna forma híbrida. No importa cuál utilicemos, necesitaremos un algoritmo para decidir qué proceso se va a ejecutar y en qué máquina. Este parágrafo tendrá como principal objetivo estudiar los algoritmos que serán utilizados, para determinar qué proceso se asigna a un determinado procesador, dando como resultado una "asignación de procesadores".

4.3.1. *Modelos de asignación*

Las estrategias de asignación de procesadores se pueden dividir en dos grandes categorías. La primera, llamada no migratoria, al crearse un proceso, toma la decisión de dónde colocarlo. Una vez colocado el proceso en una máquina, permanece allí hasta que termine; no se puede mover, así exista sobrecarga para la máquina o existan otras máquinas inactivas. La segunda es la de los algoritmos migratorios, en donde un proceso se puede trasladar aunque haya iniciado su ejecución. Las estrategias migratorias permiten tener un mejor balance de carga, pueden resultar más complejas y tendrían un efecto fundamental en el diseño del sistema.

El objetivo que tienen los algoritmos de asignación de procesos, en los procesadores, es maximizar el uso de la CPU; es decir, maximizar el uso de ciclos de CPU que se ejecutan en beneficio de los trabajos del usuario por cada hora de tiempo real. Al tratar de maximizar el uso de la CPU, se trata de evitar todo el costo que se tiene por el tiempo inactivo de la CPU. Otro objetivo importante es la minimización del tiempo promedio de respuesta.

4.3.2. *Aspectos del diseño de algoritmos de asignación de procesadores*

Las principales decisiones que debe tomar un diseñador se pueden resumir en los siguientes aspectos:

1. Algoritmos determinísticos vs. Heurísticos. Los determinísticos son adecuados cuando se sabe de antemano todo aquello acerca de los procesos; se supone que se tiene una lista completa de todos los procesos o necesidades de cómputo, de archivos, de comunicación. Con esta información sería posible hacer una asignación perfecta, creando así una mejor asignación.

   Los heurísticos son aquellos sistemas en donde la carga es totalmente impredecible. Las solicitudes de trabajo dependen de quien las esté haciendo, y pueden variar de manera drástica cada hora, e incluso cada minuto. La asignación de procesadores, en estos sistemas, utiliza técnicas ad hoc.
2. Algoritmos centralizados vs. Distribuidos. La recolección de la información, en un lugar, permite tomar una mejor decisión, pero coloca una máquina pesada en la máquina central.

El hecho de ser centralizado, en vez de maximizar el uso de la CPU, busca darle a cada procesador, de una estación de trabajo, una parte justa del poder de cómputo.

3. Algoritmos óptimos vs. Subóptimos. Se pueden obtener soluciones óptimas tanto en los sistemas centralizados como en los descentralizados, pero, por regla general, son más caras que las subóptimas, pues hay que recolectar más información y procesarla un poco más. En la vida real, la mayoría de los sistemas distribuidos buscan soluciones subóptimas, heurísticas y distribuidas, debido a la dificultad para obtener las óptimas.
4. Algoritmos iniciados por el emisor vs. Iniciados por el receptor. También es llamado política de localización. Después de que la política de transferencia ha decidido deshacerse de un proceso, la política de localización debe decidir dónde enviarlo. Esta política no puede ser local, debido a que se necesita información de la carga, en todas partes, para tener una óptima decisión: sin embargo, esta información se puede dispersar de dos formas; en la primera, los emisores son quienes inician el intercambio de información y, en la segunda, es el receptor el que toma la iniciativa.

A continuación, en la Figura 4.9, mencionaremos un ejemplo sencillo para denotar el anterior caso:

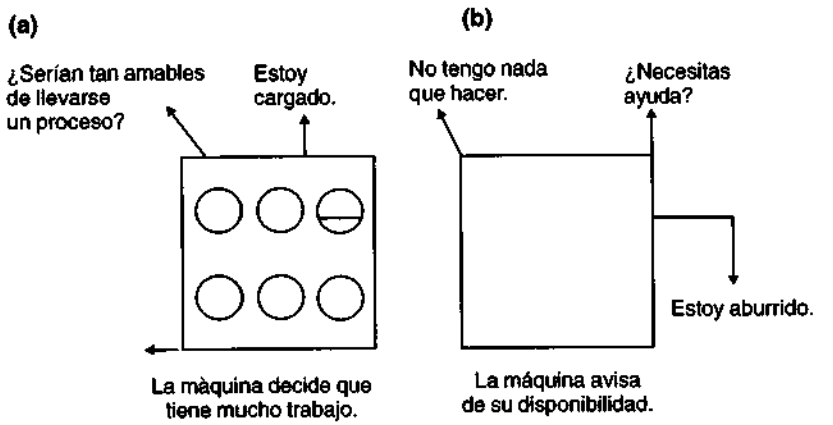


Figura 4.9 (a). *Un emisor en búsqueda de una máquina inactiva*
(b). *Un receptor en búsqueda de trabajo por realizar*

En la parte (a) se ilustra una máquina sobrecargada, la cual envía una solicitud de ayuda a las demás máquinas para que descarguen el nuevo proceso en otra. En este caso, el emisor es quien toma la iniciativa, para localizar más ciclos de la CPU.

En la parte (b), una máquina inactiva o sobrecargada que tiene poco trabajo y está preparada para más. Su principal objetivo es localizar una máquina dispuesta a darle trabajo.

5. Algoritmos locales vs. Globales. Este aspecto se relaciona con lo que se denomina política de transferencia. Debido a que cuando se está a punto de generar un proceso, hay que tomar una decisión para ver si dicho proceso se ejecuta o no en la máquina que lo genera; si esta máquina está muy ocupada, hay que transferir el nuevo proceso a otro lugar. Para el caso del algoritmo local, se hace la verificación en la máquina, para ver si ésta se encuentra por debajo de su nivel, para poder asignar el nuevo proceso; para el algoritmo global es mejor recolectar toda la información acerca de la carga, antes de decidir si la máquina puede o no recibir al proceso.

Los algoritmos locales son sencillos, pero no óptimos, mientras que los globales sólo dan un resultado mejor pero a un costo mayor.

5. SISTEMAS DISTRIBUIDOS DE ARCHIVOS

El corazón de cualquier sistema distribuido es el sistema distribuido de archivos. Como en el caso de los sistemas con un único procesador, la tarea del sistema de archivos, en los sistemas distribuidos, es almacenar los programas y los datos y tenerlos disponibles cuando sea necesario. En el caso de un sistema distribuido, es importante distinguir entre los conceptos de servicio de archivos y el servidor de archivos. El **servicio de archivos** es la especificación de los servicios que el sistema ofrece a sus clientes. Describe las primitivas disponibles, los parámetros que utilizan y las acciones que llevan a cabo.

Para los clientes, el servicio de archivos define con precisión el servicio con que pueden contar, pero no dice nada con respecto a su implantación. De hecho el servicio de archivos especifica la interfaz del sistema de archivos con los clientes.

Por el contrario, un **despachador de archivos** es un proceso que se ejecuta en alguna máquina y ayuda con la implantación del servicio de archivos. Un sistema puede tener uno o varios servidores de archivos, pero en un sistema distribuido con un diseño adecuado los clientes no deben ser conscientes de la forma de implantar el sistema de archivos. En particular, no deben conocer el número de servidores de archivos, su posición o función. Todo lo que saben es que, al llamar los procedimientos especificados en el servicio de archivos, el trabajo necesario se lleva a cabo de alguna manera y se obtienen los resultados solicitados. De hecho, los clientes ni siquiera deben darse por enterados de que el servicio de archivos es distribuido. Lo ideal es que se vea como un sistema de archivos normal, de un único procesador.

Puesto que un servidor de archivos es un proceso del usuario que se ejecuta en una máquina, un sistema puede contener varios servidores de archivos, cada uno de los cuales ofrece un servicio de archivos distinto. De esa forma es posible que una terminal tenga varias ventanas y que en algunas de ellas se ejecuten programas en UNIX y en otras programas en MS-DOS, sin que esto provoque conflictos.

Los diseñadores del sistema se encargan de que los servidores ofrezcan los servicios de archivo específicos, como UNIX o MS-DOS. El tipo y el número de servicios de archivo disponibles puede cambiar con la evolución del sistema.

5.1. Diseño de los sistemas distribuidos de archivos

En esta sección analizaremos las características de los sistemas distribuidos de archivos, desde el punto de vista del usuario.

Un sistema distribuido tiene dos componentes razonablemente distintos: el verdadero servicio de archivos y el servicio de directorios. El servicio de archivos se encarga de las operaciones en los archivos individuales, como la lectura, la escritura y la adición, en tan-

to que el servicio de directorios se encarga de crear y manejar directorios, añadir y eliminar archivos de los directorios, etc.

5.1.1. *La interfaz del servicio de archivos*

En muchos sistemas, como UNIX y MS-DOS, un archivo es una secuencia de bits sin interpretación alguna. El significado y la estructura de la información en los archivos queda a cargo de los programas de aplicación; esto es irrelevante para el sistema operativo.

Sin embargo, en los "mainframes" existen muchos tipos de archivos, cada uno con distintas propiedades. Por ejemplo, un archivo se puede estructurar como una serie de registros, con llamadas al sistema operativo, para leer o escribir en un registro particular.

Por lo general se puede especificar el registro mediante su número o el valor de cierto campo. En el segundo caso, el sistema operativo mantiene el archivo como un árbol B o alguna otra estructura de datos adecuada, o bien utiliza tablas de dispersión, para localizar con rapidez los registros.

Debido a que la mayoría de los sistemas distribuidos son planeados para ambientes UNIX o MS-DOS, la mayoría de los servidores de archivos soporta el concepto de archivo, como una secuencia de bits, en vez, de una secuencia de registros con cierta clave. Los archivos pueden tener **atributos**, que son partes de información relativas al archivo, pero que no son parte del archivo propiamente dicho. Los atributos típicos son el propietario, el tamaño, la fecha de creación y el permiso de acceso.

Por lo general, el servicio de archivos proporciona primitivas para leer y escribir alguno de los atributos.

Otro aspecto importante del modelo de archivo es si los archivos se pueden modificar después de su creación. Lo normal es que sí se puedan modificar, pero en algunos sistemas distribuidos, las únicas operaciones de archivo son CREATE y READ. Una vez creado un archivo, éste no podrá ser modificado, y se dice que este archivo es **inmutable.**

La protección en los sistemas distribuidos utiliza, en esencia, las mismas técnicas de los sistemas con un único procesador: **posibilidades y listas para control de acceso.** En el caso de las posibilidades, cada usuario tiene un cierto tipo de boleto, llamado **posibilidad,** para cada objeto al que tiene acceso. La posibilidad determina los tipos de acceso permitidos.

Todos los esquemas de **lista para control de acceso** le asocian a cada archivo una lista implícita o explícita de los usuarios que pueden tener acceso al archivo y los tipos de acceso permitidos para cada uno de ellos.

El esquema de UNIX es una lista para control de acceso, simplificada con bits que controlan la lectura, la escritura y la ejecución de cada archivo, en forma independiente para el propietario, el grupo de propietarios, y todas las demás personas.

Los servicios de archivos se pueden dividir en dos tipos: la dosificación depende de si soportan un **modelo carga/descarga** o un **modelo de acceso remoto.** En el modelo de carga/descarga, el servicio de archivo sólo proporciona dos operaciones principales: la lectura de un archivo y la escritura en un archivo.

La operación de lectura transfiere todo un archivo, de uno de los servidores de archivo, al cliente solicitante.

La operación de escritura transfiere todo un archivo, en sentido contrario; del cliente al servidor. Así, el modelo conceptual es el traslado de archivos completos, en alguna de las direcciones. Los archivos se pueden almacenar en la memoria o en un disco local.

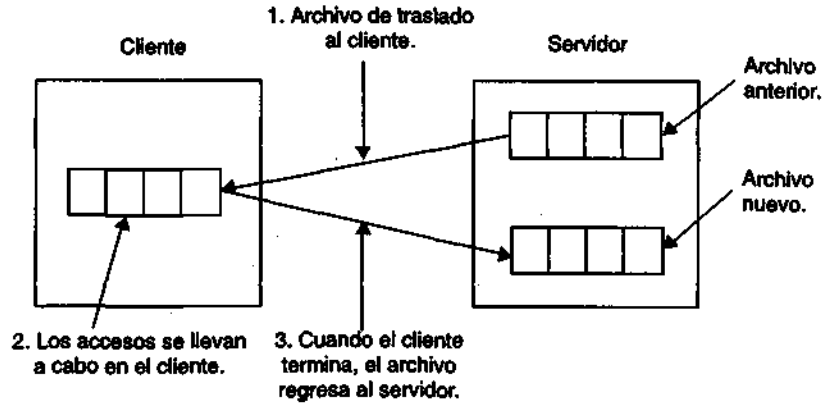


Figura 5.1. *El modelo carga/descarga.*

La ventaja de este modelo es la sencillez del concepto. Los programas de aplicación buscan los archivos que necesiten y después los utilizan de manera local. Los archivos modificados o nuevos se escriben, de regreso, al terminar el programa.

El otro tipo de servicio de archivos es el **modelo de acceso remoto.** En este modelo, el servicio de archivos proporciona un gran número de operaciones para abrir y cerrar archivos, leer y escribir partes de archivos, moverse a través de un archivo, examinar y modificar los atributos de un archivo, etc.

Mientras que en el modelo carga/descarga el servicio de archivos sólo proporciona el almacenamiento físico y la transferencia, en este caso el sistema de archivos se ejecuta en los servidores y no en los clientes. Su ventaja consiste en que no necesita mucho espacio por parte de los clientes, a la vez que elimina la necesidad de transferir archivos completos cuando sólo se necesita una parte de ellos.

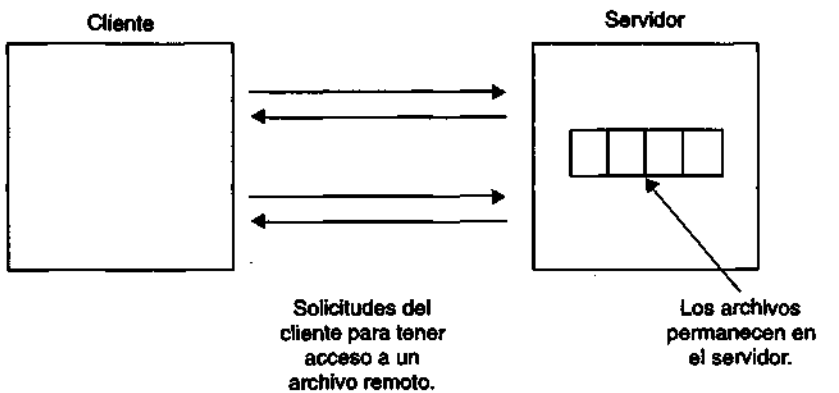


Figura 5.2. *El modelo de acceso remoto.*

5.1.2. *La interfaz del servidor de directorios*

La otra parte del servicio de archivos es el servicio de directorios, el cual proporciona las operaciones para crear y eliminar directorios, nombrar o cambiar el nombre de los archivos y mover éstos de un directorio a otro.

El servicio de directorios define un alfabeto y una sintaxis para formar los nombres de archivos y directorios. Lo común es que los nombres de archivos tengan de y hasta cierto número máximo de letras, números y ciertos caracteres especiales. Algunos sistemas dividen los nombres de archivo en dos partes, usualmente separadas mediante un punto. La segunda parte del nombre, llamada la extensión del archivo, identifica el tipo de éste. Otros sistemas utilizan un atributo explícito para este fin, en vez de utilizar una extensión dentro del nombre.

Todos los sistemas distribuidos permiten que los directorios contengan subdirectorios, para que los usuarios puedan agrupar los archivos relacionados entre sí. De acuerdo con esto, se dispone de operaciones para la creación y eliminación de directorios, así como para introducir, eliminar y buscar archivos en ellos. También, los subdirectorios pueden contener sus propios subdirectorios y así sucesivamente, lo que conduce a un árbol de directorios, el cual es conocido como **sistema jerárquico de archivos.** La Figura 5.3 muestra un árbol de cinco directorios:

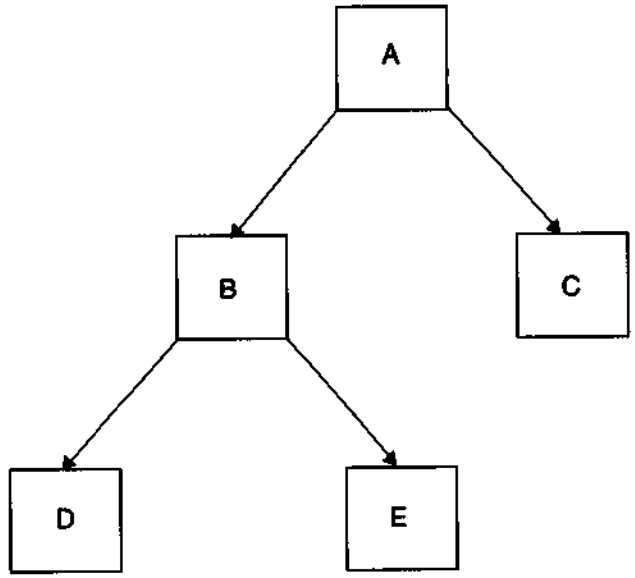


Figura 5.3. *Un árbol de directorios contenido en una máquina.*

En ciertos sistemas es posible crear enlaces o apuntadores hacia un directorio arbitrario. Estos se pueden colocar en cualquier directorio, lo que permite construir no sólo árboles sino gráficas arbitrarias de directorios, las cuales son más poderosas. La distinción entre árboles y gráficas es de particular importancia en un sistema distribuido.

La naturaleza de la dificultad se puede apreciar más claramente en la Figura 5.4. En esta gráfica, el directorio *D* tiene un enlace con el directorio *B*. El problema aparece cuando se elimina el enlace de *A* a *B*. En una jerarquía con estructura de árbol sólo se puede eliminar un enlace con un directorio, si el directorio al cual apunta es vacío. En una gráfica se permite la eliminación de un enlace mientras exista al menos otro. Mediante un contador de referencias, el cual se muestra en la esquina superior derecha de cada directorio de la figura, se puede determinar si el enlace por eliminar es el último. Después de eliminar el enlace de *A* a *B*, el contador de referencias de *B* se reduce de 2 a 1, lo cual está bien sobre el papel. Sin embargo, ahora no es posible llegar a *B* desde la raíz del sistema de archivos (A). Los tres directores *B, D* y *E* y todos sus archivos se convierten en huérfanos.

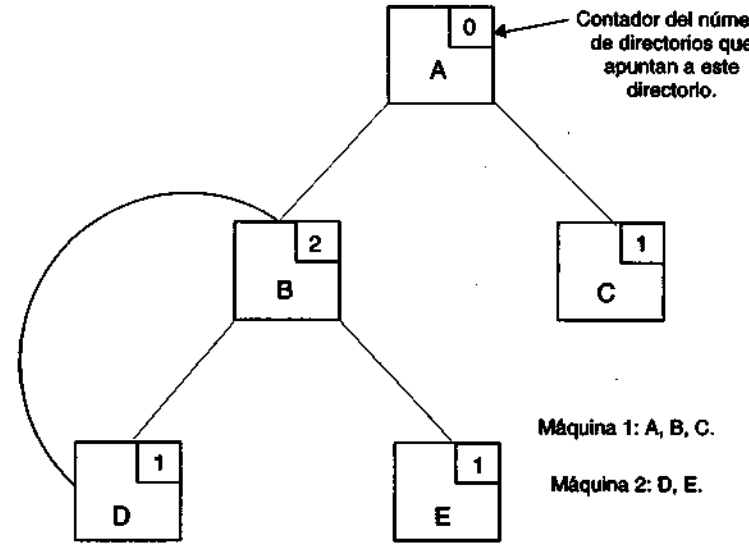


Figura 5.4. *Una gráfica de directorios en dos máquinas.*

Este problema también existe en los sistemas centralizados, pero es más serio en los distribuidos. Si todo está en una máquina, es posible descubrir los directorios huérfanos, puesto que toda la información está en un solo lugar. Se puede detener toda la actividad de los archivos y recorrer la gráfica desde la raíz, para señalar todos los directorios alcanzables. Al final de este proceso, se sabe que todos los directorios no marcados son inalcanzables. En un sistema distribuido existen varias máquinas y no se puede detener toda la actividad, por lo que es supremamente difícil tomar una foto instantánea.

Otro aspecto fundamental, en el diseño de cualquier sistema distribuido de archivos, es si todas las máquinas y procesos deben tener la misma visión de la jerarquía de los directorios.

5.1.2.1. Nombres de los archivos y directorios

Existen tres métodos usuales para nombrar los archivos y directorios en un sistema distribuido:

1. Nombre de la máquina + ruta de acceso, como */máquina/ruta o máquina:ruta.*
2. Montaje de sistemas de archivos remotos en la jerarquía local de archivos.

3. Un único espacio de nombres que tenga la misma apariencia en todas las máquinas.

Los primeros dos son fáciles de implantar, especialmente como una forma de conectar sistemas ya existentes, que no estaban diseñados para su uso distribuido. El tercer método es difícil y requiere de un diseño cuidadoso, pero es necesario si se quiere lograr el objetivo de que el sistema distribuido actúe como una única computadora.

5.1.2.2. Nombres de dos niveles

La mayoría de los sistemas distribuidos utiliza cierta forma de nombres con dos niveles. Los archivos tienen **nombres simbólicos** para el uso de las personas, pero también pueden tener **nombres binarios** internos para uso del propio sistema. Lo que los directorios hacen, en realidad, es proporcionar una asociación entre estos dos nombres. Para las personas y los programas, es conveniente utilizar nombres simbólicos, pero para el uso dentro del propio sistema, estos nombres son demasiado grandes y difíciles. Así, cuando un usuario abre un archivo o hace referencia a un nombre simbólico, el sistema busca de inmediato el nombre simbólico, en el directorio apropiado, para obtener el nombre binario, el cual utilizará para localizar realmente el archivo. A veces los nombres binarios son visibles a los usuarios y a veces no.

Un esquema más general para los nombres es que el nombre binario indique el servidor y un archivo específico en ese servidor. Este método permite que un directorio en un servidor contenga un archivo en un servidor distinto.

Otra alternativa, que a veces es preferible, es utilizar un **enlace simbólico,** el cual es una entrada de directorio asociada a una cadena (servidor, nombre de archivo), la cual se puede buscar en el servidor correspondiente para encontrar el nombre binario. El propio enlace simbólico es sólo el nombre de una ruta de acceso.

Otra idea más es utilizar las posibilidades como los nombres binarios. En este método, la búsqueda de un nombre en ASCII produce una posibilidad, la cual puede tomar varias formas. Un último método, que a veces está presente en un sistema distribuido pero casi nunca en uno centralizado, es la posibilidad de buscar un nombre en ASCII y obtener no uno sino varios nombres binarios. Por lo general, éstos representan al archivo original y todos sus respaldos.

5.1.3. *Semántica de los archivos compartidos*

Si dos o más usuarios comparten el mismo archivo, es necesario definir con precisión la semántica de la lectura y la escritura para evitar problemas.

Existen cuatro maneras de compartir archivos en un sistema distribuido. La primera que analizaremos es conocida como **semántica de UNIX** y consiste en que cada operación en un archivo es visible a todos los procesos de manera instantánea. En los sistemas con un único procesador que permiten a los procesos compartir archivos, como UNIX, la semántica establece que si una operación READ sigue después de una operación WRITE, READ regresa el valor recién escrito. De manera análoga, cuando dos WRITE se realizan en serie y después se ejecuta un READ, el valor que se lee es el almacenado en la última escritura. De hecho, el sistema impone en todas las operaciones un orden absoluto con respecto del tiempo y siempre regresa el valor más reciente. En un sistema con un único procesador, esto es muy fácil de comprender y tiene una implantación directa. En un sistema distribuido, la semántica de UNIX se puede lograr fácilmente, mientras sólo

exista un servidor de archivos y los clientes no oculten los archivos. Todas las instrucciones READ y WRITE pasan en forma directa al servidor de archivos, que los procesa en forma secuencial. Este método proporciona la semántica de UNIX, excepto por un problema menor: los retrasos en la red pueden hacer que un READ, ocurrido un microsegundo después de un WRITE, llegue primero al servidor y que obtenga el valor anterior.

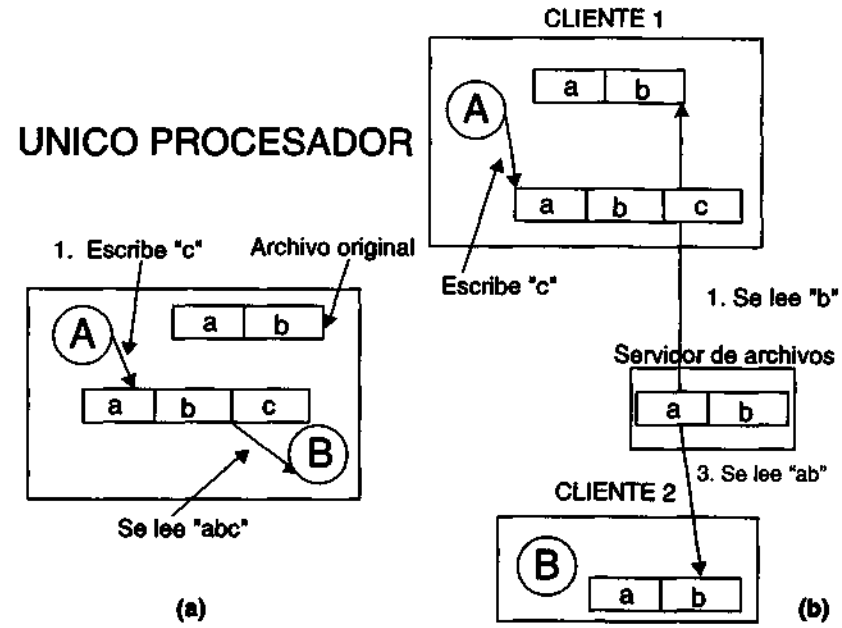


Figura 5.5 **(a).** *En un solo procesador, cuando una READ va después de un WRITE, el valor regresado por READ es el valor recién escrito.*

(b) *En un sistema distribuido con ocultamiento, el valor regresado puede ser obsoleto.*

El segundo método se llama **semántica de sesión,** el cual se caracteriza porque ningún cambio es visible a otros procesos, hasta que el archivo se cierra; en vez de pedir que un READ vea los efectos de todos los WRITE anteriores, uno puede tener una nueva regla que diga: "Los cambios a un archivo abierto sólo pueden ser vistos en un principio por el proceso (o la máquina) que modificó el archivo. Los cambios serán visibles a los demás procesos (o máquinas) sólo cuando se cierre el archivo".

Un método completamente distinto de la semántica de los archivos compartidos, en un sistema distribuido, es que todos los archivos sean **inmutables.** Así no existe forma de abrir un archivo para escribir en él. En efecto, las únicas operaciones en los archivos son CREATE y READ. No existen actualizaciones; es más fácil compartir y replicar.

Una cuarta vía para enfrentar el uso de los archivos compartidos en un sistema distribuido es usar las transacciones atómicas. Para tener acceso a un archivo o grupo de archivos, un proceso

ejecuta en primer lugar cierto tipo de primitiva BEGINTRANSACTION, para señalar que lo que sigue debe ejecutarse de manera indivisible. Después vienen las llamadas al sistema para leer o escribir en uno o más archivos. Al terminar el trabajo, se ejecuta una primitiva ENDTRANSACTION. La propiedad fundamental de este método es que el sistema garantiza que todas las llamadas contenidas dentro de la transacción se llevarán a cabo en orden, sin interferencias de otras transacciones concurrentes. Si dos o más transacciones se realizan al mismo tiempo, el sistema garantiza que el resultado final es el mismo que si se ejecutasen en cierto orden secuencial (indeterminado).

5.2. Implantación de un sistema distribuido de archivos

En la sección anterior describimos varios aspectos de los sistemas distribuidos de archivos desde el punto de vista del usuario. En esta sección veremos la forma en que se implantan dichos sistemas.

5.2.1. *Uso de archivos*

Satyanarayanan (1981) hizo un amplio estudio sobre los patrones de uso de los archivos. Vamos a presentar, a continuación, las principales conclusiones de este estudio. Para comenzar, la mayoría de los archivos está por debajo de los 10K; esta observación sugiere la factibilidad de la transferencia de archivos completos, en vez de bloques de disco, entre el servidor y el cliente. Puesto que la transferencia de archivos completos es más sencilla y eficiente, esta idea debe tomarse en cuenta. Por supuesto, algunos archivos son de gran tamaño, por lo que también se deben tomar medidas preventivas con respecto a éstos.

Una observación interesante es que la mayoría de los archivos tiene tiempos de vida cortos. En otras palabras, un patrón común es crear un archivo, leerlo y después eliminarlo.

El hecho que unos cuantos archivos se compartan, argumenta en favor del ocultamiento por parte del cliente. Como ya se ha visto, el ocultamiento complica la semántica, pero si es raro el uso de los archivos compartidos, podría ser mejor el ocultamiento por parte del cliente y aceptar las consecuencias de la semántica de sesión en favor de un mejor desempeño.

Para terminar, la existencia de distintas clases de archivos sugiere que, tal vez, se deberían utilizar mecanismos diferentes para el manejo de las distintas clases. Los binarios del sistema necesitan estar presentes en diferentes partes, pero es raro que se modifiquen, por lo que tal vez se podrían duplicar en varias partes, aun cuando esto implica una actualización ocasional compleja.

Los compiladores y los archivos temporales son cortos, no compartidos y desaparecen con rapidez, por lo que deben mantener su carácter local mientras sea posible. Los buzones electrónicos se actualizan con frecuencia, pero es raro que se compartan, por lo que su duplicación no sirve de mucho.

5.2.2. *Estructura del sistema*

En esta sección analizaremos algunas de las formas de organización interna de los servidores de archivos y directorios, con atención especial a los métodos alternativos. Comenzaremos con una sencilla pregunta:"¿Son distintos los clientes y los servidores?"

En ciertos sistemas no existe distinción alguna entre un cliente y un servidor. Todas las máquinas ejecutan el mismo software básico, de manera que una máquina que desee dar servicio de archivos al público en general es libre de hacerlo. Este ofrecimiento del servicio de archivos consiste sólo en exportar los nombres de los directorios seleccionados, de modo que otras máquinas tengan acceso a ellos. En otros sistemas el servidor de archivos y el de directorios sólo son programas del usuario, por lo que se puede configurar un sistema para que ejecute o no el software, del cliente o del servidor, en la misma máquina, como se desee. Y en el otro extremo, están los sistemas donde los clientes y los servidores son máquinas esencialmente distintas, ya sea en términos de hardware o de software. Los servidores y clientes pueden ejecutar incluso versiones diferentes del sistema operativo. Aunque la separación de funciones es un poco más transparente, no existe una razón fundamental para preferir un método por encima de los demás.

Un segundo aspecto de la implantación, en donde difieren los sistemas, es la forma de estructurar el servicio a archivos y directorios. Una forma de organización consiste en combinar ambos en un único servidor, que maneje todas las llamadas a directorios y archivos. Sin embargo, la otra posibilidad es separarlos. En este caso, la apertura de un archivo exige ir hasta el servidor de directorios, para asociar su nombre simbólico con el nombre binario, y después ir hasta el servidor de archivos, con el nombre en binario y llevar a cabo la lectura o escritura real del archivo.

El argumento en favor de la separación es que las dos funciones no tienen una relación real entre sí, por lo que es más flexible mantenerlas separadas. Por ejemplo, se puede implantar un servidor de directorios en MS-DOS y otro servidor de directorios en UNIX, donde ambos utilicen el mismo servidor de archivos para el almacenamiento físico. También es probable que la separación de funciones produzca un software más sencillo. Un contraargumento es que el hecho de contar con dos servidores requiere de una mayor comunicación.

El aspecto estructural final por considerar es si los servidores de archivos, directorios o de otro tipo deben contener la información sobre el estado de los clientes. Este aspecto tiene una controversia moderada, donde existen dos escuelas de pensamiento en competencia. Una escuela piensa que los servidores no deben contener los estados, es decir, "ser sin estado". En otras palabras, cuando un cliente envía una solicitud a un servidor, éste la lleva a cabo, envía la respuesta y elimina de sus tablas internas toda la información relativa a dicha solicitud. El servidor no guarda información alguna relativa a los clientes entre las solicitudes. La otra escuela de pensamiento sostiene que es correcto que los servidores conserven la información de estado de los clientes entre las solicitudes.

En el caso de un servidor sin estado, cada solicitud debe estar autocontenida. Debe contener todo el nombre del archivo y el ajuste dentro de éste, para que el servidor pueda realizar el trabajo. Esta información aumenta la longitud del mensaje.

Otra forma de ver la información de estado es considerar lo que ocurre si un servidor falla y todas sus tablas se pierden de manera irremediable. Al volver a arrancar el servidor, éste ya no tiene idea de la relación entre los clientes y los archivos abiertos por éstos. Fracasarán entonces los intentos posteriores por leer y escribir en archivos abiertos y la recuperación, entonces, queda a cargo totalmente de los clientes. En consecuencia, los servidores sin estado tienden a ser más tolerantes a los fallos que los que mantienen los estados, lo cual es un argumento a favor de los primeros.

En general, las ventajas de los servidores sin estado son: tolerancia a las fallas; no necesita llamadas open/close; no se desperdicia el espacio del servi-

dor en las tablas; no existe límite para el número de archivos abiertos y no hay problemas si un cliente falla. Las ventajas de los servidores con estado consisten en un mejor desempeño, los mensajes de solicitud más cortos, es posible la lectura adelantada, es más fácil la idempotencia y es posible la cerradura de archivos.

5.2.3. *Ocultamiento*

En un sistema cliente-servidor, cada uno con su memoria principal, y un disco, existen cuatro lugares donde se pueden almacenar los archivos o partes de archivos: el disco del servidor, la memoria principal del servidor, el disco del cliente o la memoria principal del cliente, como se muestra en la Figura 5.6.

El lugar más directo para almacenar todos los archivos es el disco del servidor. Ahí existe mucho espacio y los archivos serían accesibles a todos los clientes. Además, con sólo una copia de cada archivo no surgen problemas de consistencia.

El problema con el uso del disco del servidor es el desempeño. Antes de que un cliente pueda leer un archivo, éste debe ser transferido primero del disco del servidor a la memoria principal del servidor y luego, a través de la red, a la memoria principal del cliente. Ambas transferencias tardan cierto tiempo.

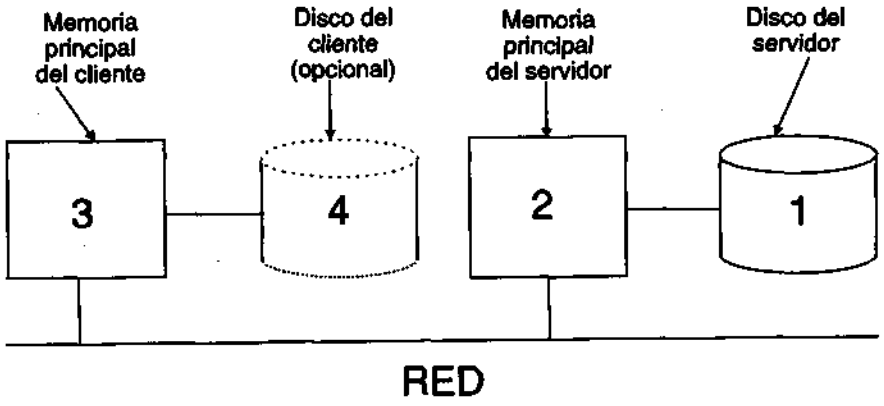


Figura 5.6. *Cuatro lugares para guardar archivos o partes de ellos.*

Se puede lograr un desempeño mucho mejor si se ocultan (es decir, se conservan) los archivos de más reciente uso en la memoria principal del servidor. Un cliente que lea un archivo ya presente en el caché del servidor elimina la transferencia del disco, aunque se deba realizar la transferencia a la red.

Puesto que la memoria principal siempre es menor que el disco, se necesita un algoritmo para determinar los archivos o partes de archivos que deben permanecer en el caché.

Este algoritmo debe resolver dos problemas. El primero es el de la unidad que maneja el caché. Puede manejar archivos completos o bloques del disco. Si se ocultan los archivos completos, éstos se pueden almacenar en forma adyacente en el disco, lo cual permite transferencias a alta velocidad entre la memoria y el disco, así como un buen desempeño en general. Sin embargo, el ocultamiento de bloques de disco utiliza el caché y el espacio en disco en forma más eficiente.

El segundo es que el algoritmo debe decidir qué hacer si se utiliza toda la capacidad del caché y hay que eliminar a alguien. Aquí se puede utilizar cualquiera de los algoritmos comunes de ocultamiento, pero como las referencias al caché son poco frecuentes, comparadas con las referencias a la memoria, por lo general es factible una implantación exacta de LRU mediante listas ligadas: Cuando hay que eliminar a alguien de la memoria, se elige al más antiguo. Si existe una copia actualizada en el disco, simplemente se descarta la copia del caché. En caso contrario, primero se actualiza el disco.

El mantenimiento de un caché en la memoria principal del servidor es fácil de lograr y es totalmente transparente a los clientes. Puesto que el servidor puede mantener sincronizadas sus copias en memoria y en disco, desde el punto de vista de los clientes sólo existe una copia de cada archivo, por lo que no hay problemas de consistencia.

5.2.4. *Réplica*

Con frecuencia los sistemas distribuidos de archivos proporcionan la réplica de archivos como servicio a sus clientes. En otras palabras, se dispone de varias copias de algunos archivos, donde cada copia está en un servidor de archivos independiente. Las principales razones para la existencia de tal servicio son:

1. Aumentar la confiabilidad al disponer de respaldos independientes de cada archivo. Si un servidor falla o se pierde en forma permanente, no se pierden los datos.
2. Permitir el acceso a un archivo, aun cuando falle un servidor de archivos. El fallo de un servidor no debe hacer que todo el sistema se detenga hasta cuando aquél se pueda volver a arrancar.
3. Repartir la carga de trabajo entre varios servidores. Al crecer el tamaño del sistema, el hecho de tener todos los archivos en un único procesador se puede convertir en un cuello de botella. Con varios archivos duplicados en dos o más servidores, se puede utilizar el que tenga menos carga.

Las dos primeras se relacionan con el mejoramiento de la confiabilidad y la disponibilidad; la tercera se refiere al desempeño.

Existen tres formas de llevar a cabo la réplica: la réplica explícita de archivos, la réplica retrasada de archivos y la réplica de archivos mediante un grupo.

CONCLUSIONES

- Los Sistemas Distribuidos son la tecnología que se va a imponer en el mundo, debido a la economía que se alcanza con su implantación en las grandes organizaciones.
- No existen todavía aplicaciones distribuidas para los pocos sistemas operativos que tratan de manejar la distribución, como Windows NT.
- La diferencia más importante entre un Sistema Distribuido y un sistema de un único procesador es la comunicación entre procesos.
- Para la organización de los procesadores se pueden tener en cuenta dos modelos básicos: el modelo de estación de trabajo y el de la pila de procesadores. Las ventajas que presenta cada uno son muy importantes, ya que le permiten tener a cada usuario su propia estación de trabajo y aprovechar las estaciones de trabajo inactivas para ejecutar procesos; en las pilas de procesadores, todas las instalaciones son un recurso compartido que puede ser usado dependiendo de las necesidades de cada usuario.

- Cuando se tiene una colección de procesadores para que éstos sean asignados a los procesos, se debe tener la ayuda de los algoritmos de asignación de procesadores.
- El sistema distribuido de archivos es el corazón de cualquier Sistema Distribuido y su tarea consiste en almacenar los programas y los datos y tenerlos disponibles cuando sea necesario.
- La implantación de un sistema distribuido de archivos implica la toma de varias decisiones: ver si el sistema es con estado o sin él, si se debe hacer el ocultamiento y la forma de manejar la réplica de archivos.
- Definitivamente nos encontramos en la era de las comunicaciones y, debido a que toda actividad administrativa se lleva a cabo por medio de comunicaciones orales o escritas, en muchos casos entre interlocutores que se encuentran a grandes distancias, se hace necesario contar con redes de terminales inteligentes que permitan a dichos interlocutores comunicarse en forma confiable y poder compartir la información de manera transparente.

BIBLIOGRAFIA

- *Sistemas operativos modernos*, Andrew S. Tanembaum. Editorial Prentice-Hall Hispanoamericana S.A.; México, 1992.
- *Internet-Working With TCP/IP.* Volumen III. Client-Server Programming and Applications.
  Douglas E. Corner and David L. Stevens. Department of Computer Sciences. Purdue University.
  Prentice-Hall Internacional, Inc.

LA ULTIMA LECCIÓN*

DISCURSO DE GRADO
Promoción Decimosegunda
Cali, 11 de febrero de 1995

ALFONSO OCAMPO LONDOÑO
Rector del ICESI

Volvemos hoy con gran alegría, pero al mismo tiempo con nostalgia, a graduar a 154 de nuestros alumnos, 95 en Pregrado en Administración de Empresas e Ingeniería de Sistemas e Informática y 59 de Postgrado, y entre éstos la primera promoción de la especialización en Ingeniería del Software. Con ellos ya conformamos un cuerpo de cerca de 3.000 egresados que hemos entregado a la sociedad, a las empresas, a sus compañías familiares y las propias.

Hemos, desde que ustedes iniciaron sus estudios, recalcado que es un gran privilegio llegar a una Universidad, ya que sólo dos de cada cien que inician la primaria lo logran y es aún mayor privilegio el de hacer formación avanzada o de postgrado. Se tiene, pues, una deuda social que tiene que pagarse, a sus padres, familiares, amigos, patrones, empresas, a quienes les han servido, y en general a la sociedad y al Estado que les han dado las condiciones o infraestructura social para poder llegar a esta importante etapa de sus vidas. Esta deuda la deben pagar creando riqueza en Colombia y participando en la mejoría de sus condiciones de competitividad en un mundo que se ha globalizado o internacionalizado y en el cual la competencia no va a ser sólo en productos, sino en la calidad del talento humano que se tiene y que es nuestra tarea. Por ello, en este acto, en un punto especial y muy emotivo, los graduandos le hacen un homenaje de gratitud a sus padres, cónyuges y amigos, que los han ayudado.

En el mundo de hoy, lo que se considera la mayor riqueza es el conocimiento y la información y por lo tanto la formación del ser humano, es la tarea más importante para hacer. Las riquezas naturales contribuyen al bienestar, solamente si hay un equipo humano que las sepa aprovechar, para que sus beneficios se extiendan a todos y no sólo a unos pocos. Lo que debe importar es el beneficio para la totalidad de la comunidad y no el aprovechar todo, sólo

* Este discurso continúa una tradición universitaria en que el rector da la primera y la última lección.

para uno mismo, aunque también éste se debe lograr.

La misión que nos hemos propuesto y que ha sido señalada por los fundadores del ICESI, es la de formar integralmente a nuestros estudiantes con un concepto de "excelencia", "alta calidad académica y cultura internacional", propender al "desarrollo del sistema democrático, la libre empresa y la propiedad privada" dentro de sus obligaciones sociales y con una "integridad", o sea una "ética" en su comportamiento y en todos sus actos. Lo que queremos formar son líderes empresariales con una mística de creación, promoción, expansión, innovación, gestión y mejoramiento de las organizaciones, con una visión positiva y amplia de su tarea en la sociedad, que conviertan sus sueños en acción y en la cual sus valores humanos y éticos sean la guía de su vida y de su trabajo en equipo para lograr una mejor calidad de vida propia en la región y el país. Con ello creemos que, con nuestros graduados, le damos a la comunidad un grupo selecto de personas conscientes de su misión y de sus obligaciones y que con seguridad van a trabajar por un desarrollo completo de la sociedad y país en que viven, que contribuirán a su felicidad y prosperidad.

Quiero, de manera especial, convocar a los egresados de hoy y a los anteriores, para que conformen un equipo de gran calidad y de múltiples conocimientos para que fortalezcan la Asociación de Egresados, dándole una gran categoría y dinamismo, para que se ayuden entre sí, en la comunidad general y empresarial en la que viven. Tienen todos los graduados un gran número de conocimientos y por lo tanto pueden hacer, si se unen, una gran Asociación para su beneficio y el de sus compañeros. En este año vamos a poner un especial esfuerzo para que esta Asociación se desarrolle, y se ha previsto para ello el que la Oficina que los ayuda lo sea de tiempo completo y abarque hasta las primeras horas de la noche. Esta Asociación debe ser también un agradable sitio de reencuentro con sus amigos de una de las más bellas épocas de la vida, que es la de la Universidad. Hagamos un esfuerzo conjunto para mantener viva la llama del afecto entre sus compañeros y la Institución que les dio la posibilidad de conocerse.

Una de las más importantes lecciones, que espero hayan aprendido en esta Universidad, que quiero recalcar hoy, es la de haber aprendido a aprender, o sea, ser un estudiante toda la vida. Esto es absolutamente necesario si se quiere progresar y avanzar. El conocimiento, vuelvo a repetirlo, es la mayor riqueza que una persona puede tener, pero tiene un continuo proceso de crecimiento. Se cree que hoy hay en el mundo tantos investigadores como en el tiempo anterior de la humanidad y que el saber se está duplicando cada cinco años, aunque en algunos casos, como en el de la informática, computadores y comunicaciones se hace mucho más rápidamente. Por lo tanto, quien se detiene en su aprendizaje, retrocede. Para quienes tienen este convencimiento y hacen los esfuerzos necesarios para actualizarse permanentemente, el futuro está asegurado y no hay que angustiarse por él. Sin embargo, debo ser claro que para tener una vida verdaderamente completa hay que desarrollar también su propia personalidad y tener una concepción intelectual, buena formación general y espiritual de la vida y de su destino final que es Dios. Este es el principal mensaje que les doy hoy en esta última lección del claustro universitario, en el cual hemos tenido el privilegio de tenerlos con nosotros.

Este año será sin duda de gran trascendencia y crucial para el desarrollo del ICESI, para lograr y mantener un puesto de excelencia y cumplir el desafío que tenemos de ser los mejores. Hemos formulado un plan de desarrollo entre los años 1994 al nuevo siglo, que hemos denominado ICESI 2000, para fortalecer el área académica con un mejoramiento continuo en docencia, investigación y servicio o asesoría y para facilitar el estudio y servicios de computación para los profesores y estudiantes y de la Institución. Tenemos para ello varios programas que son: 1o.) la preparación de docentes de tiempo completo en el exterior y en el país y en la propia Institución o en la región; 2o.) un programa intenso de mejoramiento continuo, o sea de calidad total, y en especial una comparación con las mejores universidades de Colombia y el exterior que se denomina en el lenguaje de la calidad "benchmarking" y la continuación del estudio de "congruencia curricular" para saber si estamos cumpliendo con la labor de que enseñamos, lo que decimos que hacemos, complementado con el desarrollo del diseño instruccional de cada área para lograr dicho fin; 3o.) la construcción de una Biblioteca y Centro de Cómputo modernos con un área de 3.500 m²; 4o.) la dotación y modernización de los computadores de varias de las salas, ya que tenemos el convencimiento y el propósito de que cada tres años debemos cambiarlos debido a la rápida obsolescencia de los mismos; 5o.) Utilización en docencia por medio de programas educativos y cursos del exterior, que se captarán a través de la antena parabólica que acabamos de instalar y que abrirá una nueva era de transmisión de conocimientos por medio de la reciente tecnología de la educación que, sin duda, va a convertirse en una verdadera reinvención de la enseñanza. Ya estamos afiliados a la red de Televisión Iberoamericana por medio del satélite español Hispasat y posteriormente lo haremos a otras; 6o.) mejorar todo el aspecto de comunicaciones internas y externas mediante la sistematización de la entidad, la cual abarcará todas las áreas de la Institución; 7o.) hemos solicitado la inscripción a la red de datos más grande que existe en el mundo, la Internet, para uso de estudiantes y profesores y que complementará el Dialog que ya tenemos. Esto convierte la biblioteca en una de carácter mundial y le abrirá caminos de comunicación internacional a la institución; 8o.) reorganización de la Oficina de Egresados, lo cual ya mencioné; 9o.) la organización de un padrinazgo académico, solicitando que todo Decano, Directivo, Jefe de Oficina y Profesor de tiempo completo y los de tiempo parcial que quieran, tenga a su cargo un grupo de alumnos que los siga a través del año respectivo y carrera. Esta es una experiencia que tiene y pude ver en acción en Israel en la Universidad de Tel Aviv y que ha dado excelentes resultados.

Este es un programa cuyo costo total excede de los tres mil seiscientos millones de pesos, que esperamos financiar con recursos propios y la ayuda generosa de la comunidad empresarial y privada, ya que la institución no recibe ayuda alguna del Estado y todo lo que hace es para mejorar la formación del recurso humano que necesita la empresa. Esta inversión en capital humano es la mejor que puede hacer una empresa para su presente y futuro. De acuerdo con la política acordada por el Consejo Superior y nuestra Junta Directiva, todo el dinero que se recaude por concepto de matrículas se debe invertir en dar la mejor calidad académica y llegar a la excelencia en docencia, investigación y servicio, y todo lo que requiera incremento de construcciones y espacio físico se debe tratar de conseguir con la comunidad, pues a ésta beneficia directamente y es la mejor manera de lograr un crecimiento armónico. Estas donaciones tienen exención de tributación de

acuerdo con la forma como se hace, que disminuyen el esfuerzo económico y con ellas se pueden beneficiar la entidad y el programa que se desea. La empresa del Valle del Cauca fue fundadora de esta entidad y todo lo que se ha construido físicamente ha sido por su generosa ayuda. A ésta el ICESI le ha correspondido formando sus directivos y empleados y dando asesoría y cursos especiales a sus empleados. Ha sido, pues, una inversión rentable para todos y esperamos que la comunidad del Valle, que es la más generosa del país, continúe colaborando con esta entidad que espera servirla con eficacia.

Al terminar sus estudios queremos manifestarles, de nuevo, nuestro agradecimiento por habernos seleccionado para hacer sus carreras y las especializaciones de postgrado en esta Institución y les manifestamos que ésta será siempre su casa y que toda ella estará a su servicio y pueden contar con ella en su futuro. Los esperamos de nuevo para realizar sus especialidades, el Magíster y los cursos de extensión que estaremos programando para su perfeccionamiento y actualización. También, si quieren, simplemente visitarnos y usar la biblioteca o asistir a las conferencias que se programen. Serán siempre bienvenidos.

Estamos seguros que tendrán muchos éxitos y que serán unos líderes en la comunidad empresarial y en la sociedad. Que Dios los acompañe.

RESEÑAS BIBLIOGRAFICAS

PRESIDENCIA DE LA REPUBLICA
Departamento Nacional de Planeación

El Salto Social
Bases para el Plan Nacional
de Desarrollo, 1994-1998

19x24 cm. 1-227 páginas

La economía y la sociedad colombiana han experimentado reformas profundas durante la presente década. La Constitución de 1991 consolidó la descentralización política, abrió múltiples espacios a la participación ciudadana, redefinió los derechos económicos y sociales de los ciudadanos, incorporó por primera vez los principios de protección al medio ambiente dentro de nuestra Carta Política y reformó diversas instituciones, entre ellas el sistema judicial, los mecanismos de planeación y el marco que regula la prestación de servicios públicos y la banca central. Por su parte, la apertura económica, iniciada durante la Administración Barco y consolidada durante la Administración Gaviria, generó nuevos retos al sector productivo colombiano, abrió espacios para la participación del sector privado en actividades tradicionalmente reservadas al Estado e inició un proceso activo de modernización de las instituciones estatales.

Estos cambios institucionales deben reflejarse con plenitud en la vida colombiana. Aunque el crecimiento económico ha sido satisfactorio, hay síntomas de crisis en algunos de los sectores que se esperaba serían los grandes beneficiarios de la apertura económica. Por otra parte, los cambios económicos y políticos no se han reflejado todavía en mejores niveles de vida para la mayoría de los colombianos. La persistencia de niveles alarmantes de pobreza y la ampliación de la brecha de ingresos rural-urbana son síntomas de la necesidad de volcar los esfuerzos del conjunto del país hacia un gran salto social que acelere la programación de los avances económicos al conjunto de la población.

La persistencia de la violencia está asociada en parte a este fenómeno,

aunque también a la reproducción de una cultura de intolerancia y conflicto que corroe profundamente las bases de nuestra sociedad. Por otra parte, la continua tala de nuestros bosques y depredación de nuestra gran riqueza en biodiversidad, la destrucción gradual de nuestras fuentes de agua y el envenenamiento del agua y el aire son reflejo de una crisis ambiental sin paralelo en la historia del país. Por último, las nuevas instituciones políticas no están plenamente consolidadas y, por el contrario, la velocidad de la transacción está generando en muchos casos traumatismos que afectan la provisión de los servicios del Estado.

La presente Administración tiene, por lo tanto, la doble tarea de consolidar las positivas reformas económicas y políticas de los últimos años, garantizando al mismo tiempo que sus beneficios se extiendan al conjunto de la sociedad. El Plan de Desarrollo Económico, Social y Ambiental, El Salto Social, busca, por lo tanto, consolidar la profunda transformación que viene experimentando el país y garantizar que su resultado final sea una sociedad más pacífica y equitativa, cimentada sobre un proceso de desarrollo económico dinámico y sostenible. Su meta final es, por lo tanto, "formar un nuevo ciudadano colombiano: más productivo en lo económico; más solidario en lo social; más participativo y tolerante en lo político; más respetuoso de los derechos humanos y por tanto más pacífico en sus relaciones con sus semejantes; más consciente del valor de la naturaleza y, por tanto, menos depredador; más integrado en lo cultural y por tanto más orgulloso de ser colombiano".

ICFES

Arte y Conocimiento
Resúmenes Analíticos
de Investigaciones 1981 - 1994.

Centro de Investigaciones
16x23 cm. 1-413 páginas.

De acuerdo con el documento "Colombia al filo de la Oportunidad", elaborado y publicado en 1994 por la Misión Ciencia, Educación y Desarrollo, dentro de las distintas prioridades para lograr una verdadera transformación educativa nacional, los comisionados llaman la atención sobre la necesidad de fortalecimiento e integración de los programas de doctorado, la vinculación de la investigación a la docencia y el establecimiento de un sistema coordinado entre educación y desarrollo científico y tecnológico para el país. En su conjunto, las propuestas se orientan hacia el reto de la producción responsable de conocimiento social e internacionalmente válido como elemento articulador de la Autonomía y Cultura Nacional y soporte para la democratización de la educación y la enseñanza con calidad.

Evidentemente la base del desarrollo social y económico se encuentra en la profundidad y alcance del conocimiento científico y tecnológico alcanzado por las naciones a través del movimiento permanente, aunque no lineal, de los procesos investigativos sobre interrogantes teóricos o prácticos surgidos de la observación e interpretación constante de la realidad, como actividades distintivas de la naturaleza humana. En ocasiones los bajos niveles de desarrollo científico en nuestro medio se atribuyen a la escasez de recursos e infraestructura tecnológica para la investigación, cuando en el transfondo las limitantes son de carácter cultural y actitudinal en un pueblo que aún menosprecia el valor particular y colectivo aportado por el desarrollo del conocimiento sistemático y ordenado.

Para lograr esta cultura del conocimiento como base del ordenamiento educativo y social, se hace indispensable, entre otros, la promoción de un mecanismo de divulgación y difusión de la información entre educadores e investigadores de manera que los contenidos de las asignaturas en los distintos niveles de la educación, se retroalimenten de forma permanente con los resultados de los avances científicos y tecnológicos logrados por la comunidad académica, generando en el alumno una actitud favorable hacia la búsqueda del conocimiento a través de un pensamiento crítico y analítico, relevando progresivamente los esquemas de docencia transmisionista para el aprendizaje repetitivo.

Igualmente, la comunidad de investigadores docentes requiere de instrumentos facilitadores de la consulta sobre los diferentes proyectos desarrollados o en curso dentro de sus áreas de interés, de forma que pueda orientarse el conjunto de esfuerzos hacia trabajos de integración interdisciplinaria para no continuar en la replicación de ensayos aislados sobre asuntos abordados y superados por sus colegas.

Por estos motivos, el Resumen Analítico de Investigaciones de la Corporación Universitaria Iberoamericana se constituye en un aporte a las redes de información académica dirigido a estudiantes, profesores e investigadores en los campos de la Educación Preescolar y Básica Primaria, Educación Especial y Fonoaudiología, así como a interesados en disciplinas afines a estas como la Psicología, Sociología, Filosofía, Historia y Antropología.

El Rai-Iberoamericana hace parte del conjunto de publicaciones institucionales de la serie Arte y Conocimiento, editado con el total de informes de investigación producidos desde 1981 hasta 1994 en el XV Aniversario de Fundación de la Corporación. Contiene una guía metodológica para la elaboración de resúmenes analíticos de investigación en la cual se basa el trabajo aquí presentado y con la que se aspira a fomentar la tendencia de este tipo de publicaciones en las distintas universidades y centros de investigación.

Así mismo incluye un índice de palabras claves adoptadas del listado de términos de referencia o descriptores normalizados por la UNESCO por cuyo motivo es necesario hacer algunas aclaraciones al lector. No siempre el término de referencia con el que el consultor está orientando la búsqueda se encuentra directamente en el listado de palabras claves del RAI, debido al tessaurus utilizado; por ello es conveniente remitirse en ocasiones al glosario inicial en donde se indican las palabras utilizadas para referenciar ciertos temas generales.

Por ejemplo, si el lector está interesado en consultar proyectos sobre Psicología evolutiva o del desarrollo, deberá

remitirse al índice de palabras y localizar Psicología del Niño, término de la categoría global adoptado por UNESCO. Igual ocurre cuando se pretende indagar sobre trabajos basados en programas estructurados, sean éstos de educación, rehabilitación, planeamiento curricular, capacitación, etc., en este caso deberá identificar el conjunto de resúmenes referenciados con la palabra Programas y con sus códigos trasladarse al índice de títulos de las investigaciones para seleccionar los resúmenes pertinentes. Otro ejemplo es de los estudios psicométricos o de construcción y validación de instrumentos de medición de procesos del desarrollo; en tal caso la palabra clave de referencia será Evaluación por cuanto así son registrados estos tipos de trabajos dentro de las redes de información internacionales.

Además contiene un índice de autores referido a los investigadores principales o directores de los proyectos, por cuanto el RAI cubre tanto las investigaciones institucionales desarrolladas por los investigadores de planta del centro y los docentes adscritos, como los trabajos de grado particulares elaborados por los alumnos como requisito para optar a sus títulos de técnico, tecnólogo, licenciado y profesional universitario.

Dado que la modalidad administrativa con la cual se coordina y desarrolla el proceso investigativo en la Iberoamericana admite la participación de alumnos en los proyectos institucionales y docentes como alternativa de homologación de sus trabajos de grado, en la nota de pie de página se identifican los nombres de los colaboradores para dar cumplimiento al respectivo crédito. Por este motivo en el índice de autores sólo figuran los autores o investigadores principales, dada la imposibilidad de referenciar en este listado a cada uno de los participantes en los distintos proyectos.

Posteriormente aparece el índice de títulos con el código interno del resumen para su búsqueda en el RAI y con un número de acceso al documento original cuando se lo desee consultar en la biblioteca general de la Institución. Frente a cada título se encuentra el número de página de este volumen en el que se halla el resumen analítico del título referenciado.

En síntesis, el Procedimiento de Consulta más ágil consiste en seleccionar un descriptor del tema de interés y localizarlo en el índice de palabras claves; si no se encuentra directamente, consultar entonces el glosario o en su defecto buscar un sinónimo o palabra asociada al término. Acto seguido anotar los códigos que figuran frente a la palabra clave y con estos números remitirse en orden al índice de títulos para identificar la(s) investigación(es) que contienen esta palabra. Si el título describe el tema de consulta, identificar la página del volumen en donde se encuentra y remitirse al resumen analítico de la investigación; cuando se desee ampliar o profundizar el documento, se anota el número de acceso y se solicita el informe original en la Hemeroteca o en el archivo de publicaciones del Centro de Investigaciones.

Igual procedimiento debe seguirse cuando la búsqueda se orienta por autores, en cuyo caso, conociendo previamente la línea preferencial de un determinado investigador, se identifican los códigos de sus trabajos dirigidos y se verifica la temática en el índice de títulos de investigaciones.

Cabe anotar finalmente que este volumen especial de Resúmenes Analíticos de Investigaciones constituye un esfuerzo común de trabajo de alumnos e investigadores de la Corporación Universitaria Iberoamericana con un respaldo amplio y decidido de sus directivas centrales quienes han comprendido siempre la importancia del fomento a la investigación y divulgación del conocimiento como soporte de la misión universitaria de la docencia y la extensión a la comunidad. Todas las contribuciones al perfeccionamiento y mejoramiento de esta publicación, serán acogidas y agradecidas por parte de las directivas, docentes y alumnos de la Institución.

MIGUEL LOPEZ LEORZA

Casos aplicables en políticas empresariales.

Universidad Santiago de Cali.
17x24 cm. 1 - 427 páginas.

La insistencia de muchos amigos y colegas para publicar los ejercicios y en general el material utilizado en el curso de Política Empresarial, así como los resultados altamente positivos obtenidos en años de labor académica, me llevaron finalmente a organizar las notas y apuntes de clase en un texto, como el que ahora publico a la disposición de los estudiosos del tema.

Este material trata de reunir aquellos aspectos que estudiados en las diferentes asignaturas, durante la carrera de administración, se actualizan con las políticas gubernamentales del momento y con las nuevas técnicas, para que sirvan como marco en la formulación de las políticas de los negocios.

El estímulo del Decano de Administración, doctor Harvey Rincón, y del doctor Nelson Vargas, Vicerrector Académico, hace posible que los borradores de clase sean hoy este libro de Política Empresarial, y Análisis de casos de Administración.

RYMEL SERRANO URIBE

Planificación Participativa.
Primera Edición, abril de 1994. Cooperativa de Crédito y Desarrollo.

ISBN: 958-95631-04
13.5x21 cm. 1 - 182 páginas.

Este libro se concibió como un manual metodológico de planificación participativa para uso de animadores o facilitadores del proceso de desarrollo de empresas asociativas del sector rural que, integradas por productores agropecuarios, se organizan para promover los intereses de éstos, permitiéndoles o facilitándoles un posicionamiento real y decisorio en el complejo económico del cual son principales autores por vocación, por ubicación y por destino.

El libro es, por consiguiente, un instrumento de trabajo, sustentado en documentos técnicos que son producto de la actividad teórico-práctica de un equipo profesional de consultores de Naciones Unidas y de grupos de trabajo directo, conformados por directores, ejecutivos, educadores y asociados de dieciséis empresas cooperativas y comunitarias, que han implementado y validado las técnicas de planificación participativa en un proceso continuado durante treinta y seis meses en promedio.

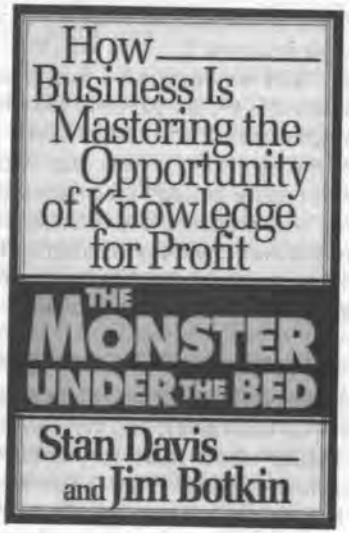
Las técnicas de planificación participativa se presentan en forma simple adaptadas en el lenguaje y en su operacionalidad a las posibilidades de aplicación en situaciones reales de las empresas rurales, y a la capacidad actual de comprensión y aprehensión por el campesino medio, quien constituye el elemento humano de las empresas asociativas en nuestro país.

Como el proceso planificador requiere de la intervención efectiva de los productores asociados en cada una de sus etapas, la metodología propuesta se presenta, antes que nada, como una Estrategia de Capacitación Participativa, orientada a ofrecer a dichos productores la posibilidad de desarrollar un conjunto de conocimientos, aptitudes y destrezas que les permita cambiar de situación y transformar, así sea en forma parcial, la realidad que los rodea.

Como la propia metodología que lo compone, este libro es inacabado; y susceptible por consiguiente de ajustes, adiciones y cambios, especialmente los que resulten de su utilización en el proceso de planificación y desarrollo de las empresas, o de la confrontación de sus contenidos y orientaciones pedagógicas con la realidad compleja y variable, de los grupos e intereses campesinos, a cuya promoción está dirigido.

Particular agradecimiento y reconocimiento al equipo técnico y profesional que ha contribuido, a través del tiempo, con sus ideas, su experiencia y su trabajo directo con las comunidades campesinas, a la formulación de esta metodología de planificación participativa; los señores Tony Shumacher, del Consejo Mundial de Cooperativas de Ahorro y Crédito, experto en Desarrollo Organizacional; Eugenio Pedro Giovenardi, Consultor de la OIT y Director Internacional Proyecto Col. 87.003; Aura Rivera Reyes, cooperadora y funcionaria del Departamento Nacional de Planeación de Colombia; y Julio Noé Cely, Orlando Solano, Jorge Iván Salazar, Jesús María Valbuena y Humberto Muñoz, quienes conformaron el equipo básico para la promoción y la aplicación de la metodología entre 1988 y 1991, período en el cual se realizó la experiencia directa con 24 empresas y con algo más de mil doscientos productores rurales.

Espero como autor de este libro y compilador de la documentación técnica y didáctica utilizada en los procesos de planificación, de los cuales tuve el privilegio de ser su orientador y coordinador, que este esfuerzo que se realiza bajo el patrocinio de Coopdesarrollo sea de utilidad práctica para el sector solidario y en particular para las empresas asociativas rurales que luchan, desde hace tantos años, por salir del círculo agobiante de la pobreza, y hacerse protagonistas conscientes de un desarrollo con bienestar.



STAN DAVIS AND JIM BOTKIN

The Monster Under the Bed.
Simon & Schuster, New York, 10029.

ISBN: 0-671-87107-2
13x22 cm. 1 -189 páginas.

Megan was five years old and worried about a monster that lived under her bed. She told a story about how the monster scared her, how she wanted it to go away, and how she solved the problem-now the monster lives under her brother's bed. She also drew pictures to accompany the story. They look like the kind displayed by parents on refrigerator doors, only Megan drew these pictures on her computer and used the computer to record her telling of the tale. It was all done with a software package geared to kids her age.

Megan didn't stop there. She wanted to share her story, so she sent it by phone to an electronic bulletin board club, where other kids her age could watch and hear it. This, in turn, was picked up by Nautilus, a CD-ROM multimedia magazine, and that was how we happened upon it on one of our computers.

When we clicked on Megan's story we were amazed. It lasted less than a minute, and we watched if five times. Here was a five-year-old child who had accomplished all the major tasks of moviemaking. She was the star, wrote the screenplay, created the visuals, did editing, was producer and director, and even did her own distribution. Her learning was integrated into the realites of her life. And to her it was all play.

The Monster Under Megan's Bed

Children have been seeing monsters under their beds and telling their families and friends about them for centuries. This is noth-ing new. What is new is the way Megan told her story –and how she learned to do it.

Just a generation ago, if Megan's mother had wanted to tell a similar tale, she would have written words and drawn pictures with paint or crayon on a piece of paper. And she would have learned to write and draw in school, under the watchful guidance of a teacher. Megan did not yet know how to write, but she knew how to use a computer. She learned how to use ir at home, with the help of the computer itself.

Similar learning is going on now for people of all ages, at home and in the workplace. The monster under Megan's bed may have been imaginary, but the electronic technology she used to describe it is undeniably real and has begun to assume its own monstrous proportions.

Countless companies and business alliances are providing electronic productos and services to millions of people like Megan and her family, bringing them entertainment and education at home and on the job. These companies are in the knowledge business– knowledge for profit –and they are revolutionizing the way we learn at the same time as they are creating a powerful new opportunity for growth in business.

ALBERTO RUIBAL HANDABAKA

Gestión Logística de la Distribución Física Internacional

Imprelibros-Carvajal S.A.
ISBN: 958-04-2584-1
15.5x22.5 cm. 1-461 páginas.

El libro se centra en los problemas que afrontan los empresarios para la *Distribución Física Internacional* (DFI) de sus productos de exportación e importación. En él se propone un enfoque de gestión empresarial que puede ser seguido por los gerentes de DFI de las compañías exportadoras e importadoras para la toma de decisión en sus negocios de comercio exterior. Dicho enfoque ha sido desarrollado dentro del concepto de *Logística Comercial Internacional* (LCI), el cual contribuye a optimizar las operaciones de comercio exterior en términos de costo, tiempo y calidad de servicio. Con ello se facilita un flujo uniforme de la carga entre todas las partes internacionales involucradas, a saber, exportadores, importadores y prestatarios de servicios (transportistas y operadores).

El objetivo principal del libro es contribuir a que las empresas exportadoras e importadoras puedan seguir la regla de oro de la DFI:

"Transportar el producto adecuado en la cantidad requerida al lugar acordado y al menor costo total para satisfacer las necesidades del consumidor en el mercado internacional *Justo a tiempo* (JAT) y con *Calidad total* (CT)".

Esto último toma en cuenta dos aspectos relevantes en el comportamiento de exportadores e importadores en el momento de tomar decisiones de venta o compra de mercancías: el tiempo de entrega, la calidad del producto (carga) y los servicios para trasladarlo a su destino final, lo que tiene importancia capital para la competitividad en relación con los demás proveedores.

La creciente globalización y liberalización de la economía mundial exige que la ventaja competitiva sea más determinante que la ventaja comparativa para aquellos países que comercian entre sí. Dentro de este contexto la DFI, como componente de la función de distribución, adquiere un rol preponderante en el desarrollo de la competitividad de los productos comerciados internacionalmente. Es así como el precio del producto al consumidor representará la suma de los costos de producción, comercialización y distribución, siendo este último mayor que los dos anteriores. En buena cuenta, el precio en destino cuantificará básicamente el valor agregado del producto más el costo de los servicios requeridos para movilizarlo de su lugar de producción a su lugar de consumo.

Una de las preguntas más frecuentes que formulan los empresarios exportadores e importadores a los asesores de comercio exterior, es la siguiente:

"¿Cuál es el precio de mi producto de exportación puesto en el local del importador extranjero; o al contrario, cuál es el precio de un producto de importación en mi local?"

Esta pregunta, aparentemente simple, prueba la necesidad de analizar los componentes del precio de venta de un producto puesto en el local del importador. Existen dos componentes básicos; uno es el precio del producto en el local del exportador y el otro es el precio del producto en el local del importador. Según los INCOTERMS, el primero se expresa como la *cotización en fábrica* (EXW) y el segundo, como la *entrega con derechos pagados* (DDP). El costo de los servicios para la movilización de la mercancía entre los dos puntos representa el costo de la DFI; la ejecución secuencial de las operaciones constituye la cadena de DFI y el período de tiempo requerido para llevar a cabo todas las operaciones se denomina tiempo de tránsito.

Los países en desarrollo ven con preocupación el desequilibrio que afecta sus balanzas de bienes y servicios, debido, entre otras razones, a los pagos por concepto de fletes, seguros y otros servicios. Estos se reflejan en los egresos del erario y constituyen un drenaje importante de divisas, representando uno de los principales renglones del comercio invisible en sus respectivas balanzas de pago. La carencia de suficiente capacidad en sus flotas mercantes y aerolíneas comerciales para transportar sus cargas de exportación e importación los obliga a contratar estos servicios de proveedores extranjeros.

FRANCISCO CAJIAO P.
Fundación FES.

Poder y Justicia en la Escuela Colombiana

Edición, Armada, Alegría de Enseñar, Colombia
ISBN: 958-9362-06-0
13.5x20.5 cm. 1-204 páginas.

Durante año y medio, entre enero de 1991 y agosto de 1992, se desarrolló el Proyecto Regional sobre Calidad de la Educación Básica (PIRCEB) que, por iniciativa de la División de Educación de la Fundación FES, se realizó con siete universidades, una escuela normal y una Secretaría de Educación departamental en los departamentos de Quindío, Antioquia, Risaralda, Valle del Cauca y la ciudad de Bogotá.

En este tiempo los maestros vinculados a las instituciones estudiaron cuatro temas relevantes de la vida escolar en relación con la formación de valores ciudadanos: el poder y la autoridad, la justicia, la tolerancia, la discriminación y la violencia. En el trabajo participaron directamente alrededor de 70 maestros rurales y urbanos, especialmente vinculados a escuelas primarias, tanto del sector oficial como del privado.

Los resultados de los procesos metodológicos obtenidos son múltiples y ya han sido publicados en otro libro de la serie FES titulado «Hacia una pedagogía de la creatividad». Por otra parte se produjo un conjunto de trabajos escritos por los maestros, algunos de los cuales están también publicados en una serie denominada «Vida Escolar en Colombia», que hasta ahora tiene seis títulos que compendian quince investigaciones. En estas publicaciones se han transcrito los estudios etnográficos realizados en muchísimas escuelas, sin modificar el material original producido por los maestros participantes en el proyecto. Entre el material producido todavía hay muchos trabajos que tienen valiosa información sobre lo que ocurre al interior de las escuelas en relación con los temas que se estudiaron.

Desde el comienzo del proyecto se preveía la necesidad de un segundo nivel de investigación que permitiera estructurar una visión de conjunto sobre los temas explorados en las distintas regiones del país, aportando elementos interpretativos a la información cualitativa recopilada durante el proceso (para este efecto se ha iniciado la organización de una base de registros etnográficos que permita a otros investigadores hacer un recorrido temático del material de entrevistas, testimonios y observaciones contenidos en los trabajos de base documental). El libro que ahora se publica corresponde a este nivel de investigación, y explora dos temas del proyecto: el poder y la justicia en la escuela colombiana. Para su elaboración se revisaron una y otra vez todos los documentos publicados y no publicados, tratando de encontrar similitudes y particularidades en los hallazgos de cada autor; intentando seleccionar los registros más representativos y, finalmente, buscando una estructura que permitiera exponer la problemática en forma coherente y suficientemente amplia como para dar un margen explicativo a la diversidad de situaciones descritas por los investigadores del PIRCEB.

Durante el año que ha tomado este trabajo se ha hecho una discusión permanente sobre los borradores reelaborados una y otra vez hasta llegar a esta versión que se quiere ofrecer a la comunidad educativa del país como tema de reflexión y como base para nuevas investigaciones que elaboren con más cuidado algunos de los tópicos apenas sugeridos en esta primera aproximación. Aunque la redacción del libro ha sido obra de Francisco Cajiao, quien figura como autor, el trabajo de preparación y discusión ha sido realizado en grupo con Rodrigo Parra Sandoval, Elsa Castañeda, Martha Luz Parodi y María Victoria Lozano. Igualmente han estado presentes todo el tiempo los maestros participantes en el proyecto, pues fue su material la referencia permanente sobre la cual se trabajó. Y no sólo el material escrito, sino la memoria de sus comentarios y anécdotas durante los talleres en los cuales se construía paso a paso el proceso de identificación de temas y problemas. Por esto queremos presentar a continuación la lista completa de los trabajos elaborados por ellos y sus nombres, acompañados de una sigla que identificará el origen de los registros utilizados en esta investigación.

Conflicto disciplinario en la escuela, sus voces y sus rastros. Secretaría de Educación Departamental. Libia Bueno Gaviria, 1992 (LB).

Derechos humanos y justicia en la escuela. Universidad Pontificia Bolivariana. Gloria Inés González, Tomás Felipe Palacio, Fernando Soto, 1992 (GIG).

La fuga de la ilusión. Diana Garzón, Nancy Boada, Fabián Molina, Zulma Lozano y Héctor Calderón. Escuela Normal M.M. Ed. FES, 1993. (FI).

Dirimir conflictos en la escuela: ¿justicia o farsa? Universidad Tecnológica de Pereira. Jaime Londoño Hurtado, 1992 (JL).

El autoritarismo disfrazado de formación. Universidad del Quindío. Ana María Rodríguez Lozano, Blanca Aurora Mejía León, 1993 (AMR).

El castigo en pasado y presente. Universidad del Quindío. María Eugenia Arias, Margarita De La Torre Andrade, 1992 (MEA).

El docente y el poder que maneja - Investigación Etnográfica desde un caso específico <->Pirceb<-> Hernán Giovani Velasco Devia, Universidad Pontificia Bolivariana, 1992 (HV).

El fantasma del poder escolar. Universidad Externado de Colombia. Luis Fernando Gutiérrez, 1992 (LFG).

El grito, una realidad en la cotidianidad escolar. Universidad del Quindío. Adyela Arias Zapata, María Consuelo Ríos Rendón, 1993 (AAZ).

El mundo de la pandilla se está tomando la escuela. Universidad Externado de Colombia. Luz Eneida Rincón Dávila, 1992 (LER).

El rostro oculto de la escuela (Calidad de la educación y formación ciudadana). Marina Camargo, 1992.

Historias de la escuela - Proyecto Priceb Fes/Upb. Juan Carlos López Díaz (JCL).

Uy, parce, ¡qué agite tan tenaz! Carlos Arturo Tobón. Universidad Pontificia Bolivariana, 1992 (CAT).

La amenaza pedagógica. Universidad del Quindío. Magda Stella Agudelo García, Luz Elena Gómez García, 1992 (MSA).

La educación especial: ¿una escuela sin remedio? Universidad Externado

de Colombia. Diana Patricia Valverde Rozo, 1992 (DV).

La escuela para el niño, un mundo dividido en dos. Universidad del Quindío. María Belinda Olave, Gloria Lasmy Serna, 1992 (GLS).

La evaluación: un procedimiento de justicia. Universidad Externado de Colombia. Enrique Alfonso Pinto, 1992 (EAP).

La magia del poder en el aula de clase. Investigación Etnográfica desde un caso específico. Universidad Pontificia Bolivariana. Jorge Luis Coronado Tovar, 1992 (JLC).

La escuela sin muros y la formación de ciudadanos. Universidad Javeriana (Cali). Amparo Escobar, Fanny Gaviria, Luz Neyra Rubio, Amparo Panesso y otros, 1992 (AE).

Escuela Nueva... Mente vieja. Universidad Tecnológica de Pereira. Milton Guillermo Fuentes Ruiz, 1992 (MGF).

La Evaluación en la Escuela: ¿Formadora o deformadora? Universidad Tecnológica de Pereira, 1992.

Tolerancia y discriminación de género en el aula de clase. Maryori Carmona G., Carmen Elizabeth Suárez. Universidad Tecnológica de Pereira, 1992 (MCG).

La Autocracia Escolar. Libia Bueno, María Eugenia Arias, Luz Eneida Rincón, Edit. FES, 1993. Serie Vida Escolar.

¿Alumnos problema o maestros problema? Edit. FES, 1991, Serie Vida Escolar. Myrian Reyna, Mireya Gutiérrez, Olga Moritz.

La FES siempre estará reconocida por la extraordinaria respuesta de las instituciones de educación superior que participaron en el proyecto, y sobre todo con el grupo de maestros que se comprometió durante este largo tiempo a generar una nueva alternativa de ver la escuela y contribuir a su desarrollo, en beneficio de cientos de miles de niños colombianos.

JUAN GUSTAVO COBO BORDA

La mirada cómplice

Centro Editorial Universidad
del Valle, 1994
ISBN: 958-670-009-7
13.5x22 cm. 1-228 páginas.

Aproximarse a la pintura desde la literatura resulta una actividad sospechosa, censurada con acritud por los profesionales de la crítica, entre ellos mi involuntario maestro Damián Bayón.

No me arrepiento ni presento demasiadas excusas. Amo la pintura, desde Altamira a los pequeños infantes de Velásquez en el Museo de Viena y disfruto reconociendo esas gloriosas revelaciones. No puedo llevarme tales cuadros a casa pero sí registrar, con insuficientes palabras, su trazo imborrable. Sólo que ahora, al corregir las pruebas de estas ocho aproximaciones, acepto que estaba tratando de comprender dos imposibles: la pintura misma y el país, Colombia, que le dio origen. No me corresponde medir los resultados. Sí reconocer cómo la pintura busca en vano aclarar lo informe y a veces naufraga en su esforzado intento, tan humano, de dar rostro a lo que aún no lo tiene o ha sido desfigurado, para ir más allá de él y convertirse tan sólo en pura pintura. Algo, por cierto, también profundamente humano.

Tal propósito anima a Obregón y a Juan Cárdenas, a Roda y a Luciano Jaramillo, a David Manzur y a María de la Paz Jaramillo. Con Rayo el afán de orden encierra también el propósito de componer una figura, tan musical como estricta. Beatriz González, por su parte, rehace y descompone desde la imagen en crisis. Al criticar lo estatuido nos revela el revés de la trama.

Pero cada uno de ellos es un pintor único. Sólo los une la misma mirada cómplice de una escritura que, al rendirles homenaje, se justifica. Y más aún, disfruta en compartir (y competir) con la vista.

Quisiera celebrar la pintura y también, ¿por qué no? a quienes han escrito sobre ella. A través de sus textos me colé sin permiso, en el sagrado recinto, hoy un tanto en ruinas, gracias a tantas ambientaciones e instalaciones sólo conceptuales. Sin embargo, los textos de Marta Traba y Damián Bayón siguen siendo incitantes y válidos. Octavio Paz, como siempre, fue una referencia decisiva.

Por su parte, dos mujeres: Celia de Birbragher, directora de **Arte en Colombia**, e Ivonne Nichols, con quien fundamos las ediciones de arte de Seguros Bolívar, me incitaron y sedujeron para que escribiera sobre arte y entregara los textos a tiempo. Nunca fue posible. Deseo que este libro, no pedido, las distraiga un poco de sus esforzados deberes como ejecutivas.

Gracias a Jaime Galarza, rector de la Universidad del Valle, y a Umberto Valverde y a Lucy Mejía disfruto de las delicias de pasear por una prestigiosa universidad sin cumplir de lleno con los requisitos académicos.

Este libro está dedicado a mi hija Paloma. Sus 22 meses aman, con fervoroso entusiasmo, los erizados gatos de Goya y las rotundas figuras de Botero. Que algún día, cuando pueda leerlo, prefiera tanto las imágenes como las letras. Las primeras deparan hallazgos y consuelos que estas páginas insuficientes sólo buscan reconocer, en público, como una forma de adensar el entramado, cada vez más vivo y polifacético, de la cultura colombiana. En ella la pintura es central: nos vimos por primera vez.

ROBERTO RUIZ

De los enteros a los dominios

Centro Editorial Universidad
del Valle, 1994
ISBN: 958-40-0005-5
15.5x24 cm. 1-147 páginas.

En este texto se hace un estudio aceptable de los números enteros en dos aspectos: el interno, es decir qué propiedades tienen y cómo se usan para facilitar el trabajo con enteros. El externo, responde esencialmente a la pregunta de si existen otras estructuras que tengan las propiedades de **Z**. Para esto se va al concepto de dominio bien ordenado y se demuestra que **Z** es el único. Esto se hace en el capítulo 1. Se dedican los capítulos 2 y 3 a estudiar aspectos muy característicos de **Z** (es decir de los dominios bien ordenados), a saber, elementos muy básicos de la teoría de números de bachillerato.

En seguida se tocan las estructuras particulares que hay en **Z**. Se inicia con grupos, lo cual ocupa los capítulos 4 y 5. En el primero se tocan grupos cíclicos esencialmente. En el segundo se desarrolla la teoría de descomposición de grupos abelianos finitamente generados. Como hasta ese punto se trabaja esencialmente con grupos abelianos, se hace ahora una parte de grupos de permutaciones llamada a alertar al lector sobre lo que sucede con grupos abelianos que suele no ser cierto en grupos no abelianos. Esto aparece en el capítulo 6.

Como se trata de avanzar sobre dominios y un ejemplo corriente es $K[x]$ donde K es un campo, se toca, con ejemplos relevantes, el concepto de campo o cuerpo. Esto aparece en el capítulo 7. Se construye el campo de fracciones de un dominio y el campo de solución de una ecuación cuadrática.

Finalmente como ejemplo de dominio de ideales principales no triviales se estudia $K[x]$ en el capítulo 8. Interesa mostrar qué es, realmente, un polinomio y también que $K[x]$ es un D.I.P. y dominio euclídeo. En el capítulo 9 se dan ejercicios generales con los cuales se espera que el estudiante refuerce la teoría y desarrolle, él solo, aunque de manera somera, temas interesantes. Tal es el caso de los dominios euclídeos.

ALOZ ROJAS

Festejos y Memorias

Centro Editorial Universidad
del Valle, 1994
ISBN: 958-670-000-3
15.5x22 cm. 1-67 páginas.

INDICE

LAUREANO GOMEZ SERRANO

Epicicios. Apuntes sobre Filosofía de la Ciencia

Centro de Investigaciones.
Universidad Autónoma de
Bucaramanga
16x22 cm. 1-108 páginas.

En esta recopilación se presentan cuatro ensayos escritos en desarrollo de las actividades académicas del post-grado de especialización en Filosofía de la Ciencia adelantado dentro del Convenio de Cooperación Académica de la Universidad de Antioquia y la Universidad Industrial de Santander.

Se pretende con ellos reseñar el curso de los debates sobre algunos problemas fundamentales de la construcción del pensamiento científico, a saber:

En el primer ensayo se presenta la ruptura del fundador de la ciencia moderna con el pensamiento aristotélico, centrándose en el desmonte de la retórica peripatética que enervaba el desarrollo del nuevo pensamiento copernicano, con el abuso del principio de autoridad.

En el segundo, se estudia el planteamiento cartesiano sobre el vacío, como un intento de superación del grave problema que presentaba el fenómeno al sistema mecanicista, y las razones que llevaron a la negación del mismo.

En el tercer ensayo se analiza la relación implícita que tiene la divergente concepción del cálculo en Newton y Leibniz, en la célebre polémica que mantuvieron los racionalistas contra los postulados de los Principios Matemáticos de Filosofía Natural.

El cuarto se ocupa del instrumental metodológico que implementa Karl Popper en la crítica a Karl Marx expuesta en la obra «La sociedad abierta y sus enemigos» con el objeto de determinar la congruencia metodológica y el resultado de las refutaciones intentadas por el ideólogo del neoliberalismo.

En el quinto se presenta la aplicación que del «modo geométrico» hace Thomas Hobbes en el análisis de la estructura del Estado y su elemento central, la «seguridad».

Los ensayos tienen cierta independencia y autonomía, pero se articulan en torno al programa de investigación desarrollado por la ciencia moderna que surge en el siglo XVI, y se expresa en los diversos campos del conocimiento, de los cuales son paradigma Galileo Galilei, René Descartes, Isaac Newton, Gottfried Leibniz, Thomas Hobbes, Karl Marx y Karl Popper.

GENTIL ROJAS LIBREROS

Lecciones breves sobre la economía colombiana
Centro Editorial Universidad del Valle, 1994

ISBN: 958-40-0001-2
23.5x16 cm. 1-155 páginas.

El análisis de la economía colombiana, para ser realista y completo, tiene que incorporar a la economía subterránea, como una variable fundamental. Ello justifica que gran parte de los artículos periodísticos que preparé llevan este mensaje al público en general y a los economistas en particular. Estando la actividad productiva, comercial y financiera, en grado considerable, por fuera de los registros y las estadísticas oficiales del mercado, el diagnóstico de los problemas y sus soluciones, es por lo tanto relativamente menos fundamentado y más complejo en Colombia que en otros países con economías más convencionales.

Un primer conjunto de los artículos de este capítulo se orienta a esclarecer los componentes de la economía subterránea. Los flujos de comercio de mercancías y capitales se ubican en el centro de ésta, teniendo como fuente financiera el activo mercado del dólar negro local. Este desapareció del ambiente sólo para ser reemplazado por el dólar subterráneo, que opera desde el extranjero y, quizás también, desde las múltiples ventanillas siniestras creadas en el sistema financiero en sustitución de la solitaria ventanilla del Banco de la República.

La economía subterránea no sólo ha desfigurado las operaciones de comercio exterior, sino que ha introducido en el funcionamiento cotidiano de la economía, al dólar como medio de pago. La dolarización es el tema del segundo conjunto de artículos de este capítulo.

El reconocimiento de este rasgo sobresaliente de la economía colombiana, en la formulación de la política de apertura, habría llevado a actuar con menos audacia y más gradualismo, en el levantamiento de las barreras a la afluencia de capitales. Como veremos en el capítulo siguiente, la avalancha de éstos pone en serias dudas la aspiración de Colombia de conquistar los mercados externos, y más bien, es una invitación al capital extranjero para que capture y penetre nuestro mercado interno con mayor profundidad que en el pasado.

ICESI

El ICESI es una corporación universitaria fundada en 1979 para satisfacer las necesidades del sector empresarial en el campo de la formación de profesionales en las diferentes áreas que aquellos requieran. Para cumplir con este propósito el ICESI ofrece los siguientes programas de Pregrado y Postgrado.

PROGRAMAS DE PREGRADO

- Administración de Empresas: Horarios Diurno y Nocturno
- Ingeniería de Sistemas e Informática: Horario Diurno

PROGRAMAS DE LA ESCUELA DE POSTGRADO

- **ESPECIALIZACIONES**
  - Administración
  - Finanzas
  - Mercados
  - Informática
  - Gerencia de Producción

- **CONCENTRACIONES**

  A. En Negocios Internacionales

  B. En Administración
  Organizaciones avanzadas
  Gerencia organizacional
  Comportamiento organizacional
  Administración agroindustrial

  C. En Mercados
  Mercados avanzados
  Administración de empresas comerciales

  D. En Finanzas
  Finanzas avanzadas
  Gerencia de impuestos

- **MAESTRIA EN ADMINISTRACION**

- **ALTA GERENCIA**

CARACTERISTICAS DISTINTIVAS DEL ICESI

- Formación integral del hombre
- Búsqueda continua de la excelencia
- Atención individualizada a los estudiantes
- Hábitos de estudio
- Núcleo de enseñanza-aprendizaje
- Programa de práctica en la empresa
- Desarrollo del espíritu empresarial
- Renovación permanente de los equipos de apoyo a la enseñanza

Informes: Apartado Aéreo 25608, Unicentro
Teléfono: 330 6822
CALI - VALLE - COLOMBIA

ICESI es una institución universitaria afiliada a la Asociación Colombiana de Universidades - ASCUN

FERIVA S.A. Cali