CAPITULO 20

SISTEMAS DISTRIBUIDOS.

20.1. - **Evolución de Arquitectura de Computadoras**.

Repasemos ahora nuevamente los conceptos de arquitecturas paralelas pero desde otra perspectiva.

El estudio de la arquitectura de computadoras involucra tanto la organización del hardware como los requerimientos de programación/software. Un programador de lenguaje ensamblador ve la arquitectura del computador como una abstracción del conjunto de instrucciones, incluyendo los códigos de operación, los modos de direccionamiento, registros, la memoria virtual, etc.

Desde el punto de vista hardware la máquina abstracta está organizada con CPUs, caches, buses, microcódigo, pipeline, memoria física, etc. Sin embargo, el estudio de la arquitectura cubre tanto la organización del conjunto de instrucciones como la de la implementación de la máquina.

En las pasadas cuatro décadas la arquitectura de las computadoras ha sufrido cambios evolutivos más que revolucionarios. Se comienza con la arquitectura Von Neumann construida como una máquina secuencial ejecutando datos escalares. La computadora secuencial fue mejorada desde la realización de operaciones de bits en forma seriada a operaciones de palabras en paralelo, y desde operaciones en punto fijo a operaciones en punto flotante. La arquitectura Von Neumann es lenta debido a la ejecución secuencial de las instrucciones del programa.

20.2. - **Lookahead, Paralelismo y Pipelining**.

Las técnicas de lookahead fueron introducidas para poder superponer las operaciones de I/E (fetch de instrucciones/ decodificación y ejecución) y habilitar el paralelismo funcional. El paralelismo funcional se logró de dos formas: una es utilizando múltiples unidades funcionales simultáneamente, y la otra es utilizar pipelining en varios niveles de procesamiento.

Esta última incluye ejecución de instrucciones pipelinizadas, cálculos aritméticos pipelinizados, y operaciones de acceso a memoria.

20.3. - **Clasificación de Flynn** (1972).

Hemos visto con anterioridad la clasificación de Flynn basada en la cantidad de flujos de datos y de instrucción que procesa un computador.

20.4. - **Computadoras paralelas/vectoriales**.

Las computadoras intrínsecamente paralelas son aquellas que procesan programas en modo MIMD. Existen dos grandes clases de estas computadoras, a saber, las *computadoras de memoria compartida* y las *computadoras con pasaje de mensajes*. La mayor distinción entre multiprocesadores y multicomputadoras reside en la compartición de la memoria y en los mecanismos de comunicación entre los procesos.

20.5. - **Atributos de un sistema para la performance**

20.5.1. - **Tasa de Reloj y CPI**

La CPU de las computadoras digitales de hoy en día está gobernada por un reloj con un *tiempo de ciclo* constante (τ en nanosegundos). La inversa de este tiempo es la *tasa de reloj* ($f = 1 / \tau$) en megahertz).

El tamaño de un programa esta determinado por la *cantidad de instrucciones* (I_c). Las instrucciones de diferentes máquinas pueden requerir diferentes ciclos de reloj para ejecutarse. Sin embargo, los *ciclos por instrucción* (CPI) es un importante parámetro para medir el tiempo necesario para ejecutar dicha instrucción.

20.5.2. - **Factores de performance**.

Sea I_c como lo definimos antes. El tiempo de CPU (T en segundos/programa) necesario para ejecutar el programa se estima hallando el producto :

$$T = I_c * \text{CPI} * \tau \qquad (1)$$

Definimos un ciclo de memoria como el tiempo necesario para completar una referencia a memoria. Usualmente un ciclo de memoria es k veces el ciclo de procesador τ. El valor de k depende de la velocidad de la memoria y del esquema de interconexión de memoria-procesador.

El CPI de una instrucción tipo puede dividirse en dos componentes correspondientes al total de ciclos del procesador y de ciclos de memoria necesarios para completar la ejecución de la instrucción. Dependiendo del tipo

de instrucción el ciclo completo de la misma puede involucrar de uno a cuatro referencias a memoria (una para el fetch de la instrucción, dos para los operandos, y otra para almacenar el resultado). Reescribimos entonces la fórmula (1) de la siguiente forma :

$$I_c * (p + m * k) * \tau \qquad (2)$$

donde *p* es la cantidad de ciclos de procesador necesarios para decodificar y ejecutar la instrucción, *m* es la cantidad de referencias de memoria necesarias y *k* es la relación entre el ciclo de memoria y el ciclo de procesador.

20.6. - **Atributos de Sistema**.

Los cinco factores de performance anteriores están influidos por atributos del sistema tales como la arquitectura del conjunto de instrucciones, la tecnología del compilador, el control y la implementación de la CPU y la jerarquía de la memoria y de la cache de acuerdo a como se muestra en la siguiente tabla:

ATRIBUTOS DEL SISTEMA	**FACTORES DE PERFORMANCE**				
	Cantidad de instrucciones Ic	Promedio de Ciclos pos instrucción, CPI			Tiempo de ciclo de procesador, τ
		Ciclos de Procesador por instrucción, *p*	Referencias a Memoria por instrucción, *m*	Latencia de acceso a Memoria, *k*	
Arquitectura del set de instrucciones	**X**	**X**			
Tecnología del compilador	**X**	**X**	**X**		
Control e implementación del Procesador		**X**			**X**
Jerarquía de Memoria y cache				**X**	**X**

20.6.1. - **Tasa MIPS**.

Sea *C* el número total de ciclos de reloj necesarios para ejecutar un programa dado. Entonces el tiempo de CPU en (2) puede estimarse como

$$T = C * \tau = C / f.$$

Más aún,

$$CPI = C / I_c$$

y $T = I_c * CPI * \tau = I_c * CPI / f.$

La velocidad del procesador se mide a menudo en *millones de instrucciones por segundo* (MIPS),

$$\text{tasa MIPS} = I_c / (T * 10^6) = f / (CPI * 10^6) = f * I_c / (C * 10^6) \qquad (3)$$

Basándonos en esta ecuación se puede decir que el tiempo de CPU en la ecuación (2) también puede escribirse como

$$T = (I_c * 10^{-6}) / MIPS$$

Se concluye que los MIPS de un procesador son directamente proporcionales a la velocidad del reloj e inversamente proporcionales al CPI.

20.6.2. - **Tasa Throughput**.

Otra medida importante está relacionada con cuántos programas por unidad de tiempo un sistema puede ejecutar, esto se llama *throughput del sistema* W_s (en programas/segundo). En un sistema multiprogramado el throughput del sistema es a menudo inferior al *throughput del procesador* W_p al que definimos como:

$$W_p = f / (I_c * CPI) \qquad (4)$$

Nótese que de la ecuación (3) $W_p = (MIPS * 10^6) / I_c$. El throughput del procesador es una medida de cuántos programas pueden ejecutarse por segundo basándose en la tasa MIPS y el promedio de longitud del programa (I_c). La razón del porqué $W_s << W_p$ se debe a los overheads adicionales del sistema causados por las E/S, el compilador y el sistema operativo cuando varios programas se intercalan para su ejecución en un ambiente multiprogramado.

20.6.3. - **Ejemplo**

Considérese la utilización de una VAX/780 y una IBM RS/6000 para ejecutar un cierto programa de benchmark. Las características de las máquinas y la performance declarada se muestran en la siguiente tabla:

Máquina	Reloj	Performance	Tiempo de CPU
VAX 11/780	5 MHz	1 MIPS	12x segundos
IBM RS/6000	25 MHz	18 MIPS	x segundos

Estos datos indican que el tiempo de CPU medido para la VAX es 12 veces mayor que el medido para la RS/6000. Los códigos objeto corridos en ambas máquinas difieren en su longitud debido a la máquina y al compilador utilizado. Los otros overhead se ignoran.

Basándose en la ecuación (3) la cantidad de instrucciones del código objeto en la RS/6000 es 1,5 veces mayor que el código que corre en la VAX. Más aún, el promedio CPI en la VAX/780 se asume que es de 5 en tanto que en la RS/6000 es 1,39 ejecutando el mismo programa de benchmark.

No podemos calcular el throughput del procesador W_p a menos que conozcamos la longitud del programa y el promedio CPI de cada código.

20.7. - **MULTIPROCESADORES Y MULTICOMPUTADORAS**

Existen dos categorías de computadores paralelos. Estos modelos físicos se diferencian en el hecho de tener memoria compartida o distribuida.

20.7.1. - **Multiprocesadores de memoria compartida**.

Existen tres modelos que se diferencian en como la memoria y los periféricos se comparten o distribuyen, a saber:

- **UMA** (uniform memory access)
- **NUMA** (no uniform memory access)
- **COMA** (cache-only memory access)

20.7.2. - **El modelo UMA**

La memoria se comparte uniformemente entre los procesadores. Todos tienen igual tiempo de acceso a todas las palabras de memoria. Cada procesador puede tener una cache privada. Los periféricos también se comparten de la misma forma. Se los denomina *sistemas fuertemente acoplados*. Para coordinar los eventos paralelos, la sincronización e intercomunicación entre procesos se utilizan variables en la memoria común.

Cuando todos los procesadores tienen igual acceso a todos los periféricos el sistema se denomina *simétrico* (MP).

En un multiprocesador *asimétrico* solo uno o un subconjunto de los procesadores tienen la capacidad ejecutiva. El procesador ejecutivo o maestro ejecuta el sistema operativo y maneja las E/S. Los otros procesadores no pueden manejar las E/S y se los llama *attached processors (APs)*. Los procesadores attached ejecutan código bajo la supervisión del procesador maestro.

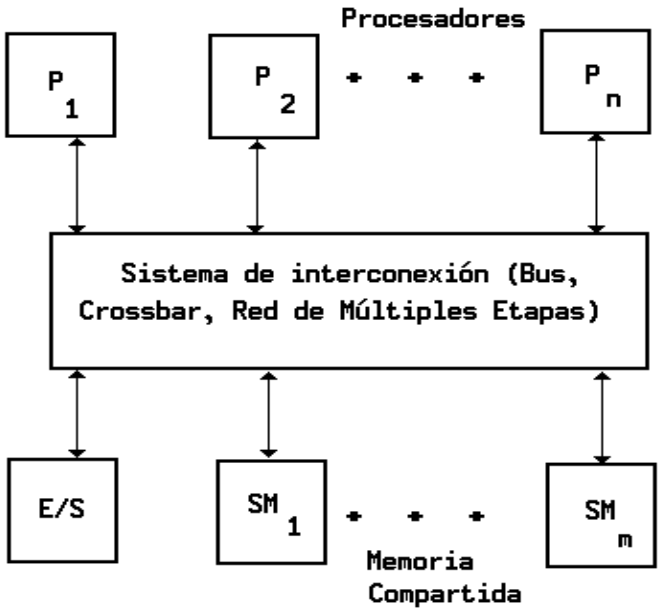

Fig. 20.1. - El modelo UMA. Sequent S-81.

20.7.3. - **El modelo NUMA**

Es un sistema de memoria compartida en donde el tiempo de acceso a memoria varía dependiendo de la ubicación de la palabra de memoria.

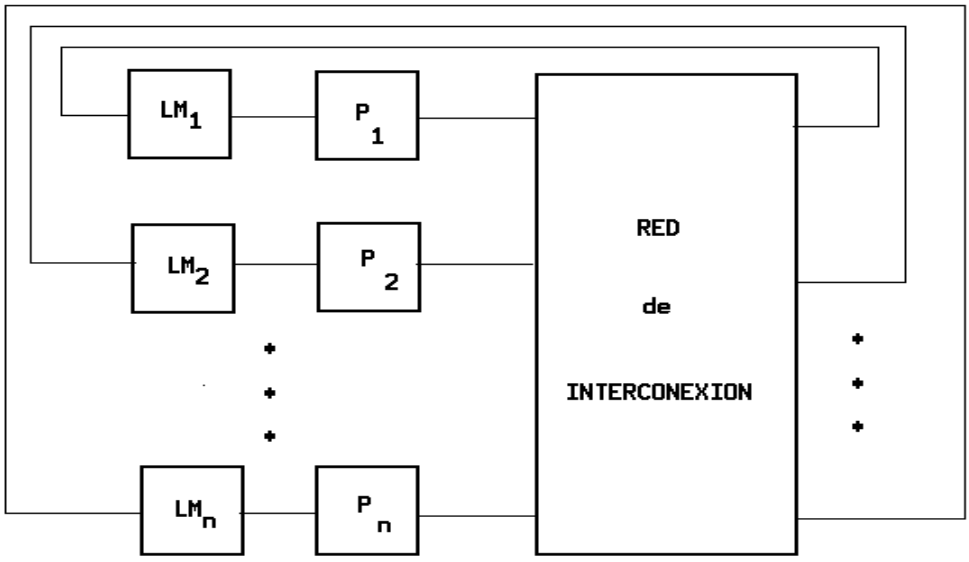

Fig. 20.2. - Memoria locales compartidas BBN Butterfly.

La memoria está físicamente distribuida entre todos los procesadores, se llaman *memorias locales*. Es más rápido acceder un dato en la memoria local, para acceder información en una memoria remota existe una demora debida a la red de interconexión (ej. la BBN TC-2000 Butterfly).

Además de estas memorias distribuidas se puede agregar memoria globalmente accesible por todos los procesadores. En este caso existen tres patrones de acceso a memoria:

- el más rápido es acceder la memoria local,
- un poco más lento es acceder la memoria global,
- y por último el más lento de todos es acceder la memoria remota de otro procesador (ej. Cedar de la universidad de Illinois)

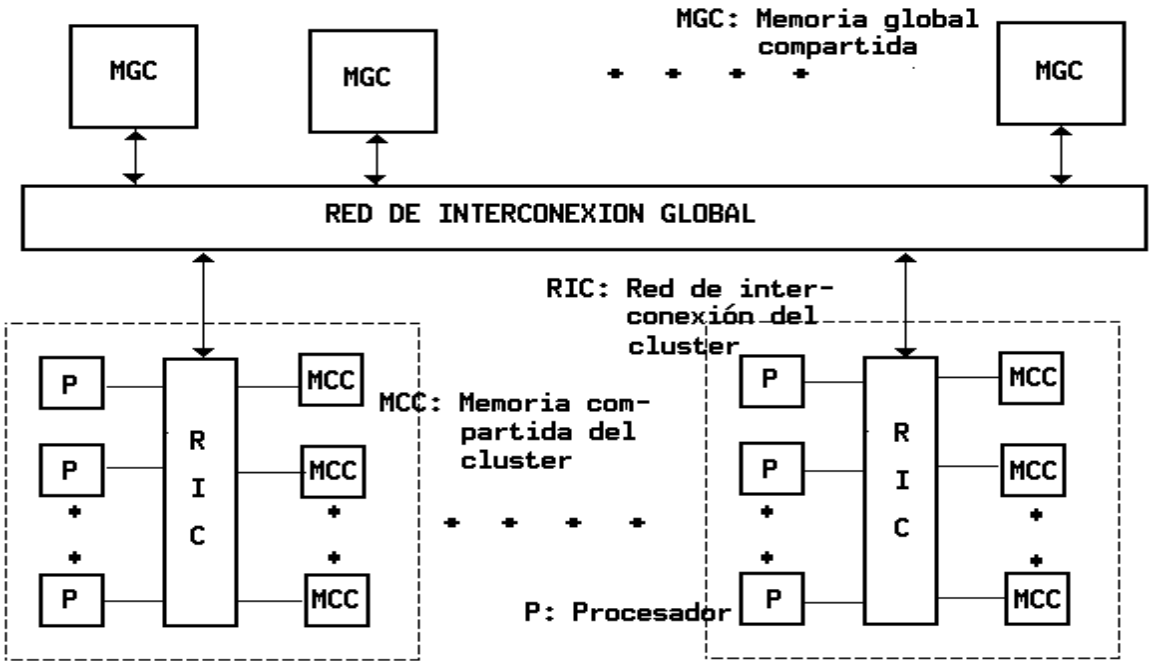

Fig. 20.3. - Cluster jerárquico (la Cedar).

20.7.4. - **El modelo COMA**

El modelo COMA utiliza solo memorias de tipo cache. (ej. la KSR-1 de Kendall Square Research). Son un caso particular de la NUMA en el cual la memoria distribuida se convierte en memorias cache. No existen jerarquías de memoria en cada nodo procesador. Todas las caches forman un espacio de direccionamiento global. El acceso a una cache remota se facilita a través de los directorios distribuidos de cache los cuales en ciertas arquitecturas pueden estructurarse en forma jerárquica.

Una de las mayores falencias que poseen estos multiprocesadores es la falta de escalabilidad debido a la centralización de la memoria compartida. Al construir sistemas MPP se logra mayor escalabilidad pero son menos programables debido a la adición de los protocolos de comunicación.

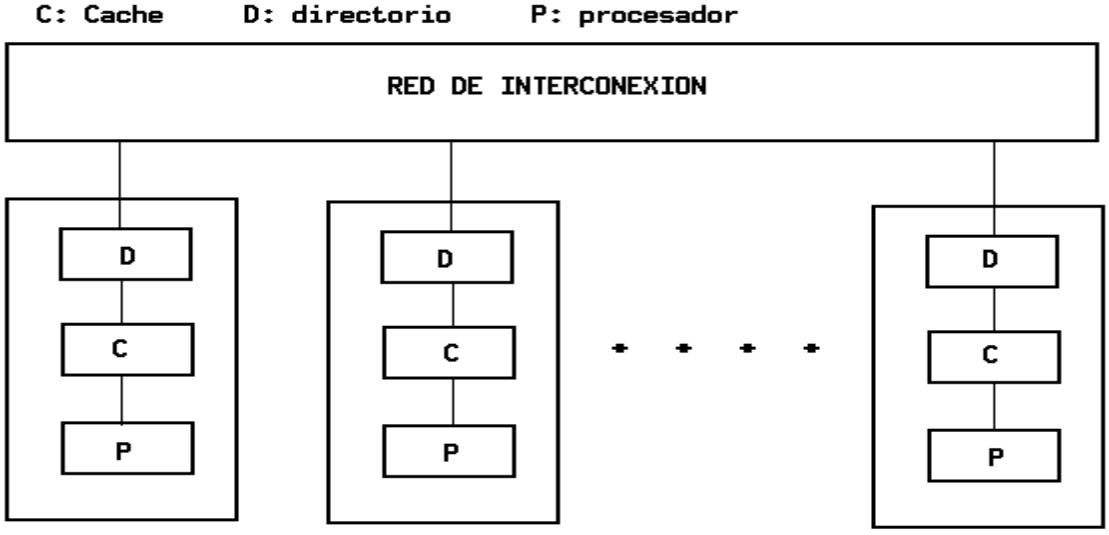

Fig. 20.4. - El modelo de multiprocesador COMA. La KSR-1.

20.8. - **Multiprocesadores de Memoria Distribuida**

Consisten de múltiples computadores llamados a menudo *nodos* interconectados por una red de pasaje de mensajes. Cada nodo es un sistema computador autónomo con su procesador, su memoria y a veces periféricos de E/S.

La red de pasaje de mensajes provee un mecanismo de comunicación punto-a-punto. Todas las memorias locales son privadas y son solo accesibles por el conjunto de procesadores locales del nodo, por esta razón se las denomina máquinas **NORMA** (no remote memory access). La comunicación entre los nodos se logra a través de la red de pasaje de mensajes.

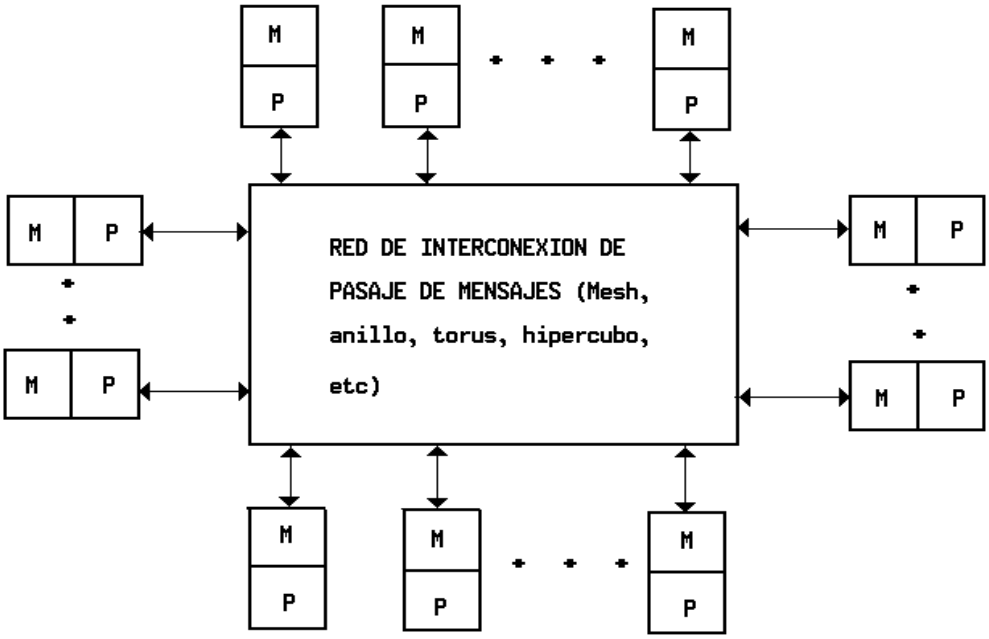

Fig. 20.5. - Multicomputador genérico. Pasaje de mensajes.

20.9. - **Generaciones de Multicomputadoras**

Las multicomputadores de pasaje de mensajes provienen de dos generaciones de desarrollo y existe una tercer generación emergente.

La *primera generación* (1983-1987) se basó en tecnología de la placa del procesador (board) utilizando arquitectura hipercubo y control de mensajes por software. Ejemplos de esta generación son la Caltech Cosmic y la Intel iPSC/1.

La *segunda generación* (1988-1992) se implementó sobre arquitecturas mesh-connected, ruteo de mensajes por hardware y un entorno software para distribuir medianamente el cómputo. Ejemplos de esta generación son la Intel Paragon y la Parsys SuperNode 1000.

La *tercera generación* (1993-1997) se espera que provea multicomputadoras de un refinamiento mayor tales como la MIT J-Machine y la Caltech Mosaic implementadas con mecanismos de transmisión de comunicación y los procesadores en el mismo chip VLSI.

20.10. - **UNA TAXONOMÍA DE COMPUTADORAS MIMD**

Las computadoras paralelas aparecen tanto como configuraciones SIMD y MIMD. Las SIMD parecen más apropiadas para aplicaciones específicas.

La tendencia arquitectural para las futuras computadoras de propósito general apunta a las MIMD con memorias distribuidas proveyendo un espacio de direccionamiento globalmente compartido.

Gordon Bell (1992) proveyó una taxonomía de máquinas MIMD que se muestra en la siguiente tabla.

Este autor considera los multiprocesadores de memoria compartida como que poseen un único espacio de direccionamiento.

Las multicomputadoras o multiprocesadores escalables deben utilizar memoria compartida distribuida.

Los multiprocesadores no escalables utilizan una memoria compartida central.

Las multicomputadoras utilizan memorias distribuidas con múltiples espacios de direccionamiento. Son escalables con memoria distribuida.

Las multicomputadoras centralizadas aún no han aparecido.

MIMD	**Multiprocesadores único espacio de direccionamiento Memoria compartida**	Multiprocesadores de memoria distribuida (escalables)	Procesadores con enlace dinámico de direcciones (KSR)
			Procesadores con enlace estático de direcciones, multi-anillo (estándar propuesto IEEE SCI)

			Procesadores con enlace estático de direcciones con uso de cache (Alliant, DASH)
			Enlace estático de programas (BBN, Cedar, CM*)
	Multicomputadores Múltiples espacios de direccionamiento Pasaje de mensajes	Multiprocesadores de memoria centralizada (no escalables)	Cross-point o múltiples etapas (Cray, Fujitsu, Hitachi, IBM, NEC, Tera)
			Multianillo, multibus
			Multibus (DEC, Encore, NCR, Sequent, SGI, Sun)
		Multicomputadores distribuidos (escalables)	Conexión Mesh (Intel)
			Butterfly / Árbol ancho (CM5)
			Hipercubo (NCUBE)
			LANs rápidas para alta disponibilidad y clusters de alta capacidad (DEC, Tandem)
			LANs para procesamiento distribuido (workstations, PCs)
		Multicomputadores centralizados	

20.11. - **INTRODUCCION A LOS SISTEMAS DISTRIBUIDOS**

20.11.1. - **Introducción**

Desde 1945 hasta el 85 las computadoras eran grandes y caras incluso las minicomputadoras. Sin embargo a partir de la década de los 80 hubo dos importantes avances tecnológicos:

- el desarrollo de poderosos microprocesadores (similares a los de los mainframes)
- el desarrollo de las redes de área local de alta velocidad (LAN)

Estos sistemas permitieron conectar varios computadores transmitiendo información entre ellos a muy alta velocidad. El resultado neto de estas dos tecnologías recibe el nombre genérico de Sistemas Distribuidos en contraste de los Sistemas Centralizados.

20.11.2. - **Ventajas de los S. D. con respecto de los Sistemas Centralizados**

Economía : Los microprocesadores ofrecen mejor proporción de precio/rendimiento que los mainframe.

Velocidad : Un Sistema Distribuido puede tener un mayor poder de cómputo que un mainframe ya que al agregar otra terminal, se incrementa dicha potencia mientras que en el mainframe hay que cambiar el procesador por otro más veloz, y esto no siempre es posible.

Distribución inherente : Algunas aplicaciones poseen una distribución inherente a sí mismas, por ejemplo las diferentes sucursales de un supermercado operan casi totalmente en base a operaciones y decisiones locales y necesitan comunicarse entre sí solo en lo que respecta a intercambio de información.

Confiabilidad : Si una máquina falla, el sistema sobrevive en un gran porcentaje.

Crecimiento incremental : Se puede añadir poder de cómputo en pequeños incrementos agregando más computadoras.

20.11.3. - **Ventajas de los S. D. con respecto a las PC.**

Datos compartidos : Permite el acceso por varios usuarios a una base de datos en común. Facilita la coherencia y consistencia de la información.

Dispositivos compartidos : Permiten que varios usuarios compartan periféricos caros como ser impresoras láser, etc. Inclusive remotamente.

Comunicación : Facilita la comunicación de persona a persona. Por ejemplo correo electrónico.

Flexibilidad : Distribuye la carga de trabajo entre las máquinas disponibles en forma más eficaz en cuanto a los costos.

20.11.4. - **Desventajas**

Software : Existe poco software para sistemas distribuidos en la actualidad.
Redes : La red se puede saturar o causar otros problemas como ser pérdida de mensajes, ruido en la línea, etc.
Seguridad : Si bien el poder compartir los datos es una ventaja se torna también en un problema la privacidad de los mismos.

20.12. - CONCEPTOS DE HARDWARE

La clasificación de hardware más citada es la de Flynn que si bien se detiene en los tipos SISD, SIMD, MISD y MIMD, para un sistema distribuido, se usa la arquitectura MIMD, dividiéndola en aquellas que tiene memoria compartida, usualmente denominadas *Multiprocesadores* (Sistemas fuertemente acoplados) y aquellas que no, llamada *Multicomputadoras* (Sistemas débilmente acoplados).

La diferencia fundamental es que en los sistemas multiprocesadores existe un espacio de direcciones virtuales compartido por todas las CPU y en la multicomputadoras cada máquina tiene su propia memoria.

Cada una de estas categorías se pueden subdividir en base a la arquitectura de la red de interconexión: *bus* o *conmutador* (switch).

Los sistemas fuertemente acoplados tienden a ser usados como sistemas paralelos y los débilmente acoplados como sistemas distribuidos.

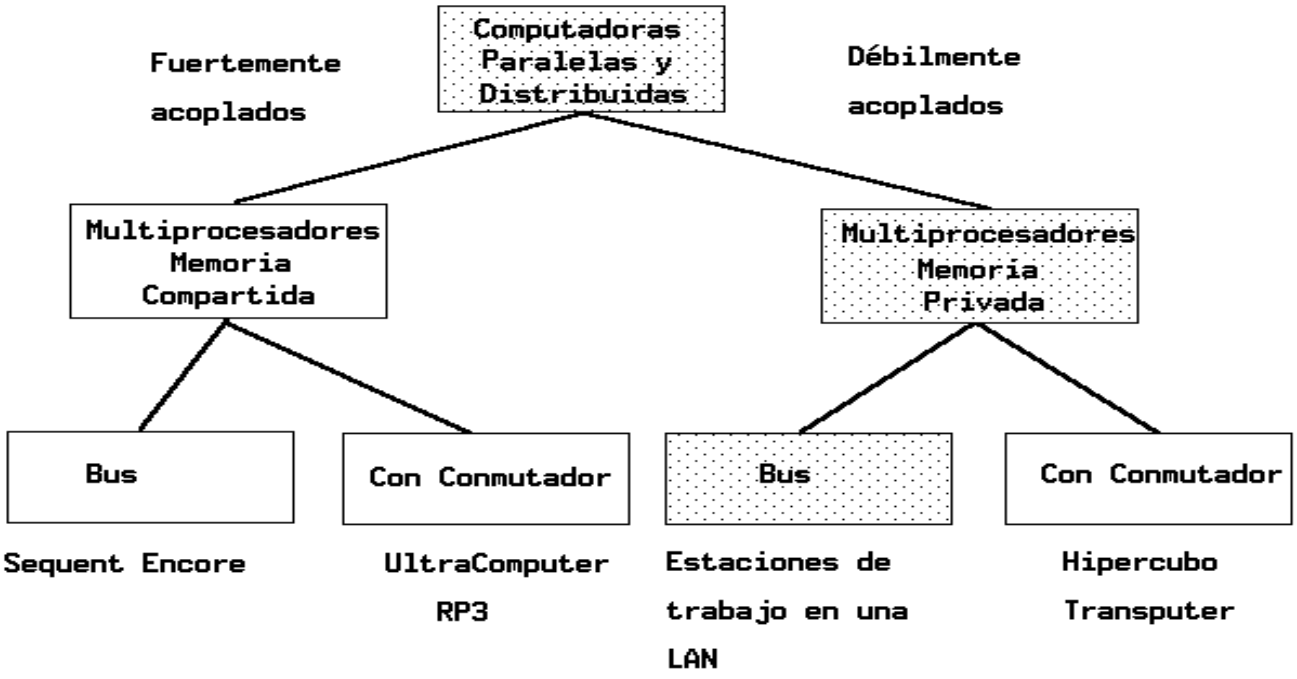

Fig. 20.6. - Clasificación de Computadoras.

20.12.1. - Multiprocesadores con base en buses

Estos sistemas constan de un cierto número de CPUs conectados a un bus común, junto con un módulo de memoria. Esta memoria es *Coherente* ya que si una CPU escribe algo sobre una dirección de memoria y luego otra CPU la lee, obtiene el valor actualizado.

El problema de este esquema es que con 4 o 5 CPU's el bus estará por lo general sobrecargado y el rendimiento disminuirá, para solucionar esto se prevén memorias cachés entre el CPU y el bus. El tamaño tipo de un caché varía desde 64 K hasta 1 M dando como resultado una tasa de hallazgos del 90 % o más.

Para solucionar el problema de la coherencia del caché cuando dos CPU's acceden al mismo dato y lo copian en sus respectivos cachés para luego actualizarlo se ideó que cuando una palabra es escrita en el Cache, también sea escrita en la memoria. Se denomina a este caché, *caché de escritura* (el tráfico en el bus solo es para escritura).

Además todos estos caches monitorean el bus constantemente. Cada vez que un caché observa una dirección de memoria de la cual él tiene una copia puede actualizarla en base al dato que está circulando en el bus. Se denomina *Cachés Monitores* a este tipo de cachés ya que observan al bus y se actualizan automáticamente.

La mayoría de los multiprocesadores basados en buses utilizan esta tipo de arquitectura con cachés de escritura y cachés monitores u otra muy similar, de esta forma es posible colocar desde 32 hasta 64 CPU's en el mismo bus.

20.12.2. - Multiprocesador con conmutador

Para poder conectar más de 64 procesadores es necesario un método distinto de conectar cada CPU a la memoria. Una posibilidad es dividir la memoria en módulos y conectarlos a las CPU con un conmutador Crossbar Switch (ver capitulo 7).

La ventaja de este esquema es que muchas CPU pueden acceder simultáneamente a la memoria, siempre y cuando no accedan al mismo módulo en cuyo caso deberán esperar.

La desventaja es que teniendo n CPU y n Memorias, se necesitan n^2 conmutadores. Si n es grande el costo puede ser prohibitivo.

Otra red posible es la red omega que necesita N etapas de conmutación en la cual con n CPU's y n memorias se necesitan $\log_2 n$ etapas de conmutación cada una de las cuales tiene n/2 conmutadores para un total de (n $\log_2 n$) / 2 conmutadores.

Igualmente si la cantidad de conmutadores es muy alta, por ejemplo n = 1024 con 10 etapas de conmutación la velocidad requerida de cada conmutador para responder a un requerimiento puede obligar a que cada conmutador posea un tiempo de respuesta tal que elevaría el costo individual alcanzando un costo total de la red también prohibitivo.

Otra implementación que intenta reducir un poco los costos es mediante los sistemas jerárquicos en los cuales cada CPU tiene asociada una memoria con rápido acceso y puede acceder pero más lentamente a la memoria de los demás (NUMA). La desventaja de este esquema es que la colocación de los programas y de los datos en el esquema NUMA se convierte en un factor crítico cuando se intenta lograr que la mayoría de los accesos sean a la memoria local.

Como conclusión final se puede decir que la construcción de un multiprocesador grande fuertemente acoplado y con memoria compartida es difícil y cara.

20.12.3. - Multicomputadoras con base en buses (Sistema Distribuido)

Este sistema es de fácil construcción pero tiene el problema de la forma en que se comunican las CPU. Es necesario cierto esquema de interconexión pero como solo es entre CPUs el volumen de tráfico será varios ordenes menor que si se utiliza la red de interconexión para el tráfico CPU - Memoria (por ejemplo de entre 10/100 Mb/seg en el primer caso y 300 Mb/seg en el otro).

Fig. 20.7. - Estaciones de trabajo en una LAN.

20.12.4. - Multicomputadoras 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. Ejemplo Retícula, también llamada Red con Vecinos Cercanos o Mesh-Connected, Hipercubo, etc. Estas arquitecturas se adecuan a problemas de naturaleza bidimensional. Actualmente existen hipercubos disponibles comercialmente en el mercado de hasta 16384 CPU's.

20.13. - **CONCEPTOS DE SOFTWARE**

Daremos una introducción a los distintos tipos de sistemas operativos para los multiprocesadores y multicomputadoras. Los sistemas operativos no se pueden clasificar tan fácil como el hardware. A pesar de que el software es algo vago se pueden distinguir dos tipos de sistemas operativos: los débilmente acoplados y los fuertemente acoplados. Esto es un tanto análogo al hardware débilmente y fuertemente acoplado.

En las siguientes secciones analizaremos algunas de las combinaciones más comunes de software y hardware.

20.13.1. - **Sistemas operativos de redes** (Software débilmente acoplado en Hardware débilmente acoplado)

Es una situación 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 ellas como un *sistema operativo de red*.

Una función importante del sistema operativo de red es permitir a los usuarios hacer un login en forma remota con otra computadora, por ejemplo Internet nos brinda el Telnet para este propósito. Una vez que se ha

establecido la comunicación, el software de la red crea una unión transparente y bidireccional tal que todos los caracteres ingresados por el usuario son mandados al proceso que se está ejecutando en forma remota y todos los datos que salen del proceso son enviados de vuelta al usuario. Esto nos permite realizar operaciones sobre la máquina remota como si fuésemos cualquier usuario local.

Otra función que permiten los sistemas operativos de red es proveer un mecanismo de transferencia de archivos. Esto se implementa por ejemplo contando con una máquina que actúa como **servidor** de archivos y que provee a las otras máquinas de los usuarios **clientes** de servicios para leer o escribir estos archivos.

En los servidores de archivos la información se estructura con alguna jerarquía y los clientes pueden importar o montar esta jerarquía aumentando la ya existente en sus propias máquinas.

Internet brinda este mecanismo de transferencia de archivos por medio del FTP (File Transfer Protocol).

1) FTP remote machine
2) Posicionarse en el directorio donde está el archivo y GET arch_origen arch_destino

El mecanismo FTP es implementado de una manera similar al telnet. Hay un daemon en la máquina remota que observa peticiones de conexión al port del sistema FTP. Comandos que brinda:

GET	Copia un archivo de la máquina remota a la local.
PUT	Copia un archivo de la máquina local a la remota.
LS // DIR	Lista archivos en el directorio actual de la remota.
CD	Cambia de subdirectorio en el nodo remoto.

El sistema operativo a usar en este ambiente debe controlar las estaciones de trabajo en lo individual, a los servidores de archivos y también encargarse de la comunicación entre ellos.

20.13.2. - **Sistema multiprocesador de tiempo compartido** (Software fuertemente acoplado en Hardware fuertemente acoplado)

La característica clave de este tipo de sistema es la existencia de una sola cola para la ejecución, es decir, una lista de todos los procesos en el sistema que no están bloqueados en forma lógica y listos para su ejecución.

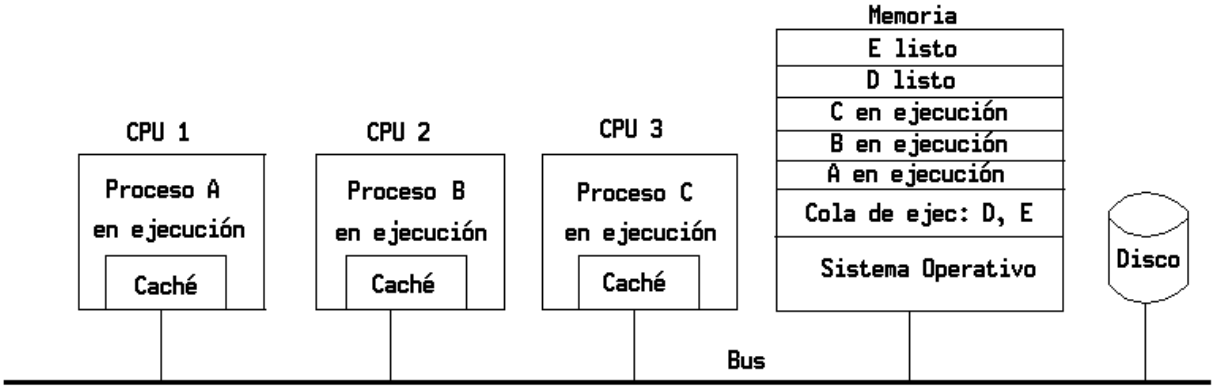

Fig. 20.8. - Un multiprocesador con una cola de ejecución.

La cola de ejecución es una estructura de datos contenida en la memoria compartida.

Existen varias máquinas específicas en esta categoría como ser las dedicadas a bases de datos. Operan como un sistema de tiempo compartido pero con varias CPU's.

Un área en donde este tipo de multiprocesador difiere de una red o un sistema distribuido es en la organización del sistema de archivos. Usualmente el sistema operativo cuenta con un sistema de archivos tradicional.

En muchos de estos sistemas se dedica un CPU a la ejecución del sistema operativo y los otros atienden las tareas de los usuarios. Nótese que en estos casos el CPU que ejecuta el S.O. se convierte a menudo en un cuello de botella.

20.13.3. - **Sistema realmente distribuido** (Software fuertemente acoplado en Hardware débilmente acoplado).

El objetivo es crear la ilusión en la mente de los usuarios de que toda la red es un solo sistema de tiempo compartido.

Definición de Sistema Distribuido: Es aquel que se ejecuta en una colección de máquinas sin memoria compartida pero que aparece ante sus usuarios como una sola computadora (uniprocesador virtual).

Características:

- Debe existir un Mecanismo de comunicación global entre los procesos (cualquiera puede hablar con cualquiera). No tiene que haber distintos mecanismos en distintas máquinas o distintos mecanismos para la comunicación local o la comunicación remota.
- Debe existir un esquema global de protección

- La administración de procesos debe ser la misma en todas partes (crear, destruir, iniciar, detener)
- Debe existir un sistema global de archivos y debe tener la misma apariencia en todas partes

Elemento	S.O. De Red	S.O. Distribuido	S.O. Multiprocesador
Se ve como un uniprocesador virtual ?	No	Si	Si
Mismo Sistema Operativo en todas ?	No	Si	Si
Cuantas copias del S.O. existen ?	N	N	1
Cómo se logra la Comunicación ?	Archivos compartidos	Pasaje de mensajes	Memoria compartida
Requiere acuerdos de Protocolos de la red?	Si	Si	No
Existe una única cola de ejecución ?	No	Si	Si
Existe una semántica bien definida para archivos compartidos ?	Generalmente No	Si	Si

20.14. - **SISTEMAS DISTRIBUIDOS**

Definiremos a los Sistemas Distribuidos como sistemas "débilmente acoplados", donde los procesadores no comparten memoria común ni reloj y donde cada procesador tiene su propia memoria local.

La comunicación entre ellos se realiza por medio de vías de comunicación (buses, líneas telefónicas, etc.).

Los objetivos de un Sistema Distribuido son:

- compartir recursos,
- mayor capacidad de procesamiento, y
- comunicación.

La mayor capacidad de procesamiento se da cuando es posible dividir una tarea en varias subtareas concurrentes. Si es posible que éstas ejecuten en distintos computadores se obtiene un alto grado de paralelismo.

Otra situación se da cuando una computadora que se encuentra sobrecargada de trabajo lo deriva a otros procesadores.

Compartir recursos significa la posibilidad de utilización de un recurso no disponible en forma local, pero sí hacer posible su uso en forma remota. Además sería posible la utilización de recursos, descompuestos en un sitio, por otros de igual capacidad en forma remota.

La Comunicación significa el intercambio de información, y abarca desde distribuir procesos hasta el simple correo electrónico.

20.14.1 - **MODOS DE PROCESAMIENTO**.

Los Sistemas Distribuidos permiten La Migración de Datos, La Migración de Procesos y La Migración de Trabajos.

- **Migración de Datos** : cuando un usuario remoto requiere un archivo existen dos maneras de satisfacerlo. Una es enviarle el archivo entero y la otra es enviarle la porción que necesita, si luego necesita más se le enviará. En el primer caso cuando haya modificado el archivo lo reenviará completo, en el segundo reenviará las porciones modificadas.
- **Migración de Procesos** : en muchos casos es preferible, cuando los archivos son muchos o de gran volumen, transferir los procedimientos. Aquí también tenemos dos alternativas : una es que un procedimiento invoque una necesidad a otro en el sitio remoto y que éste la satisfaga (Remote Procedure Call) y la otra es que un proceso envíe un mensaje al sitio remoto y allí se genere un nuevo proceso que satisfaga sus necesidades.
- **Migración de Trabajos** : significa la transferencia directa de Trabajos de un nodo a otro, con lo cual se puede lograr :
  - balancear el sistema
  - subdivisión de tareas que ejecutan en forma concurrente
  - uso de mejor o nuevo hardware
  - uso de mejor o nuevo software

20.15. - **ASPECTO DEL DISEÑO**

20.15.1. - Transparencia

Se trata de lograr la imagen de un único sistema, que todas las personas piensen que la colección de máquinas sea un sistema de tiempo compartido de un solo procesador. Se puede lograr en dos niveles distintos:

1) Ocultar la distribución a los usuarios.
El usuario no tiene por qué saber que la ejecución, por ejemplo, de una serie de compilaciones se realiza en paralelo en distintas máquinas.
2) Transparencia al sistema para los programas.
Lograr diseñar la interfaz de llamadas al sistema de modo que no sea visible la existencia de varios procesadores.
El concepto de transparencia se puede aplicar a distintos aspectos de un sistema distribuido :

Transparencia de localización	Los usuarios no pueden indicar la localización de los recursos
Transparencia de migración	Los recursos se pueden mover de una localización a otra sin cambiar sus nombres
Transparencia de réplica	Los usuarios no pueden indicar el número de copias existentes. El S.O. puede hacer copias de los archivos sin que los usuarios lo noten.
Transparencia de concurrencia	Varios usuarios pueden compartir recursos de manera automática. El S.O. asegura un acceso secuencial no concurrente.
Transparencia de paralelismo	Las actividades pueden ocurrir paralelamente sin que los usuarios lo sepan.

20.15.2. - Flexibilidad

Es importante que el sistema sea flexible ya que las decisiones de diseño que parecen hoy razonables podrían demostrar luego ser incorrectas.

Existen dos escuelas de pensamiento en las estructuras de un SO:

- una mantiene que cada máquina debe ejecutar un núcleo tradicional que proporcione la mayoría de los servicios (*núcleo monolítico*);
- 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 a nivel usuario (*micronúcleo*).

El primero es el SO centralizado básico actual aumentado con capacidades de red y la integración de servicios remotos. La mayoría de las llamadas al sistema se realizan mediante interrupciones al núcleo en donde se realiza el trabajo para después devolver el resultado deseado al proceso del usuario. La mayoría de las máquinas tienen disco y administran sus propios sistemas locales de archivos (extensiones e imitaciones de Unix).

El segundo (microKernel) es el más flexible pues casi no hace nada. Proporciona cuatro servicios mínimos
1) Un mecanismo de comunicación entre procesos.
2) Cierta administración de la memoria.
3) Una cantidad limitada de planificación y administración de procesos de bajo nivel.
4) Entrada/Salida de bajo nivel.

No provee (a diferencia del monolítico) el sistema de archivos, el sistema de directorios, la administración completa de los procesos ni gran parte del manejo de las llamadas al sistema.

Todos los servicios del SO se implementan por lo general como servidores a nivel usuario. Es precisamente esta capacidad de añadir, eliminar y modificar servicios lo que da al microKernel su flexibilidad.

La única ventaja del núcleo monolítico es el rendimiento. Las interrupciones al núcleo y la realización de todo el trabajo allí puede ser más rápido que el envío de mensajes a los servidores remotos.

20.15.3. - Confiabilidad

La idea es que si una máquina falla alguna otra se encargue del trabajo. La confiabilidad global del sistema es mayor que la confiabilidad de un servidor individual. Uno de los aspectos de la confiabilidad es la disponibilidad que se refiere a la fracción de tiempo que se puede utilizar el sistema. Otra herramienta para el mejoramiento de la disponibilidad es la redundancia (duplicar piezas de hard y soft). Un sistema altamente confiable debe ser altamente disponible, y además los datos confiados al sistema no deben perderse o mezclarse de manera alguna (todas las copias deben ser consistentes).

Otro aspecto de la confiabilidad general es la seguridad. Los archivos y otros recursos deben ser protegidos contra el uso no autorizado. Esto es similar a los métodos vistos pero es más severo en los sistemas distribuidos. No es sencillo determinar de quién proviene una solicitud "mensaje", lo que requiere un cuidado considerable.

Otro aspecto es la tolerancia de fallas. Si un servidor falla y vuelve a arrancar de manera súbita, si el servidor tenía tablas con información importante respecto de las tareas en curso lo menos que puede ocurrir es que la recuperación será difícil .

20.15.4. - Desempeño o Performance

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, por ejemplo :

- Tiempo de respuesta.

- Rendimiento (cantidad de trabajos por hora).
- Uso del sistema.
- Cantidad consumida de la capacidad de la red.

El problema del desempeño se complica por el hecho de que la comunicación (factor esencial en un sistema distribuido) es algo lenta por lo general.

La mayoría de este tiempo se debe a un manejo inevitable que realizan los protocolos en ambos extremos en lugar del tiempo que tardan los bits en viajar por el medio de transmisión.

La tolerancia a fallas también afecta el desempeño.

20.15.5. - Escalabilidad

La mayoría de los sistemas distribuidos están capacitados para trabajar con unos cuantos cientos de CPU's. No sería una buena idea tener un solo servidor de correo para cincuenta millones de usuarios, aunque tuviera la suficiente capacidad de CPU y almacenamiento para ello, la capacidad de la red dentro y fuera de él sí sería un problema. Además no toleraría bien los fallos. Las tablas centralizadas son tan malas como los componentes centralizados. Los algoritmos centralizados también son una mala idea, el problema es la recolección de la información para el algoritmo.

En los algoritmos descentralizados:

1) Ninguna máquina tiene la información completa acerca del estado del sistema.
2) Las máquinas toman decisiones solo en base a la información disponible de manera local.
3) El fallo de una máquina no arruina el algoritmo.
4) No existe una hipótesis implícita de la existencia de un reloj global. (ver capítulo de Sincronización)

20.15.6. - **Ventajas y Desventajas**

Puntos a favor:

- buena proporción precio/desempeño y se pueden ajustar bien a las aplicaciones distribuidas,
- pueden ser altamente confiables y aumentar su tamaño en forma gradual al aumentar la carga de trabajo.

Puntos en contra:

- software más complejo
- potenciales cuellos de botella en la comunicación
- la seguridad es débil

Los modernos sistemas de cómputos tienen con frecuencia varios CPU. Se pueden organizar con multiprocesadores (memoria compartida) o multicomputadoras (sin memoria compartida), ambos se pueden basar en un bus o en un conmutador, los primeros suelen ser fuertemente acoplados mientras que los segundos débilmente acoplados.

Los sistemas distribuidos deben diseñarse con cuidado, teniendo en cuenta la transparencia, la flexibilidad, confiabilidad, desempeño y escalabilidad. En los sistemas operativos de red, los usuarios conocen la existencia de múltiples máquinas y es necesario para acceder a estos recursos un login en la máquina remota apropiada, o transferir datos de la remota a su propia máquina. Mientras que en los sistemas operativos distribuidos, los usuarios no necesitan tener en cuenta las múltiples máquinas; ellos acceden a prestaciones remotas de la misma forma que a las locales.

El software se divide en tres clases:

- Sistemas operativos de red (estación de trabajo independiente)
- Sistemas operativos distribuidos (convierten toda la colección de hardware y software en un único sistema integrado).
- Multiprocesadores con memoria compartida (no son sistemas distribuidos).

20.16. - **COMUNICACIONES**

La mayor y más importante diferencia entre un sistema distribuido y un uniprocesador es el mecanismo de comunicación entre procesos. Para que la comunicación sea coherente, existen protocolos.

20.17. - **ESTRATEGIAS DE DISEÑO**

Niveles de un Protocolo

Cuando A se quiere comunicar con B, primero arma el mensaje y luego ejecuta un System Call y el S.O. toma el mensaje y lo envía a través de la red. Para esto A y B se tienen que poner de acuerdo si los bits se toman de izquierda a derecha, si se mandan en ASCII o EBCDIC, etc.

Para facilitar el trabajo con los distintos niveles y aspectos correspondientes a la comunicación, la organización internacional de estándares (International Standards Organization -ISO-) ha desarrollado un modelo de refe-

rencia que identifica en forma clara los distintos niveles, les da nombres estandarizados y señala cuál nivel debe realizar cada trabajo. Este modelo se llama el *modelo de referencia para interconexión de sistemas abiertos* (Day y Zimmerman, 1983), lo cual se abrevia por lo general como **ISO OSI** o a veces solo como **modelo OSI**.

El modelo OSI está diseñado para permitir la comunicación de los sistemas abiertos. Se dice que un sistema es abierto cuando está preparado para comunicarse con cualquier otro sistema abierto mediante estándares que gobiernan el formato, contenido y significado de los mensajes enviados y recibidos. Básicamente un protocolo es un acuerdo en cómo debe llevarse la comunicación. El modelo OSI distingue entre dos tipos de protocolos:

Orientados a conexión: Establecen una conexión explícita entre Sender y Receiver y posiblemente negociación de qué protocolo van a utilizar (Ej. Teléfono)

Orientados sin conexión: No se realiza ningún tipo de negociación previa, se envían los datos cuando el Sender está listo.

El modelo OSI se divide en siete capas/niveles. 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 *interfase* con la otra capa por encima de ella. La interfase es un conjunto de operaciones que juntas definen el servicio que la capa está preparada para ofrecer a sus usuarios. Estas capas son:

1) *Física* : Responsable de los detalles físicos
2) *Data Link* : Responsable de la comunicación de la red (protocolos)
3) *Network* : Responsable de los paquetes (asegura la trayectoria)
4) *Transport* : Responsable del transporte de paquetes con un orden
5) *Sesión* : Sirve como interfase entre el usuario y el servicio de transporte.
6) *Presentación* : Homogeneización de datos y de dispositivos (criptografía, compresión, etc.)
7) *Aplicación* : Responsable del manejo de datos. Concierne al soporte de aplicación del usuario.

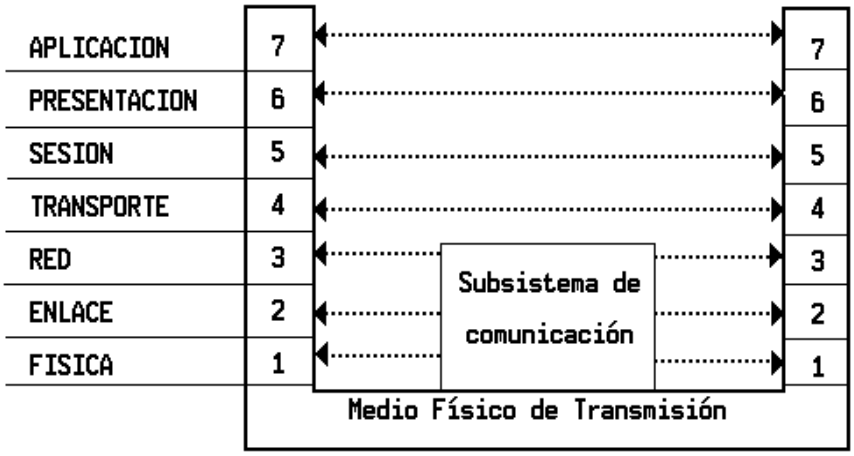

Fig. 20.9. - El modelo en capas OSI/ISO.

20.17.1. - Hardware Layer o Capa Física

Es la que se encarga de transmitir los ceros y unos a la máquina vecina, cuántos bits por segundo transmite, el tipo de conector de red, si la transmisión es bidireccional simultáneamente o no, etc. El protocolo de la capa física se encarga de la estandarización de las interfases eléctricas mecánicas y de señalización. Por ejemplo uno de los estándares desarrollados para esta capa es la comunicación serial RS-232.

20.17.2. - Data Link Layer o Capa de Enlace

Esta capa detecta errores de transmisión es decir agrupa un conjunto de bits en frames (tramas, marcos) y chequea que cada frame sea recibido correctamente (pone información extra en el mensaje y agrega un checksum (suma de verificación), además todas las tramas tienen un número de secuencia como etiqueta.

20.17.3. - Network Layer o Capa de Red

Para que un mensaje llegue del emisor al receptor, tienen que hacer un cierto numero de saltos y en cada uno de ellos elegir un nuevo nodo al cual ir (hops). Cómo elegir un camino se llama Routing (ruteo) y es el principal trabajo de esta capa (asegura la trayectoria). Para esto hay que tener en cuenta el tráfico y cuántos mensajes existen encolados para salir por una línea en cada nodo.

Un protocolo conocido orientado a conexión de esta capa es el **X.25** que es el utilizado en grandes redes (WANs). Al inicio el usuario de X.25 envía una solicitud de llamada a un destino, el cual puede aceptar o rechazar la conexión propuesta. Si se acepta la conexión, quien hace la llamada obtiene un identificador de conexión para usarlo en las solicitudes posteriores. En muchos casos la red escoge una ruta al receptor durante esta configuración y la utiliza para el tráfico posterior.

Otro protocolo pero sin conexión es el **IP** (Protocol Internet) y es parte de los protocolos DoD (Departamento de Defensa de los Estados Unidos). Un paquete IP se envía sin configuración alguna.

Cada paquete tiene una ruta hacia su destino independiente de los otros paquetes, e inclusive pueden tomar distintos caminos para llegar a destino. No se selecciona ruta alguna ni se recuerda ésta, como ocurre a menudo en X.25.

Estas tres capas anteriores se encuentran involucradas indefectiblemente en todos los nodos del camino que toma el mensaje, las cuatro siguientes solo en el emisor y receptor (origen y destino).

20.17.4. - Transport Layer o Capa de Transporte

El trabajo de esta capa es proveer confiabilidad a nivel de proceso, es decir estar seguro que tarde o temprano va a llegar a destino, para esto el mensaje que recibe de la capa sesión lo fracciona, le asigna un número secuencial y después envía todos. Se puede decir que es responsable del transporte de paquetes manteniendo su orden (fragmentación de paquetes) para que estos signifiquen un mensaje.

Estos tipos de protocolos pueden ser implementados sobre X.25 o IP. En el primer caso, los paquetes llegarán en orden mientras que en el segundo, la capa de transporte debe reordenarlos antes de pasarlos a la aplicación que va a recibirlos.

El protocolo de transporte oficial ISO tiene cinco variantes conocidas como TP0 a TP4. El protocolo de transporte DoD orientado a conexión se llama **TCP** (Transmission Control Protocol). Es similar a TP4. La combinación TCP/IP es muy utilizada en universidades y en la mayoría de los sistemas UNIX. La serie de protocolos DoD también soportan un protocolo de transporte sin conexión llamado **UDP** (Universal Datagram Protocol) que en esencia es igual a IP con ciertas adiciones menores. Los programas del usuario que no necesitan un protocolo orientado a conexión utilizan por lo general UDP.

20.17.5. - Session Layer o Capa de Sesión

Es una refinación de la capa de transporte, mantiene el registro de quién está hablando y provee facilidades de sincronización. Esto último permite a los usuarios colocar Checkpoints en las transmisiones largas (por si se cae la red). Generalmente esta capa no se implementa.

20.17.6. - Presentation Layer o Capa de Presentación

Esta capa es la encargada del significado de los bits, permite la homogeneización de datos, facilita la comunicación entre las máquinas con distintas representaciones internas, o sea que ambas partes entiendan lo mismo de un mensaje. En esta capa se pueden definir registros que contengan campos como los mencionados y que entonces el emisor notifique al receptor que un mensaje contiene un registro particular en cierto formato. Ej. Criptografía, Compresión y Conversión de datos.

20.17.7. - Application Layer o Capa de Aplicación

Esta capa es la que interactúa con los usuarios, brindando una colección de protocolos misceláneos para actividades comunes como ser File Transfer (FTP), Correo Electrónico, Remote Logins, Telnet's. Los más conocidos de estos protocolos son el X.400 de correo electrónico y el servidor de directorios X.500.

20.18. - **ESTRATEGIAS DE RUTEO**

Se llama ruteo a una sucesión de nodos o terminales o gateways que deben ser recorridos para el envío de mensajes a través de la red.

Las estrategias empleadas pueden ser:

Rutas Fijas : El camino entre dos nodos es fijo o sea que el envío de los mensajes se realiza por el mismo camino.

Circuito Virtuales : El camino entre dos nodos es fijo durante una sesión de mensajes y puede variar en otras. O sea que cuando se toma un camino para enviar no se lo libera hasta su fin, pero si después es necesario volver a mandar mensajes al mismo nodo, se puede tomar otro camino distinto, logrando con esto tal vez evitar un embotellamiento de paquetes.

Ruteo Dinámico : El camino entre dos nodos se determina en el momento de enviarse el mensaje (cada mensaje). Esto implica que los paquetes consecutivos viajan por distintos caminos.

20.19. - **ESTRATEGIA DE COMUNICACIÓN**

Estas estrategias indican cómo son conectados dos nodos a través de la red. Pueden ser:

Conmutación de circuitos : Se establece una conexión física fija en cada sesión de comunicación, de forma similar a la que se utiliza en el sistema telefónico.

Conmutación de mensajes : Se establece una conexión fija durante toda la transmisión de un mensaje.

Conmutación de Paquetes : Los mensajes son "cortados" en paquetes y cada paquete puede seguir caminos distintos por la red.

20.19.1. - **Disciplina de Prioridades y Protocolos**

En cuanto al acceso al medio (línea de transmisión) siempre es posible definir una Disciplina de Prioridades que establezca el orden en el cual cada nodo accederá a la línea de transmisión.

Los diferentes tipos de Disciplinas de prioridades se basan en métodos del estilo del Round-Robin o Prioridades (que pueden ser estáticas o dinámicas) explicados ya en capítulos anteriores

Asimismo definimos como Protocolo al conjunto de reglas de control que garanticen al nodo de mayor prioridad su acceso al medio independientemente de la Disciplina de Prioridades que se utilice. En consecuencia un mismo Protocolo puede servir a diferentes Disciplinas de Prioridades.

20.19.1.1.- **Ranuras**

El tiempo de uso del medio de transmisión puede ser dividido en intervalos denominados Ranuras o Slots de una cierta duración fija o variable.

La Ranura de Tiempo (Slot Time) es el período de tiempo límite durante el cual podría ocurrir una colisión. Para determinar el tiempo de ranura es muy importante el tiempo de Propagación del mensaje en la línea de conexión.

La disciplina de prioridades y el protocolo de acceso deben hacer posible asignar siempre una ranura a un determinado nodo a efectos de aprovechar al máximo el medio de transmisión.

20.20. - **CLASIFICACION DE PROTOCOLOS**

Los protocolos pueden clasificarse según la forma en la cual administran el acceso al medio de transmisión como Protocolos de acceso Controlado y Protocolos de acceso Contencioso.

20.20.1. - **Protocolos de Acceso Controlado**

20.20.1.1. - **Protocolos con Mecanismo de Reserva**

Uno de los tipos de protocolo de acceso controlado es el que implementa Mecanismo de Reserva. En tales protocolos la ranura se compone de dos partes, una de las cuales está destinada a contener las reservas que van realizando los nodos de las ranuras futuras y la otra parte de la ranura contiene el mensaje.

La ranura de reserva está compuesta por miniranuras cada una de las cuales corresponde a un nodo de la red. Cuando un nodo desea hacer uso del medio expresa su intención de hacerlo indicándolo en la miniranura que le está asignada. El tiempo de la miniranura se establece como el tiempo total de propagación de la reserva desde un extremo a otro de la red.

En consecuencia cuando termina el período de reserva todos los nodos conocen cuáles son los que desean transmitir y por lo tanto ceden el derecho de acceso para la transmisión al nodo de mayor prioridad según la Disciplina de Prioridades establecida.

20.20.1.2. - **Protocolo de Paso de Token (Token Passing)**

En esta clase de protocolos el acceso al medio se controla por medio del pasaje de un nodo a otro de una ficha o token.

El pasaje de una ficha real implica necesariamente una topología de anillo el cual puede ser real (anillo, token ring) o virtual.

El mensaje que se transmite por la línea se denomina trama. En el protocolo con token en la trama existe un encabezamiento que indica si lo que sigue a continuación es un mensaje válido o si la trama está vacía y por ende disponible para que un nodo transmita un mensaje.

El nodo que detecta que la trama está vacía y desea transmitir un mensaje coloca el encabezamiento de mensaje válido, le agrega el mensaje en cuestión y transmite la trama en el medio. Cuando el mensaje ha dado la vuelta completa al anillo el nodo que originalmente lo transmitió lo retira del anillo y devuelve al mismo la trama con encabezamiento de trama vacía cediendo así el token al siguiente nodo en el anillo.

En el caso del anillo virtual o lógico los nodos están conectados a una barra o bus bidireccional (la denominada topología de Bus compartido o Shared Bus). El anillo queda definido porque cada nodo conoce a su predecesor y a su antecesor. El protocolo de pasaje de token para estas topologías se denomina Token Bus.

El protocolo para la topología Token Bus fue reglamentado por la IEEE en su norma 802.4, en tanto que el protocolo para Token Ring corresponde a la norma IEEE 802.5.

20.20.1.3. - **Protocolo de Paso de Token con Slots**

Una forma de que circulen varios mensajes en el anillo consiste en dividir el anillo en varias ranuras. El mecanismo de funcionamiento es en todo similar al del Paso de Token anterior al cual se le debe agregar, por el hecho de que varios mensajes están circulando simultáneamente por el anillo, el hecho de que cada nodo debe ir llevando la cuenta de la cantidad de mensajes que han pasado desde que él envió el suyo a efectos de poder detectar la vuelta del mensaje original que despachó (debe conocer la cantidad de mensajes que circulan en el anillo).

20.20.2. - **Protocolos de Acceso Contencioso**

En estos protocolos un nodo sensa la línea de transmisión y cuando percibe silencio en la misma transmite su mensaje. Puede suceder que dos o más nodos detecten silencio en la línea y transmitan ambos sus propios mensajes, en ese caso se produce lo que se denomina una *colisión*. Tal colisión es escuchada por los nodos que han transmitido los mensajes, los cuales abortan sus respectivos intentos para reintentarlo nuevamente luego de un tiempo generado aleatoriamente. En la Fig. 20.10 puede verse la lógica de funcionamiento de tal mecanismo.

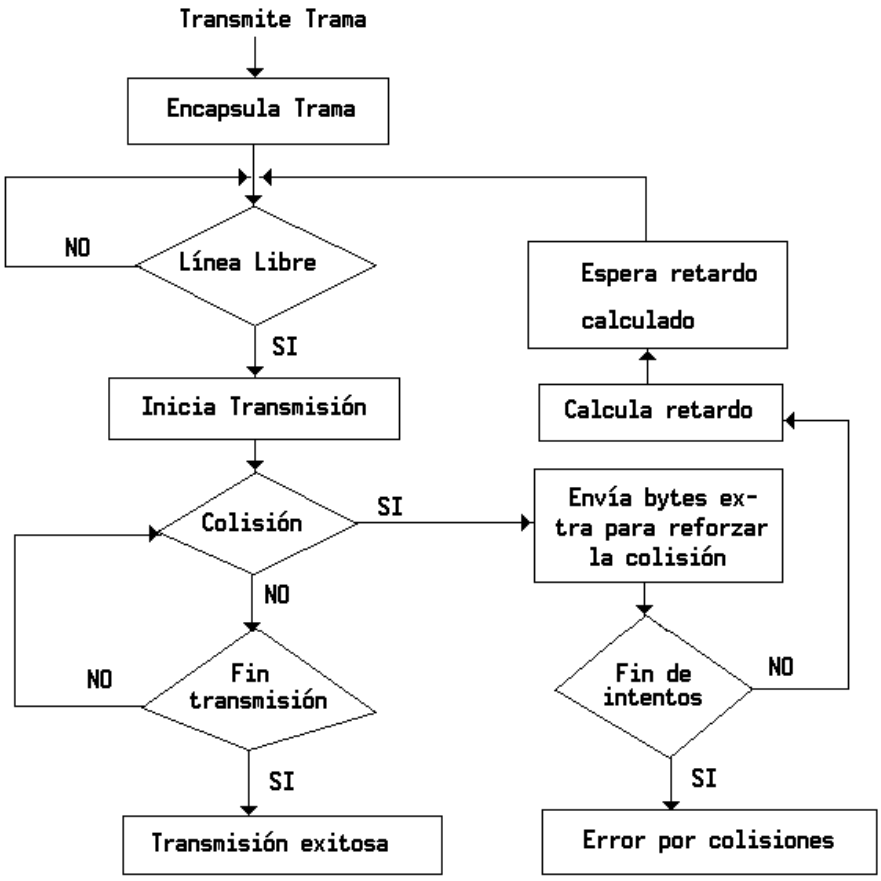

Fig. 20.10. - Comunicación en Protocolos Contenciosos.

20.20.2.1. - **Protocolo CSMA/CD**

En los protocolos de Acceso Múltiple con Sensado de Portadora (del inglés Carrier Sense Multiple Access, CSMA, con Detección de Colisiones -with Collision Detection-) las estaciones escuchan o sensan el canal de transmisión y usan esa información para determinar la posibilidad o no de transmitir.

Existen dos variantes: no-persistente y p-persistente.

20.20.2.1.1. - **CSMA no-persistente**

En este protocolo el nodo escucha la línea si no está ocupada transmite y si está ocupada posterga la transmisión de acuerdo a una cierta política preestablecida de demoras. Estos protocolos pueden ser ranurados o no.

En los protocolos ranurados un nodo puede transmitir solo al comienzo de una ranura. Un mensaje generado en el transcurso de una ranura solo puede transmitirse al inicio de la siguiente. En este caso los mensajes no colisionan o colisionan totalmente.

20.20.2.1.2. - **CSMA p-persistente**

En este protocolo la estación sensa el canal. Si está ocupado demora la transmisión como en el no-persistente. Pero si la línea está libre transmite con probabilidad p. Con probabilidad 1-p espera un tiempo fijo (en general el tiempo de propagación de la señal entre los extremos de la red) al final de lo cual repite el procedimiento anterior. Cuanto menor es p menor es la probabilidad de colisiones.
Ethernet que es un protocolo contencioso es un CSMA 1-persistente.

20.20.2.1.3. - **CSMA/CA**

Otra variante del protocolo CSMA es el CSMA/CA (with Collision Avoidance) en el cual luego de una transmisión se reservan ranuras específicas para cada uno de los nodos en las cuales pueden transmitir sin colisión.

20.21. - **TIPOS DE SISTEMAS**.

Para clasificar los tipos de sistemas existentes un enfoque de acuerdo a la repartición geográfica parece lo más adecuado.

Redes Globales (GAN - Global Area Network): designan generalmente las llamadas redes geográficas, o sea redes que pueden abarcar un país o varios.

Existen dos grandes tipos :

- Las redes de **Teleproceso** que históricamente son las más antiguas, basadas en un computador central al cual se conectan terminales remotas vía módem en topologías estrella o bus compartido.
- Las redes generales de **Conmutación de Paquetes** : basadas en la técnica de conmutación de paquetes que se multiplexan y pueden viajar por diferentes caminos hacia el nodo destino.

Redes Locales (LAN - Local Area Network) : que designan redes de computadores que no están a demasiada distancia entre sí, generalmente con aplicaciones en la oficina e industria.

Redes de Campo (FAN - Field Area Network o SAN - Small Area Network) : realizan la conexión de microcontroladores que operan directamente sobre dispositivos asociados a robots, autómatas, procesos industriales, etc.

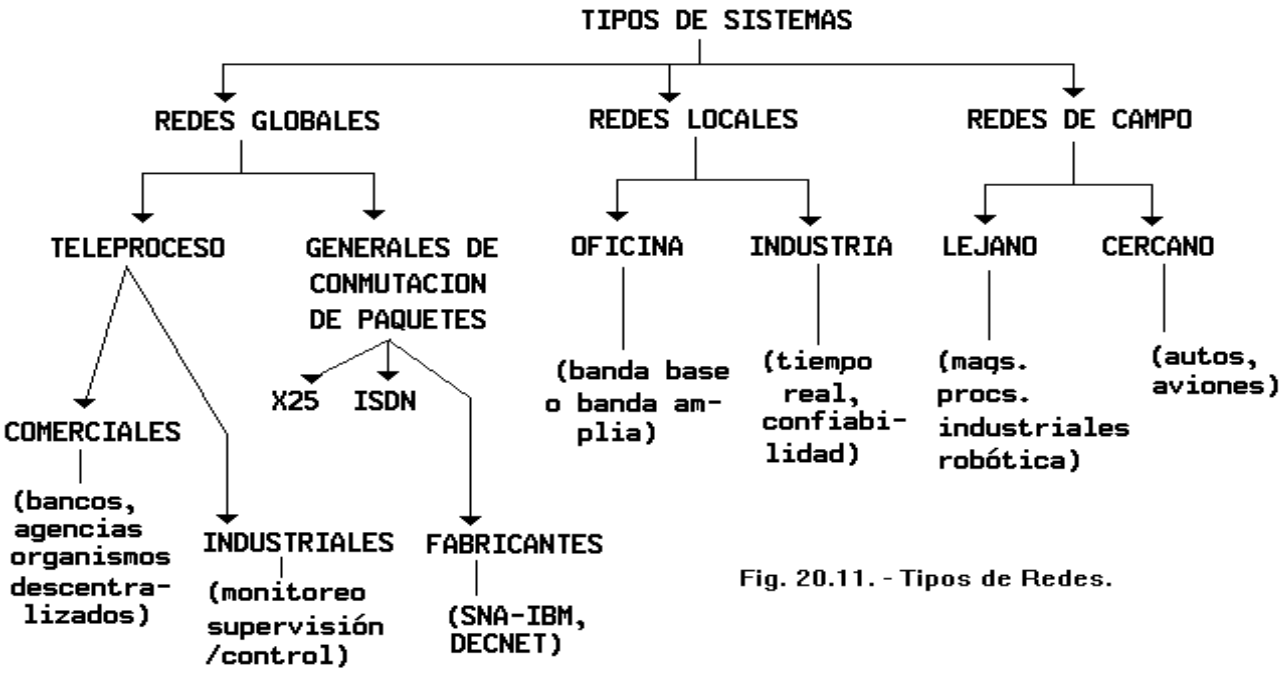

Fig. 20.11. - Tipos de Redes.