

MODULO: 10

PROCESAMIENTO DISTRIBUIDO

CONTENIDO:

- Conceptos sobre procesamiento distribuido en un Sistema de Cómputo
- Conceptos sobre Arquitecturas utilizadas en el procesamiento distribuido
- Conceptos sobre comunicaciones y protocolos
- Modelos de procesamiento distribuidos

OBJETIVO DEL MÓDULO: Describir conceptualmente los principios y los fundamentos del procesamiento distribuido, las comunicaciones y sus problemas en Sistemas de Cómputos.

OBJETIVOS DEL APRENDIZAJE: Después de estudiar este módulo, el alumno deberá estar en condiciones de:

- Explicar y reconocer las diferencias entre procesamiento centralizado y distribuido.
- Enumerar y explicar las ventajas y desventajas de los Sistemas Distribuidos.
- Comprender y explicar la organización y los modelos de diseños de comunicación utilizados en los Sistemas Distribuidos
- Comprender la comunicación utilizando los distintos protocolos.
- Comprender las distintas arquitecturas y el hardware que se utilizan en el procesamiento distribuido.
- Conocer y explicar la terminología específica empleada en éste módulo.

Metas:

En este Módulo presentamos una introducción a los sistemas distribuidos. Las bases del procesamiento multiprocesador y redes de computadores.

Luego brindaremos básicamente los conceptos de comunicaciones entre maquinas daremos la definición de *protocolos* y *arquitecturas de protocolos*, que son utilizados para permitir el intercambio de datos en sistemas distribuidos y los distintos modelos actualmente implementados.

Por último presentaremos dos anexos importantes para comprender la evolución del protocolo IP y su utilización y la consistencias de los caché utilizadas en el intercambio de datos entre procesos.

10.1. TIPOS DE PROCESAMIENTOS:

Existen básicamente cuatro modalidades en el procesamiento de los datos. Uno es el centralizado, otro es el paralelo, otro es el de Tiempo Real y el último es el distribuido como se observa en la figura 3.18 del módulo 3.

10.1.1. INTRODUCCIÓN A LOS SISTEMAS DISTRIBUIDOS

Gracias al desarrollo de poderosos microprocesadores y al avance de las redes de área local de alta velocidad (LAN), han surgido nuevos sistemas de cómputos compuestos por un conjunto de CPUs comunicados por redes. Estos sistemas se denominan sistemas distribuidos, en contraste con los sistemas centralizados que poseen una única CPU, periféricos y terminales.

10.1.2. DEFINICIÓN

Un sistema distribuido es un sistema de procesamiento descentralizado de datos que ejecutan sobre un conjunto de procesadores débilmente acoplados, interconectados por una red cuyos nodos están dispersos geográficamente.

10.1.3 OBJETIVO

Se busca un procesamiento rápido en paralelo y de este modo de incrementar la eficiencia para lograr un mayor rendimiento.

Técnicamente existen dos formas para incrementar la eficiencia del Hardware:

TECNOLOGICAMENTE: se puede mejorar las prestaciones mediante el uso de tecnología sofisticada (por ejemplo en vez de emplear silicio en la construcción de los Chips usar zafiro que tiene mejores propiedades), pero es antieconómico.

ARQUITECTONICAMENTE: la mayoría de los desarrollos de arquitecturas de computadoras trabajan sobre esto ya que es más aplicable y económico.

En un sistema débilmente acoplado los procesadores no comparten memoria ni reloj, en cambio cada uno tiene su propia memoria local. Los procesadores se comunican por medio de diversas líneas de comunicación, como canales de alta velocidad o líneas telefónicas.

Los procesadores en un sistema distribuido pueden variar en cuanto a tamaño y función e incluir pequeños microprocesadores, estaciones de trabajo, minicomputadoras y grandes sistemas de computación de propósito general.

10.1.4. VENTAJAS DE LOS SISTEMAS DISTRIBUIDOS

Las ventajas son Múltiples. Enumeramos solamente aquellas mas relevantes (basado en la arquitectura de multiprocesadores) propuestos por las tecnologías computacionales y el mercado:

- Costo & rendimiento: esto tal vez sea el motivo más importante.
- Mayor rendimiento y potencia de cálculo: es un objetivo muy interesante.
- Tolerante a fallas: si falla algo se tolera es decir siempre se esta en procesamiento.
- Flexibilidad y reconfiguración en cuanto a equipamiento nuevo, software nuevo, hardware nuevo, etc. (es muy adaptable).
- Crecimiento modular en cuanto a equipamiento: se puede ir comprando de a poco. A esto se lo llama ESCALABILIDAD.
- Especialización Funcional: soluciona los problemas específicos en cuanto a las áreas de la organización.

Existen las siguientes razones principales para construir sistemas distribuidos:

Economía

Estos sistemas tienen en potencia una proporción *precio/desempeño* mucho mejor que la de un único sistema centralizado. Se puede afirmar que un conjunto de microprocesadores proporciona un mejor índice de *precio/rendimiento* que un mainframe puede brindar.

Distribución Inherente

Existen muchas aplicaciones que son distribuidas en sí mismas, para lo cual necesitan de una plataforma que soporte este tipo de procesamiento. Por ejemplo, cuando se conectan todas las sucursales de un banco se tiene un sistema distribuido.

Confiabilidad

Si en un sistema distribuido hay un error en una instalación, las restantes pueden potencialmente continuar trabajando.

El sistema en sí mismo debe detectar una falla en una instalación y tomar las medidas necesarias para recuperarse. A partir de aquí las aplicaciones ya no utilizan los servicios de esa instalación migrándose a otro lugar para continuar su procesamiento; además si hay otra instalación que pueda hacerse cargo de las funciones correspondiente a la que presentó la falla, el sistema debe asegurar que la transferencia de las funciones se lleve a cabo correctamente. Por último, cuando la instalación que falló se recupera debe haber mecanismos para integrarla de nuevo en el sistema sin problemas. En este tipo de sistemas, la falla de un chip no afecta al sistema; a lo sumo se descompondrá una máquina y el resto seguirá funcionando correctamente. Este factor es muy importante tener en cuenta en aplicaciones críticas como ser: reactores nucleares, sistemas de aviación, pozos petroleros, etc..

Crecimiento por Incrementos modulares

En algunas oportunidades ocurre que una compañía compra un mainframe con la intención de procesar todo su trabajo en él. Si la compañía prospera y la carga de trabajo aumenta, el mainframe no será adecuado en cierto momento. Las únicas soluciones posibles son el reemplazo del mainframe con otro mas grande o añadir un segundo. Por el contrario, con un sistema distribuido, podrían añadirse simplemente más procesadores o nodos al sistema, lo que permite un desarrollo gradual conforme a las necesidades.

Compartir recursos (Procesadores, Datos y Periféricos)

Si varias instalaciones con capacidades diferentes están conectadas entre si, entonces los usuarios de una instalación puede usar los recursos disponibles en otra. En general, el compartir recursos proporciona mecanismos para utilizar archivos en instalaciones remotas, procesar información en una base de datos distribuida, imprimir archivos en instalaciones remotas y otras operaciones.

Otras veces los usuarios necesitan compartir ciertos datos. Por ejemplo, los empleados del sector sueldo y jornales necesitan tener acceso a la base de datos maestra de personal. Si se le diera a cada empleado una copia particular de toda la base de datos, no funcionaría bien, ya que se estaría duplicando los datos y la consistencia se perdería pronto. Los datos compartidos son absolutamente esenciales en este tipo de aplicaciones, de modo que se concluye a que las máquinas deben estar conectadas entre sí mediante algún tipo de conexión de red.

No solo los datos se pueden compartir, sino que los periféricos son otros candidatos. Muchas veces se comparten los periféricos caros como impresoras láser color, equipos de fotocomposición y dispositivos de almacenamiento masivos.

Flexibilidad

Los sistemas distribuidos permiten distribuir la carga de trabajo entre las computadoras de una manera más eficaz, y la perdida de alguna máquina se resuelve brindando a los usuarios otros equipos para que ejecuten sus trabajos. Además, si hay áreas que necesitan mas potencia de calculo o una especialización en su procesamiento (ejemplo comunicaciones), con este tipo de equipamiento es relativamente mas fácil ampliar o especializar para que las aplicaciones se ejecuten mas eficientemente.

Aceleración de cálculos

Si un calculo puede dividirse en varios subcálculos que pueden ejecutarse concurrentemente, entonces la disponibilidad de un sistema distribuido permite, justamente distribuir estos cálculos entre

varias instalaciones y ejecutarlos en forma concurrente. Además si una instalación esta sobrecargada de trabajos, algunos de ellos pueden moverse a otras instalaciones con menor carga.

Comunicación

Si varias instalaciones están conectadas por medio de una red, los usuarios de las distintas instalaciones tienen la oportunidad de intercambiar información. Tiene la posibilidad de realizar transferencia de archivos e intercambiar mensajes.

La ventaja es que estas funciones pueden efectuarse a grandes distancias en forma totalmente transparente para los usuarios.

10.1.5. DESVENTAJAS DE LOS SISTEMAS DISTRIBUIDOS

Software

La principal desventaja es el software. En la actualidad, no se tiene mucha experiencia en el diseño, implantación y uso del software distribuido. Los especialistas se plantean distintos interrogantes, como por ejemplo:

¿Qué tipo de sistemas operativos, lenguajes de programación y aplicaciones son adecuadas para estos sistemas? ¿Cuánto deben saber los usuarios sobre la distribución? ¿Qué proporción de trabajo debe realizar el sistema operativo y cuánto los usuarios?.

A medida que se realice más investigación sobre este tema, el problema del software disminuirá gradualmente, mientras tanto el software será la desventaja predominante.

Las redes

Otra desventaja son las redes de comunicación. Estas pueden perder mensajes, lo cual implica poseer un software especial para su manejo, pero puede verse sobrecargado.

También puede saturarse la red, lo cual implicaría reemplazarse o añadir otra. En ambos casos, hay que realizar conexiones de nuevos cables (incurriendo en nuevos costos), reemplazar las tarjetas de interfase de red, etc. Una vez que el sistema llega a depender de la red, la pérdida o saturación de ésta puede disminuir considerablemente las ventajas que el sistema distribuido debía lograr.

La seguridad

La ventaja de que los datos puedan ser compartidos fácilmente, puede aparejar serios problemas. Partamos de que si los usuarios pueden tener acceso a los datos en todo el sistema, entonces pueden tener acceso a datos con los que no tiene nada que ver.

Es fácil distinguir que la política que corre un gran riesgo es *la seguridad*, y es por ello que se considera como una gran desventaja.

10.2. Hardware para el procesamiento distribuido

El hardware para el Procesamiento Distribuido lo dividiremos en 2 grupos:

1. Redes computacionales
2. Arquitectura de procesadores

Daremos una ampliación del hardware para el procesamiento distribuido en la arquitectura Cliente servidor, en el cual nos referiremos a sus principales características.

Los avances tecnológicos en las redes de área local y la creación de microprocesadores de 32 y 64 bits lograron que computadoras mas o menos baratas tuvieran el suficiente poder en forma autónoma para desafiar en cierto grado a los mainframes, y a la vez se dio la posibilidad de intercomunicarlas, sugiriendo la oportunidad de partir procesos muy pesados en cálculo en unidades más pequeñas y distribuirlas en los varios microprocesadores para luego reunir los sub-resultados, creando así una máquina virtual en la red que exceda en poder a un mainframe. El sistema integrador de los microprocesadores que hacen ver a las varias memorias, procesadores, y todos los demás recursos como una sola entidad en forma transparente se le llama sistema operativo distribuido.

10.2.1 Redes computacionales

El procesamiento para que sea distribuido necesita interconectarse. Esta interconexión se realiza mediante redes de comunicaciones. Cada red tiene una arquitectura. Describiremos brevemente estas arquitecturas.

Concepto de Red.

Una red consiste en dos o más computadoras unidas que comparten recursos como archivos, CD-Roms o impresoras y que son capaces de intercomunicarse entre ellas mediante mensajes. Las redes están unidas por cable, líneas de teléfono, ondas de radio, satélite, etc.

Objetivos.

Su objetivo principal es lograr que todos sus programas datos y equipo estén disponible para cualquiera de la red que lo solicite, sin importar la localización física del recurso y del usuario.

Otro de sus objetivos consiste en proporcionar una alta fiabilidad, al contar con fuentes alternativas de suministro, es decir que todos los archivos podrían duplicarse en dos o tres máquinas, de tal manera que si una de ellas no se encuentra disponible, podría utilizarse una de las otras copias. Igualmente la presencia de varios CPU significa que si una de ellas deja de funcionar, las otras pueden ser capaces de encargarse de su trabajo, aunque su rendimiento en general sea menor.

El ahorro económico debido a que los computadores pequeños tiene una mejor relación costo / rendimiento, en comparación con la que ofrece las máquinas grandes.

Proporciona un poderoso medio de comunicación entre personas que se encuentran en lugares distantes entre sí.

Tipos de conexiones en red

Las instalaciones del sistema pueden conectarse físicamente de varias maneras y cada configuración tiene sus ventajas y desventajas. Analizaremos las configuraciones mas comunes según los siguientes criterios:

Costo básico: ¿Cuánto cuesta enlazar las distintas instalaciones del sistema?

Velocidad de comunicaciones: ¿ Cuánto se tarda en enviar un mensaje de la instalación A a la B?

Confiabilidad: Si falla una instalación o enlace del sistema, ¿es posible que aun sigan comunicándose las demás instalaciones?

Las distintas conexiones se representan como grafos cuyos nodos corresponden a las instalaciones. Una arista del nodo A al nodo B corresponde a una conexión directa entre las dos instalaciones.

Conexión total:

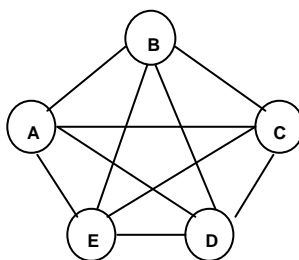


Fig. 10.01 Red de conexión total

En una red de conexión total (Fig. 10.01) cada instalación esta enlazada directamente con todas las demás instalaciones del sistema. El costo básico de esta instalación es muy elevado, ya que debe haber una línea de comunicación entre cada par de instalaciones. Pero, los mensajes entre instalaciones pueden enviarse con gran rapidez, dado que un mensaje solo requiere viajar por un enlace. Además estos sistemas son muy confiables ya que deben averiarse muchos enlaces para particionar el sistema.

Conexión parcial:

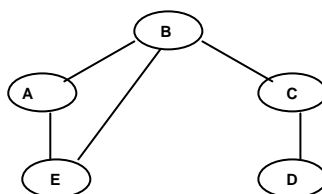


Fig 10.02 Red de conexión parcial

En una red conexión parcial (Fig. 10.02) hay un enlace directo entre algunos, pero no todos, los pares de instalaciones. Por lo tanto el costo básico de esta configuración es menor al de una red de conexión total. Pero es posible que un mensaje enviado de una instalación a otra tenga que pasar por varias instalaciones intermedias, lo que hace mas lenta la comunicación. Ejemplo E envía un mensaje a D.

Además, el sistema de conexión parcial no es tan confiable como un sistema de conexión total porque la falla de un enlace puede particionar la red. Para que esta situación se minimice, cada instalación se interconecta con, por lo menos otras dos.

Jerarquía: En una red de jerarquía (Fig. 10.03) las instalaciones se organizan como un árbol. Cada instalación, excepto la raíz, tiene un solo padre y varios hijos.

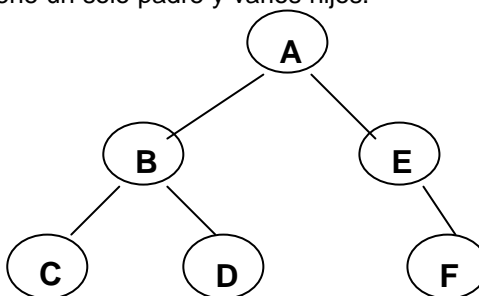


Fig 10.03 Red jerarquica con estructura de arbol.

El costo base de esta configuración es menor al esquema de conexión parcial. En este entorno, un padre y un hijo se comunican directamente y los hermanos solo pueden hacerlo a través de su padre común.

Si falla un instalación padre, sus hijos quedan incomunicados entre sí y con otros procesadores. Por lo general la falla de cualquier nodo particiona la red en varios subárboles disjuntos, excepto a nivel de una hoja.

Topología de redes computacionales:

Se conoce como topología de una red a la forma de interconectar los nodos de la red.

a) Estrella:

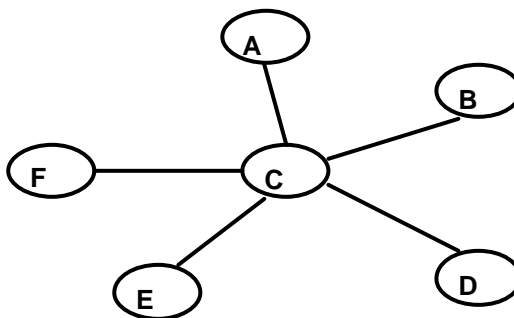


Figura 10.04 Red en estrella

En una red estrella (Figura 10.04) una de las instalaciones del sistema esta conectada con todas las demás, ninguna de las otras instalaciones se conecta con otra. El costo básico de esta configuración es lineal, dependiendo del numero de instalaciones. También es bajo el costo de comunicaciones ya que un mensaje de la instalación A a la B requiere a lo sumo dos transferencias. No obstante, este sencillo esquema no asegura la rapidez porque la instalación central se puede convertir en un cuello de botella. Por lo tanto, aunque el número de transferencias sea bajo, el tiempo necesario para transmitir el mensaje puede ser elevado.

Si falla la instalación central la red se particiona por completo desapareciendo la red.

Ventajas de la Topología Estrella:

- ✓ Gran facilidad de instalación
- ✓ Posibilidad de desconectar elementos de red sin causar problemas.
- ✓ Facilidad para la detección de fallo y su reparación.

Inconvenientes de la Topología de Estrella.

- ✓ Requiere más cable que la topología de BUS.
- ✓ Un fallo en el concentrador provoca el aislamiento de todos los nodos a él conectados.
- ✓ Se han de comprar hubs o concentradores.

b) Anillo: En una red en anillo (fig 10.05) cada instalación esta físicamente conectada a exactamente otras dos. El anillo puede ser unidireccional o bidireccional.

En una arquitectura unidireccional una instalación solo puede transmitir información a uno de sus vecinos y todas las instalaciones deben enviar la información en la misma dirección. En una arquitectura bidireccional una instalación puede transmitir información a ambos vecinos. El costo básico de un anillo es lineal con el número de instalaciones, pero el costo de comunicación puede ser elevado, ya que un mensaje de una instalación a otra viaja por el anillo hasta llegar a su destino. En un anillo bidireccional deben fallar dos enlaces para que se particione la red; en un anillo unidireccional una sola falla en una instalación particionará la red.

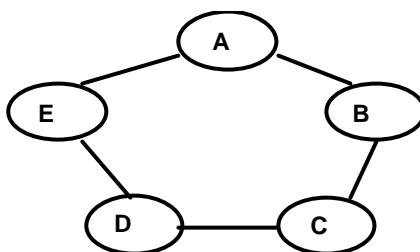
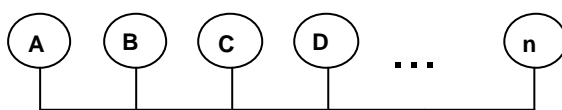


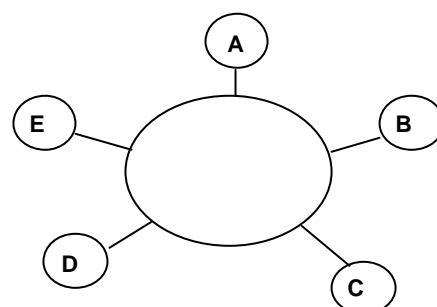
Fig.10.05 Red en anillo

c) Canal multiacceso o bus: En una red de canal multiacceso (Fig. 10.06) existe un solo enlace compartido (el canal). Todas las instalaciones del sistema se conectan directamente a ese enlace que se puede organizar como una línea recta o bus, o como un anillo. Las instalaciones pueden comunicarse directamente por medio de este canal. El costo básico de la red es lineal con el número de nodos y el costo de comunicaciones es bastante bajo a menos que el enlace se convierta en un cuello de botella. La falla de una instalación no afecta a la comunicación entre las demás, pero si falla el enlace la red queda completamente particionada.



a) Red en canal lineal

Fig.10.06 Canal multiacceso o bus



b) Red en canal anular o estrella

Ventajas de la topología de BUS.

- ✓ Es Más fácil conectar nuevos nodos a la red
- ✓ Requiere menos cable que una topología estrella.

Desventajas de la topología de BUS.

- ✓ Toda la red se caería se hubiera una ruptura en el cable principal.
- ✓ Se requiere terminadores.
- ✓ Es difícil detectar el origen de un problema cuando toda la red cae.
- ✓ No se debe utilizar como única solución en un gran edificio.

d) Topología de Árbol / Tree.

La topología de árbol combina características de la topología de estrella con la BUS. Consiste en un conjunto de subredes estrella conectadas a un BUS. Esta topología facilita el crecimiento de la red.

Ventajas de la Topología de Árbol.

- ✓ Cableado punto a punto para segmentos individuales.
- ✓ Soportado por multitud de vendedores de software y de hardware.

Desventajas de la Topología de Árbol.

- ✓ La medida de cada segmento viene determinada por el tipo de cable utilizado.
- ✓ Si se viene abajo el segmento principal todo el segmento se viene abajo con él.
- ✓ Es más difícil su configuración.

TIPOS DE REDES

Existen predominantemente cuatro tipos de redes: las redes de escritorio (Desk Area Network – DAN), *redes locales* (Local Area Network – LAN), *Las redes metropolitanas* (Metropolitan Area Network – MAN) y las *redes de área distante* (Wide Area Network – WAN). La principal diferencia entre ellas es su distribución geográfica: las redes locales están compuestas por procesadores distribuidos en un área geográfica pequeña como ser un edificio; las metropolitanas en una ciudad y las redes de área distante están formadas por varios procesadores autónomos distribuidos en áreas geográficas lejanas. Las principales son las LAN y las WAN, mientras las DAN están en una etapa de investigación.

Redes locales: Las redes locales (LAN, local área network) surgieron como sustitutos de los grandes sistemas de computadora central. En ese momento, fue evidente que era más económico tener varias computadoras pequeñas, cada una con sus propias aplicaciones, que un solo sistema de gran tamaño. Como cada computadora pequeña necesita un complemento de dispositivos periféricos y como existen ambientes donde se comparten los datos, la conexión de estos pequeños sistemas mediante una red fue un paso lógico.

Los enlaces de comunicación son más rápidos y tienen menor tasa de error que las redes de área distante. Los enlaces más comunes para interconectar estos equipos son: el par trenzado, el cable coaxial de banda base, el cable coaxial de banda ancha y la fibra óptica; las configuraciones más habituales son las redes de canal multiacceso, anillo y estrella.

Una red local típica puede consistir en varios minicomputadores diferentes, dispositivos periféricos compartidos, estaciones de trabajo y uno o más procesadores especializados (compuertas) que proporcionan acceso a otras redes. (Fig. 10.07). Para construir las redes locales es común usar un esquema ethernet donde el medio de comunicación es un cable coaxial multiacceso. Los mensajes entre los nodos de la red se envían en paquetes. Como no existe un controlador central es fácil agregar nuevas instalaciones a la red.

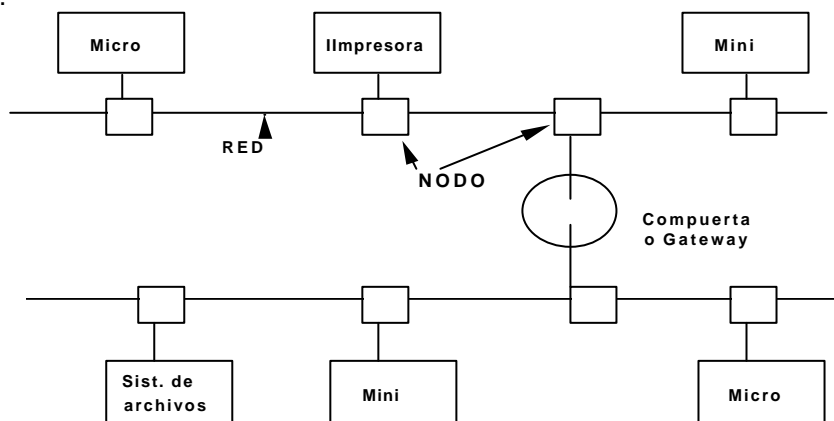


Fig. 10.07 Red Local (LAN).

Un protocolo es un conjunto de normas que rigen la comunicación entre las computadoras de una red. Estas normas especifican el tipo de cables que se utilizará, que topología tendrá la red, que velocidad tendrán las comunicaciones y de qué forma se accederá al canal de transmisión.

Los estándares más populares son:

- a) Ethernet
- b) LocalTalk
- c) Token Ring

a) Ethernet.

Ethernet es hoy en día el standard para las redes de área local. Ethernet se define como un modo de acceso múltiple y de detección de colisiones, es el conocido carrier sense multiple access/collision detection (CSMA/CD). Cuando una estación quiere acceder a la red escucha si hay alguna transmisión en curso y si no es así transmite. Es el caso de que dos redes detecten probabilidad de emitir y emitan al mismo tiempo, se producirá una colisión por esto queda resuelto con los sensores de colisión que detectan esto y fuerzan una retransmisión de la información.

Velocidades de Transmisión:

Tipo de Ethernet	Velocidad (Mbps)	DISTANCIA MEDIA
10 base 5	10	500 m
Coaxial Grueso 10 Base 8	10	185
Coaxial Fino 10 Base T	10	100
UTP 10 Base F	10	2000
Fibra Óptica.		

b) Local Talk.

El protocolo LocalTalk fue desarrollado por Apple Computer, Inc. Para computadores Macintosh. El método de acceso al medio es el SCMA/CA. (Carrier Sense Multiple Access with Collision Avoidance) Este método se diferencia en que el computador anuncia su transmisión antes de realizarla. Mediante el uso de adaptadores LocalTalk y cables UTP especiales se puede crear una red de computadores a través del puerto serie. El sistema operativo de estos establece relaciones punto a punto sin necesidad de software adicional aunque se puede crear una red cliente servidor con el software AppleShare.

Con el protocolo LocalTalk se pueden utilizar topologías bus, estrella o árbol usando cable UTP pero la velocidad de transmisión es muy inferior a la de Ethernet.

c) Token Ring.

El protocolo Token Ring fué desarrollado por IBM a mediados de los 80. El modo de acceso al medio está basado en el traspaso del testigo o token passing. En una red Token Ring los computadores se conectan formando un anillo. Un testigo o token electrónico para de un computador a otro.

Cuando se recibe este testigo se está en disposición de emitir datos. Estos viajan por el anillo hasta llegar a la estación receptora. Las redes Token Ring se montan sobre tipologías estrella cableada o star-wired con par trenzado o fibra óptica. Se puede transmitir información a 4 o 16 Mbs. Esta tecnología está siendo desplazada actualmente por el auge de Ethernet.

Resumen protocolos:

Protocolo /Topología	Cable	Velocidad
Ethernet	Par Trenzado/Coaxial/ Fibra Optica	10 Mbps
Bus Star Tree / Fast Ethernet	Par Trenzado /Fibra Optica	23 Mbps
Bus Star Local Talk	Par Trenzado	23 Mbps
Bus Star Token Ring	Par trenzado	4 Mbps - 16 Mbps
Star Wired Ring		

Redes de área distante (WAN):

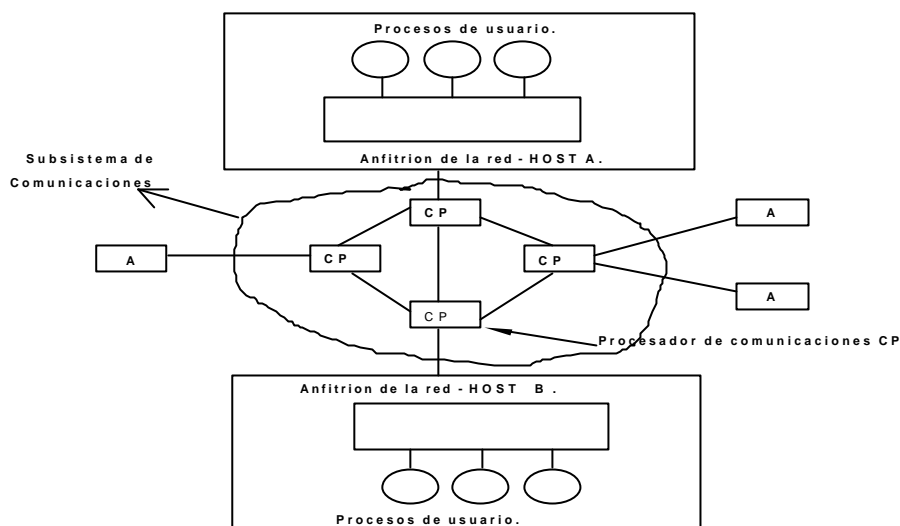


Fig. 10.08. Procesadores de comunicación en un red de área distante (WAN).

Las redes de área distante (WAN, wide area network) están distribuidas en extensas área geográficas por los que las enlaces de comunicaciones son bastante lentos y poco confiables. Los enlaces típicos son líneas telefónicas, enlaces por microondas y canales de satélite, y se controlan mediante procesadores de comunicación (CP) especiales (Fig. 10.08), responsables de definir la interfase por medio de la cual las instalaciones se comunican en la red, así como de transferir informaciones entre instalaciones.

Como ejemplo específico podemos considerar la red WAN Internet. El sistema permite que se comunique entre sí instalaciones geográficamente separadas, llamadas anfitriones (**Host**). Las computadoras anfitriones normalmente difieren en cuanto a tipo, velocidad, sistema operativo y generalmente se encuentran en redes locales, las cuales están conectadas a Internet por redes regionales. Las redes regionales están enlazadas por encaminadores (**Routers**) para formar la red mundial. Los routers conectan las redes en el nivel de red del modelo ISO, y controlan la ruta que sigue cada paquete por la red. Este ruteo puede ser dinámico, para aumentar la eficiencia de las comunicaciones, o estático para reducir los riesgos de seguridad. La mayoría de las anfitriones son direccionados lógicamente con un nombre compuesto por varias partes. Los nombre avanzan de la parte más específica a la más general de la dirección.

Las redes de área distantes generalmente son más lentas que las locales.

Red de Área Metropolitana / MAN (Metropolitan Area Network).

Las redes de área metropolitana cubren extensiones mayores como pueden ser una ciudad o un distrito. Mediante la interconexión de redes LAN se distribuyen la información a los diferentes puntos del distrito. Bibliotecas, universidades u organismos oficiales suelen interconectarse mediante este tipo de redes.

Red DAN (Desk Area Network)

Son computadoras que tienen una Red interna a la cual se conectan los procesadores, las memorias y las interfases de los periféricos, mediante una fibra óptica sin un motherboard.

Este concepto aún no se produce comercialmente y es un área de investigación.

Cableado de la red.

El Cable es el medio a través del cual fluye la información a través de la red. Hay distintos tipos de cable de uso común en redes LAN. Una red puede utilizar uno o más tipos de cable, aunque el tipo de cable utilizado siempre estará sujeto a la topología de la red, el tipo de red que utiliza y el tamaño de esta. Estos son los tipos de cable más utilizados en redes LAN:

- Cable de par trenzado sin apantallar /UTP Unshielded twisted pair.
- Cable de par trenzado apantallado / STP Shieles twisted.
- Cable Coaxial.
- Cable de fibra óptica.
- LAN sin cableado.

a) Cable de par trenzado sin apantallar / Unshielded Twisted Pair (UTP).

Este tipo de cable es el más utilizado. Tiene una variante con apantallamiento pero la variante sin apantallamiento suele ser la mejor opción para una PYME.

La calidad del cable y consecuentemente la cantidad de datos que es capaz de transmitir varían en función de la categoría del cable. Las graduaciones van desde el cable de teléfono, que solo transmite la voz humana a el cable de categoría 5 capaz de transferir 100 Mega bites por segundo.

Categorías UTP

TIPO	USO
Categoría 1	Voz (cable de teléfono).
Categoría 2	Datos a 4 Mbps (Local Talk).
Categoría 3	Datos a 10 Mbps (Ethernet).
Categoría 4	Datos a 20 Mbps/ 16 Mbps Token Ring).
Categoría 5	Datos a 100 mbps (Fast Ethernet).

La diferencia entre las distintas categorías es la tirantez. A mayor tirantez mayor capacidad de transmisión de datos. Se recomienda el uso de cables de categoría 3 a 5 para la implementación de redes en PYMES o sea pequeñas y medianas empresas.

Es conveniente sin embargo utilizar cables de categoría 5 ya que estos permitirán migraciones de tecnologías 10Mb a tecnología 100 Mb.

Conector UTP

El estándar para conectores de cable UTP es el RJ-45. Se trata de un conector de plástico similar al conector del cable telefónico. La sigla RJ se refiere al Estándar Registered Jack, creado por la industria telefónica. Este estándar se encarga de definir la colocación de los cables en su pin correspondiente.

b) Cable de par trenzado pantallar / Shielded Twisted Pair (STP).

Una de las desventajas del cable UTP es que es susceptible a las interferencias eléctricas. Para entornos con este tipo de problemas existe un tipo de cable UTP que lleva apantallamiento, esto significa protección contra interferencias eléctricas. Este tipo de cable es usado por lo general en redes de topología Token Ring.

c) Cable Coaxial.

El cable coaxial contiene un conductor de cobre en su interior. Este va envuelto en un aislante para separarlo de un apantallado metálico con forma de rejilla que aísla el cable de posibles interferencias externas.

Aunque la instalación de cable coaxial es más complicada que la del UTP, este tiene un alto grado de resistencia a las interferencias, también es posible conectar distancias mayores que con los cables de par trenzado.

Tipos de Cable Coaxial:

- ✓ Cable coaxial fino o thin coaxial.
- ✓ Cable coaxial grueso o thick coaxial.

Es posible escuchar referencias sobre el cable coaxial fino como thinnet o 10Base2. Estos hacen referencia a una red de tipo Ethernet con un cable coaxial fino. Donde el 2 significa que el mayor segmento es de 200 metros, siendo en la práctica reducido a 185 m. El cable coaxial es muy popular en las redes con topología BUS.

También se referencia el Cable Coaxial grueso como thicknet o 10Base5. Este hace referencia a una red de tipo Ethernet con un cableado coaxial grueso, donde el 5 significa que el mayor segmento posible es de 500 metros. El cable coaxial es muy popular en las redes con topología de BUS. El cable coaxial grueso tiene una capa plástica adicional que protege de la humedad al conductor de cobre. Esto hace este tipo de cable una gran opción para redes de BUS extensas, aunque hay que tener en cuenta que este cable es difícil de doblar.

Conector para Cable Coaxial:

El más usado es el conector BNC, cuyas siglas son Bayonet-Neill-Concelman. Los conectores BNC pueden ser de tres tipos:

d) Cable de Fibra Óptica.

El cable de fibra óptica consiste en un centro de cristal rodeado de varias capas de material protector. Lo que se transmite no son señales eléctricas sino luz con lo que se elimina la problemática de las interferencias. Esto lo hace ideal para entornos en los que haya gran cantidad de interferencias eléctricas. También se utiliza mucho en la conexión de redes entre edificios debido a su inmunidad a la humedad y a la exposición solar.

Con un cable de fibra óptica se pueden transmitir señales a distancias mucho mayores que con cables coaxiales o de par trenzado. Además la cantidad de información capaz de transmitir es mayor por lo que es ideal para redes a través de las cuales se desee llevar a cabo videoconferencias o servicios interactivos. El costo es similar al cable coaxial o al cable UTP pero las dificultades de instalación y modificación son mayores. En algunas ocasiones escucharemos 10BaseF como referencia a este tipo de cableado; estas siglas hablan de una red Ethernet con cableado de fibra óptica.

Las principales características son:

- ✓ El aislante exterior está echo de teflón o PVC.
- ✓ Fibras Kevlar ayudan a dar fuerza al cable y hace más difícil su ruptura.
- ✓ Se utiliza un recubrimiento de plástico para albergar a la fibra central.
- ✓ El centro del cable está echo de cristal o de fibras plásticas.

e) LAN sin cableado.

No todas las redes se implementan sobre un cableado, algunas utilizan señales de radio de alta frecuencia o haces infrarrojos para comunicarse. Cada punto de la red posee una antena desde la que emite y recibe. Para largas distancias se pueden utilizar teléfonos móviles o satélites.

Este tipo de conexión está especialmente indicada para su uso con portátiles o para edificios viejos en los que es imposible instalar un cableado.

Las desventajas de este tipo de redes son su alto costo, su susceptibilidad a las interferencias electromagnéticas y la baja seguridad que ofrecen. Además son más lentas que las redes que utilizan cableado.

Clasificación de las Redes según su uso.

Las redes según sea la utilización por parte de los usuarios puede ser:

Redes dedicadas o exclusivas: Son aquellas que por motivo de seguridad, velocidad o ausencia de otro tipo de red, conectan dos o más puntos de forma exclusiva. Este tipo de red puede estructurarse en redes punto a punto o redes multipunto.

Redes punto a punto: Permiten la conexión en línea directa entre terminales y computadoras. La ventaja de este tipo de conexión se encuentra en la alta velocidad de transmisión y la seguridad que presenta al no existir conexión con otros usuarios. Su desventaja sería el precio muy elevado de este tipo de red.

Redes multipunto: Permite la unión de varios terminales a su correspondiente computadora compartiendo una única línea de transmisión. La ventaja consiste en el abaratamiento de su costo, aunque pierde velocidad y seguridad. Este tipo de redes requiere amplificadores y difusores de señal o de multiplexores que permiten compartir líneas dedicadas.

Redes compartidas: Son aquellas a las que se une un gran número de usuarios, compartiendo todas las necesidades de transmisión e incluso con transmisiones de otras naturalezas. Las redes más usuales son las de conmutación de paquetes y las de conmutación de circuitos.

Redes de conmutación de paquetes: Son redes en las que existen nodos de concentración con procesadores que regulan el tráfico de paquetes. **Paquete:** es una pequeña parte de la información que cada usuario desea transmitir. Cada paquete se compone de la información, el identificador del destino y algunos caracteres de control.

Redes de conmutación de circuitos: Son redes en las que los centros de conmutación establecen un circuito dedicado entre dos estaciones que se comunican.

Redes digitales de servicios integrados (RDSI): Se basan en desarrollos tecnológicos de conmutación y transmisión digital. La RDSI es una red totalmente digital de uso general capaz de integrar una gran gama de servicios como son la voz, datos, imagen y texto. La RDSI requiere de la instalación de centrales digitales.

Clasificación de las Redes según los servicios que prestan:

Las redes según los servicios que satisfacen a los usuarios se clasifican en:

Redes para servicios básicos de transmisión: Se caracterizan por dar servicio sin alterar la información que transmiten. De este tipo son las redes dedicadas, la red telefónica y las redes de conmutación de circuitos.

Redes para servicios de valor añadido: Son aquellas que además de realizar la transmisión de información, actúan sobre ella de algún modo. Pertenecen a este tipo de red, las redes que gestionan mensajería, transferencia electrónica de fondos, acceso a grandes bases de datos, videotex, teletex, etc.

Clasificación de las Redes según la relación empresaria:

Las redes según el servicio que se realice en torno a la empresa puede subdividirse en:

Redes intraempresa: Son aquellas en las que el servicio de interconexión de equipos se realiza en el ámbito de la empresa.

Redes interempresa: Son las que proporcionan un servicio de interconexión de equipos entre dos o más empresas.

Clasificación de las Redes según la propiedad o pertenencia:

Las redes según la propiedad a la que pertenezcan pueden ser:

Redes privadas: Son redes gestionada por personas particulares, empresas u organizaciones de índole privado. A ellas sólo tienen acceso los terminales de los propietarios.

Redes públicas: Son las que pertenecen a organismo estatales, y se encuentran abiertas a cualquier usuario que lo solicite mediante el correspondiente contrato.

Ej: Redes telegráficas, redes telefónicas, redes especiales para transmisión de datos.

Componentes de una Red

Las redes de computadores se montan con una serie de componentes de uso común y que es mayor o menor medida aparece siempre en cualquier instalación.

Servidores: Los servidores de archivos conforman el corazón de la mayoría de las redes. Se trata de computadores con mucha memoria RAM, uno o varios discos duros enormes y una rápida tarjeta de red. El sistema operativo de red se ejecuta sobre estos servidores así como las aplicaciones compartidas.

Un servidor de impresión se encargará de controlar el tráfico de red ya que este es el que accede a las demandas de las estaciones de trabajo y el que les proporcione los servicios que pidan las impresoras, archivos, Internet, etc. Es preciso contar con un computador con capacidad de guardar información de forma muy rápida y de compartirla con la misma rapidez.

Estaciones de Trabajo: Son los computadores mas poderosos que una PC conectados al servidor. Las estaciones de trabajo no han de ser tan potentes como el servidor, simplemente necesita una tarjeta de red, el cableado pertinente y el software necesario para comunicarse con el servidor. Una estación de trabajo puede carecer de disquetera y de disco duro y trabajar directamente sobre el servidor. Prácticamente cualquier computador puede actuar como estación de trabajo.

Tarjeta de Red: La tarjeta de red o NIC (Network Interface Card) es la que conecta físicamente el computador a la red. Las tarjetas de red más populares son por supuesto las tarjetas Ethernet, existen también conectores Local Talk así como tarjetas TokenRing.

Tarjeta Ethernet con conectores RJ-45: Los conectores LocalTalk se utilizan para computadores Mac, conectándose al puerto paralelo. En comparación con Ethernet la velocidad es muy baja, de 230KB frente a los 10 o 100 MB de la primera.

Las tarjetas de Token Ring, son similares a las tarjetas Ethernet aunque el conector es diferente, por lo general es un DIM de nueve pines.

Concentradores o Hubs: Un concentrador o Hub es un elemento que provee una conexión central para todos los cables de la red. Los hubs son cajas con un número determinado de conectores, habitualmente RJ.45 más otro conector adicional de tipo diferente para enlazar con otro tipo de red.

Los hay de tipo inteligente que envían la información solo a quien ha de llegar mientras que los normales envían la información a todos los puntos de la red siendo las estaciones de trabajo las que decidan si se quedan o no con esa información.

Están provistos de salidas especiales para conectar otro Hub a uno de los conectores permitiendo así ampliaciones de la red.

Repetidores: Cuando una señal viaja a lo largo de un cable va perdiendo fuerza a medida que avanza. Esta pérdida de fuerza puede causar pérdida de información. Los repetidores amplifican la señal que reciben permitiendo así que la distancia entre dos puntos de la red sea mayor que la que un cable solo permite.

Puentes o Bridges: Los Bridges se utilizan para segmentar redes grandes en redes más pequeñas, destinado a otra red pequeña diferente mientras que todo el tráfico interno seguirá en la misma red. Con esto se logra reducir el tráfico de la red.

10.2.2. Arquitectura de Procesadores para Sistemas Distribuidos

Clasificación de los Sistemas de Cómputos

Con el correr del tiempo, se han propuesto diversos esquemas de clasificación para los sistemas compuestos de varios CPU. Ninguno de ellos ha sido exitoso ni tampoco se han adoptado. La clasificación más acertada es la enunciada por Flynn (1972), aunque es un poco precaria. Flynn eligió dos parámetros para su topología: el número de *flujo de instrucciones* y el número de *flujo de datos*.

SISD (Single Instruction stream Single Data stream)	Computadora con un solo flujo de instrucciones y un flujo de datos. Son aquellos computadores compuestos por una sola CPU, desde computadoras personales hasta mainframes.
SIMD (Single Instruction stream Multiple Data stream)	Computadora con un flujo de instrucciones y varios flujos de datos. La idea es ordenar los procesadores con una unidad de instrucción que busca cada instrucción, y que alimenta a varias unidades de datos para que la lleven a cabo en paralelo.
MISD (Multiple Instruction stream Single Data stream)	Computadoras con un flujo de varias instrucciones y un solo flujo de datos. Hasta el momento, las computadoras conocidas que se basan en este modelo son las pipelines.
MIMD (Multiple Instruction stream Multiple Data stream)	Conjunto de computadoras independientes, cada una con su propio Program Counter, Datos y Programas.

Tabla 10.01 La clasificación de arquitecturas dada por Flynn

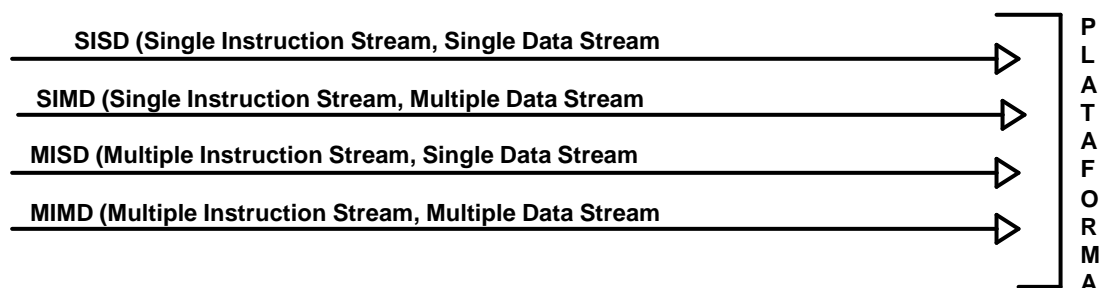


Figura 10.09. La clasificación de arquitecturas dada por Flynn

Luego de distinguir las distintas categorías de sistemas de cómputos, deberíamos introducir a los sistemas distribuidos en alguno de estos conjuntos. No resulta difícil ver que los sistemas distribuidos son todos MIMD pero antes trataremos el tema de las plataformas que ofrecen los distintos proveedores y su interoperabilidad.

Hay un conjunto de plataformas provistas por distintos proveedores para resolver las arquitecturas. Cada plataforma es "propietaria" de su fabricante.

El problema es la heterogeneidad y la interoperabilidad de estas plataformas.

Plataformas para la interoperatividad

Las arquitecturas se puede concebir como una máquina que tenga:

- Multiprogramación sobre una sola CPU.
- Multiprocesadores con memoria compartida (muy usual).
- Computadores secuenciales multiprogramados en RED.

Al primero se lo llama Sistema Centralizado y a la segunda Sistema de Procesamiento paralelo y al restante Sistemas Distribuidos.

Para representar gráficamente cada una de estas plataformas conviene detallar primero el espacio de direccionamiento privado de cada proceso que está compuesto por el programa mas sus datos y el contexto de ejecución como se indica el la Figura 10.10..

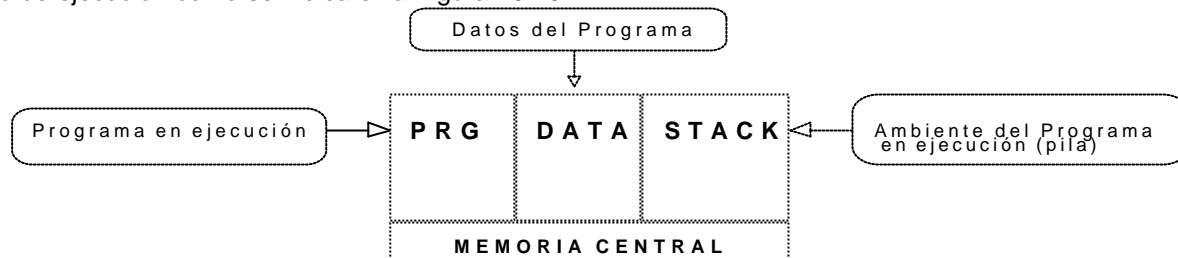


Fig. 10.10 Funcionamiento del espacio de direccionamiento de un proceso.

A continuación describiremos las arquitecturas en función de cómo se ejecutan los procesos:

A) Arquitectura multiprogramada o Sistema Centralizado:

Multiprogramación supone múltiples procesos en un ambiente de una sola máquina.

Hay abundante experiencia acumulada sobre procesos asincrónicos. por ejemplo en Administración, Cooperación, Sincronización, IPC, etc.

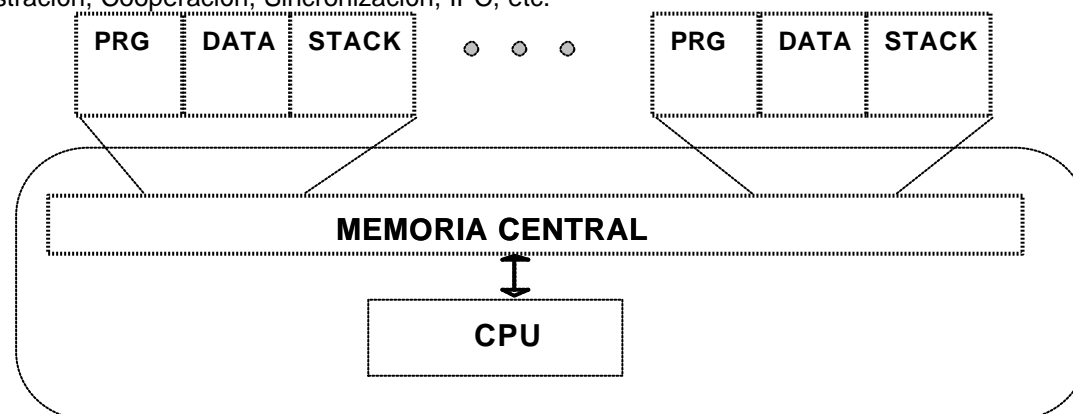


Fig. 10.11 Arquitectura multiprogramada o Sistema Centralizado

VENTAJAS:

- Muy simple
- Muy conocido
- Fácil de operar

DESVENTAJAS:

- Secuencialidad (una sola CPU), no se cumplen las ventajas del procesamiento distribuido.
- Los procesos utilizan los servicios provistos por un Sistema Operativo, entonces la ejecución del Sistema Operativo es overhead del procesamiento.
- Solo válido para ese ambiente computacional.

B) sistemas de Procesamiento paralelo:

Generalmente se usan como servidores (Servers) en redes. Se implementan protocolos en distintas capas (layers). Básicamente se agrupan en 2 esquemas:

1) Arquitectura basada en multiprocesadores homogéneos con memoria compartida

La memoria compartida es una mejora al sistema anterior, es a lo que se tiende actualmente desde el punto de vista funcional. Su mayor problema consiste en que el sistema tiene que manejar varias CPUs ejecutando sobre la misma memoria.

Si los procesadores son iguales se dice que son homogéneos y los procesos se administran con una sola cola de listos para ejecutar.

Si los procesadores son distintos, el sistema se llama heterogéneo, y se necesitan varias colas de listos para ejecutar (una por cada tipo de procesador).

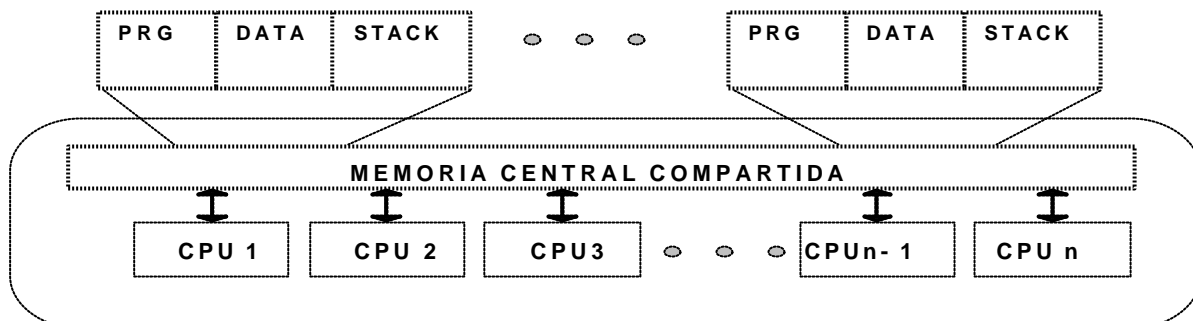


Fig. 10.12. Arquitectura basada en multiprocesadores con memoria compartida

VENTAJAS:

- Soporta interoperabilidad pues es mas flexible.
- Es más rápido (procesan más CPUs).

DESVENTAJAS:

- Problemas de sincronización en concurrencias.
- Depende del S.O. (Scheduling y Servicios).
- La planificación es mas compleja.

2) Arquitectura basada en multiprocesadores heterogéneos con memoria compartida

En este caso, los procesadores no son iguales por eso se dice que son heterogéneos y los procesos se administran con tantas colas de listos para ejecutar como tipo de procesadores haya en el sistema.

Las ventajas y desventajas son las mismas que en el caso de homogéneo.

C) Arquitectura con computadores secuenciales multiprogramados en red:

La red es un vínculo físico de comunicación. Puede ser cables, micro-ondas, fibras ópticas, etc.

VENTAJAS:

- Independencia de máquinas
- Se cumplen los objetivos del procesamiento distribuido.
- Un solo S.O. para los sistemas homogéneos.
- Un solo Software de Red.

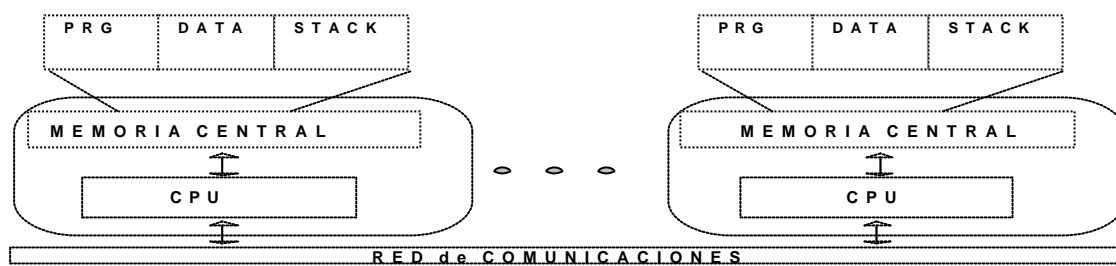
DESVENTAJAS:

- Aumento de complejidad
- Es más difícil compartir la información que en sistemas Multiprogramados o multiprocesadores
- Solo se puede compartir información mediante mensajes (que están en memoria común - Buffers).
- Problemas con protocolos.

Se pueden tener máquinas diferentes y redes distintas, y esto hace mas difícil la interoperatividad.

La compatibilidad se logra mediante software llamado middleware¹. Para esto se necesita homogeneizar los protocolos.

¹ Middleware se verá en el módulo 11.



10.13 Arquitectura con computadores secuenciales multiprogramados en red

CLASIFICACIÓN DE SISTEMAS MIMD

Aunque todos los sistemas distribuidos constan de varias CPUs, existen diversas formas de organizar el hardware, en particular en la forma de interconectarlas y comunicarse entre sí.

Los sistemas distribuidos entran dentro de la categoría MIMD definida por Flynn. MIMD significa Múltiples flujos de Instrucción y Múltiples flujos de Data. Este modelo se refiere a un grupo de computadoras independientes cada una con su propio contador de programa, programa y datos.

Ya que concluimos a que todos los sistemas distribuidos caen en este conjunto, abordaremos el tema con mayor profundidad, dividiendo a éste en dos grandes grupos: los multiprocesadores que poseen memoria compartida (o sea que existe un solo espacio de direcciones virtuales compartido por todas las CPUs), y las multicomputadoras que no comparten memoria (cada máquina posee su propia memoria particular como por ejemplo un conjunto de PC conectadas mediante una red.).

Los multiprocesadores o multicomputadoras a su vez pueden subdividirse según la arquitectura de red de conexión. Este enlace puede ser mediante bus o conmutador.

La conexión mediante un bus se realiza por medio de una única red, bus o cable u otro medio que conecta todas las máquinas. Este esquema se ejemplifica en la Fig. 10.17.

La conexión mediante un conmutador no tienen una sola columna vertebral, sino que su conexionado se realiza por medio de cables individuales entre cada máquina. Los mensajes se mueven a través de los cables y se hace una decisión explícita de conmutación en cada etapa, para dirigir el mensaje por el cable correcto de salida.

THIGHTLEY COUPLED:

Otra dimensión de nuestra taxonomía es que en ciertos sistemas las máquinas están *fuertemente acopladas* y en otras están *débilmente acopladas*. En un sistema fuertemente acoplado, el retraso que se experimenta al enviar un mensaje de una computadora a otra es pequeño y la tasa de transmisión de los datos es alta. En un sistema débilmente acoplado ocurre lo contrario. El retraso de los mensajes entre las máquinas es grande y las tasa de transmisión de datos es baja.

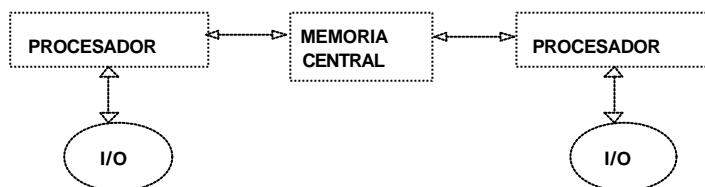


Fig. 10.14 Procesadores fuertemente acoplados

Los sistemas fuertemente acoplados tienden a utilizarse mas como sistemas paralelos y los débilmente acoplados como sistemas distribuidos.

LOWLEY COUPLED

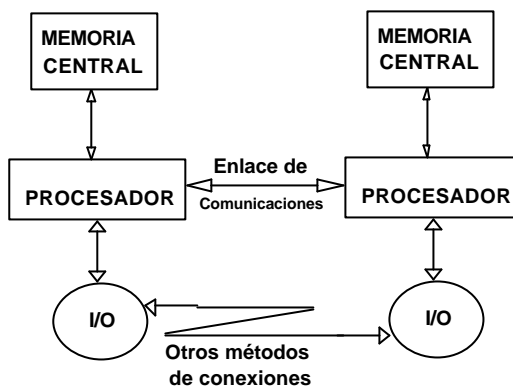


Fig. 10.15 Multiprocesamiento debilmente acoplado

En general, los multiprocesadores tienden a estar mas fuertemente acoplados que las multicomputadoras porque pueden intercambiar datos a la velocidad de sus memorias.

Clasificación de los sistemas según el método de acceso a la memoria:

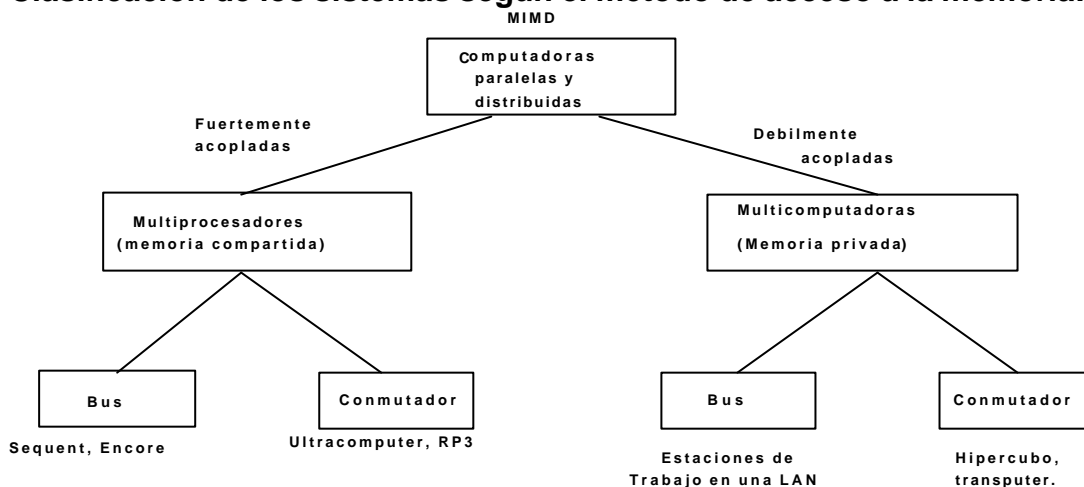


Fig. 10.16. Division arquitectonica de los sistemas paralelos y distribuidos.

UMA (uniform memory access): Sistema en que todos los procesadores pueden acceder a toda la memoria disponible con la misma velocidad (de forma uniforme). Ejemplo: BUS compartido.

NUMA (non uniform memory access): Sistema en que hay diferencias de tiempo en el acceso a diferentes áreas de Memoria. El procesador accede de una forma no uniforme. Depende de la proximidad del procesador y de la complejidad del mecanismo de conmutación.

NORMA (non remote memory access): no existen memorias compartidas.

Multiprocesadores en base a Bus

Los multiprocesadores con base a bus, están compuestos de una cantidad finita de CPUs, conectados a un bus común mediante una interfase, conectado a una Memoria Central. El bus esta compuesto de líneas de direcciones, líneas de datos y líneas de control. Un bus típico tiene 32 o 48 o 64 líneas de direcciones, 32 o 64 líneas de datos y de 20 a 30 líneas de control, todo lo cual opera en paralelo.

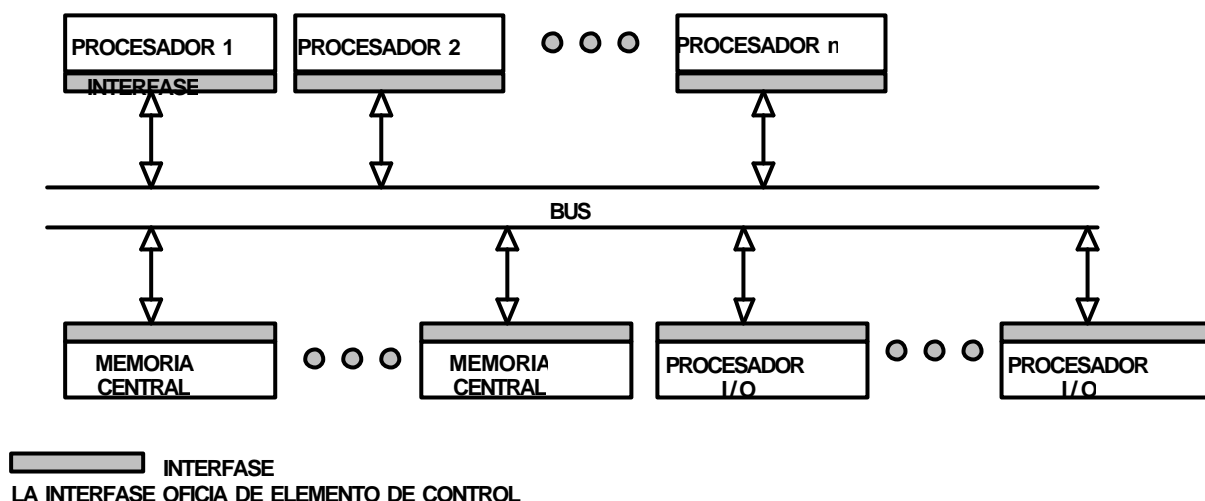


Fig. 10.17 Procesadores fuertemente acoplados

La lectura de una palabra de memoria se lleva a cabo mediante la siguiente dinámica: la CPU solicitante coloca la dirección de la palabra en la línea de direcciones y una señal determinada en las líneas de control, reflejando una operación de lectura. La memoria responde en base a la señal y coloca el valor de la palabra en las líneas de datos para permitir la lectura de ésta por parte de la CPU.

En otras palabras, dado que solo existe una memoria, si la CPU A escribe una palabra en la memoria y después la CPU B lee esa palabra unos microsegundos después, B obtendrá el valor recién escrito por A. Una memoria con esta propiedad es *coherente*. La coherencia juega un papel muy importante en los sistemas operativos distribuidos².

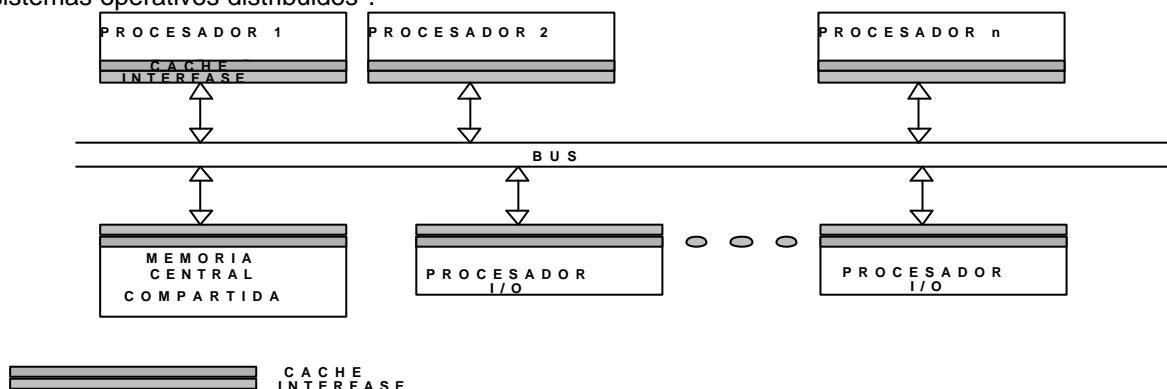


Fig. 10.18 Procesadores conectados mediante caché.

El problema con este esquema es la sobrecarga del bus y la disminución del rendimiento respectivo del mismo. Por ejemplo con solo disponer de 4 o 5 procesadores, el bus estará por lo general sobre cargado y el rendimiento disminuirá en forma drástica. La solución es agregar una memoria *caché* de alta velocidad entre cada CPU y el bus, como se muestra en la figura 10.18. El caché guarda las palabras de acceso reciente. Todas las solicitudes de la memoria se encuentran en la caché, ésta responde a la CPU y no se hace solicitud alguna al bus. Si la caché es bastante grande la posibilidad de éxito será alta y el tráfico en el bus por cada CPU disminuirá en forma drástica, lo que permitirá un mayor número de CPUs en el sistema.

Sin embargo, el uso de cachés también acarrea un serio problema. Supongamos que dos CPU, A y B, leen la misma palabra en sus respectivas cachés. A escribe entonces sobre la palabra. Cuando B lee esta palabra obtiene el valor anterior y no el valor recién escrito por A. La memoria es entonces inconsistente.

Para solucionar esto las cachés realizan un monitoreo constante del bus. Cada vez que una caché observa una escritura a una dirección de memoria presente en ella, puede eliminar ese dato o actualizarlo con el nuevo valor. Este tipo de caché recibe el nombre de *caché monitor*.

² Recomendamos leer el anexo de este modulo sobre la memoria caché.

MULTIPROCESADORES CON CONMUTADORES DE CRUCES

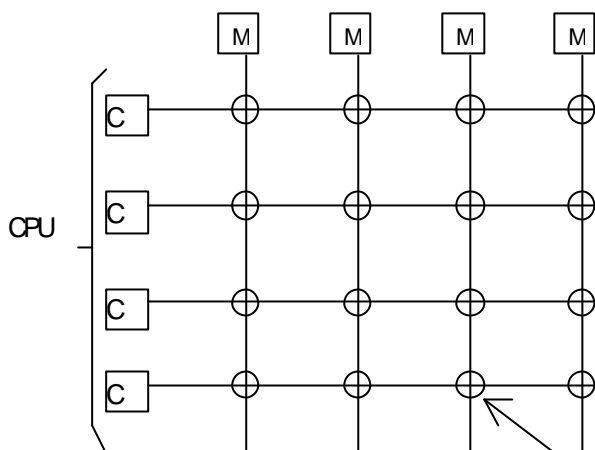


Fig. 10.19 Conmutador de cruce.

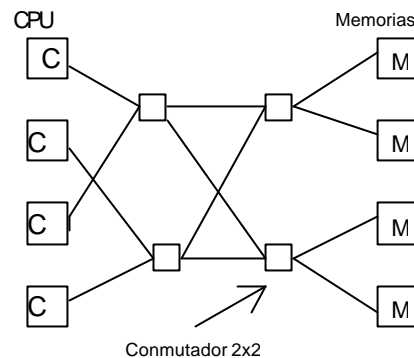


Fig.10.20. Red omega de conmutación

Para conectar muchos procesadores con muchas memorias existe un método diferente de conexión de cada CPU con la memoria central. Esta arquitectura de conexión se lleva a cabo mediante conmutadores de cruce como el que se presenta en la figura 10.19.

Se divide la memoria central en módulos más pequeños. Los módulos de memoria y las CPU están conectados por medio de un conmutador de cruce (cross switch). En cada intersección existe un conmutador de punto de cruce (crosspoint switch) electrónico que el hardware puede abrir o cerrar.

Cuando una CPU necesita tener acceso a una memoria específica, el conmutador de cruce que los conecta se cierra momentáneamente, para realizar el acceso solicitado.

La ventaja que posee el conmutador de cruce es que muchas CPUs pueden tener acceso a Memoria Central al mismo tiempo, pero si dos CPUs intentan acceder al mismo módulo de memoria uno de ellos deberá esperar.

La desventaja del conmutador de cruce es que con n CPUs y n módulos de memorias, se necesitan n^2 conmutadores de cruce. Por lo tanto, se deduce que si n es grande se necesitarían muchos conmutadores, lo cual es desmesurado y caro.

A causa de la desventaja que poseen los conmutadores de cruce, los especialistas han investigado y hallaron otras redes de conmutación que necesiten menos conmutadores. Un ejemplo de éstas son las redes *Omega*. Esta red contiene conmutadores de 2x2, cada uno de los cuales tiene dos entradas y dos salidas. Cada conmutador puede dirigir cualquiera de las entradas en cualquiera de las salidas (Fig. 10.20).

En general, con n CPUs y n memorias, la red omega necesita de n etapas de conmutación, cada una de las cuales tiene $\log_2 n$ conmutadores, para un total de $n \log_2 n$ conmutadores. Aunque este número es mucho menor que n^2 para n grande, sigue siendo significativo.

Otro problema existente es el retraso. Si $n=1024$, existen 10 etapas de conmutación de CPUs a la memoria y otras 10 para que la palabra solicitada regrese.

Se ha intentado reducir el costo mediante sistemas jerárquicos. Cada CPU tiene asociada cierta memoria. Cada CPU puede tener un rápido acceso a su propia memoria local, pero será más lento el acceso a la memoria de las demás. Este diseño se denomina NUMA (Acceso no uniforme a memoria). Aunque las máquinas NUMA poseen un mejor tiempo promedio de acceso que las máquinas basadas en redes omega, tienen una complicación: la distribución de los programas y datos se convierte en un punto crítico, para lograr que la mayoría de los accesos sean a la memoria local.

MULTICOMPUTADORAS CON BASE EN BUSES

Cada CPU tiene una conexión directa con su propia memoria local. El único problema es la forma en que las CPUs se comuniquen entre sí. Aquí también se necesita un esquema de interconexiones. Podemos ver en la figura 10.21 una multicomputadora basada en buses. Es similar arquitectónicamente al multiprocesador basado en bus, pero como tendrá menor tráfico no necesita un bus de base plano de alta velocidad.

Podríamos decir que este modelo está compuesto por una colección de estaciones de trabajo en una LAN, en contraste con la colección de tarjetas de CPU que se insertan en bus rápido.

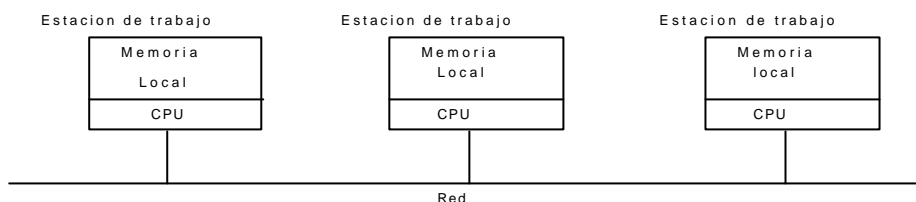


Fig. 10.21. Una multicomputadora que consta de estaciones de trabajo en una LAN

Multicomputadoras con Redes de interconexión

Se han propuesto varias redes de interconexión, pero todas tienen la propiedad de que cada CPU tiene acceso directo y exclusivo a su propia memoria particular.

La figura 10.22b y c muestra dos topología populares, una retícula y un hipercubo. Las retículas se basan en las tarjetas de circuitos impresos. Se adecuan mejor a problemas con una naturaleza bidimensional inherente, como la teoría de grafos o la visión. (ej.: Ojos de robot).

El hipercubo es un cubo de n dimensiones. El hipercubo de la figura 10.22a es de cuatro dimensiones.

La Fig. 10.22c se puede pensar como dos cubos ordinarios, cada uno de los cuales cuenta con 8 vértices y 12 aristas. Cada vértice es una CPU. Cada arista es una conexión entre dos CPUs. Se conectan los vértices correspondientes de cada uno de los cubos.

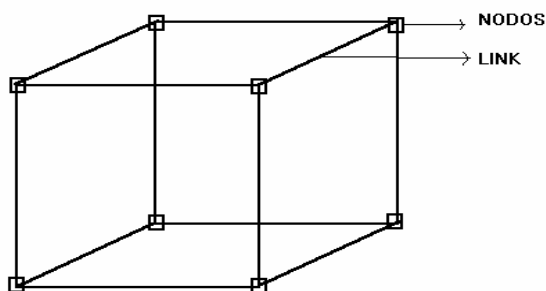


Figura 10.22-a Arquitectura hipercubo

Para el caso de un hipercubo de n dimensiones, cada CPU tiene n conexiones con otras CPUs. Así la complejidad del cableado aumenta en proporción logarítmica con el tamaño. Puesto que solo se conectan los vecinos mas cercanos, muchos mensajes deben realizar varios saltos antes de llegar a su destino.

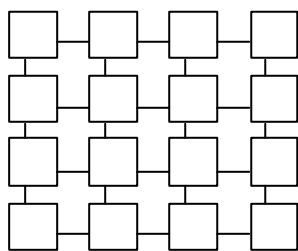


Fig. 10.22-b.

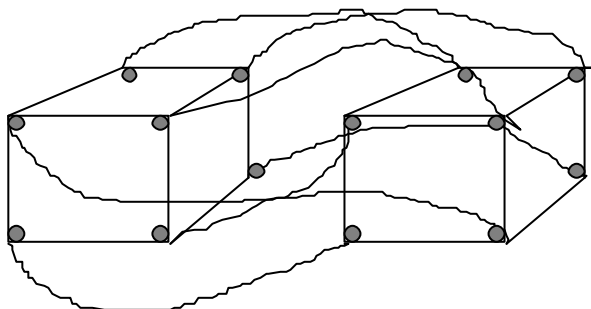


Fig. 10.22.-c.

SISTEMAS DE MULTIPROCESADOR CON TIEMPO COMPARTIDO

Ofrecen la imagen de un único sistema pero lo hacen mediante la vía de centralizar todo, por lo que en realidad este caso es un único sistema. Los multiprocesadores con tiempo compartido NO son sistemas distribuidos.

Este tipo de sistemas es una combinación de software fuertemente acoplado en un hardware también fuertemente acoplado. Aunque existen varias máquinas de propósito especial en esta categoría, los ejemplos más comunes de propósito general son los multiprocesadores, que operan como un sistema de tiempo compartido de UNIX, solo que con varias CPUs en vez de una sola. Para el mundo exterior, un

multiprocesador con 32 CPUs idénticas de 5 MIPS³ actúa de manera muy parecida a una sola CPU de 160 MIPS. Esta es la imagen de un único sistema analizada antes. Solo que la implementación de éste en un multiprocesador es más sencillo, puesto que todo el diseño se puede centralizar. Este esquema es muy utilizado en equipos servidores escalables.

La característica de este tipo de sistemas es la existencia de una sola cola de 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 ejecuciones una estructura de datos contenida en la memoria compartida.

Como por ejemplo considere la figura 10.23, con 4 CPU y 6 procesos listos para su ejecución. Los seis procesos están dentro de la memoria compartida y por el momento se ejecutan 4 de ellos: el proceso A en la CPU 1, el proceso B en la CPU 2, el C en la CPU3 y el proceso D en la CPU 4. Los procesos E y F están en la memoria esperando para ser ejecutados.

Supongamos que el proceso B se bloquea en espera de entrada/salida o que su quantum de tiempo haya terminado. Es la CPU 2 quien debe suspenderlo y encontrar otro proceso para llevarlo a ejecución. Lo normal sería que la CPU 2 comenzará la ejecución del dispatcher del sistema operativo (localizado en la memoria compartida). Después de realizar el context switch y guardarlo en memoria central, entrará a una región crítica para ejecutar sobre B el planificador y buscar otro proceso para su ejecución. Es esencial que el planificador se ejecute como una región crítica, con el fin de evitar que dos CPUs seleccionen el mismo proceso para su ejecución inmediata.

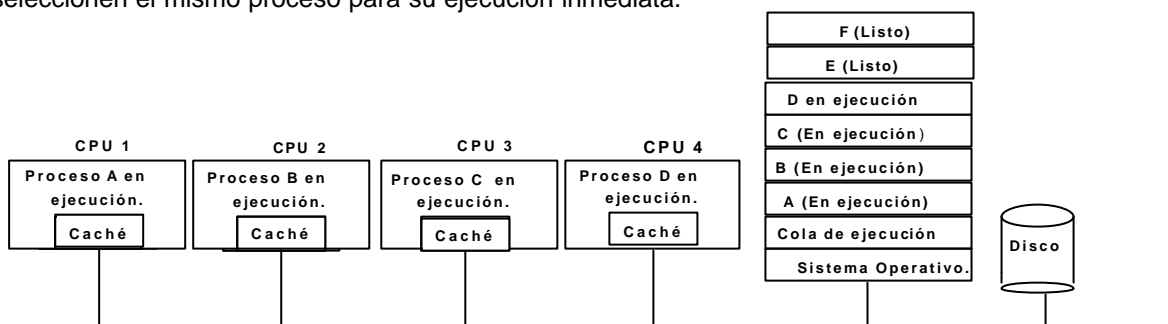


Fig. 10.23. Un multiprocesador con una sola cola de ejecución.

Puesto que ninguna de las CPUs tiene memoria local y todos los programas se almacenan en la memoria global compartida, no importa sobre qué CPU se ejecute un proceso. Si un proceso con un tiempo de ejecución grande se planifica muchas veces antes de terminar, en promedio, gastará la misma cantidad de tiempo que si se ejecutase en una CPU una sola vez.

Un aspecto colateral es que si un proceso se bloquea en espera de E/S en un multiprocesador, el sistema operativo tiene la opción de suspenderlo o bien dejarlo que realice una espera ocupada. Si la mayoría de la E/S se lleva a cabo en menos tiempo del que tarda un cambio entre los procesos, entonces es preferible una espera ocupada. Algunos sistemas dejan que el proceso conserve su procesador por unos cuantos milisegundos, con la esperanza de que la E/S termine pronto, pero si esto no ocurre antes de que se agote el tiempo, se realiza una conmutación de procesos sobre esa CPU.

Un área en donde este tipo de multiprocesador difiere de manera apreciable de una red o un sistema operativo distribuido es la organización del sistema de archivos. Cuando cualquier proceso ejecuta una llamada al sistema operativo, éste se ejecuta mediante semáforos, monitores, o cualquier otro sistema de sincronización equivalente para bloquear las demás CPUs, mientras se ejecutan las regiones críticas o se tiene acceso a las tablas del Sistema. De este modo, por ejemplo al hacer una llamada WRITE, la caché central se bloquea, los nuevos datos entran a la caché y luego se retira el bloqueo.

Se deja expresa constancia, que los métodos utilizados en el multiprocesador para lograr la apariencia de un uniprocador virtual no se pueden aplicar a máquinas sin memoria compartida. Las colas centralizadas de ejecución y los cachés de bloque solo funcionan cuando todas las CPUs tienen acceso a ellas con muy poco retraso. Aunque estas estructuras de datos se podrían simular en una red de máquinas, los costos de comunicación hacen que este método sea prohibitivamente caro.

10.3.Comunicaciones

³ MIPS significa Millones de Instrucciones por Segundo

En los puntos anteriores explicamos las arquitecturas de procesadores y sus interconexiones. También dimos los conceptos de redes con las que se vinculan los centros de procesamiento. Ahora explicaremos como viajan los datos y los mensajes a través de estos elementos.

Básicamente las comunicaciones se establecen mediante una Arquitectura de Comunicaciones que contempla los nodos (de una red) interconectados (vinculados) por las que viajan los mensajes y los datos de los procesos localizados físicamente que pretenden comunicarse. Entonces tenemos varios problemas: la adecuación de los mensajes y datos para su transmisión, que se resuelve mediante una arquitectura de protocolos por un lado. Por otro lado se requiere establecer una vinculación física y lógica para que los datos y mensajes tengan un origen y un destino. Esto implica lograr una ruta y una conexión. Veamos como se resuelve estos problemas.

Necesidad de una Arquitectura de Comunicaciones

Planteamos la necesidad de transferir por ejemplo un archivo entre dos estaciones de trabajo conectadas a una red. El procedimiento sería el siguiente:

1. El sistema de origen debe informar a la red sobre la identidad del sistema de destino deseado.
2. El sistema de origen debe determinar que el sistema destino está listo para recibir datos.

La aplicación de transferencia de archivos del sistema origen debe determinar que el programa de gestión de archivos del sistema destino está preparado para aceptar y guardar el archivo de un usuario particular.

Para una comunicación con éxito, cada entidad del sistema global debe tener una dirección única. En realidad, se necesitan dos niveles de direccionamiento (*ver Service Access Point - SAP*).

Los módulos del mismo nivel en computadores diferentes se comunican uno con otro: por medio de protocolos

Puede decirse que la aplicación emisora genera un bloque de datos y lo pasa al nivel de transporte. El nivel de transporte puede dividir este bloque en partes más pequeñas para hacerlo más manejable, a cada una de las cuales le añade una **cabecera de transporte** conteniendo información de control del protocolo. La combinación de datos del nivel superior siguiente y la información de control se conoce como una unidad de datos del protocolo (PDU); **PDU de transporte** en este caso. La cabecera de cada

PDU de transporte contiene información de control a usar por el protocolo de transporte del computador B.

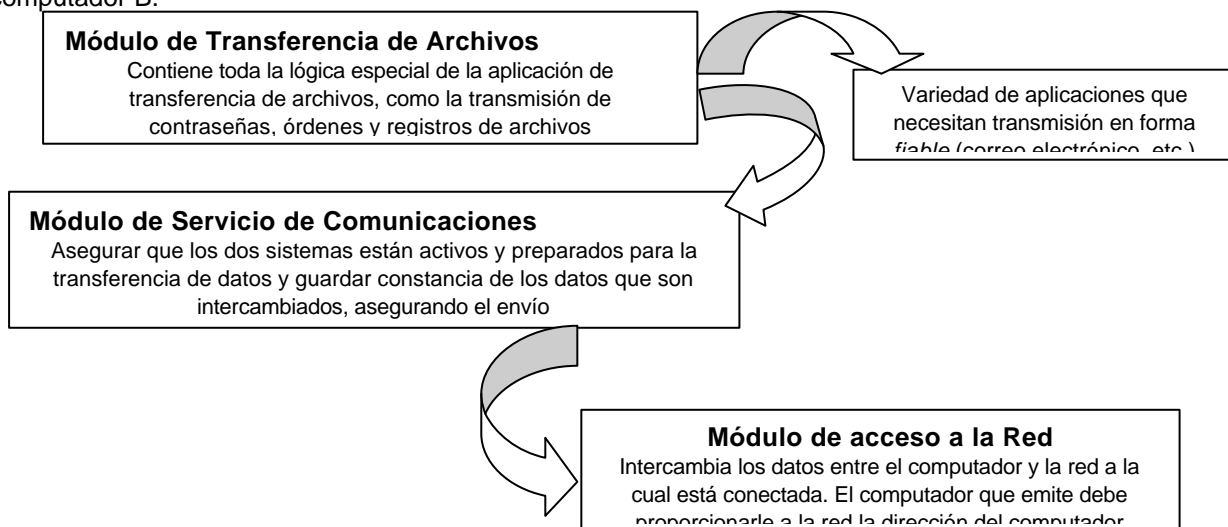


Fig. 10.24 Interacción de los módulos que intervienen en una comunicación de datos.

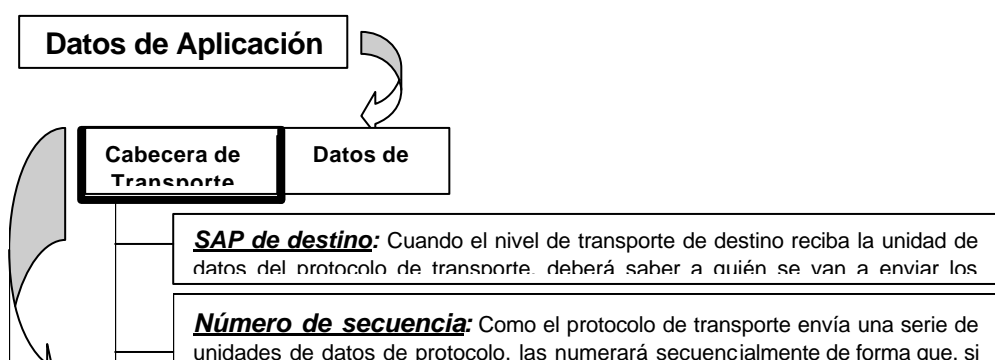


Fig. 10.25. Preparación de los datos para su

La siguiente etapa consiste en que el nivel de transporte entregue cada PDU al nivel de red junto con instrucciones para transmitirlos al computador de destino. Para cumplir con este requisito, el protocolo de acceso a la red debe presentar los datos a la red mediante una solicitud de transmisión. El protocolo de acceso a la red añadirá una cabecera (header) a los datos que reciba del nivel de transporte, para crear una PDU de acceso a la red.

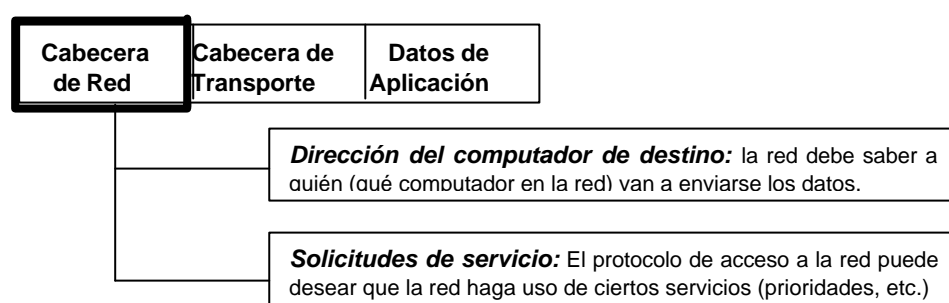


Fig. 10.26. Preparación de los datos para el acceso a la red de comunicaciones

10.3.1 Protocolos de Comunicación y Arquitecturas de Protocolos

Para intercambio de comunicación entre dos máquinas, se necesita un camino para los datos. Este camino puede ser lograda mediante una vinculación Directa o Indirecta.

Como se explicó en el punto anterior, cuando las computadoras, terminales u otros dispositivos de proceso de datos intercambian datos, el procedimiento puede ser muy complejo. Considérese como ejemplo la transferencia de un archivo entre dos computadoras conectados a una red. Las siguientes son algunas de las tareas normales a desarrollar:

- El sistema de origen debe informar a la red sobre la identidad del sistema de destino deseado.
- El sistema de origen debe determinar que el sistema de destino está listo para recibir datos.
- La aplicación de transferencia de archivos del sistema origen debe determinar que el programa de gestión de archivos del sistema destino está preparado para aceptar y guardar el archivo de un usuario particular.
- Si los formatos de archivos que se usan en los dos sistemas son incompatibles, uno de los dos debe llevar a cabo una función de traducción de formato.

Esta claro que debe existir un alto grado de cooperación entre los dos computadores. En vez de implementar la lógica de esta cooperación como un único módulo, la tarea se dividirá en subtareas, cada una de las cuales se implementa por separado.

Hay ciertos casos que en vez de existir un único módulo que lleve a cabo las comunicaciones se dispone de un conjunto estructurado de módulos que implementa las funciones de la comunicación. Dicha estructura se conoce como **arquitectura de comunicaciones** que se ocupa de las operaciones necesarias para que la comunicación sea exitosa o informa si no pudo lograrla (ver Fig. 10.24).

Además de la vinculación para establecer una comunicación entre computadoras se necesita dos cosas mas:

- Protocolos
- Arquitectura de protocolos

Definimos como **PROTOCOLO** al conjunto (Set) de reglas que gobiernan el intercambio de datos entre dos entidades. Un protocolo implica 3 cosas: la Sintaxis, la Semántica y el Timing o velocidad de las comunicaciones.

Definimos como **ARQUITECTURA de Protocolos** a la estructura que conforma a un juego de protocolos que implementan las funciones de comunicación.

10.3.2. Una Arquitectura de Comunicaciones Simple

En términos generales, se dice que las comunicaciones implican a tres agentes:

- Procesos: son las entidades que se comunican.
- Hosts o sistemas computacionales: es donde se encuentran los Procesos:
- Redes o Networks: interconectan a los Hosts y transmiten los datos entre ellos.

Las aplicaciones que son objeto de preocupación son las aplicaciones distribuidas, en las que entra en juego el intercambio de datos entre dos sistemas, algunos ejemplos son el correo electrónico y la transferencia de archivos. Estas y otras aplicaciones se ejecutan en computadores multiprogramados, que soportan muchas aplicaciones simultáneas.

La transmisión de datos de un Proceso a otro Proceso parece natural organizar las comunicaciones en los siguientes tres niveles independientes:

- Nivel de acceso a la red
- Nivel de transporte
- Nivel de aplicación

El nivel de acceso a la red se ocupa del intercambio de datos entre el computador y la red a la que está conectada. El computador que emite debe proporcionarle a la red la dirección del computador destino, de forma que la red pueda encaminar los datos al destino apropiado. El software específico utilizado en este nivel dependerá del tipo de red usada; se han desarrollado diferentes estándares para la comunicación de circuitos, la conmutación de paquetes, redes de área local y otras más. Las características son las siguientes:

Nivel de acceso a Red:

- Aplicación y Red tienen que ser Independientes
⇒Capa específica para manejo de Red.
 - Intercambio de datos entre Host y Red .
 - Host le provee destino (igual Host que destino).
 - Host (fuente) puede pedir ciertos servicios.
 - Protocolo de este nivel DEPENDE del tipo de Red:
 - Conmutación de circuito.
 - Conmutación paquetes
 - LAN
 - Etc...

Sin importar la índole de las aplicaciones que intercambian datos, suele pedirse que los datos se intercambian de forma fiable. Los mecanismos para ofrecer fiabilidad son, en esencia, independientes de la naturaleza de las aplicaciones. De esta forma, parece razonable reunir estos mecanismos en un nivel común compartido por todas las aplicaciones; este nivel es conocido como nivel de transporte. Las características son las siguientes:

Nivel de Transporte: Se asegura del transporte de los procesos INDEPENDIENTEMENTE de la naturaleza de los procesos.

- Seguridad del intercambio de datos:
 - Todos los datos llegan a destino (Proceso destino).
 - Todos los datos llegan en el mismo orden en que fueron enviados.

Finalmente el nivel de aplicación contiene la lógica necesaria para soportar varias aplicaciones de usuario. Para cada clase diferente de aplicación, como la transferencia de archivos se necesitará un módulo separado y particular para la misma. La característica principal es la siguiente:

Nivel de Proceso: Protocolos necesarios por la VARIEDAD de Aplicaciones.

- Por cada tipo de Aplicación se necesita un protocolo propio a esa Aplicación.

Por otra parte para lograr una comunicación con éxito, cada entidad del sistema global debe tener una dirección única. En realidad, se necesitan dos niveles de direccionamiento. Cada computador de la red debe tener una **dirección única** en la red para que ésta envíe los datos al computador correcto. Cada aplicación de un computador debe tener una dirección que sea única dentro de su computador, lo que permite que el nivel de transporte envíe datos a la aplicación correcta. Estas últimas direcciones se conocen como puntos de acceso al servicio (SAP), implicándose el hecho de que cada aplicación individual accede a los servicios del nivel de transporte.

Ejemplo de Mensaje de SAP1(Service Access Point 1) Host A a SAP2 Host B de la figura:

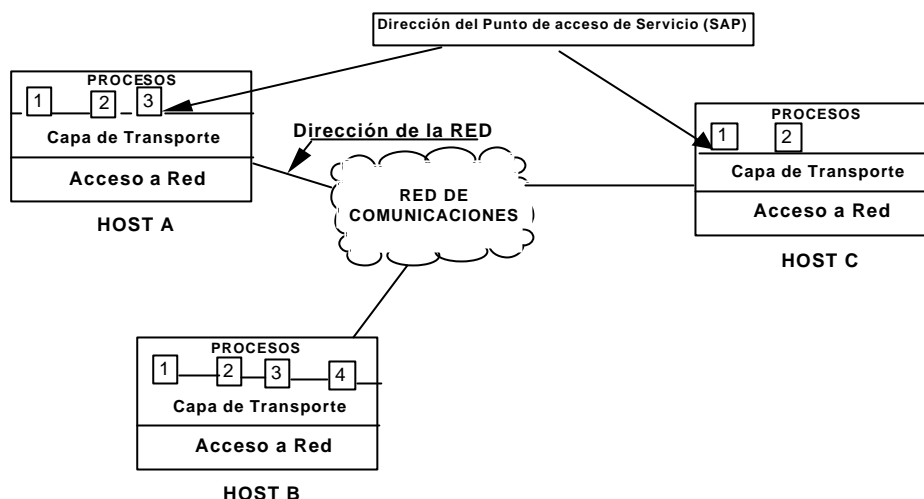


Fig. 10.27. Transmisión de datos en una red de comunicaciones

Para una comunicación exitosa se necesitan DOS NIVELES DIRECCIONAMIENTO:

- Cada Host con una única dirección (Nivel de Red).
- Cada Proceso debe tener una única dirección en un Host (Nivel Transporte).

Por ejemplo: dos procesos que se desean comunicar deben hacer:

1) Proceso en A manda bloque de datos por su capa de transporte con instrucciones de que vaya para Host B SAP2.

2) La Capa de Transporte

- Divide bloque de datos de Proceso en pedazos más manejables.
- Le pone a cada pedazo un Header de Transporte con información (Ver Fig, 10.28).
- Se pasa PDU(Protocol Data Unit) a la de Red diciéndole que tiene que ir al Host B (solamente, no dice SAP2).

3) Capa de Red le pone Header de Red y hace un pedido (request) de transmisión.

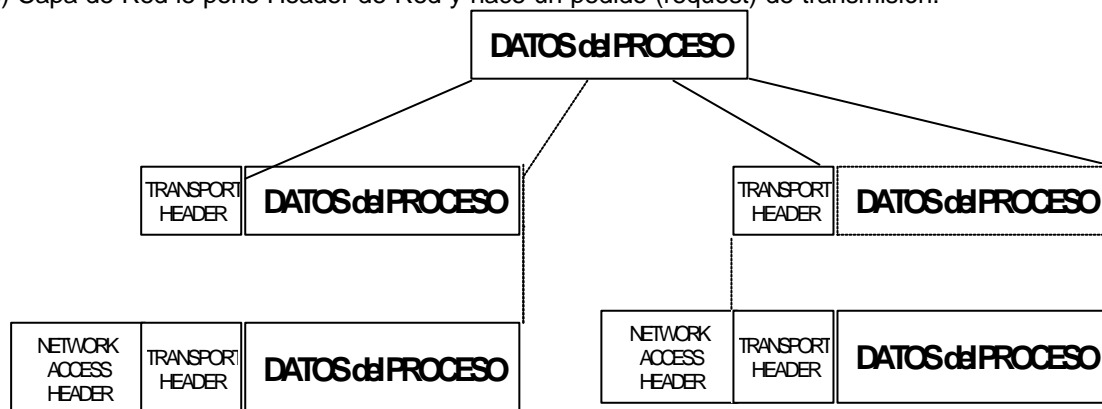


Fig. 10.28. La estructura de un paquete de datos para una transmisión.

Se presentan cuatro aspectos importantes de la comunicación en cuanto al trabajo interno:

- **Nombre:** ¿Cómo hacen los procesos PARA UBICARSE.?
- **Estrategia de Ruteo:** ¿Cómo hacen los mensajes PARA PASAR POR LA RED?
- **Estrategia de Comunicación:** ¿Cómo hacen dos Procesos PARA MANDARSE UNA SECUENCIA DE MENSAJES? .
- **Contención:** La Red es un recurso compartido, ¿CÓMO RESOLVER EL CONFLICTO DE SU DEMANDA?.

Estas cuatro preguntas, en la arquitectura de comunicaciones, llevan a plantearse un nuevo conjunto de interrogantes en los aspectos del diseño que deben ser resuelto mediante diferentes estrategias.

Al diseñar un red de comunicaciones es necesario tener en cuenta cuatro aspectos:

- **Estrategias de ruteo:** ¿Como se enviarán los mensajes por la red?
- **Estrategias de conexión:** ¿Como envían dos procesos una serie de mensajes?

- **Conflictos:** Dado que la red es un recurso compartido, ¿Como solucionamos las demandas de uso conflictivas?
- **Estrategias de diseño:** ¿Cual es el diseño global para la comunicación entre aplicaciones?

10.3.3. Estrategias de ruteo de mensajes

Cuando la instalación A quiere comunicarse con un proceso en la instalación B, ¿Como se envía el mensaje?. Si solo hay una ruta física entre A y B, el mensaje debe ser enviado por esa ruta, pero si existen varias rutas físicas se plantean opciones de ruteo. Cada maquina debe tener una tabla de rutas posibles que indica las alternativas de rutas que pueden emplearse para enviar un mensaje a otras instalaciones. Esta tabla puede incluir información acerca de la velocidad y costo de las diversas rutas. La tabla de ruteo se puede modificar manualmente o dinámicamente en base a caminos alternativos o velocidad, incluso se puede establecer un Costo de cada camino para una mejor decisión.

Los tres esquemas de ruteo mas comunes son:

- **Ruteo fijo:** Se especifica por adelantado una ruta de A a B y no cambia a menos que una falla en el hardware invalide esta ruta. Se mantiene hasta el final de la comunicación. Su aspecto negativo es que no puede adaptarse a los cambios de la carga, del trafico de la ruta pero los mensajes se entregaran en el orden en que fueron enviados.
- **Circuito virtual:** Se determina un camino entre dos nodos (de A a B) que va a durar una sesión puede cambiar entre sesión y sesión.
- **Ruteo Dinámico:** Se elige el camino solo cuando un mensaje es enviado. Cada nodo intermedio decide donde mandarlo. La ruta para enviar un mensaje de la instalación A a la Instalación B se elige en el momento de enviar el mensaje. Generalmente se elige la ruta menos congestionada en ese momento. Aunque es el ruteo mas rápido tiene el problema de que los mensajes pueden llegar en cualquier orden; esto puede arreglarse adosando un número de secuencia a cada mensaje.

10.3.4. ESTRATEGIAS DE CONEXIÓN

Cuando procesos ya tienen camino, pueden empezar una SESIÓN. Existen varias maneras de conectar pares de procesos; los esquemas mas comunes son:

- **Conmutación de circuitos:** Si dos procesos quieren comunicarse entre ellos se establece un enlace físico permanente. Este enlace se asigna para todo el tiempo que dura la comunicación y ningún otro proceso puede utilizarlo durante ese periodo. Las desventajas de este esquema es que requiere tiempo de preparación, pero provoca menos tiempo de procesamiento adicional para enviar cada mensaje.
- **Conmutación de mensajes:** Si los procesos quieren comunicarse se establece un enlace temporal durante el tiempo que dura la transferencia de un mensaje. Los enlaces físicos se asignan dinámicamente según se requiera y durante un periodo breve. Cada mensaje es un bloque de datos, junto con cierta información (fuente, destino, y códigos de corrección de errores) que permite a la red de comunicaciones entregar correctamente el mensaje a su destino. Tiene como aspecto negativo que requiere menos tiempo para la preparación del mensaje pero necesita mas tiempo de procesamiento adicional por cada mensaje.
- **Conmutación de paquetes:** Los mensajes generalmente tienen longitud variable. Para simplificar el diseño del sistema habitualmente se implanta la comunicación con mensajes de longitud fija llamados paquetes. Es posible que un mensaje lógico tenga que dividirse en varios paquetes, cada uno de los cuales se envía por separado a su destino y puede seguir rutas diferentes por la red. Para formar el mensaje hay que reensamblar los paquetes conforme llegan. Los mensajes entran en memoria no necesitan bajar a disco. Al igual que la conmutación de mensajes requiere menos tiempo de preparación y pierde mas tiempo de procesamiento adicional con el agregado de que debe dividir los mensajes en paquetes y luego reagruparlos.

10.3.5. CONFLICTOS

Un enlace puede conectar varios nodos. Es posible que estos quieran transmitir simultáneamente información por un enlace. En este caso se mezcla la información transmitida y hay que descartar la que no corresponde a ese nodo. Es necesario notificar el problema a los nodos para que estos puedan retransmitir la información.

Se han desarrollado varias técnicas para evitar las colisiones repetidas en un enlace.

CSMA/CD (Carrier Sence Multipple Access/ Collision Detection): Antes de transmitir un mensaje por un enlace, una maquina debe escuchar para determinar si en ese momento se esta transmitiendo otro mensaje en ese mismo instante por el enlace. Esta técnica se denomina detección de portadora con acceso múltiple. Si el enlace esta libre, la instalación puede comenzar a transmitir de lo contrario debe esperar y seguir escuchando hasta que el enlace quede libre. Si dos o mas nodos comienzan a transmitir en el mismo instante entonces ambos registraran una detección de colisión y dejaran de transmitir. Cada nodo intentará hacerlo de nuevo, después de un cierto periodo aleatorio de tiempo.

El problema principal con esta estrategia es que cuando el sistema esta muy ocupado pueden ocurrir muchas colisiones y degradarse el rendimiento.

Paso de testigo (Token Passing): Un tipo de mensaje único, conocido como testigo, circula continuamente por el sistema. Un nodo que desea transmitir información espera a que llegue el testigo; lo saca de la red y comienza a transmitir sus mensajes. Cuando termina vuelve a transmitir el testigo lo que permite que otro nodo reciba y quite el testigo para comenzar la transmisión de sus mensajes.

El sistema debe detectar si se ha perdido el testigo y en tal caso generar uno nuevo.

Ranuras de mensaje (Message slots): Por el sistema circulan constantemente varias ranuras de mensaje de longitud fija. Cada ranura puede contener un mensaje de longitud fija e información de control (origen, destino y si la ranura esta vacía o llena). Un nodo que esta listo para transmitir debe esperar que llegue una ranura vacía, en la cual insertará su mensaje ajustando la información de control adecuada. La ranura con su mensaje prosigue entonces por la red y al llegar a un nodo, este examina la información de control para ver si le corresponde o no el mensaje en la ranura. Si no es para ese nodo, la ranura con el mensaje vuelve a circular; de lo contrario, el nodo toma el mensaje, restablece la información de control indicando que la ranura esta vacía, y luego puede enviar un mensaje propio o liberar la ranura. Como una ranura solo puede contener mensajes de longitud fija, en ocasiones hay que dividir el mensaje lógico en paquetes mas chicos, cada uno de los cuales se envía en una ranura diferente.

10.3.6. ESTRATEGIAS DE DISEÑO

Al diseñar una red de comunicaciones debemos tratar con la complejidad inherente de la coordinación de operaciones sincrónicas que se comunican en un entorno potencialmente lento y propenso a errores.

La tarea del diseño es definir una serie de niveles y servicios desempeñados por cada uno. La división debe agrupar lógicamente a las funciones y debe disponer suficientes niveles para hacer que el manejo de cada uno no sea muy complicado.

Básicamente hay dos modelos:

- Modelo OSI
- Modelo TCP/IP

El concepto de Sistema Abierto

La interconexión de sistemas abiertos se basa en el concepto de aplicaciones distribuidas cooperativas. Una aplicación distribuida es cualquier actividad en la que interviene el intercambio de información entre dos sistemas abiertos. Algunos ejemplos de dichas actividades son los siguientes:

- Un usuario en un terminal de un computador se conecta a una aplicación de proceso de transacciones de otro computador
- Un usuario envía un mensaje de correo electrónico a otro usuario de otro computador
- Un programa de control de procesos envía una señal de control a un robot

El objetivo del esfuerzo de OSI es definir un conjunto de estándares que habilitará a los sistemas abiertos ubicados en cualquier lugar del mundo para cooperar, interconectándolos mediante servicios de comunicaciones estándares y ejecutando protocolos OSI estándares.

Un sistema abierto puede implementarse de cualquier forma que esté de acuerdo con un conjunto mínimo de estándares que permitan conseguir la comunicación con otros sistemas abiertos.

El modelo OSI (Open System Interconnection)

Una técnica de estructuración muy aceptada y elegida por la ISO es la división en niveles. Cada nivel desempeña un subconjunto de las funciones necesarias para comunicarse con otro sistema. En el mejor de los casos los niveles deben definirse de forma que los cambios en uno no provoquen cambios en los otros niveles. De este modo, se ha descompuesto un problema en un conjunto de subproblemas más manejables.

Entonces, podemos simplificar el diseño dividiendo el problema en varias niveles, de acuerdo con lo establecido por la Organización Internacional de Normas (ISO). Esta división en niveles se trata del Modelo OSI (Modelo Internacional de Sistemas Abiertos) que se presenta en la Fig.10.210.

La tarea de la ISO fue definir una serie de niveles y servicios desempeñados por cada uno. La división debe agrupar lógicamente a las funciones y debe disponer suficientes niveles para hacer que el manejo de cada uno no sea muy complicado.

Los niveles OSI son:

Nivel Físico: Es la interfase física entre un dispositivo de transmisión de datos y el medio de transmisión, y se las reglas con que los bits se pasan de uno a otro. Se ocupa de transmitir un flujo no estructurado de bits por el enlace físico, donde incluye parámetros como el nivel de voltaje de la señal y la duración de los bits. Contiene cuatro elementos importantes:

- **Mecánicos:** Están relacionados con el punto de contacto, y usualmente es un conector con un número específico de patillas que soportan una serie de cables portadores de señal a través de la interfase.
- **Eléctricos:** Corresponde a los niveles de voltaje y la temporización de cambios en el voltaje con que se definen los bits. Determinan la velocidad de los datos y las distancias que pueden alcanzarse.
- **Funcionales:** Determinan las funciones que se desempeñan asignando un significado a cada cable.
- **Procedimientos:** Especifican la secuencia de sucesos en la transmisión de datos.

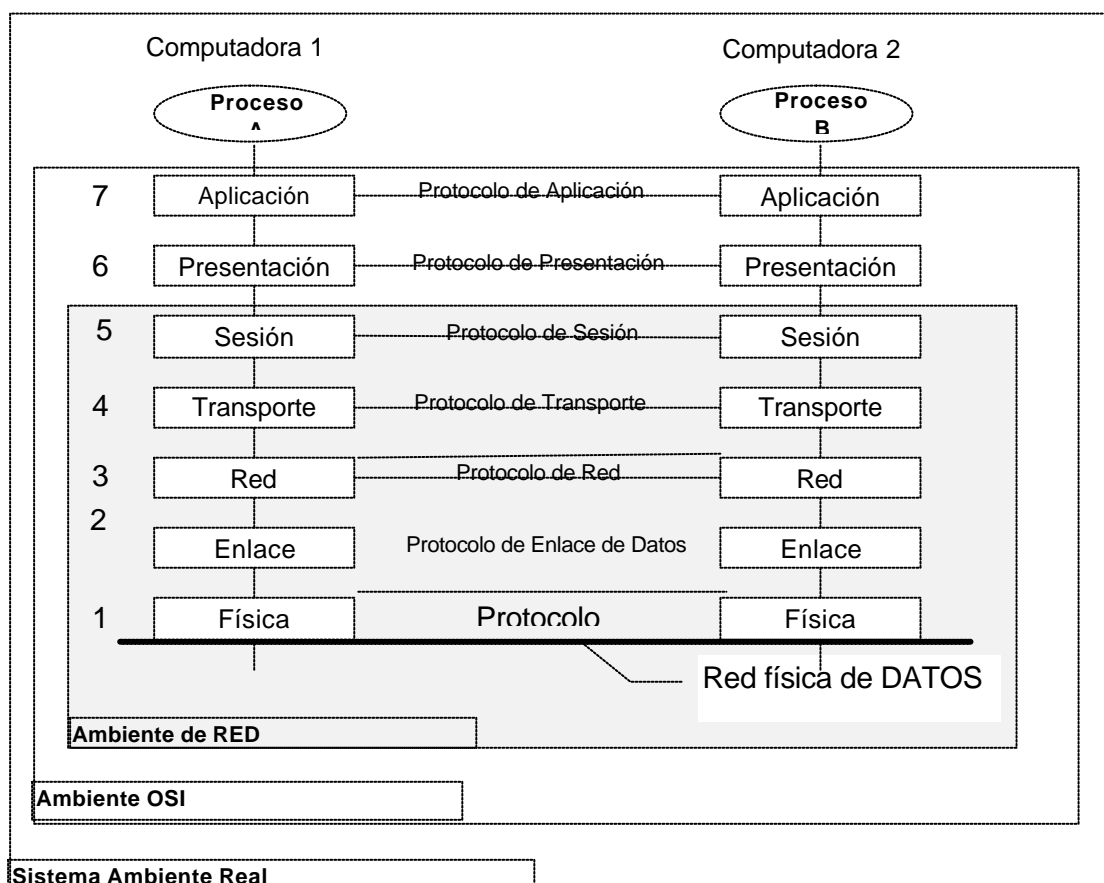


Fig. 10.29 El modelo OSI

Nivel de enlace de datos: Intenta hacer el enlace físico confiable y permite activar, mantener y desactivar el enlace. Los principales servicios que brinda a los niveles superiores son la detección y el control de errores. Lo que se hace para garantizar la fiabilidad es pasar información de control junto con los datos, por el enlace. Los datos se transmiten en bloques llamados **tramas**, donde cada trama tiene una cabecera, un pie y un campo opcional de datos. La cabecera y el pie contienen información de control que se usa en la gestión de enlace.

Nivel de Red: Brinda los medios para transferir información a través de algún tipo de red de comunicaciones. Libra a los niveles superiores de conocer información sobre la transmisión de datos

subyacente y las tecnologías de conmutación utilizadas en la conexión de sistemas. Es responsable de establecer, mantener y finalizar las conexiones con la red. El sistema de computadoras dialoga con la red para especificar la dirección de destino y pedir ciertos servicios de red, como las prioridades.

Nivel de transporte: Ofrece un mecanismo fiable para el intercambio de datos entre computadoras. Así se verifica que los datos enviados no tienen errores, vienen en secuencia, sin pérdidas ni duplicaciones. Puede también ocuparse de optimizar el uso de los servicios de red y de satisfacer a las demandas de calidad del servicio. Para ello se emplean números de secuencia, códigos de detección de errores, y la retransmisión cuando vencen los plazos de tiempo. La complejidad y tamaño del protocolo de transporte depende de la fiabilidad de la red subyacente y los servicios del nivel de red.

Nivel de Sesión: Brinda mecanismos para controlar el diálogo entre dos sistemas finales. A veces este nivel no se utiliza o se utiliza poco, dependiendo de las aplicaciones. Los servicios que ofrece son:

- **Disciplina de diálogo:** Puede ser simultáneo en ambos sentidos (full duplex), alterno en cada sentido (half duplex), o simple.
- **Agrupación:** el flujo puede ser marcado para definir grupos de datos.
- **Recuperación:** El nivel de sesión puede ofrecer un mecanismo de puntos de control, de modo que si se produce un fallo entre dos puntos, se pueda retransmitir la información desde el último control.

Nivel de Presentación: Define la sintaxis y la semántica de los datos, para diferenciar los formatos a intercambiar entre las aplicaciones. Por ejemplo el cifrado o la compresión de datos se realizan en este nivel.

Nivel de Aplicación: Proporciona a los programas de aplicación el acceso al entorno OSI. Es la interfase para el usuario. Tiene mecanismos útiles para respaldar las aplicaciones distribuidas. En este nivel residen además las aplicaciones de propósito general, como la transferencia de archivos, el correo electrónico, y el acceso por terminal a computadoras remotas.

Hay otra estrategia de diseño para las comunicaciones conocido como TCP/IP que es la mas difundida en estos momentos.

El Modelo TCP/IP (*Transmission Control Protocol / Internet Protocol*)⁴

El TCP/IP es el resultado de la investigación del protocolo y la conducta del desarrollo en la red experimental ARPANET, fundada por la agencia de investigaciones avanzadas de proyectos de defensa (DARPA) que es generalmente referida como el conjunto de protocolos TCP/IP. Este conjunto de protocolos consiste en una colección de protocolos que son usados como el estándar de Internet por la arquitectura global de Internet (IAB- Internet Architectur Board).

Definición del Department Of Defense Military Standard Protocols:

- Norma N° 1777: Proveer un servicio para terminales para comunicarse entre una o más redes. No asume la confiabilidad de la red.
- 1778: Proveer un servicio de transferencia confiable entre terminales. Equivalente a la capa 4 del modelo OSI.
- 1780: Proveer una simple aplicación para la transferencia de ASCII, EBCDIC y archivos binarios.
- 1781: Proveer facilidades para un simple correo electrónico.
- 1782: Proveer un simple controlador remoto de terminales.

No existe un modelo oficial del protocolo TCP/IP, como sí en el caso de OSI. Sin embargo, basándonos en los protocolos estándares que han sido desarrollados sobre UNIX podemos organizar las tareas de comunicaciones para el TCP/IP en cinco capas independientes:

⁴ Recomendamos leer el Anexo 10.A sobre este tema.

FTP SMTP TELNET
TCP
IP
NET ACCESS

Modelo construido en base a tres conceptos mas el Acceso a Red:

- Capa de proceso o de aplicación (FTP, SMTP, TELNET, etc...).
- Capa Host to host o capa de transporte (TCP).
- Capa Internet (IP).
- Capa de Acceso a Red y
- Capa física

La capa física cubre la interfase física entre los dispositivos de transmisión de datos (por ejemplo las computadoras o workstations) y el medio de transmisión o la red. Esta capa concierne las características del medio de transmisión, la naturaleza de las señales, la tasa de datos y los problemas relacionados. Esta capa no pertenece a la arquitectura TCP/IP.

La capa de acceso a la red (Network Access Layer) es una capa aparte, abstrae las particularidades de la Red. Conciene el intercambio de datos entre un sistema final y la red a la cual está adjuntada. La computadora que manda debe proveer a la red con la dirección de la computadora de destino de modo tal que la red pueda realizar una ruta de datos apropiada para ese destino. La computadora que manda debe invocar ciertos servicios, como una prioridad, que deberá ser proveídos por la red. El software usado en esta capa depende del tipo de red usada. Protocolo de este Layer depende de Red, puede ser X.21, X.25, IEEE 802.x u otro. Esta capa tampoco pertenece a la arquitectura TCP/IP.

En el caso de dos computadoras quieren conectarse a través de la red se necesitarán procedimientos específicos para permitir el paso de los datos o mensajes a las múltiples redes interconectadas. Esta es la función de la capa de Internet.

El **protocolo de Internet (IP)** es usado en esta capa para proveer la función de ruteo a través de las múltiples redes. Un **"router"** es un procesador que conecta dos redes y su función primaria es relevar los datos desde una red a otra en su ruta desde la fuente al sistema final de destino. La conexión entre Redes distintas suelen denominarse GATEWAYS (GW). El Network Access rutea datos entre dos dispositivos en la misma Red. En el caso en que dos Host que se quieran comunicar estén en diferentes Redes, se necesita un procedimiento que le permita a los datos atravesar una o varias Redes. Esa es la función de IP, o sea que conecta dos Redes en un Gateways.

El IP obtiene los Segmentos del TCP en un Host, a partir del segmento construye el DATAGRAMAS y los manda al Gateways a través del encabezado de la capa física. A partir de aquí hay dos posibilidades para el paquete:

- 1) Host de Destino está conectado a la Red que está conectada a ese Gateways, entonces se manda al Host.
- 2) Hay que pasar uno o más Gateways para llegar a la Red en que está el Host, entonces se manda al próximo Gateways y este se ocupará de retransmitirlo.

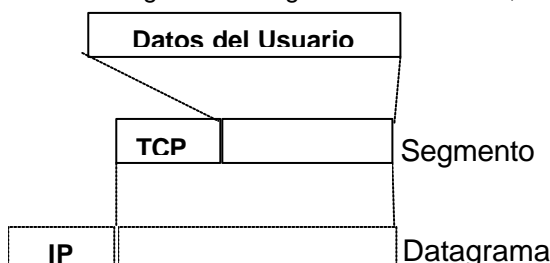
Existen mecanismos para proveer la confianza de que los datos arribarán a destino de la misma forma de que han sido mandados. Estos mecanismos conforman la capa de transporte o de Host a Host.

El **protocolo de control de transmisión (TCP)** es el más comúnmente usado para proveer esta funcionalidad.

La capa de aplicación contiene la lógica necesaria para soportar varias aplicaciones usuario. Para cada tipo de aplicación diferente, como por ejemplo una transferencia de archivo, un módulo separado es necesario particular para esa aplicación.

El TCP descompone en pequeños pedazos la información que debe mandar de una computadora a otra a través de la red para así lograr que la información sea más manejable. A cada una de estas piezas se la adjunta una cierta información de control contenida en la **cabecera TCP**, formando un **segmento**. Cada entidad debe tener una dirección única y cada Host en la subred debe tener una dirección global única en la Internet, la cual será usada para el ruteo y la entrega. Estas direcciones son llamadas **"puertos"**. Cada cabecera del TCP contiene un mínimo de 20 Bytes o 160 bits. La cabecera incluye una dirección "fuente" de 32 bits y la dirección de destino. El campo de control (**checksum**) de la cabecera es usado para detectar errores en la cabecera.

El TCP secuencia numeralmente los segmentos que serán mandados a un puerto de destino particular de tal manera que si los segmentos llegan desordenados, entonces la entidad TCP del destinatario los reordena.



de
Destino

Fig. 10.32 Encabezado del IPv6

El encabezado de IPv6 contiene 40 By. El encabezado de TCP 20 By. El de fragmentación 8 bytes. El resto de los encabezados es de longitud Variable.

El IPv6 resuelve los problemas del IPv4 e incorpora las siguientes mejoras:

- Espacio de direcciones expandido de 128 bits.
- Mecanismos de opciones de encabezado de paquetes separados en lugar de los Datagramas:
 - Encabezado de IP.
 - Encabezado de opciones salto - por - salto (información para cada Router de la trayectoria cantidad de Bytes del Paquete).
 - Encabezado de ruteo (información de cada nodo visitado por Paquete).
 - Encabezado de fragmentación (información de fragmentación y reensamblados).
 - Encabezado de autenticación (integridad y autenticación).
 - Encabezado encapsulador de seguridad de carga.
 - Encabezado de opciones de destino (información para nodo destino).
 - Encabezado TCP.

Esto hace que los routers no tengan que examinar o procesar muchos de estos encabezados. Encabezados de opciones a lo largo de la trayectoria del paquete y con ello simplifica y mejora la velocidad del procesamiento.

Además incorpora Autoconfiguración de recursos. Por ejemplo soporte de tráfico especializado como ser Vídeo en Tiempo Real y también incorpora capacidad de funciones de seguridad, como ser: Autenticación y privacidad.

Aplicaciones del TCP:

Un número de aplicaciones han sido estandarizadas para operar en el TCP. Mencionaremos tres de ellas:

➤ **Protocolo de transferencia de correo simple (SMTP):** Provee una facilidad para el correo electrónico. Provee de un mecanismo para la transferencia de mensajes entre Hosts separados. El SMTP incluye una lista de correo, recipientes de vuelta y el forward. Una vez que el mensaje fue escrito el SMTP lo acepta y hace uso del TCP para mandarlo al módulo SMTP de otro Host. El módulo SMTP hará uso del correo electrónico local para almacenar el mensaje entrante en la casilla del usuario.

➤ **Protocolo de transferencia de archivos (FTP):** Es usado para mandar archivos desde un sistema a otro por debajo del comando del usuario. El texto y los archivos binarios son acomodados y el protocolo provee características para controlar el acceso al usuario. Cuando un usuario quiere traer una transferencia de archivos, el FTP crea una conexión TCP al objetivo del sistema para el intercambio de mensajes de control. Esto permite que la identidad (ID) y la contraseña del usuario sean transmitidas y permite al usuario especificar el archivo y las acciones sobre el archivo deseados. Una vez que la transferencia de archivos fue aprobada una segunda conexión es creada para la transferencia de datos. Cuando la transferencia fue completada la conexión de control es usada para marcar la finalización y estará dispuesto a aceptar nuevos comandos de transferencia de archivos.

➤ **Telnet:** provee una capacidad de entrada remota la cual permite al usuario de una terminal o una computadora personal (P.C.) ingresar a una computadora remota y funcionar como si estuviera conectada directamente con esa computadora. El protocolo fue hecho para trabajar con terminales en modo simple. Telnet esta implementado actualmente en dos módulos: Una es el módulo usuario Telnet, que interactúa con el módulo de la terminal de entrada/salida (I/O) para comunicarse con una terminal local. Convierte las características de las terminales reales al standard de la red y viceversa. La otra es módulo servidor Telnet, que interactúa con una aplicación, actuando como un administrador de un portal de una terminal de tal forma que las terminales remotas parezcan como locales para la aplicación. El tráfico de la terminal entre el usuario y el servidor Telnet es realizado por la conexión TCP.

UNIX y TCP/IP:

El Kernel de UNIX tiene incorporado:

- TCP/IP
- Variedad de Interfases para Acceso a Red llamados puertos.

- Mecanismos de programación estándar (Sockets)

Estructura del UNIX Usuario FTP:

Al hacer un pedido (request) con FTP:

- (1) Se crea un Proceso.
 - (2) Este Proceso abre una Conexión TCP con el Destino.
 - (3) se Crea un Segundo Proceso que asiste con el manejo de la Transferencia.
El Primer Proceso trata con la Traserferencia de Datos.
 - (4) Mientras que el Segundo Proceso trata con las respuestas para el Control de la Conexión.
- Al Terminar, el Server Cierra la Conexión y Avisa al Usuario.

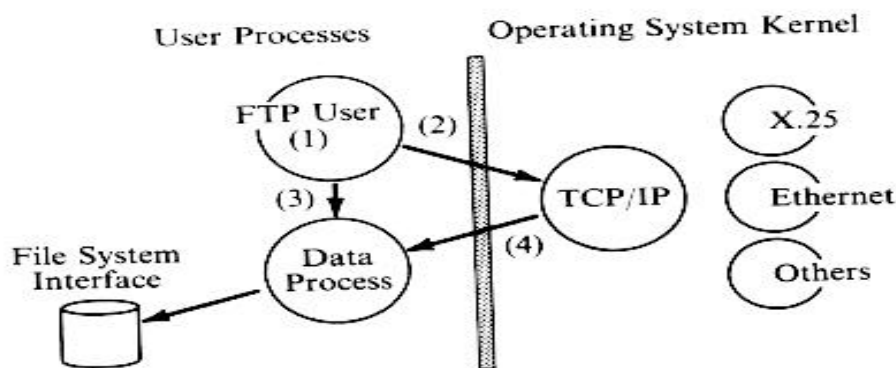


Fig. 10.33 Transmisión por medio de FTP en UNIX

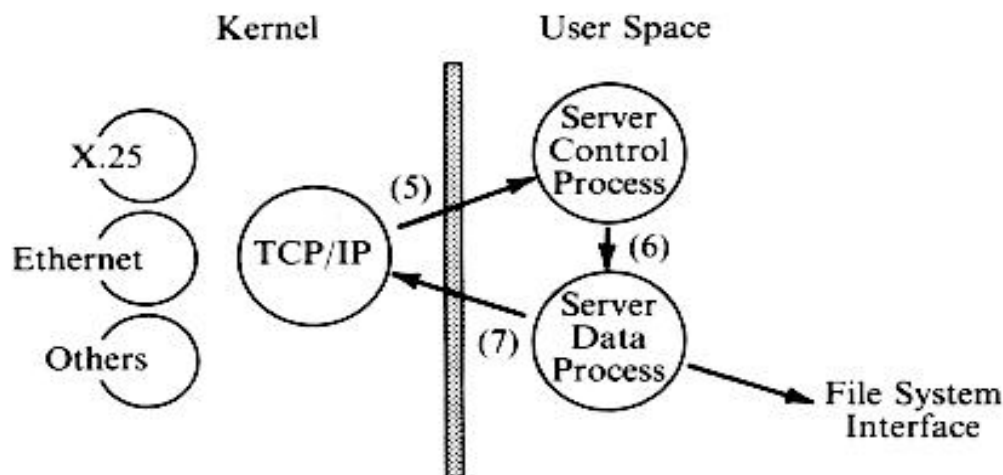


Fig. 10.34. Recepción mediante FTP en UNIX

Estructura del UNIX Server FTP:

El Server nunca sabe cuando le va a llegar un Pedido de Servicio por lo que hay una aplicación que espera siempre por si llega algo. Esto se conoce como "Demonio" (Daemon).

Al llegar un request (por el Port 21: puerto para FTP Request) ocurre lo siguiente:.

- (5) Se crea un Proceso que toma el Control de la Conexión.
Proceso que se va a dedicar al último Request llegado.
(Aplicación se desvincula de la situación y sigue esperando nuevos Request).
- (6) Proceso crea Segundo Proceso para establecer y manejar (7) la Conexión de Datos.
- (7) Este proceso se ocupa de la Conexión de Datos y su transferencia.

10.4. EJEMPLOS DE MODELOS DE SISTEMAS

Los procesos se ejecutan en procesadores. En un sistema tradicional, solo existe un único procesador. En un sistema distribuido, con varios procesadores, este es un aspecto fundamental. Los

procesadores de este tipo de sistemas, se pueden organizar de varias formas como se explicó en los puntos anteriores. Pero se puede definir cuatro tipos:

- Modelo Cliente Servidor
- Modelo de estación de trabajo.
- Modelo de la pila de procesadores
- Cluster de procesadores.

10.4.1. El Modelo Cliente - Servidor.

Llevado a un plano muy básico, el la arquitectura OSI, se basa en establecer una vía de comunicación entre máquinas por medio de la cual los bits serán enviados, uno detrás de otro.

Como vimos anteriormente este método tiene la gran desventaja de generar mucho overhead por mensaje al tener que circular por cada una de las capas, dado que esto es proporcional a la cantidad de mensajes que circulen por la red. Una red más rápida tiende a “relentizarse” más que una de menor velocidad. Es por esto que si bien los modelos OSI y TCP/IP son adecuados para redes WAN no lo son tanto para las LAN, en especial para aquellas de gran performance con una gran cantidad de computadoras.

El modelo de cliente - servidor disminuye este problema, además de brindar una forma de estructurar estos sistemas, ya que no se ocupa solamente de la forma en que se enviarán (y recibirán) los datos como los modelos anteriores.

La estructura de este tipo de sistema consiste en varios *servidores* los cuales prestan servicios a diversos *clientes* conectados a ellos, ambos utilizando el mismo microkernel (en la mayoría de los casos).

El modelo cliente servidor trabaja sobre un entorno de red donde el control de los datos está establecido como un nodo del servidor y está disponible para accederlo, pero no para la actualización en otros nodos.

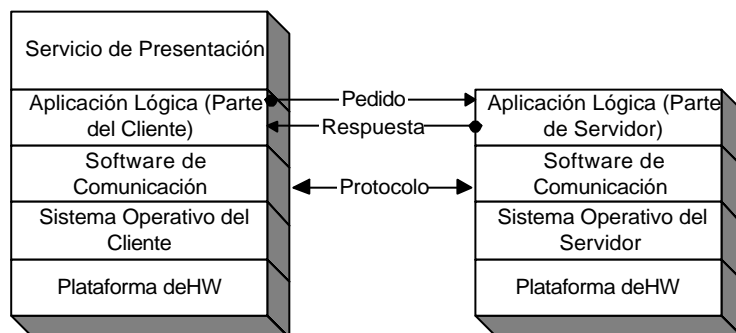


Fig. 10.35 Arquitectura genérica Cliente –

Servidor Las máquinas clientes generalmente son simples P.C. del usuario o workstations que proveen una interfase amigable al usuario final. La estación basada en el cliente generalmente presenta el tipo de interfase gráfica más confortable al usuario, incluyendo el uso de ventanas y mouse. Las aplicaciones basadas en el cliente son fáciles de usar.

Cada servidor en el ambiente cliente/servidor provee un conjunto de servicios compartidos por los usuarios a los clientes. El tipo más comúnmente usado como servidor es el de la base de datos. El servidor debe permitir a mucho clientes compartir el acceso a la misma base de datos.

Para evitar el overhead generado por el uso de protocolos multicapa este modelo emplea un protocolo “sin conexión” de tipo pedido/respuesta (RRP - Request/Reply Protocol). Este protocolo utiliza únicamente dos llamadas al sistema, una para enviar y otra para recibir los mensajes, los cuales (como se desprende de su nombre: send y receive) son o un pedido por parte de una estación para ejecutar un proceso o la respuesta a un pedido por parte de un servidor. No es necesaria la utilización de confirmaciones dado que el mismo mensaje de respuesta sirve a la estación como confirmación de que el pedido había llegado y fue aceptado.

Las principales ventajas de esta aproximación son su simplicidad y eficiencia; la simplicidad se logra en forma inherente al no ser necesario el establecer una comunicación previa al envío de los paquetes y por ende tampoco es necesario “cancelar” esa comunicación al terminar el envío/recepción de los mismos.

Dada la mayor simplicidad en la forma de enviar los mensajes este protocolo reduciría la cantidad de capas necesarias solamente a tres:

- una capa física, la cual cumpliría el mismo rol que en los protocolos anteriormente vistos (OSI);
- una capa de enlace de datos encargada de hacer llegar los paquetes y de recibirlos; y
- una capa que podemos denominar de “pedido/respuesta” que sería la encargada de monitorear qué pedidos y qué respuestas son consideradas válidas.

Las capas de red, sesión, presentación y aplicación se vuelven prácticamente innecesarias.

Si bien mencionamos únicamente dos tipos de mensajes posibles, en realidad se usan más, en el cuadro siguiente están explicados los distintos mensajes que se utilizan junto con la Fig. 10.36 mostrando la comunicación entre un cliente y un servidor.

REQ	Request	Cliente	Servidor	El cliente solicita un servicio.
REP	Reply	Servidor	Cliente	Respuesta del servidor para un cliente.
ACK	Acknowledge	Ambos	Ambos	El paquete enviado ha sido recibido.
AYA	Are you Alive?	Cliente	Servidor	Prueba si el servidor sigue en funcionamiento.
IAA	I am alive	Servidor	Cliente	Indica que el servidor sigue en funcionamiento.
TA	Try Again	Servidor	Cliente	El servidor no tiene más espacio.
AU	Address Unknown	Servidor	Cliente	No se encuentra ningún proceso usando esa dirección.

Tabla 10.2 Tipos de mensajes utilizados en un protocolo cliente - servidor.

Ejemplos de comunicación entre clientes y servidores.

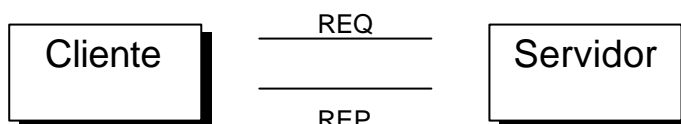


Fig. 10.36 a Comunicación simple entre Cliente y Servidor

En este caso vemos la combinación de mensajes más sencilla: un cliente envía un pedido a un servidor, el cual al haberlo cumplido envía los resultados de vuelta al cliente.

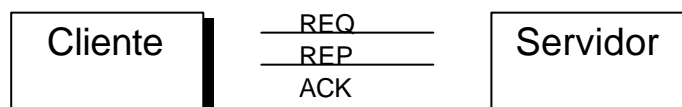


Fig. 10.33 b Comunicación entre Cliente y Servidor con reconocimiento específicos

Este ejemplo muestra al cliente enviando un mensaje al servidor por medio del cual indica que se ha recibido la respuesta en forma correcta, si el mensaje de confirmación no llega a destino el servidor puede asumir que la respuesta enviada no llegó correctamente por lo que la retransmite hasta recibir la confirmación.

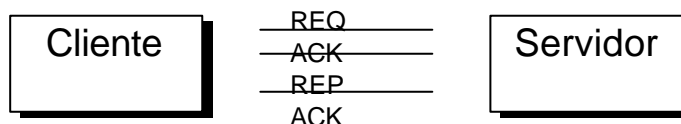


Fig. 10.36 c Comunicación entre Cliente y Servidor con reconocimientos específicos

Aquí la cantidad de mensajes enviados aumenta ya que el servidor confirma que ha recibido el pedido por parte del cliente con un mensaje particular (a diferencia de los primeros esquemas, donde se tomaba como confirmación la respuesta que enviaba el servidor).

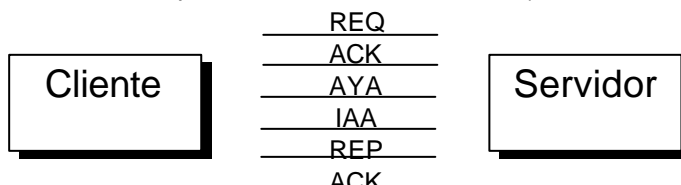


Fig. 10.36 d Comunicación entre Cliente y Servidor con verificaciones específicas

Este ejemplo puede ser visto como el del punto anterior, con la diferencia de que el cliente realiza un chequeo del servidor mediante el mensaje AYA, si el servidor puede recibir esto, envía a su vez un mensaje al cliente para que continúe esperando el resultado.

Este caso de clientes “impacientes” puede darse en el caso de servidores lentos o pedidos muy largos, de no recibir el cliente la confirmación por parte del servidor reintenta una cierta cantidad de veces antes de declarar a ese servidor como “inaccesible”.

Ampliamos este modelo por ser el mas popular dentro del procesamiento distribuido. Veamos con mas detalles el comportamiento de cada componente de este modelo:

Servidores (SERVERS):

Se obtiene un gran beneficio de una Red (especialmente LAN) es la de compartir los RECURSOS CAROS como ser:

- Almacenamientos
- Impresoras
- Modem
- Etc...

Surge la idea de SERVER. Las máquinas no necesitan tener el mismo S.O. o ser del mismo tipo. Un server es una Computadora dedicada que controla uno o más recursos a la cual se puede acceder mediante un Hardware y Software para interactuar con la Red (modelo OSI). La Aplicación puede Acceder directamente, por ejemplo ejecutando write, a los recursos. Para ello Red Lógica prepara los mensajes para Server quien devolverá el servicio.

Es una Arquitectura PODEROSA y FLEXIBLE:

PODEROSA por que:

- Puede controlar cualquier tipo de Recurso.
- Nuevos Servers se le pueden sumar en cualquier momento (+ Red Lógica de c/ Cliente).

FLEXIBILIDAD:

- No depende de un tipo de computadora *por que* existe una Red Lógica.

La principal Virtud de un SERVER es su TRANSPARENCIA para la aplicación. Se usan las mismas ordenes para leer, escribir, imprimir, comunicarse, etc. ya sea en forma Remota o en forma Local.

Características de los Servidores

El rol de una aplicación servidor es como su nombre lo indica, servir a múltiples clientes quienes tienen interés en los recursos compartidos poseídos por el servidor. Los servidores cumplen las siguientes características:

- **Protocolos Asimétricos** : Hay una relación de uno a muchos entre clientes y el servidor. El cliente siempre inicia el diálogo por requerimientos de servicios. El programa servidor gasta la mayoría de su tiempo esperando los requerimientos de los clientes, que en forma de mensaje, llegan a través de una sesión de comunicaciones. Algunos servidores asignan una sesión dedicada a cada cliente. Otros crean un pool dinámico de sesiones reusables. Otros también proveen una mezcla de los dos entornos. Por supuesto para que tenga éxito, el servidor debe responder a sus clientes y estar siempre preparado para soportar horas pico (rush hour traffic) donde se produce la mayor cantidad de requerimiento.
- **Ejecutar muchos requerimientos al mismo tiempo**: No es muy conveniente que los clientes sean atendidos por un programa servidor que soporte un solo hilo (single-threaded), dado que se corre el riesgo de un solo cliente retenga todos los recursos del sistema y produzca inanición a otros procesos clientes. El servidor debe estar capacitado para atender múltiples servicios en forma concurrente y a la vez proteger la integridad de los recursos compartidos.
- **Poner atención a los clientes "VIP"**: EL programa servidor debe ser capaz de proveer diferentes niveles y prioridad de servicios para sus clientes.
- **Iniciar y correr tareas en forma background**: El servidor debe ser capaz de correr tareas disparadas en background para ejecutar alguna actividad no relacionada con un programa principal en ejecución. Por ejemplo puede disparar la tarea de bajar registros de un servidor de base de datos fuera de las horas pico.
- **Siempre correr**: El programa servidor es típicamente una aplicación cuya misión es critica. Si el servidor se cae, esto impacta en todos los clientes que dependen de sus servicios. El programa servidor y el entorno en el cual corre deben ser muy robustos.
- **Transparencia de ubicación**: El servidor es un proceso el cual puede residir en la misma máquina que el cliente o en diferentes máquinas conectadas por red. Los software Cliente/Servidor

generalmente enmascaran la ubicación del servidor al cliente, redireccionando la llamada al servicio solicitado.

- **Intercambio basado en mensajes:** Los clientes y servidores son sistemas débilmente acoplados los cuales interactúan a través de mecanismos de pasaje de mensajes. El mensaje es el mecanismo de distribución de requerimientos y respuestas de servicios.
- **Encapsulación de servicios:** El servidor es un “especialista”. Un mensaje le dice al servidor que servicio fue solicitado, luego él determina de que manera concluirá el trabajo solicitado. Todos los pedidos se realiza a través una interfase de servicios. Los servicios que brinda el servidor pueden ser actualizado sin afectar a los clientes.
- **Escalabilidad:** los sistemas Cliente/Servidor pueden crecer horizontal o verticalmente. Horizontalmente significa que se pueden sumar o quitar terminales clientes con muy poco impacto en la performance. Verticalmente significa migrar a maquinas servidoras más grandes y rápidas o a multiservidores.
- **Integridad:** El código del servidor y los datos del servidor se mantienen en forma centralizada lo cual resulta en un mantenimiento más barato y el resguardo de los datos que se comparten. A la vez que los clientes permanecen como independientes.

Hay distintos tipos de servidores. Daremos algunos ejemplos:

Server de Impresión:

Se ocupa de los requerimientos de impresión de un número de Usuarios. Sus principales virtudes son:

- Ahorro de dinero.
- Más fácil
- Más Rápido.

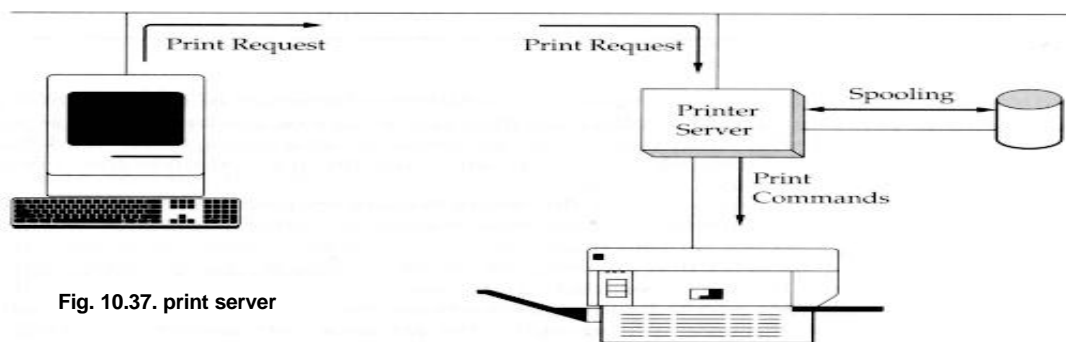


Fig. 10.37. print server

Usa SPOOLing que es una técnica que combina Hardware y Software para administrar la impresión simulando que siempre se dispone de impresora para la demanda de impresión que efectúan los procesos. Redirecciona los pedidos de I/O de la Impresora a un Disco. La secuencia es...

- 1) Llega un pedido de impresión.
- 2) Se graba documento o archivo en un disco manteniendo una cola FIFO.
- 3) Server toma los archivos de a uno y los manda a imprimir

SPOOL soluciona dos problemas:

- Si no alcanza el espacio en memoria para guardar un archivo muy grande, entonces el archivo es traído de a bloques desde el disco.
- Si la impresora está imprimiendo y hay varios pedidos, se los encola esperando a que la impresora se libere.

Server de comunicación:

El Server tiene conectados uno o más modems. Su trabajo consiste en:

- 1) El User hace un pedido de comunicación entregando el número de Teléfono
- 2) El Server activa el Modem, marca el número, y devuelve el resultado.
- 3) Si hubo éxito se le provee una Conexión al usuario.

Se puede Mandar y Recibir datos como si fuera Local

Server de Disco y Archivos:

Por el enorme crecimiento de las LANs y de puestos en las LANs se necesita cada vez mas espacio para almacenar datos, programas, backups, etc. Esto se resuelve mediante dos tipos de servidores; Uno para Archivos y otro para discos. El interés en File Servers radica en dos conceptos para ese conjunto de Archivos que se tienen que compartir.:

- Administra un recurso compartido.
- Soporta Aplicaciones del usuario.

Ofrece un servicio común a los otros sistemas en la LAN que puede ser, por ejemplo: Compartir Espacio de Almacenamiento, o ocuparse del Backup Automático y su Recuperación. Esta última tarea tiene las siguientes ventajas y desventajas:

- contra:
 - Falla del medio de Almacenamiento.
 - Errores del Usuario.
 - Virus.
- Tiene que ser Frecuente
- Recuperación tiene que poder usarse en cualquier momento.
- Movilidad del User
 - Que no necesite depender de un lugar físico.
 - Links a otros File Servers.

El término "File Server" es usado muy ampliamente:

- (Dos extremos): 1) Almacenamiento Compartido.
 2) File System compartido.

1) Almacenamiento Compartido: (Disk Server)

- Usuario tiene más espacio.
- Usuario lo usa para Escribir y Leer Archivos.
- No hay control del File Sharing.

2) Servers con File Server:

- Usuario tiene más espacio.
- Server controla Acceso Concurrente a los Archivos.
- Marca los derechos y las restricciones de los Accesos.

Consistencia del CACHE:

Existe cierta degradación de los tiempos de I/O por la transferencia en la Red. Esto se resuelve mediante el uso de CACHEs locales en los Sistemas de la Red. En el Caché se guarda los últimos Archivos accedidos. Cuando un proceso quiere acceder a un Archivo, busca:

- 1^{ro} En su Caché.
- 2^{do} En el disco local.
- 3^{ro} En el Caché del Server.
- 4^{to} En el disco del Server.

Definimos a un Caché Consistente, cuando el CACHE tiene una copia exacta de los datos remotos.

Se da Inconsistencia cuando los datos remotos son cambiados y la copia en el Caché no se actualizaron inmediatamente.

Hay otros tipos de servidores. Por ejemplo:

- Servidores de Mail: Almacenan y remiten los mensajes de e-mail correspondiente a los usuarios.
 Dos tipos determinados de servidores de correo encontrados en Internet son
 - Servidores POP (reciben y retienen los e-mails para los clientes)
 - Servidores SMTP (acepta e-mails de clientes y lo remite por la red)
- Servidores de base de datos: Estos extraen datos de una base de datos según el requerimiento del cliente.
- Servidores Web: Proporciona paginaciones Web para los cliente que navegan usando el protocolo http.

Servicios del sistema operativo para un servidor.

En un entorno de computación distribuida el sistema operativo debe brindar funciones básicas y extendidas. No hay una regla estricta que determine lo que corresponde a los servicios básicos y los extendidos.

Los servidores requirieren un alto nivel de concurrencia y su código correrá mas eficientemente si las tareas son asignadas como parte del mismo proceso en lugar de programas separados (estas tareas son conocidas como corutinas o hilos). Los hilos del mismo proceso son mas rápidos para crearlos, mas rápidos para hacer cambio de contexto y tienen un acceso mas fácil a la información compartida. A continuación se enumeran los servicios básicos que un servidor necesita:

- **Programación de tareas:** Es mucho mas fácil y seguro escribir programas servidores multitareas en entornos donde el sistema operativo administre automáticamente el cambio de contexto entre tareas asignando a cada una su tiempo de ejecución.
- **Prioridad de tareas:** El sistema operativo debe atender tareas en base a prioridades. Esto ultimo permite a los servidores diferenciar el nivel de servicios en base a la prioridad del cliente.
- **Semáforos:** El sistema operativo debe proveer mecanismos de sincronización para mantener un orden cronológico cuando se ejecutan tareas concurrentes que comparten recursos.
- **Comunicación entre procesos (IPC):** EL sistema operativo debe proveer los mecanismos que permita el intercambio de datos entre procesos independientes.
- **Hilos:** Estos son unidades de concurrencia provistos dentro del programa en sí. Los hilos son usados para crear programas servidores muy concurrentes orientado a eventos. Cada evento espera ser asignado a un hilo que esta bloqueado hasta que el evento ocurre, mientras otro hilo puede ocupar la CPU para ejecutar su trabajo.
- **Protección entre tareas:** El sistema operativo debe proteger que las tareas no se interfieran mutuamente ni interfieran a los recursos que las tareas tienen asignadas. Ninguna tarea debe ser capaz de colgar todo el sistema. La protección se extiende al sistema de archivos y las llamadas al sistema operativo.
- **Sistema de Archivo Multiusuario de Alta Performance:** El sistema de archivos debe soportar múltiples tareas y proveer la "cerradura" para proteger la integridad de los datos. Los programas servidores trabajan generalmente con muchos archivos a la vez. El sistema de archivos debe soportar una gran cantidad de archivos abiertos sin producir un deterioro en la performance del sistema.
- **Administración de Memoria Eficiente:** El sistema de memoria debe soportar eficientemente muchos programas y gran cantidad de objetos de datos. El sistema operativo debe proveer los mecanismos necesarios para swapear desde o hacia el disco bloques de programas.
- **Extensiones de Enlace Dinámico:** Los servicios del sistema operativo deben ser extensibles. Se debe proveer un mecanismo que permita agregar nuevos servicios en tiempo de ejecución sin tener que recompilar el sistema operativo.

Servicios Extendidos

Los servicios extendidos deben proveer el software avanzado que explotará la potencialidad de la red permitiendo el acceso flexible a la información compartida y hace que el sistema sea mas fácil de administrar y mantener. A continuación se enumeran los servicios extendidos que un servidor espera del sistema operativo

- **Comunicaciones:** Entre los servicios extendidos que un sistema operativo debe proveer se encuentra un conjunto de protocolos de comunicación que permitan al servidor comunicarse con un gran numero de plataformas cliente. Además el servidor debe ser capaz de comunicarse con otros servidores en caso de necesitar asistencia para proveer un servicio.
- **Extensiones de Red:** En cuanto a las extensiones de red, el sistema operativo debe proveer facilidades para extender por ej.: los servicios de archivos e impresión por toda la red. Idealmente las aplicaciones deben ser capaz de acceder en forma transparente a cualquier dispositivo remoto como si lo hiciera localmente.
- **Servicios de transacciones y base de datos:** Con esta extensión de servicios el sistema operativo debe proveer un robusto sistema de administración de base de datos multiusuario (DBMS). Este

DBMS debe permitir la ejecución de SQL⁵ para el soporte de decisiones y procedimientos almacenados en el servidor usado en transacciones. Los procedimientos almacenados son creado por programadores pero fuera del sistema operativo. La funciones mas avanzadas incluye un Monitor de Transacciones (TM) para administrar los procedimientos almacenados (o las transacciones) como una unidad atómica de trabajo que se ejecuta en uno o mas servidores.

- **Servicios de Autenticación y Autorización:** El sistema operativo debe proveer la herramienta que los clientes se puedan identificar ante los servidores. Los sistemas de autorización determinan si el cliente tiene permiso para obtener un servicio remoto.
- **Administración de Sistema:** El sistema operativo debe brindar una plataforma integrada de administración de sistemas y de red. El sistema debe ser administrado como un servidor simple o como múltiples servidores asignados a dominios. La administración de sistema incluye herramientas de configuración, funciones para monitorear la performance de todos los elementos del sistema generar alertas cuando se producen conflictos, herramientas para evitar virus o intrusos.

Características de los Clientes

Todas las aplicaciones cliente tienen algo en común, requieren los servicios del servidor. Podemos distinguir tres categorías de clientes:

Clientes No-GUI:

Este tipo de aplicación cliente genera requerimientos al servidor con una mínima interacción de los humanos. Entre los clientes no-GUI se distinguen las siguientes subcategorías:

- Clientes no GUI que no necesitan multitarea: Incluyen maquinas automáticas, equipos fax, terminales inteligentes.
- Clientes no GUI que necesitan multitarea: Ejemplos robots, equipo de prueba, programas deamon. Estos clientes requieren servicios de multitarea muy granular dirigido a eventos y de tiempo real.

Clientes GUI (Graphic User Interface) :

Son aplicaciones donde los requerimientos al servidor se llevan a cabo por la interacción de un humano a través de una interfase gráfica. Un ejemplo de este tipo de aplicaciones clientes son las aplicaciones comerciales del tipo OLTP (On Line Transaction Processing) que requieren un alto volumen de datos y generalmente implica tareas repetitivas.

Clientes con interfase orientada a objetos OOUI (Object Oriented User Interface):

Estos clientes presentan una interfase de usuario orientada a objeto que provee un acceso a la información en forma visual donde los requerimientos al servidor generalmente no son repetitivos.

Los clientes OOUI tienen un insaciable apetito de comunicación, Los objetos pertenecientes a estos clientes necesitan comunicarse entre ellos y con los servidores externos. Las comunicaciones son, por necesidad, de tiempo real, interactivos y altamente concurrente.

Servicios del sistema operativo para el Cliente.

Los tres tipos de clientes descriptos tienen distintos requerimientos de su sistema operativo.

Requerimientos del SO	Cliente noGUI Sin multitarea	Cliente noGUI Con multitarea	Cliente GUI simple	Cliente OOUI
Mecanismos de pedido/respuesta (pref. con transparencia local/remoto)	Si	Si	Si	Si
Mecanismos para transferencia de archivos, registros de base de datos, textos, etc.	Si	Si	Si	Si
Programación multitarea	No	Si	Deseable	Si
Prioridad de tareas	No	Si	Deseable	Si
Comunicación entre procesos	No	Si	Deseable	Si
Hilos para la comunicación background con el servidor	No	Si	Si	Si
Robustez del SO incluyendo protección entre tareas y llamadas al SO reentrantes	No	Si	Deseable	Si
GUI avanzado con diálogos interactivos drag and drop, soporte multimedia	No	No	Si	Si

Tabla 10.3. Requerimientos de los clientes al S.O.

Todos las aplicaciones clientes necesitan algún tipo de mecanismo que permita requerir servicios al servidor. Las tres categorías trabajan mejor en un entorno multitarea robusto. Los clientes GUI y OOUI trabajan mejor con mecanismos de hilos para el manejo de requerimientos background. Usando hilos

⁵ SQL significa Structured Query Language o sea Lenguaje estructurado de consulta.

separados para la interfase de usuario y el procesamiento background, el programa puede responder a las entradas de usuarios mientras otro hilo se ocupa de la interacción con el servidor. A continuación se listan las necesidades de los clientes.

Clientes “gordos” o Servidores “gordos”

Las aplicaciones cliente /servidor pueden ser diferenciadas por como se distribuyen la aplicación entre cliente y servidor.

El modelo de servidor gordo establece la mayor parte de la aplicación corriendo en el servidor. Mientras que el modelo de cliente gordo es a la inversa. Como ejemplo de servidor gordo podemos citar un servidor de transacciones. Como ejemplo de cliente gordo podemos citar aquellos que solicitan servicios de servidores de archivos o de base de datos.

La forma mas tradicional es la de clientes gordos, donde el grueso de la aplicación corre en el cliente cuyas aplicaciones son usados para soporte en la toma de decisiones o para software personal, mientras que el servidor se dedica a la organización y almacenamiento de datos. El uso de clientes gordos está muy relacionada con un tipo de arquitectura que recibe el nombre de arquitectura Two-tier.

El modelo de servidores gordos es la mas fácil de administrar y desplegar en la red porque la mayoría del código corre en el servidor. Los servidores tratan de minimizar el intercambio en la red creando servicios de niveles abstractos. Los servidores de transacciones, por ejemplo, encapsulan la base de datos. En lugar de exportar simplemente datos, exportan los procedimientos (o los métodos en terminología orientada a objeto) que operan sobre los datos. El cliente en un modelo de servidor gordo provee la interfase GUI e interactúa con el servidor a través de RPC⁶.

Cada modelo cliente/servidor tiene su uso. En muchos casos, los modelos se complementan uno a otro y no es inusual tenerlos coexistiendo en una aplicación. Por ejemplo un software de gestión podría requerir todo en un servidor que combine servicios de archivos, base de datos, transacciones y de aplicación.

Tipos de Arquitecturas Cliente/Servidor

Aunque las arquitecturas cliente/servidor pueden llegar a ser muy complejas, se distinguen dos arquitecturas básicas para el desarrollo de aplicaciones. Arquitecturas Two-Tier y Arquitecturas Three-Tier. La elección de alguna de ellas dependerá del alcance, la complejidad, el tiempo disponible para completar un proyecto, el valor y la obsolescencia del sistema.

Arquitectura Two-Tier (dos-nexos)

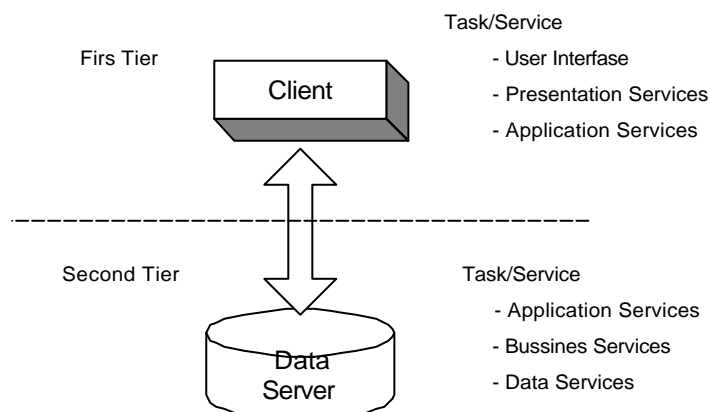


Fig. 10.38 Arquitectura Two-tier

La arquitectura two-tier fue desarrollada en los 80 basado en el diseño de servidores de archivos. Esta arquitectura cliente/servidor es una buena solución de computación distribuida cuando el grupo de trabajo no excede los 100 usuarios intercalando a través de una red LAN. La arquitectura two-tier consta de tres componentes principales distribuido en dos capas donde el cliente requiere servicios y el servidor los provee, los componente son:

- **Una Interfase de Usuario del Sistema:** Tal como servicios de administración de sesión, entrada de texto o dialogo.
- **Administración de Procesamiento:** Tal como desarrollo, puesta en marcha y monitoreo de procesos.

⁶ RPC significa Remot Procedure Call que es invocar la ejecución de un procedimiento en forma remota.

- **Administración de Datos:** Tales como servicios de archivos y base de datos

El diseño de two-tier asigna la interfase de usuario del sistema exclusivamente al cliente y ubica la administración de datos en el servidor y reparte la administración de procesamiento entre el cliente y el servidor creando esas dos capas que se representan en las arquitecturas two-tier.

Las arquitecturas two-tier se recomiendan exclusivamente para procesamiento de datos que no implica tiempo crítico de proceso, donde las operaciones del sistema no son muy complejas. Por ejemplo sistemas de soporte de decisión donde la carga de transacciones es liviana. Este tipo de arquitectura funciona bien en entornos relativamente heterogéneos con reglas de procesamiento (business rules) que no cambian frecuentemente. Generalmente en este tipo de arquitectura la mayor parte de procesamiento la lleva a cabo en la porción de aplicación cliente. Mientras que la aplicación servidor se dedica a lo relacionado con la administración y acceso a datos

Ventajas de las arquitecturas two-tier (dos-nexos)

Históricamente, el caso de arquitecturas two-tier se basa en la simplicidad. El middleware y las herramientas de programación del servidor (store procedures), si los hubiese, deberían ser comprados de un solo vendedor (ej. el proveedor DBMS⁷). En contraste a las arquitecturas three-tier donde las herramientas de programación, el middleware y los DBMS son ofrecidos por distintos vendedores. De esa manera el desarrollo de arquitecturas two-tier requiere menos complejidad en el desarrollo de las funciones que se deben poner en el servidor y en el cliente.

Desventaja de las arquitecturas two-tier (dos-nexos)

Escalabilidad: La arquitectura two-tier solo puede crecer hasta no mas de 100 usuarios en la red. Mas allá de este numero la performance del sistema se ve afectada, esto es porque en esta arquitectura el cliente y el servidor intercambian mensajes continuamente ("keep-alive"), aun cuando no se esta realizando ningún trabajo, lo que origina la saturación de la red.

Implementar procedimientos almacenados (store procedures) o reglas de negocios puede limitar la escalabilidad porque cuanto mas reglas o procedimientos son instalados en los servidores mayor es el poder de procesamiento necesitado. Consecuencia de ello se reduce el numero de usuarios que pueden ser soportados.

Interoperabilidad: La arquitectura two-tier limita su interoperabilidad cuando usa procedimientos almacenados para implementar lógica de procesamiento compleja (como el manejo de integridad en una base de datos distribuida) porque los procedimientos almacenados son normalmente implementados usando un lenguaje propio del motor de base de datos. Esto significa que intercambiar o interoperar con mas de un tipo de motor de base de datos necesita del desarrollo de aplicaciones que oculten las diferencias.

Administración y configuración del sistema: Las arquitecturas two-tier pueden ser difícil de administrar y mantener porque cuando las aplicaciones residen en el cliente, los paquetes de actualización deben ser llevados, instalados y testeados en cada cliente. La falta de uniformidad en las configuraciones del cliente y la falta de controles sobre las siguientes cambios en las actualizaciones incrementa la sobrecarga de administración.

Trabajos Batch: Las arquitecturas two-tier no es efectiva para correr programas batch. El cliente esta parado hasta que el trabajo batch finaliza, aun si el trabajo batch se ejecuta en el servidor.

Arquitecturas Three-Tier (Tres-Nexos)

Estas arquitecturas aparecidas en los años 90 emergieron para remediar las limitaciones de las arquitecturas two-tier. En arquitectura three-tier existe un tercer nexo (tier) entre el cliente y los componentes de manejo de datos del servidor. Este nexo medio provee administración de procesos donde se emplean reglas y lógicas de negocio y se puede soportar varios cientos de clientes (a diferencia de Two-tier que soporta no mas de 100). Las arquitecturas three-tier se usan cuando se necesita un diseño cliente/servidor efectivo, incrementando performance, flexibilidad, mantenimiento, reusabilidad y escalabilidad, ocultando la complejidad del procesamiento.

⁷ DBMS significa Data Base Management System.

Hay una gran variedad de formas de implementar este nexo medio, tales como Monitores de Procesamiento de Transacciones (TP), Servidores de mensajes o Servidores de Aplicaciones. El nexo medio puede ejecutar: encolado (queuing), ejecución de aplicación (application execution), y uso de base de datos (database staging). Por ejemplo, si el nexo medio provee encolado, el cliente puede despachar su requerimiento a la capa media y deshacerse porque el nexo medio accederá a los datos y retornará la respuesta al cliente. Además, la capa media ofrece la programación (scheduling) y priorización de las tareas en progreso. La flexibilidad en el particionamiento puede ser tan fácil como un “drag and drop” de módulos de aplicaciones en diferentes computadoras con una arquitectura three-tier similar.

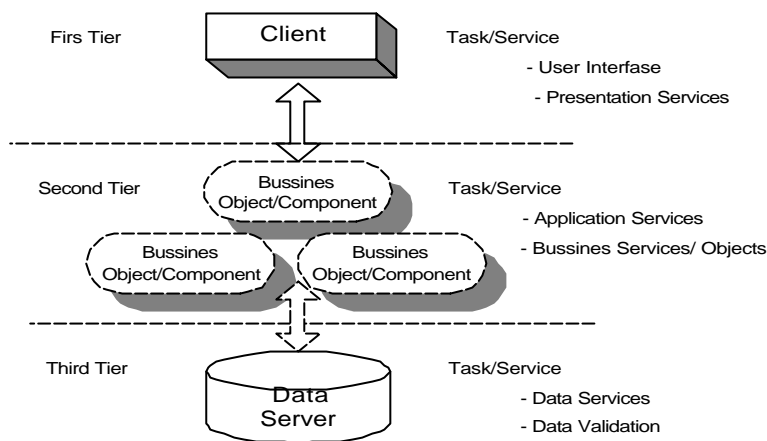


Fig. 10.310. Arquitectura Three-tier

Arquitecturas Three-tier con tecnología de Monitor de procesamiento de Transacciones (TP).

El tipo más básico de arquitectura three-tier contiene una capa media que consiste en un Monitor de Procesamiento de Transacciones (TP) que provee el encolado de mensajes (message queuing) programación de transacciones (transaction scheduling), y servicios de priorización (prioritization service) donde el cliente se conecta a este monitor en lugar de el servidor de datos. La transacción es aceptada por el monitor, el cual la encola y toma la responsabilidad de administrar su ejecución hasta que esté terminada, liberando del todo al cliente. Cuando la capacidad es provista por vendedores de middleware se refiere a “TP Pesados” porque puede servir a una gran cantidad de clientes.

Los Monitores TP pueden proveer además:

- La habilidad para actualizar diferentes y múltiples DBMS en una sola transacción
- Conectividad a una variedad de fuente de datos incluyendo archivo planos, DBMS no relacionales.
- La habilidad para asignar prioridad a las transacciones.

La desventaja de los monitores TP es que aún no se crearon herramientas visuales para su programación por lo cual son escritos en lenguajes de bajo nivel.

Arquitecturas Three-Tier y Servidor de Mensajes.

La mensajería es otra manera de implementar arquitecturas three-tier. Los mensajes son priorizados y procesados asincrónicamente. Los mensajes constan de un encabezado que contiene información de prioridad y la dirección y número identificador. El servidor de mensajes se conecta a una base de datos relacional o cualquier otra fuente de datos. La diferencia entre la tecnología de un Monitor TP y un servidor de mensajes es que éste último se enfoca en la inteligencia de los mensajes mientras que el Monitor TP tiene la inteligencia en el Monitor y trata a las transacciones como paquetes de datos bobos.

Arquitecturas Three-Tier en Servidores de Aplicaciones.

Los servidores de aplicaciones three-tier asignan de forma repartida el cuerpo de la aplicación lógica de manera que corra parte en el cliente y parte en el servidor. El servidor de aplicaciones no es el encargado de proveer una interfase gráfica (GUI) pero tiene a su cargo la lógica de negocios, un motor para recuperar datos. Diseñar servidores de aplicaciones es bueno cuando la seguridad, escalabilidad y los costos son las mayores consideraciones.

Arquitecturas Three-Tier en ORB

Actualmente la industria está trabajando en el desarrollo de estándares para mejorar la interoperabilidad y determinar cual será el común de ORB (Object Request Broker). El desarrollo de

sistemas cliente/servidor usando tecnologías que soporten objetos distribuidos promete grandes mejoras si se logra soportar interoperabilidad entre distintos lenguajes y plataformas con total transparencia estas tecnologías se las conoce como

- Common Object Request Broker Architecture (CORBA)
- COM/DCOM

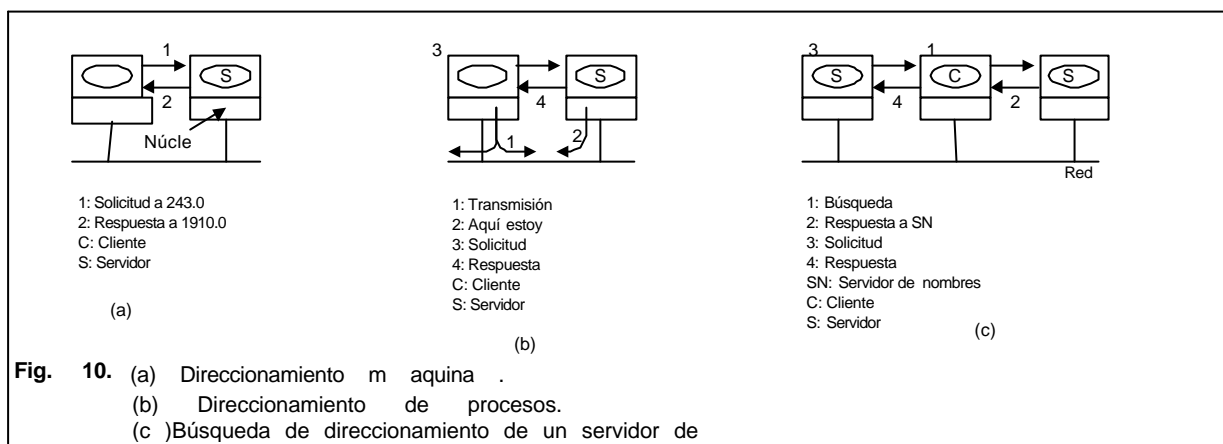
Ventajas de Arquitecturas Three-Tier

La principal ventaja de las arquitecturas three-tier se basa en la flexibilidad que proveen los productos de comunicación de propósitos generales o productos middleware (ej. Sockets, SPI-C, RPCs, monitores TP, sistema de mensajería) comparado a los procedimientos almacenados (Stored Procedures). Un servidor de aplicaciones three-tier corre bajo el control del Sistema Operativo o en un entorno del Middleware y de esa manera son mas poderosos que los servidores de aplicaciones two-tier que corre limitado por los procedimientos almacenados. Los Procedimientos almacenados y las facilidades de DBMS que los ejecutan en tiempo de corrida, mejoraron en los últimos años pero aún son menos flexibles que algunos productos middleware en ciertos punto claves.

Desventajas de Arquitecturas Three-Tier

La principal desventaja de estas arquitecturas es que el entorno de desarrollo es mucho mas difícil de usar que las herramientas de desarrollo en entorno visual.

Direccionamientos en el modelo cliente servidor



Para que el cliente pueda enviar un mensaje a un servidor, debe conocer la dirección de éste. Si sólo existe un proceso en la máquina destino, el núcleo sabrá que hacer con el mensaje recibido. Sin embargo, si existen varios procesos en una máquina, el núcleo no sabrá a quién darle el mensaje. Por lo tanto, existen distintas maneras para el direccionamiento de los procesos, a saber: (ver, Fig. 10.40).

- Direccionamiento máquina.proceso
- Direccionamiento de procesos con transmisión
- Direccionamiento por medio de un servidor de nombres

Direccionamiento máquina.proceso

En este sistema se envía mensajes a los procesos en vez de a las máquinas. Aunque este método elimina la ambigüedad acerca de quién es el verdadero receptor, presenta el problema de cómo identificar los procesos. Una manera consiste en utilizar nombres de dos partes para especificar tanto la máquina como el proceso. Por ejemplo 240.1 identifica el proceso 1 de la máquina 240.

El núcleo utiliza el número de máquina para que el mensaje sea entregado de manera correcta a la máquina adecuada, a la vez que utiliza el número de proceso en esa máquina para determinar a cuál proceso va dirigido el mensaje. Cada máquina puede numerar sus procesos a partir de 0. No se necesitan coordenadas globales, puesto que nunca existe confusión entre el proceso 0 de la máquina 243 y el proceso 0 de la máquina 1910. El primero será 243.0 y el segundo será 1910.0.

Direccionamiento de procesos con transmisión

En este método de direccionamiento, cada proceso elige su propio identificador de un espacio de direcciones grande y disperso. La probabilidad de que dos procesos elijan el mismo número es muy pequeña y el método puede utilizarse en sistemas más grandes. Sin embargo, existe el núcleo no sabe a que máquina enviar el mensaje. En una LAN que soporte transmisiones, el emisor puede transmitir un a

paquete especial de localización con la dirección del proceso destino. Puesto que es un paquete de transmisión, será recibido por todas las máquinas de la red. Todos los núcleos verifican si la dirección es la suya y, en caso que lo sea, regresa un mensaje con su dirección en la red (número de máquina). El núcleo emisor utiliza entonces esa dirección y la captura, para evitar el envío de otra transmisión la próxima vez que necesite el servidor.

Aunque este método es eficiente, la transmisión provoca una carga adicional en el sistema.

Direcccionamiento por medio de un servidor de nombres

La carga adicional del método anterior se puede evitar mediante una máquina adicional para la asociación a alto nivel de los nombres de servicios con las direcciones de la máquina.

Al utilizar este sistema, se hace referencia a los procesos del tipo de los servidores mediante cadenas en ASCII, las cuales son las que se introducen en los programas y no los números en binario de las máquinas o los procesos. Cada vez que se ejecute un cliente, en su primer intento por utilizar el servidor, el cliente envía una solicitud de cuestionamiento a un servidor especial de asociaciones, el cual se conoce como servidor de nombres, para pedirle 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.

Cada uno de estos métodos tiene problemas. El primero no es transparente; el segundo genera una carga adicional al sistema y el tercero necesita un componente centralizado, el servidor de nombres.

10.4.2. Modelo de estación de trabajo

El sistema consta de estaciones de trabajo, que pueden estar dispersas en un EDIFICIO o un pequeño área y conectadas entre sí por medio de una LAN de alta velocidad. Algunas pueden estar en oficinas con lo que, de manera implícita, cada una de ellas se dedica a un único usuario. En ambos casos, en un instante dado, una estación de trabajo puede tener un único usuario conectado a ella y tener entonces un solo dueño.

En ciertos sistemas, las estaciones de trabajo pueden tener discos locales y en otras no.

Si las estaciones de trabajo 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, se envían a un servidor de archivos, el cual realiza el trabajo y envía de regreso los resultados como respuestas.

Las estaciones de trabajo sin discos, proporcionan simetría y flexibilidad. Un usuario puede utilizar cualquier estación de trabajo y entrar al sistema. Puesto que todos sus archivos están dentro del servidor de archivos, una estación de trabajo sin disco, es tan buena como cualquier otra.

Por el contrario, si todos los archivos se almacenan en archivos locales, el uso de la estación de trabajo de otra persona, implica que se tendrá un fácil acceso a los archivos de esa estación, pero el uso de los propios requerirá de un esfuerzo adicional y claramente distinto.

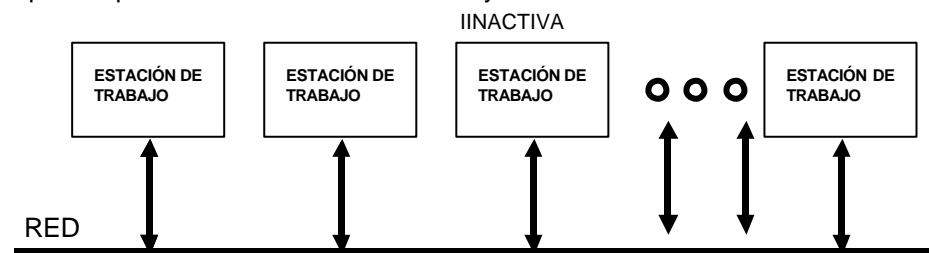


Fig. 10.41 Una red de estaciones de trabajo personal, cada una con un sistema local de archivos. Si la estación de trabajo tiene sus propios discos, estos se pueden utilizar al menos de 4 maneras:

- *Paginación y archivos temporales*
- *Paginación, archivos temporales y binarios del sistema*
- *Paginación, archivos temporales, binarios del sistema y ocultamiento de archivos*
- *Un sistema local de archivos completo*

El primer diseño se basa en que es conveniente mantener los archivos del usuario en los servidores centrales de archivos, también se necesitan discos para la paginación y los archivos temporales. En este modelo, los discos locales se utilizan exclusivamente para la paginación y los archivos temporales no compartidos que se pueden eliminar al final de la sesión.

El segundo modelo, es una variante del primero. Los discos locales también contienen los programas en binario (ejecutables). 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. Sin embargo, si ocurre que la máquina no funciona cuando se envía el programa, lo perderá y continuará ejecutando la versión anterior. Por ello se necesita cierta administración para mantener un registro de las versiones de cada programa.

Un tercer método, consiste en utilizar de forma explícita los discos locales como Cachés, además de utilizarlos para la paginación. En este modo de operación, los usuarios pueden cargar archivos desde los servidores de archivos hasta sus propios discos, y leerlos y escriben en ellos de manera local y después regresar los archivos modificados al final de la sesión. El objetivo de esta arquitectura es mantener centralizado el almacenamiento a largo plazo, pero reducir la carga en la red al mantener los archivos en forma local mientras se utilizan.

Una desventaja es poder mantener consistentes los Caché.

En el cuarto método, cada máquina puede tener su propio sistema de archivos auto-contenidos, con la posibilidad de montar o tener acceso a los sistemas de archivos de otras máquinas. Esta organización proporciona un tiempo de respuesta uniforme, y garantizado para el usuario y pone poca carga en la red. La desventaja es que es difícil compartir algunos aspectos y el sistema resultante se parece mas a un sistema operativo de red que a un verdadero sistema operativo distribuido y transparente.

Uso del disco	Ventajas	Desventajas
(Sin disco)	Bajo costo, fácil mantenimiento del hardware y el software, simetría y flexibilidad	Gran uso de la red; los servidores de archivos se pueden convertir en cuellos de botella
Paginación, archivos de tipo borrador	Reduce la carga de la red comparado con el caso sin discos	Un costo alto debido al gran número de discos necesarios
Paginación, archivos de tipo borrador, binarios	Reduce todavía más la carga sobre la red	Alto costo; problemas de consistencia del caché
Paginación, archivos de tipo borrador, binarios, ocultamiento de archivos	Una carga aun menor en la red; también reduce la carga en los servidores de archivos	Alto costo; problemas de consistencia del caché
Sistema local de archivos completo	Escasa carga en la red; elimina la necesidad de los servidores de archivos	Pérdida de transparencia

Tabla 10.4 Uso de los discos en las estaciones de trabajo

Uso de estaciones de trabajo inactivas

El primer intento por permitir el uso de las estaciones de trabajo inactivas, fue el programa rsh proporcionado junto con el UNIX de Berkeley. Este programa se llama con el comando:

rsh machine command

en donde el primer argumento es el nombre de una máquina y el segundo, un comando para ejecutarse en ella. Lo que hace **RSH** es ejecutar el comando específico en la máquina dada. Aunque se utiliza ampliamente, este programa tiene serios problemas. El primero de ellos es que, el usuario, debe indicar la máquina que desea utilizar, lo que lo coloca al mismo en el predicamento de mantener un registro de las máquinas inactivas.

El segundo problema, es que el programa se ejecuta en el ambiente de la máquina remota el cual es, distinto del ambiente local. Por último, si alguien debe entrar a una máquina inactiva, que ejecuta un proceso remoto, el proceso continua su ejecución y el nuevo usuario debe aceptar el desempeño menor o encontrar otra máquina.

¿Cómo encontrar una estación de trabajo inactiva ?

Una estación donde ninguna persona esta conectada puede ejecutar docenas de procesos.

Por otro lado, un usuario que entra al sistema por la mañana, pero que después no toque la computadora durante horas, no coloca una carga adicional en dicho sistema. Los distintos sistemas, toman diversas decisiones a cerca del significado de la palabra INACTIVO, pero por lo general, se dice que la estación de trabajo esta inactiva cuando nadie toca el teclado o el mouse durante varios minutos y no se ejecuta algún proceso iniciado por el usuario.

Los algoritmos que se utilizan para localizar las estaciones de trabajo inactivas, se pueden dividir en dos categorías:

- Controladas por el servidor
- Controlada por el cliente

En la primera categoría, cuando una estación de trabajo esta inactiva y por lo tanto se convierte en un servidor potencial, anuncia su disponibilidad. Puede hacer esto al proporcionar su nombre, dirección en la red y propiedades en un archivo.

Otra alternativa, consiste en que la estación inactiva anuncia el hecho de que no tiene trabajo, al colocar un mensaje que se transmite en toda la red. Todas las demás estaciones registran esto.

La ventaja de esto, es un menor costo en la búsqueda de una estación de trabajo y una mayor redundancia. La desventaja es pedir a todas las máquinas que se encarguen de mantener el registro.

Existe un peligro potencial de que aparezcan condiciones de competencia. Si dos usuarios llaman al mismo tiempo al comando REMOTE y ambos descubren que la misma máquina está inactiva, ambos

intentarán iniciar procesos al mismo tiempo. Para detectar y evitar esta situación, el programa REMOTE verifica la estación de trabajo inactiva, la cual, si continua libre, se elimina a sí mismo del registro y da la señal de continuar.

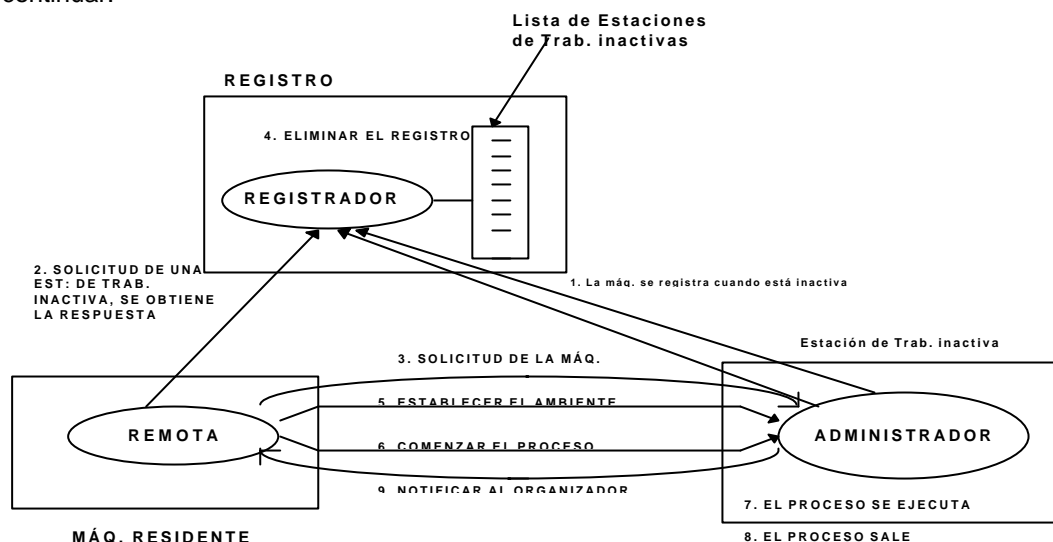


Fig. 10.42 Un algoritmo basado en registros para la búsqueda y uso de las estaciones de trabajo inactivas

La otra forma, utiliza un método controlado por el cliente. Al llamar a REMOTE, transmite una solicitud donde indica el programa que desea ejecutar, la cantidad de memoria, si requiere o no un chip coprocesador de punto flotante. Al regresar la respuesta, REMOTE elige una de ellas y la configura.

La búsqueda de la estación de trabajo, es solo el primer caso. Ahora hay que ejecutar el proceso allí. El truco consiste en configurar el proceso remoto de modo que vea el mismo ambiente que tendría en el caso local (estación de trabajo de origen) y llevar a cabo el computo de la misma forma que en el caso local.

Necesita la misma visión del sistema de archivos, el mismo directorio de trabajo y las mismas variables del ambiente (SHELL). después de configurar esto, el programa puede iniciar su ejecución. El problema comienza cuando se ejecuta la primera llamada al sistema. El núcleo, si el sistema no tiene disco y todos los archivos se localizan en el servidor de archivos, puede enviar simplemente la solicitud al servidor apropiado, que es lo mismo que hubiera hecho la máquina de origen si el proceso se hubiese ejecutado en ella. Si el sistema tuviera discos locales, cada uno con su propio sistema de archivos, entonces hay que dirigir la solicitud de regreso a la máquina de origen para su ejecución.

Una última pregunta que podemos originar, es, ¿Que hacer si regresa el poseedor de la máquina (es decir, alguien entra al sistema o un usuario previamente inactivo toca el teclado o el ratón)? Lo mas fácil es no hacer nada.

Si otras personas intentan ejecutar programas en su estación de trabajo al mismo tiempo en que se desea utilizarla, se diluye la respuesta garantizada.

Otra posibilidad, es eliminar el proceso intruso. La forma mas sencilla de lograr esto es hacerlo de manera abrupta y sin previo aviso. La desventaja de esta estrategia es que se perderá todo el trabajo y el sistema de archivos tendrá un estado caótico. Es mejor darle al proceso una advertencia, mediante una señal que permita detectar una catástrofe inminente y hacer esto con suavidad. Si no sale después de unos cuantos segundos, termina.

Un método completamente distinto, es hacer que el proceso emigre a otra máquina, ya sea la de origen o alguna estación de trabajo inactiva.

En ambos casos, al irse el proceso, debe dejar la máquina en el mismo estado en que la encontró, para evitar molestar al dueño. Entre otros aspectos, el proceso no solo debe irse, sino también sus hijos y los hijos de estos. Además, hay que eliminar los buzones, conexiones en la red y otras estructuras de datos, relativas a todo el sistema.

Si existiera un disco local, hay que eliminar los archivos temporales y, de ser posible, restaurar los archivos que hayan sido eliminados de su Caché.

10.4.3. El modelo de la Pila de procesadores

Que ocurre cuando es posible proporcionar de 10 a 100 veces mas CPU que el número de usuarios activos . Una solución consiste en dar a cada uno un multiprocesador. Sin embargo, este es un diseño ineficiente.

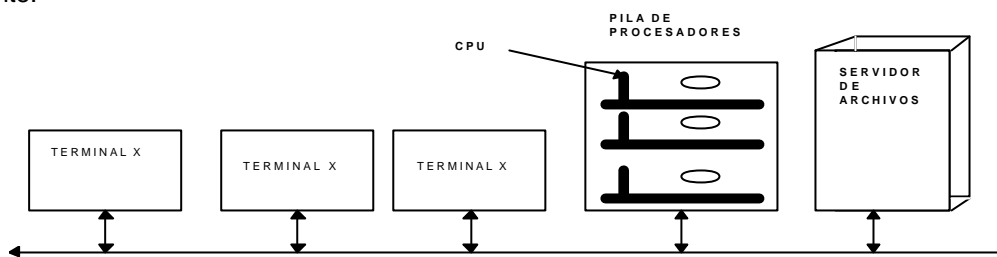


Fig. 10.43 Sistema basado en el modelo de la pila de procesadores

Otro método, consiste en construir una pila de procesadores, repleta de CPUs, en una sala de servidores o máquinas, las cuales se pueden asignar de manera dinámica a los usuarios según la demanda. A las estaciones de los usuarios, se les dan terminales gráficas de alto rendimiento, como los terminales X. Esta idea se basa en la observación de que lo que realmente desean muchos usuarios es una interfase gráfica de alta calidad y buen desempeño. Este método es mucho mas parecido al tiempo compartido tradicional que al modelo de la computadora personal, aunque se construye con microprocesadores de bajo costo. La motivación para la idea de la pila de procesadores, proviene de llevar un paso mas adelante la idea de las estaciones trabajo sin disco. Si el sistema de archivos se debe concentrar en un pequeño número de servidores de archivos, para economizar la escala, debe ser posible hacer lo mismo para los servidores de computadoras.

Si colocamos todas las CPUs en un gabinete de gran tamaño dentro de la sala de servidores o de máquinas, se pueden reducir los costos de suministro de energía y otros costos de empaquetamiento, lo cual produce un mayor poder de computo por una misma cantidad de dinero. Además, permite el uso de terminales X mas baratas.

De hecho, convertimos todo el poder de computo en ESTACIONES DE TRABAJO INACTIVAS., a las que se puede tener acceso de manera dinámica. Los usuarios pueden obtener tantas CPUs como sea necesario, durante periodos cortos, después de los cual regresan a la pila, de modo que otros usuarios puedan disponer de ellos. Todos los procesadores pertenecen por igual a todos. Un sistema de colas, es una situación donde los usuarios generan en forma aleatoria solicitudes de trabajo a un servidor. Cuando esta ocupado, los usuarios se forman para el servicio y se procesan según su turno.

Los sistemas de cola son útiles. Es posible modelarlos de forma analítica. Llamamos λ a la tasa de entradas total de solicitudes por segundo de todos los usuarios combinados. Sea μ la tasa de procesamiento de solicitudes por parte del servidor. Debemos tener $\mu > \lambda$. Si el servidor puede manejar 100 solicitudes /seg., pero los usuarios generan de manera continua 110 solicitudes/seg. , la cola crecerá sin limite alguno.

Se puede demostrar que:

$$T = \frac{1}{\mu - \lambda}$$

Supongamos que tenemos n multiprocesadores personales, cada uno con cierto número de CPUs y que cada uno forme su propio sistema de colas, con una tasa de llegada de solicitudes λ y una tasa de procesamiento de los CPU μ . El tiempo promedio de respuesta T estará dado por la expresión anterior.

Consideremos ahora lo que ocurre si reunimos todas las CPUs y los colocamos en una única pila de procesadores. En vez de tener n pequeños sistemas de colas ejecutándose en paralelo, ahora solo tenemos uno grande, con una tasa de entrada $n\lambda$ y una tasa de servicio $n\mu$. Llamemos T_1 al tiempo promedio de respuesta de este sistema combinado, tenemos que:

$$T_1 = \frac{1}{n\mu - n\lambda} = \frac{T}{n}$$

Este sorprendente resultado, nos dice que, si reemplazamos n pequeños recursos por uno grande que sea n veces mas poderoso, podemos reducir el tiempo promedio de respuesta n veces.

Hasta aquí hemos supuesto de manera implícita que una pila de n procesadores es igual a un único procesador que es n veces mas rápido que un único procesador. En realidad, esta hipótesis solo se justifica si todas las solicitudes se pueden dividir de manera que se puedan ejecutar en todos los procesadores en forma paralela. Si un trabajo solo se puede dividir en 5 partes, entonces el modelo de pila de procesadores solo tiene un tiempo de servicio efectivo 5 veces mejor que el de un único procesador y no n veces mejor.

Si se parte de la hipótesis de que ningún procesador pertenece a alguien, obtenemos un diseño con base en el concepto de *Solicitud de las Maquinas a la Pila*, su uso posterior y su regreso al terminal.

Tampoco hay necesidad de regresar algo a una máquina de origen puesto que esta no existe. Tampoco hay peligro de que el poseedor regrese, puesto que no existen poseedores.

Por últimos, si todas las personas hacen una sencilla edición y de manera ocasional envían uno o 2 mensajes de correo electrónico, tal vez sea suficiente con estaciones de trabajo personales. Si por otro lado, los usuarios están implicados en un gran proyecto de desarrollo del SOFTWARE y ejecutan con frecuencia, por ejemplo, MAKE, o intente obtener la inversa de grandes matrices ralas, o llevar a cabo enormes simulaciones, etc..., se requiere de un gran número de estaciones de trabajo y la solución es una pila de procesadores.

10.4.4. Clusters

Entre los sistemas que implementan la alta disponibilidad tenemos el conocido como **clustering** formado por un grupo independiente de servidores que se presentan ante la red como un sistema único.

El clustering es la alternativa al multiprocesamiento simétrico (SMP) como una salida para proveer alto rendimiento y alta disponibilidad y por su particular atractivo para las aplicaciones del servidor. Un **cluster** es un grupo de máquinas interconectadas juntas trabajando como un único recurso de computadora unificado que puede crear la ilusión de estar en una sola máquina. Este grupo de máquinas es un sistema que puede ejecutarse por si solo, aparte del cluster. Cada computadora en un cluster es referida como un **nodo**.

Existen varios beneficios para el clustering:

- **Escalabilidad absoluta:** Se pueden crear grandes clusters que sobrepasan por mucho el poder de la máquina solitaria más potente. Un cluster puede estar formado por docenas de máquinas, cada una de las cuales puede ser un multiprocesador.
- **Escalabilidad incremental:** Cada cluster se configura de manera tal que sea posible el agregado de sistemas nuevos al cluster.
- **Alta disponibilidad:** Ya que cada nodo en un cluster es una máquina solitaria, la falla de uno de los nodos no resulta en la caída del sistema.
- **Relación precio/ rendimiento más conveniente:** Se puede armar un cluster con igual o mayor poder de cómputo que el de una máquina solitaria a un costo mucho más bajo.

Implementar clustering junto a niveles RAID que respalden los datos aseguran la disponibilidad de los sistemas. Las características más sobresalientes son:

- Múltiples servidores independientes funcionando como un sistema servidor único.
- Los servidores tienen un nombre común. Se maneja como un sistema único
- Los servidores están disponibles a todas las máquinas conectadas a la red
- Pueden tolerar fallas de componentes.
- Se pueden agregar componentes sin interrumpir a los usuarios.

Las aplicaciones cliente-servidor recaen en la disponibilidad de los servicios de la red. Estos servicios son proporcionados por los recursos. Si los recursos no están disponibles debido a fallas en aplicaciones o fallas del hardware, el trabajo del usuario es interrumpido. Clustering incrementa la disponibilidad de estos recursos del servidor.

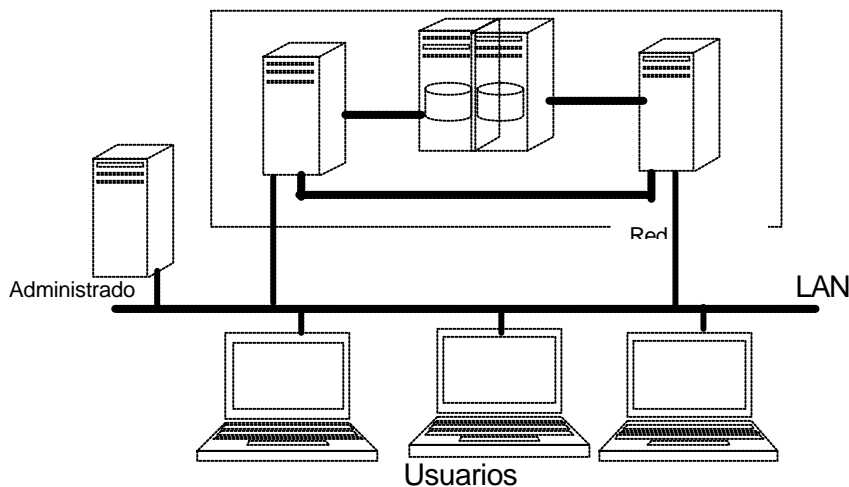


Fig. 10.44 Clustering

Recursos de aplicación, de entrada/salida y CPU pueden ser añadidos, para expandir eficientemente la capacidad del sistema sin interrupción del servicio del usuario. Esto se traduce en un acceso confiable a los recursos del sistema e información, así como protección de la inversión de los recursos de hardware y software.

En un ambiente de servidores comunes, se utilizan herramientas administrativas para identificar los servidores en la red, monitorear sus contenidos y actividades. Sin embargo, en un ambiente de cluster, la administración de aplicaciones y servicios puede ser centralizada a través del uso de herramientas de administración y monitoreo de redes. (ver Fig. 10.44)

Un cluster puede ser configurado de varias maneras: la configuración más simple se basa en si las computadoras en un cluster comparten acceso al mismo disco o no. En caso de ser así, habrá un enlace de alta velocidad para el intercambio de mensajes entre los nodos, y además habrá un subsistema de disco, el cual está conectado directamente a varias computadoras. El subsistema más comúnmente usado es el RAID.

En caso contrario (los nodos no comparten acceso al disco), también habrá un enlace de alta velocidad entre los nodos para coordinar la actividad entre los nodos. Este enlace puede ser una LAN, la cual es compartida con otras máquinas no pertenecientes al cluster o puede ser una facilidad de interconexión dedicada.

Problemas de diseño de sistemas operativos de clusters

La explotación correcta de una configuración de hardware de cluster requiere realizar algunas mejoras por sobre los sistemas operativos ordinarios.

Respuestas ante fallas

Este tema puede ser tratado de varias maneras:

- **Clusters altamente disponibles:** Ofrece una alta probabilidad de que todos los recursos estén en servicio. Si se produce una falla, las consultas en progreso se pierden. En caso de reintentarse las consultas, las mismas se satisfarán en otra computadora del cluster.
- **Clusters tolerantes a fallas:** Se garantiza la permanente disponibilidad de los recursos, lo cual se logra mediante el uso de discos compartidos redundantes.
 - **Failover:** Función que cambia aplicaciones y recursos de datos desde un sistema caído a otro en funcionamiento.
 - **Failback:** Restauración de aplicaciones y recursos de datos hacia el sistema original, una vez que éste se recupera de la falla.

Balance de carga del Cluster

Todo cluster requiere de la capacidad de balancear efectivamente la carga a lo largo de las computadoras disponibles. Cuando se agrega una nueva computadora al cluster, la facilidad de balance de carga incluye automáticamente dicha computadora en las aplicaciones de planeamiento.

Clusters vs. Multiprocesamiento simétrico

El SMP se destaca por sobre los clusters en los siguientes aspectos:

- Configuración y administración más simple.
- Es más similar al modelo de monoprocesamiento original en el cual se escribe la mayoría de las aplicaciones.
- Ocupa menos espacio físico y consume menos energía.
- Es más estable.

En cambio, los clusters se destacan por sobre el SMP en términos de escalabilidad absoluta e incremental. También son superiores con respecto a la disponibilidad, debido a que todos los componentes del sistema pueden hacerse redundantes.

10.5. Bibliografía recomendada para este Módulo

1. Distributed System, edited by Sape Mullander. (Second Edition); Addison wesley, 1994, 601 pages.
2. Open System. Gary J. Nutt
3. Operating Systems. (Second Edition), Stallings Willams; Prentice Hall, Englewood Cliff, NJ. 1995, 702 pages.

4. Operating Systems Concepts (Fifth Edition), Silberschatz, A. and Galvin P. B; Addison Wesley 1998, 850 Pages.
5. Operating Systems Concepts and Design (Second Edition), Milenkovic, Millan; Mc Graw Hill 1992
6. Modern Operating Systems, Tanenbaum, Andrew S.; Prentice - Hall International 1992, 720 pag.
7. Sistemas Operativos Conceptos y diseños. (Segunda Edición); Milenkovic Milan; Mc Graw Hill; 1994. 827 páginas

GLOSARIO DE TÉRMINOS EN IDIOMA INGLÉS

File	client - server	peer to peer	mainframe
on-line	Local Area Network	Wide Area Network	Host
Off-line	Routers	Frames	Middleware
File System	Client -Server computing	Scheduling	Data
Kernel - Microkernel	Single Instruction Stream	Single Data Stream	Multiple Instruction Stream
First In First Out	Multiple Data Stream	Stack	Open System
Print Server	Thightly coupled	Lowley coupled	Uniform Memory Access
Not Uniform Memory Access	Not Remote Memory Access	Cross switch	Cross Point switch
Crosstalk	Link	write	Network File System
Threads	Mailbox	Deadlock	Master -Slave
Overhead	Task	Context switch	Run time system
Garbage collection	Make	Daemon	Dispatcher
Universal	Time out	Fault Tolerance	Distributed Computing
Coordinated			
Message Passing	Shared Memory	Crash	Received omission
Send omission	General omission	Arbitrary with message authentication	Timing failure
Broadcast Network	Strongly Broadcast	Broadcast especification	FIFO Broadcast
Reliable Broadcast	Causal Broadcast	Atomic Broadcast	Agreement
Deliver	Sender	Validity	Integrity
FIFO Order	Causal Order	Total Order	State Machine approach to
Timed broadcast	Atomic commitment	Uniform agreement	Non blocking Atomic commitment
Copy approach	Primary Backup	Two phase commitment	Tree phase commitment
Others	Spooling	Image	Starter
Long term scheduler	Memory allocation	Acknowledge	service call
error cheking	Call / Return	Pack / Unpack	Execute Procedure
client stub	Server stub	Disk server	File server
caché	Upload /Download	Remote access	Lazy replication
Pipeline	Workstation	Transaction	Begin / End
Clock	Load Sharing	Network	Layer
Session	File transfer	Check point	Recovery
Star topology	Ring topology	Bus /tree topology	set
frame	checksum	mailbox	native mail
full	half	Message Passing	

American Standard Code for information interchange

GLOSARIO DE TÉRMINOS EN CASTELLANO

Tipos de procesamientos	Procesamiento Centralizado
Procesamiento Distribuido	Procesamiento Cooperativo
Procesamiento Cliente - Servidor	Procesamiento Par a Par
Objetivos del Procesamiento Distribuido	Fuertemente acoplado - Débilmente Acoplado
Ventajas y desventajas de los Sistemas Distribuidos	Escalabilidad
Redes	Topología de Redes
Tipo de redes	Comunicaciones
Estrategias de ruteo de mensajes	Estrategias de conexión
Conflictos	Estrategias de diseño
El modelo OSI	Interoperatividad
Middleware	Plataformas
S.O. para Multiprocesadores	Software de comunicaciones (S.O. de redes)
S.O. realmente distribuidos	Migración de : Datos, Cómputos y Procesos

Características de un S.O. Distribuido	Consistencia de datos y caché
Multiprocesamiento con tiempo compartido	S.O. Distribuidos
procesos y procesadores en Sistemas Distribuidos	Los hilos (threads)
Paquetes de hilos	Modelos de Sistemas
Asignación de Procesadores /algoritmos	Sincronización en Sistemas Distribuidos/Algoritmos
La Mutua Exclusión en Sistemas Distribuidos/Algoritmos	Las transacciones en Sistemas Distribuidos/Algoritmos
Los Deadlock en Sistemas Distribuidos	Los problemas en Sistemas Distribuidos
Las comunicaciones en Sistemas Distribuidos/Protocolos	Estructuras de redes
Protocolos de comunicaciones y arquitecturas	Las migraciones en Sistemas Distribuidos
Las comunicaciones en el procesamiento Distribuido	

ACRÓNIMOS USADOS EN ESTE MÓDULO

E/S	Entrada / Salida	I/O	Input / Output
DMA	Direct Memory Access	O.K.	okey
PCB	Process Control Block	PID	Process Identifier
IEEE	Institute of Electronic and Electric Engineers	SYSCALL	System Call
RAM	Random Access Memory	LAN	Local Area Network
ROM	Read Only Memory	WAN	Wide Area Network
CPU	Central Processing Unit	ethernet	ether network
S.O.	Sistema Operativo	internet	inter network
ISO	International Standard Organization	CP	Communication Processor
OSI	Open System Interconnection	SQL	Structured Query Language
r/w/x	Read / Write/Execute	RPC	Remote Procedure Call
VMS	Virtual Machine System	B.D.	Base de Datos
SISD	Single Instruction stream, Single Data Stream	SIMD	Single Instruction stream, Multiple Data Stream
MISD	Multiple Instruction stream, Single Data Stream	MIMD	Multiple Instruction stream, Multiple Data Stream
PRG	PRoGram	SW	SoftWare
API	Application Programming Interface	HW	HardWare
UMA	Uniform Memory Access	NUMA	Not Uniform Memory Access
NORMA	NOt Remote Memory Access	NFS	Network File System
TCB	Thread Block Control	MUTEX	Mutua Exclusión
SCSI	Small Computer System Interface	RAID	Redundant Array of Inexpensive Disk
rsh	remote shell	PD	Procesamiento Distribuido
USR	User	RRP	Request/Ready Protocol
REQ	Request	TCP	Transport Control Protocol
REP	Reply	IP	Internet Protocol
ACK	Ack	FTP	File Transfer Protocol
AYA	Are you Alive?	SMTP	Simple Mail Transfer Protocol
IAA	I am alive	TELNET	Teletype Network
TA	Try Again	GW	Gate Way
AU	Address Unknown	PH	Physical Header
FIFO	First In, First Out	SD	Sistemas Distribuidos
DFS	Distributed File System	NFS	Network File System
FS	File System	IPC	Inter Process Communication
SAP	Service Access Point	CC	Conmutador Central
PDU	Protocol Data Unit		
ASCII	American Standard Code for Information Interchange	EBCDIC	Extended Binary-Coded Decimal Interchange Code

ANEXO 10.a: “IPv6 el Nuevo Protocolo de Internet”

por: Hernán Novodvorski

Índice

Prólogo.....	698
1. Introducción al Protocolo de Internet (IP).....	698
1.1. ¿Cómo se llega a IPv6?.....	700
1.2. Otros puntos débiles.....	701
2. Formatos y Funciones de IPv6.....	704
2.1. Formato de la cabecera IPv6.....	704
2.2. Cabeceras Extendidas	705
2.3. Direcciones de IPv6.....	709
3. Seguridad en IPv6.....	713
3.1. Asociaciones de Seguridad	713
3.2. Autenticidad.....	714
3.3. Autenticidad más privacidad	716
3.4. Administración de Clave IPv6	717
4. Capacidades de Calidad de Servicio	718
5. La Transición a IPv6.....	719
5.1. El método de transición con capa dual	720
5.2. Escenarios de Transición	720
Conclusión.....	722
Bibliografía	722

Prólogo

Mediante este trabajo, se pretende hacer un “análisis” de las características del Protocolo de Internet versión 6 (IPv6), mostrar sus diferencias con el Protocolo de Internet anterior (versión 4 – IPv4), y poder entender la importancia de su uso en las redes de la actualidad. Este documento está dividido en cinco partes: el primero es una introducción al Protocolo de Internet (IP), explica como se llega a IPv6, y tiene un resumen de sus principales funciones; el segundo trata sobre aspectos más técnicos cómo formato de las cabeceras, direccionamiento, etc.; el tercero se centra en temas relacionados con las seguridad (mecanismos, etc.); el cuarto es una introducción a las Capacidades de Calidad de Servicio (manejo especial de paquetes, “Tiempo Real”, etc.) y el quinto muestra la transición entre el uso de IPv4 e IPv6.

1. Introducción al Protocolo de Internet (IP)

IP provee la funcionalidad de interconectar sistemas finales a través de múltiples redes. Para este propósito, IP es implementado en cada sistema final y en los routers (encaminadores), que son dispositivos que proveen conexión entre redes. Los datos de alto nivel en un sistema final de origen son encapsulados en una unidad de datos de protocolo IP (PDU) para la transmisión. Este PDU luego se pasa a través de una ó mas redes y routers para alcanzar el sistema final de destino.

El router debe ser capaz de hacer frente a una variedad de diferencias entre las redes, incluyendo :

- **Esquemas de Direccionamiento** – Las redes pueden usar esquemas diferentes para asignar las direcciones a los dispositivos. Por ejemplo, una red de area local IEEE 802 (LAN) usa direcciones binarias de 16-bit ó 48-bit para cada dispositivo conectado; una red pública de intercambio de paquetes X.25 usa direcciones decimales de 12 dígitos (codificadas como 4 bits por dígito para una dirección de 48-bit). Se debe proveer alguna forma de direccionamiento global de red, así como también un servicio de directorio.

- **Tamaño máximo de paquete** – Los paquetes de una red pueden tener que ser divididos en piezas más pequeñas para ser transmitidos a otra red. Por ejemplo, Ethernet impone un tamaño máximo de paquete de 1500 bytes; un tamaño máximo de paquete de 1000 bytes es común en redes X.25. Un paquete que es transmitido en un sistema Ethernet y es tomado por un router para retransmisión en una red X.25 puede tener que segmentar el paquete de entrada en dos más pequeños.
- **Interfases** – Las interfases de hardware y software para varias redes distintas. El concepto de router debe ser independiente de estas diferencias.
- **Confiabilidad** – Varios servicios de red pueden proveer desde un circuito virtual end-to-end hasta un servicio no confiable. La operación de los routers no debería depender del supuesto que la red es confiable.

Con la naturaleza cambiante de Internet y las redes de negocios, el protocolo actual de Internet (IP), que es el fundamento del Protocolo de Control de Transmisión de red (TCP)/IP, se está volviendo rápidamente obsoleto.

Hace poco tiempo, Internet y la mayoría de las demás redes de TCP/IP han provisto soporte principalmente para simples aplicaciones distribuidas, tales como transferencia de archivos, correo electrónico, y acceso remoto usando TELNET ; pero hoy, Internet se está convirtiendo de a poco en un medio rico en aplicaciones multimedia, conducido por la inmensa popularidad de la World Wide Web. Al mismo tiempo, las redes corporativas se ramificaron de las simples aplicaciones de e-mail y transferencia de archivos y se volvieron medios complejos de client/server y, más recientemente, Intranets que imitan las aplicaciones disponibles en Internet.

Todos estos desarrollos han ganado la capacidad de las redes basadas en IP para proveer las funciones y los servicios necesarios. Un medio con trabajo de Internet necesita soportar tráfico en tiempo real, esquemas de control de congestión flexible, y características de seguridad. Ninguno de estos requerimientos son fácilmente obtenidos con el IP existente.

Sin embargo, el manejo del poder por atrás del desarrollo de una nueva red IP es el hecho inflexible que el mundo está desperdiciando de las direcciones IP para los dispositivos de red. La longitud de dirección fija de 32 bits de IP es inadecuada para el crecimiento explosivo de las redes.

Para identificar cada una de las interfases de los sistemas finales y equipos routers de la Internet se emplean *direcciones IP* de 32 bits, que usualmente se suelen representar textualmente en grupos de 8 bits en notación decimal (por ejemplo, 130.206.16.75). En IPv4 se definen tres tipos de direcciones: la más común o *unicast* (asociada a una única interfase), *multicast* (asociada a un grupo de interfases) y *broadcast* (asociada a todos los interfaces). Para asignar direcciones unicast a equipos se dispone de más de 3.750 millones de direcciones (0.0.0.0-223.255.255.255) que se agrupan de forma topológica en forma de una parte de red (los bits más significativos o más a la izquierda), que es común a todos los miembros de la unidad topológica pudiendo ser ésta una red local, una organización o un proveedor y una parte local (los bits menos significativos) que es diferente para cada ente individual.

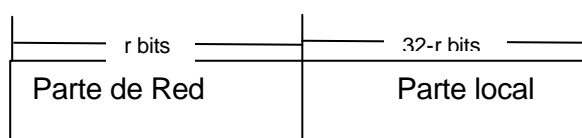


Fig. 10.a.1. direcciones de IP

El ruteo en la Internet se lleva a cabo bajo la suposición que todos los equipos que tienen identificadores con una misma parte de red son miembros de una misma unidad topológica sujeta a una estructura administrativa única, lo que permite inferir que el ruteo *interno* es homogéneo y completo. El resultado de todo esto es que todos los equipos de la red son vistos desde el *exterior* como un único elemento de cara al ruteo, lo cual reduce en gran manera la cantidad de información que deben mantener, almacenar y procesar los equipos routers o routers. En adelante, a la parte de red de una dirección se la denomina *prefijo*, que servirá para representar diferentes estructuras topológicas mediante la sintaxis:

direccion-IP-de-32-bits/tamaño-del-prefijo

Esto indicará qué parte de la dirección es significativa en términos de ruteo. Por ejemplo, 130.206.16.75/32 representa una única dirección de red mientras que 130.206.16.0/24 representa el conjunto de direcciones 130.206.16.0-130.206.16.255. Es importante resaltar la diferencia entre los procedimientos de ruteo internos a un dominio o *intra-dominio*, de los que éste dominio tiene respecto a los demás con los que tiene conexión directa o indirecta (*inter-dominio*). Existe una jerarquía de ruteo que se refleja en la mayor o menor generalidad de los prefijos empleados para anunciar el dominio.

1.1. ¿Cómo se llega a IPv6?

El problema de la asignación de direcciones

A la hora de diseñar un método de asignación de direcciones en los albores de la Internet, cuando estaba conectada apenas una docena de centros, se pensó en un esquema basado en el tamaño de las organizaciones y de ahí nació el modelo de clases en la que sólo se daba cabida a tres tipos de prefijo de longitud determinada según fuera una gran organización (clase **A**, prefijo 8 bits), de tamaño mediano (clase **B**, prefijo 16 bits) o pequeño (clase **C**, prefijo 24 bits).

Se tiene pues 128 prefijos correspondientes a clases A (0.0.0.0/8-127.0.0.0/8), 16384 de clases B (128.0.0.0/16-191.255.0.0/16) y algo más de 2 millones de clases C (192.0.0.0/24-223.255.255.0/24).

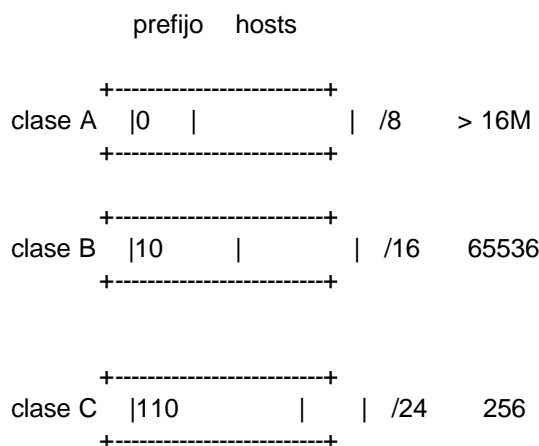


Fig. 10.a.2 Direcciones de Internet

La asignación de direcciones comenzó a hacerse de manera centralizada por un único centro de registro (SRI-NIC) satisfaciendo casi todas las solicitudes sin necesidad de mayor trámite. Este modelo de asignación de direcciones, cuando la Internet comenzó a crecer de forma espectacular, trajo algunas de estas consecuencias:

- **Mal aprovechamiento del espacio de direcciones.** Cada centro tendía a pedir una clase superior a la requerida, normalmente una clase B en vez de una o varias clases C, por puro optimismo en el crecimiento propio o por simple vanidad.
- **Peligro de agotamiento de las direcciones de clase B.** Las más solicitadas debido a la escasez de posibilidades de elección. La alerta sonó cuando se había agotado el 30% de esta clase y la demanda crecía exponencialmente.
- **Síntomas de saturación en los routers de los backbones.** Al imponerse restricciones severas en la asignación de clases B, las peticiones de múltiples clases C se hicieron masivas, lo que hizo que aumentara de forma explosiva el número de prefijos que los routers habían de mantener en sus tablas, llegándose a alcanzar los límites físicos impuestos por la capacidad de memoria y de proceso.

Por lo tanto existe un doble problema: agotamiento de direcciones y colapso de routers debido a la explosión de rutas. En una situación en la que la población conectada a la red se duplicaba (en términos de equipos y redes conectadas) en periodos que oscilaban en torno a los 6 meses había que tomar medidas urgentes, y he aquí algunas de las que se tomaron:

- Imposición de políticas restrictivas de asignación de direcciones por parte de los centros de registros (ya descentralizados del primitivo NIC), como ya se ha apuntado antes.
- Modificación de los protocolos exteriores de ruteo para soportar prefijos de red variables. Este método se conoce como **CIDR** (Classless Inter-Domain Routing) y consiste en la agregación de prefijos que son adyacentes para dar lugar a prefijos menores (y por tanto más generales), con lo que se reducía el número de entradas en las tablas de los routers. Por ejemplo, el RIPE-NCC ha asignado a RedIRIS el bloque 193.144.0.0/15 (131.072 direcciones) para delegar entre sus miembros, anunciándose todo el bloque como un único prefijo.
- Iniciación de la asignación, según el modelo CIDR o sin-clase, de partes del espacio de direcciones que estaban reservadas, como 64.0.0.0/8 - 126.0.0.0/8 que suponen

aproximadamente 1/4 del total del espacio asignable. Estas direcciones se asignarán en bloques de prefijo variable.

Para entender bien el problema hay que tener en cuenta que el período de tiempo en el que los fabricantes duplican la capacidad de proceso de sus equipos y la de sus memorias es de aproximadamente dos años. La capacidad de los routers que deben mantener en sus tablas una información completa o *full-routing* sobre la topología de la Internet (equipos conectados a los backbones principales o de dominios conectados a múltiples proveedores *-multi-homed-*) se habría hoy día superado, con el colapso consiguiente de la Internet, si CIDR no hubiese sido puesto en funcionamiento a principios de 1994. Como dato ejemplificador, RedIRIS cuenta en su dominio interno con más de 650 redes pero sólo anuncia al exterior 32 prefijos, en su mayoría clases B *históricas*, con lo que se consigue una agregación óptima.

En cualquier caso, hay que entender que tanto CIDR como las políticas restrictivas de asignación de direcciones son sólo medidas temporales, dirigidas a afrontar problemas concretos y que no resuelven (en algunos casos hasta agravan) los problemas crónicos detectados en la Internet en gran parte debidos a su tremendo éxito. Así, se han llegado a plantear iniciativas como la devolución de direcciones, la obligatoriedad de cambiar de direcciones al cambiar de proveedor, la asignación dinámica de direcciones, el uso de traductores de direcciones (NAT) que transformen un espacio privado de direcciones en otro perteneciente al proveedor, o incluso el cobrar una cantidad elevada por cada prefijo (no perteneciente al espacio del proveedor) que un cliente desee que su proveedor anuncie.

1.2. Otros puntos débiles

De forma recurrente se ve como se achaca a la Internet el ser un medio de comunicación inseguro. Este es un tema con muchas aristas y que debe ser examinado en cada una de sus partes. Dado el carácter puramente académico de la Internet en sus comienzos, los asuntos relativos a la seguridad fueron, como desgraciadamente suele ocurrir en la práctica, relegados a posterior estudio hasta que los primeros ataques globales hacen sonar la alarma y empieza a producirse un notable esfuerzo en incorporar mecanismos de seguridad a las aplicaciones existentes.

Sin embargo, el problema de seguridad en el nivel de red sigue sin ser tenido en cuenta y comienza a producirse una serie de ataques cada vez más sofisticados y basados en la suplantación de la identidad de máquinas conectadas a la red, dando la posibilidad de violar un acceso prohibido o dando la posibilidad de escudriñar (o desviar) la información a intrusos. Como respuesta surgen mecanismos de barrera como los *firewalls*, pero los protocolos siguen sin incorporar medidas específicas de seguridad.

Pero esto es sólo una parte del problema. La seguridad integral comprende servicios tanto de confidencialidad como de autenticación, integridad y no rechazo para los que se requieren técnicas criptográficas que estén sujetas a diferentes normativas de exportación y uso en determinados países, lo que hace complicado su uso generalizado en un medio que se tiene por libre (en cuanto a la naturaleza de la información intercambiada y su formato) y homogéneo (en cuanto al tipo de protocolos/aplicaciones empleados). Se corre el peligro de fracturar la Internet en zonas donde se puedan intercambiar información de forma segura y otras en que no, bien por considerarse tecnología de uso militar, bien por el derecho que se guardan algunos gobiernos a poder intervenir -e interpretar- las comunicaciones de sus ciudadanos.

En otro orden de cosas, se está asistiendo al nacimiento de servicios de transmisión de información en tiempo real dentro de la Internet. Ejemplos de ello son las aplicaciones para comunicaciones de voz a través de la red y el **Mbone**, red virtual superpuesta a la Internet, basada en el concepto de *IP Multicast*. Un defecto claro de IPv4 es la falta de caracterización de los distintos flujos de información que viajan por la red, dando la misma consideración a un tráfico que podría considerarse como de relleno como las NetNews, sujeto a la redundancia de un mecanismo corrector de transporte, que a una transmisión de voz en tiempo real en la que la pérdida de un número significativo de paquetes puede alterar o incluso imposibilitar la interpretación de la información. El advenimiento de este tipo de servicios en la era de las autopistas de la información presenta una clara limitación al uso de la Internet tal y como se la concibe actualmente en contraposición a otro tipo de tecnologías orientadas a la conexión como ATM que parecen más adecuadas para los servicios en tiempo real.

IPv6

Los problemas mencionados anteriormente han sido tenidos en cuenta por la comunidad de Internet desde que se empezaron a predecir sus consecuencias (en mayor o menor grado de pesimismo catastrofista) lo que motivó que sus técnicos elaboraran una serie de propuestas conducentes a la creación de un nuevo protocolo para la red Internet, conocido como *IP Next Generation (IPng)*. A finales de 1994 El **IESG** (Internet Engineering Steering Group), después de examinar las propuestas finales y en base a unos criterios que habían sido elaborados con anterioridad, emitió una recomendación para IPng que

posteriormente se publicaría como RFC 1752 con la categoría de *Proposed Standard*. Dicha recomendación fue desarrollada por los grupos de trabajo del **IETF** (Internet Engineering Task Force).

En esta recomendación se asigna para IPng el número de versión 6 y pasa a denominarse formalmente como **IPv6**. La decisión del IETF tiene como aspecto más importante la elección de **SIPP** (Simple IP Protocol Plus) como base para la elaboración de IPv6 con aspectos de otros contendientes, en particular de **TUBA** (TCP & UDP with Bigger Addresses) del que se toma el modelo de transición, uno de los aspectos más importantes recogidos entre los criterios de selección. Mientras que SIPP representa un paso evolutivo en el protocolo IP, tomando de la versión 4 los elementos que se emplean y desechando los que no se usan, simplificando al mismo tiempo el protocolo para optimizar la transmisión; TUBA proponía el empleo del protocolo de ISO para redes no orientadas a la conexión (CLNP) y el uso de direcciones NSAP normalizadas.

Volviendo a IPv6, se encuentra como primera característica (como era fácil suponer) unas direcciones ampliadas (128 bits) junto con unas facilidades de ruteo mejoradas. El espacio de direcciones no sólo aumenta sino que su estructura se hace más adecuada para un ruteo óptimo mediante una jerarquía de prefijos que implican distintas formas de asignación. He aquí algunos ejemplos de estas nuevas direcciones en su representación textual (nótese la presentación hexadecimal y la eliminación de ceros no significativos):

- **unicast** (direcciones equivalentes)
4800:F400:0045:EA00:0000:0000:04E7
4800:F400:45:EA00:0:0:0:4E7
4800:F400:45:EA00::4E7
- **multicast**
FF02:0:0:0:0:0:0:2E5 ó FF02::2E5
- **compatible IPv4**
0:0:0:0:0:0:130.206.16.75
::130.206.16.75

Otras características destacables del nuevo esquema de direccionamiento son:

- Da cabida a distintos modelos de asignación, como el *basado en el proveedor* (direcciones unicast de equipos que se conectan a través de un proveedor), el *geográfico* (para puntos neutros de interconexión, por ejemplo) o *local*, sin conexión a la red global. Hay prefijos preasignados a centros de registro como Internic, RIPE-NCC (Europa) y APNIC (Asia-Pacífico) que a su vez registrarán prefijos para los proveedores de su ámbito. También se pueden representar direcciones IPX y NSAP con las nuevas direcciones IPv6 de 128 bits.
- Un nuevo tipo de dirección: *anycast*, que identifica un grupo de direcciones y que al ser empleado, se referirá al nodo *más próximo* al originador.
- Las direcciones de *broadcast* no existen en IPv6. Son casos particulares de direcciones *multicast*. Estas últimas incorporan un campo de *ámbito*, en sustitución del parámetro TTL que se usaba para determinar el rango de actuación de una sesión multicast.
- Autoconfiguración. Uno de los aspectos fundamentales de IPv6 es la incorporación de mecanismos que permitan la conexión automática (modelo *plug and play*) de equipos a la red. Pueden construirse direcciones globales usando como parte local la dirección MAC de un equipo y obteniendo el prefijo a través de un servidor de la red.

Las cabeceras de los paquetes han sido simplificadas en IPv6 eliminando los campos no utilizados y añadiendo el concepto de cabeceras de extensión. Estas permiten seleccionar facilidades especiales de ruteo, fragmentación y seguridad así como el manejo de opciones, que han sido eliminadas de la cabecera IPv6. Cada cabecera incluye un campo que define el tipo de cabecera que le sigue a continuación, hasta llegar a la de transporte, con lo que se agiliza el proceso de los paquetes.

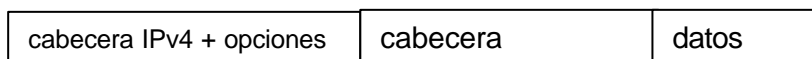


Fig. 10.a.3 datagrama IP versión 4

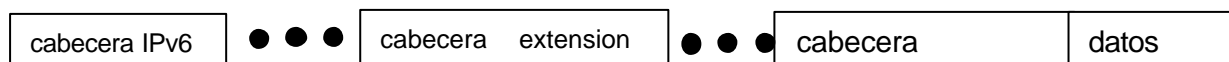


Fig.10.a.4 datagrama IP versión 6

En IPv6 la fragmentación/ensamblado de paquetes es una tarea de los sistemas finales con lo que de nuevo se facilita la vida a los routers para que se entreguen a su misión primordial, encaminar paquetes. La unidad máxima de transmisión (MTU) de un enlace es ahora como mínimo de 576 bytes (68 en IPv4).

El ruteo propuesto permitirá la selección de proveedor por parte del originador así como la movilidad mediante el empleo de cabeceras de extensión de ruteo en la que se especifique un camino elegido (que será recorrido a la inversa al regreso) mediante direcciones unicast o anycast.

La transmisión de información en tiempo real se podrá realizar mediante el empleo de dos campos en la cabecera IPv6. La *prioridad*, que distingue los diferentes tipos de datagramas según la clase de servicio y la *etiqueta de flujo*, que permite diferenciar y asignar distintos estados a distintos flujos originados por la misma fuente.

Por fin, la seguridad en el nivel de red será una realidad mediante el empleo de cabeceras de extensión de *autenticación*, que proporciona servicios de verificación de identidad e integridad y de *encapsulado de seguridad* que proporciona servicios de confidencialidad. En el primer caso no se plantean problemas de exportación de tecnología por tratarse de un procedimiento de verificación (se cifra una función del contenido, pero no el contenido en sí) mientras que en el segundo caso todos los malos presagios salen a relucir. Más sorprendente es aún el hecho de que, de acuerdo con las especificaciones, todo producto que pretenda ser conforme a IPv6 ha de incluir necesariamente ambos mecanismos de seguridad a pesar de los impedimentos expuestos.

El tránsito hacia IPv6

Uno de los aspectos más importante de todos los contemplados durante el proceso IPng es el de la migración o paso de una red basada en la arquitectura IPv4 a IPv6. Un criterio que guió la selección de candidatos para IPng fue el de rechazar cualquier propuesta, por brillante que fuera, que no incluyera un plan viable de transición de la base actual instalada de equipos IPv4 a IPv6. La transición ha de hacerse de forma gradual, sin afectar a los servicios que se prestan en la actualidad.

En IPv6 el método propuesto se basa primordialmente en la coexistencia de ambos protocolos. Los nuevos sistemas que vayan incorporando IPv6 (computadores y routers) deberán mantener asimismo la plena capacidad de procesar paquetes IPv4. Para conectar las islas IPv6 que irán emergiendo mediante la infraestructura Internet actual se emplearán técnicas de encapsulado (túneles IPv6) en los datagramas IPv4 de forma similar a como funciona en la actualidad el Mbone. A medida que los routers vayan incorporando IPv6, los túneles se irán desmantelando para dar paso a una infraestructura ya basada en IPv6.

Respecto a las aplicaciones actuales, la migración a IPv6 requiere que todas las referencias a direcciones de 32 bits sean sustituidas por las nuevas de 128. Desgraciadamente eso supone cambiar prácticamente todas las aplicaciones existentes. Además, un nuevo tipo de registro deberá ser definido en el DNS para realizar la transformación de los nombres de dominio actuales (que no requieren cambio) en direcciones de 128 bits. Este nuevo registro es el **AAAA** y la traducción inversa de direcciones a nombres se realizará dentro de la nueva zona ip6.int dispuesta a tal efecto.

En resumen, los cambios de IPv4 a IPng caen principalmente en las siguientes categorías :

- Ruteo expandido y capacidad de direccionamiento.

IPng incrementa el tamaño de dirección de IP de 32 a 128 bits, para soportar más niveles de jerarquía de direccionamiento y un número mucho mayor de nodos direccionables, y auto configuración simple de direcciones.

- Se define un nuevo tipo de dirección denominado "dirección anycast", para identificar los conjuntos de nodos donde un paquete enviado a una dirección anycast es entregado a uno de los nodos. El uso de direcciones anycast en la ruta fuente de IPng le permite a los nodos controlar el camino que sigue su flujo de tráfico.
- Simplificación del formato del encabezado

Algunos campos de encabezado IPv4 han sido hechos opcionales, para reducir el costo por proceso común de manejo de paquetes y mantener el costo por ancho de banda del encabezado IPng tan bajo como sea posible sin importar el tamaño aumentado de las direcciones. Aunque las direcciones IPng son cuatro veces más grande que las direcciones IPv4, el encabezado IPng es solo dos veces el tamaño del encabezado IPv4.

- Soporte mejorado para opciones.
Los cambios en el modo de codificar las opciones del encabezado de IP permiten seguir más eficiencia, menos límites rígidos en la longitud de las opciones, y mayor flexibilidad para introducir nuevas opciones en el futuro.
- Capacidades de Calidad de Servicio
Se adiciona una nueva capacidad para permitir el etiquetado de paquetes pertenecientes a un flujo de tráfico particular para el cual el emisor solicita un manejo especial, tal como calidad de servicio no por defecto o servicio en "tiempo real".

- Capacidad de Autenticidad y Privacidad
IPng incluye la definición de extensiones que proveen soporte para la autenticidad, la integridad de datos, y la confidencialidad. Esto se incluye como un elemento básico de IPng y será incluido en todas las implementaciones.

El protocolo IPng consta de dos partes, la cabecera básica de IPng y las cabeceras de extensión IPng.

2. Formatos y Funciones de IPv6

La unidad de datos del protocolo IPv6 (conocida como paquete) tiene el siguiente formato general

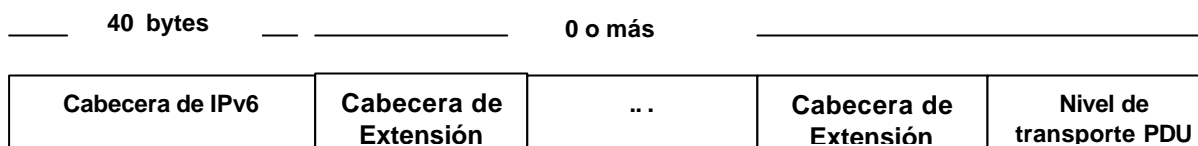


Fig. 10.a.5. Formato del paquete IPv6

La única cabecera que se necesita se denomina simplemente cabecera IPv6. Es de tamaño fijo con una longitud de 40 bytes, comparada con los 20 bytes para la porción obligatoria de la cabecera IPv4.

2.1. Formato de la cabecera IPv6

El formato de la cabecera IPv6 es el siguiente:

El formato de la cabecera II VO es el siguiente:			
Versión	Prioridad	Etiqueta de flujo	
Longitud carga		Siguiente cabecera	Límite de saltos
Dirección origen			
"			
"			
"			
Dirección destino			
"			
"			
"			

Fig. 10.a.6. Cabecera del IPv6

La cabecera de IPv6 tiene una longitud fija de 40 bytes y consta de los siguientes campos :

- | | |
|-------------------------|----------------------|
| a) Versión | e) Siguiete cabecera |
| b) Prioridad | f) Límite de saltos |
| c) Etiqueta de flujo | g) Dirección origen |
| d) Longitud de la carga | h) Dirección destino |

a) Versión.

Este campo ocupa 4 bits, e indica la versión de IP. Para el formato descrito, la versión es la 6, para IPv6 (también llamada IPng, Internet Protocol Next Generation).

b) Prioridad.

Este campo ocupa 4 bits, e indica la prioridad que el remitente desea para los paquetes enviados, respecto a los demás paquetes enviados por él mismo. Los valores de prioridad se dividen en dos rangos, de 0 a 7, paquetes para los cuales el remitente espera una respuesta en caso de congestión (p.e. tráfico TCP). Y de 8 hasta 15, paquetes que no deben ser respondidos en caso de congestión, el valor más bajo (8), se usaría cuando el remitente está dispuesto a que sus paquetes sean descartados en caso de congestión (p.e. Vídeo en alta calidad). Y el valor más alto (15), cuando el remitente está muy poco dispuesto a que algún paquete sea descartado (p.e. Audio de baja calidad).

c) Etiqueta de flujo.

Este campo ocupa 24 bits, y es usado por el remitente para indicar que sus paquetes sean tratados de forma especial por los routers, como en servicios de alta calidad o en tiempo real. En este punto, se entiende el flujo como un conjunto de paquetes que requieren un tratamiento especial.

Todos los paquetes pertenecientes al mismo flujo deben tener valores similares en los campos dirección origen, dirección destino, prioridad, y etiqueta de flujo.

d) Longitud de la carga.

Este campo ocupa 16 bits, e indica la longitud del resto del paquete que sigue a la cabecera, en bytes. Si su valor es cero, indica que el tamaño de la carga vendrá especificado como *Carga Jumbo*, en una opción *salto a salto*.

e) Siguiente cabecera.

Este campo ocupa 4 bits, e identifica el tipo de cabecera que sigue a la cabecera IPv6. Es coherente con los valores del campo *protocolo* en IPv4.

f) Límite de saltos.

Este campo ocupa un byte. Es decrementado en una unidad por cada nodo que redirige el paquete hacia su destino. El paquete es descartado si el valor del campo llega a cero. Este campo sustituye al campo *tiempo de vida*, de IPv4.

g) Dirección origen.

Este campo ocupa 128 bits, y corresponde a la dirección de origen.

h) Dirección destino.

Este campo ocupa 128 bits, y corresponde a la dirección de destino.

2.2. Cabeceras Extendidas

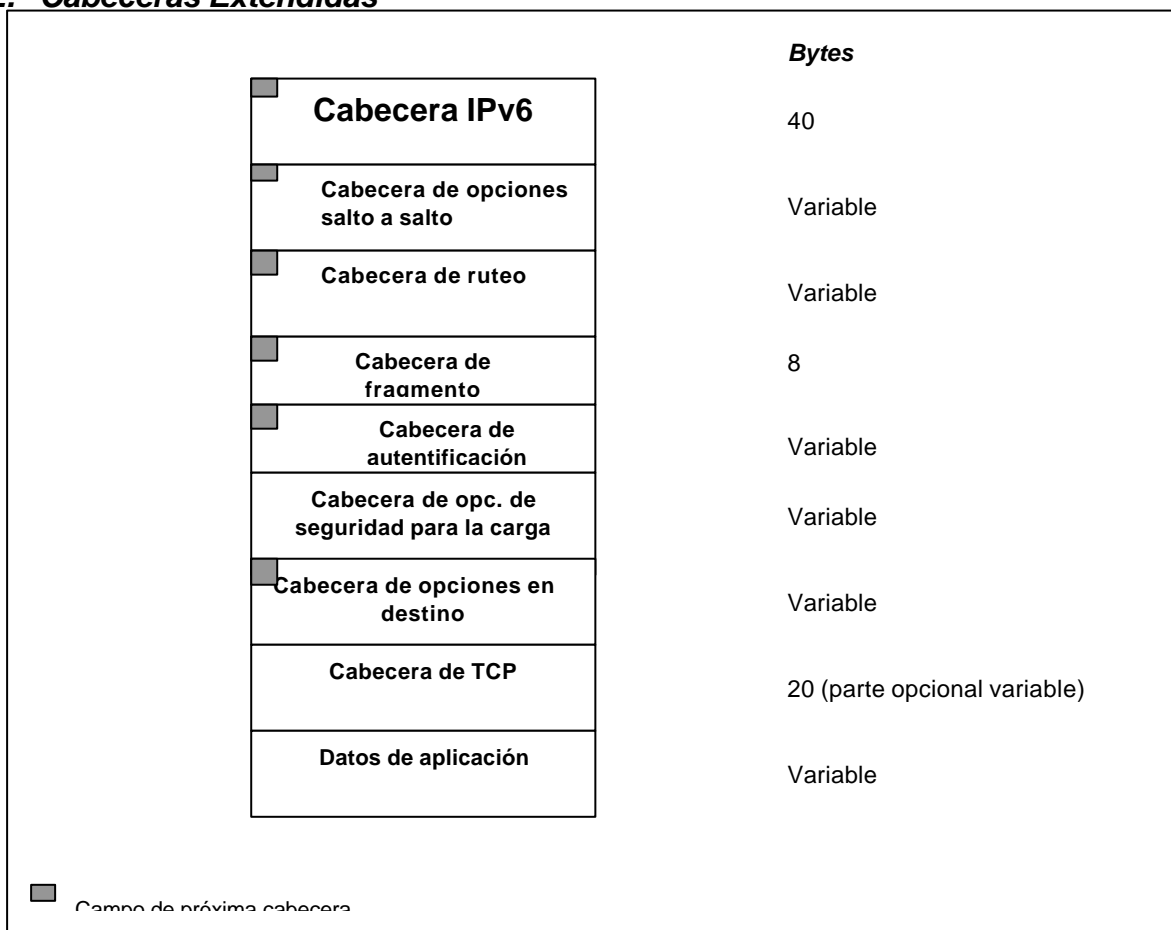


Fig. 10.a.7 Paquete IPv6 con todas las cabeceras de extensión

- Cabecera de opciones salto a salto: Define las opciones especiales que requiere el proceso salto a salto.
- Cabecera de ruteo : Provee ruteo extendido, similar al ruteo fuente de IPv4.
- Cabecera de fragmento : Contiene información de fragmentación y reensamble.
- Cabecera de autenticación : Provee autenticidad e integridad al paquete.
- Cabecera de opciones de seguridad para la carga : Provee privacidad.
- Cabecera de opciones en destino: Contiene información especial para ser examinada por el nodo destino.

El standard IPv6 recomienda que, cuando se usan muchas cabeceras de extensión, las cabeceras de IPv6 aparezcan en el siguiente orden :

- Cabecera IPv6 : Obligatoria, siempre debe aparecer primero.
- Cabecera de opciones salto a salto.
- Cabecera de opciones de destino : Para que las opciones se procesen por el primer destino que aparece en el campo de dirección de destino IPv6 más los destinos siguientes listados en la cabecera de ruteo.
- Cabecera de ruteo.
- Cabecera de fragmento .
- Cabecera de autenticación .
- Cabecera de opciones de seguridad para la carga .
- Cabecera de opciones en destino : Para que las opciones solo se procesen por el destinatario final del paquete.

La figura 10.a.7 muestra un ejemplo de un paquete IPv6 que incluye una instancia de cada cabecera. Nótese que la cabecera IPv6 y cada cabecera de extensión incluyen un campo para la cabecera siguiente. Este campo identifica el tipo de cabecera que sigue inmediatamente. Si la próxima cabecera es una extensión, entonces éste campo contiene el tipo de identificador de esa cabecera; sino el campo contiene el identificador de protocolo del protocolo de capa más alta que usa IPv6 (generalmente un protocolo de nivel de transporte), que usa los mismos valores que el campo de protocolo de IPv4.

En la figura 10.a.7., el protocolo de capa más alta es TCP, por lo que los datos de la capa más alta llevados por el paquete IPv6 constan de una cabecera TCP seguida por un bloque de datos de aplicación.

Opciones TLV (Type-Lenght-Value)

Dos de las cabeceras definidas (*salto a salto* y *opciones de destino*), llevan un número variable de opciones, las cuales a su vez tienen longitud variable (TLV, type-lenght-value). Estas opciones tienen la siguiente estructura :

Tipo de opción	Longitud datos opción	Datos
8 bits	8 bits	Long. Variable

Fig. 10.a.8

- Tipo de opción.
Este campo ocupa 1 byte, y actúa como identificador de cada opción específica.
- Longitud de los datos.
Este campo ocupa 1 byte, e indica la longitud del campo de datos, medida en bytes.
- Datos.

Este campo tiene una longitud variable, y contiene los datos específicos de cada opción.

La secuencia de opciones (ya que pueden aparecer varias), debe ser procesada en el orden en que aparezcan. El receptor no puede examinar la cabecera en busca de una opción y procesarla antes que las anteriores.

El campo *Tipo de opción* está codificado de tal forma que los dos bits de mayor peso especifican las acciones a tomar en caso que el nodo no reconozca la opción :

- 00 Descartar la opción y seguir procesando el paquete
- 01 Descartar el paquete entero
- 10 Descartar el paquete, y enviar un mensaje de error ICMP al origen.
- 11 Descartar el paquete, y enviar un mensaje de error ICMP al origen, si y sólo si
- el paquete no tiene una dirección destino *multicast*.

El tercer bit de mayor peso especifica si los datos específicos de la opción pueden cambiar durante el recorrido del paquete. Esto es útil cuando existe una cabecera de autenticación. Cualquier mecanismo de autenticación deberá tomar como 0's los datos que puedan cambiar en ruta. Los valores son :

- 0 Datos de la opción NO pueden cambiar en ruta.
- 1 Datos de la opción pueden cambiar en ruta.

Las diferentes opciones pueden tener diferentes requerimientos de alineamiento, estos requerimientos se especifican en la forma $xn+y$. Por ejemplo, $2n$ significa que la opción debe encontrarse desplazada del comienzo de la cabecera en un número de bytes múltiplo de 2.

Para mantener el alineamiento, existen dos opciones de relleno. Si sólo se requiere un byte, este se coloca todo a ceros. Si se necesita más de un byte, se usa la siguiente estructura:

Todo a 1's	Longitud relleno	Todo a 0's
8 bits	8 bits	(longitud relleno)-2 bytes

Fig. 10.a.9

La longitud se especifica en bytes, sin tener en cuenta el preámbulo y el campo de longitud, por tanto, el número de bytes en el relleno propiamente dicho es longitud-2.

Cabecera de opciones Salto a Salto

La cabecera extendida de *Opciones Salto a Salto* se usa para contener información que deberá ser examinada por cada nodo que encamine el paquete hacia su destino. Este tipo de cabecera se identifica con valor 0 en el campo *siguiente cabecera*.

Su formato es el siguiente :

Siguiente cabecera	Longitud opciones	Opciones
8 bits	8 bits	Long. Variable

Fig. 10.a.10

- Siguiente cabecera.

Este campo ocupa 1 byte, e identifica el tipo de cabecera existente inmediatamente después de la de *Opciones Salto a Salto*.

- Longitud opciones.

Este campo ocupa 1 byte, e indica la longitud de la cabecera, en bytes, sin incluir los ocho primeros.

- Opciones.

Este campo es de longitud variable. Además de las opciones con la estructura descrita, existe una opción especial, la *Carga Jumbo*. Con la siguiente estructura :

194 (identificador)	4 (long. Opciones)	Longitud Carga Jumbo
8 bits	8 bits	32 bits

Fig. 10.a.11

La opción *Carga Jumbo*, es utilizada para enviar paquetes con cargas superiores a los 65535 bytes. La longitud especificada por la *Carga Jumbo* es el tamaño total del paquete, excluyendo la cabecera IPv6, e incluyendo la cabecera de *Opciones Salto a Salto*.

La longitud determinada debe ser siempre superior a 65535, si se recibe un paquete con una *Carga Jumbo* que indique un tamaño de paquete igual o menor a 65535, ICMP se encargará de enviar un error.

Cada paquete cuya longitud esté especificada por un opción *Carga Jumbo*, debe tener a 0 el campo *longitud de la carga* en la cabecera IPv6, además, la opción *Carga Jumbo* no puede ser usada en un paquete conteniendo un fragmento. El incumplimiento de cualquiera de estas restricciones provocará un error ICMP.

Cabecera de ruteo

La *Cabecera de Ruteo*, es utilizada por el remitente para indicar uno o más nodos que el paquete debe visitar en su recorrido. Su formato es el siguiente :

Siguiente cabecera	Longitud cabecera	Tipo enrutamiento	Nodos restantes	Datos
8 bits	8 bits	8 bits	8 bits	Long.Variable

Fig. 10.a.12

- Siguiente cabecera.

Este campo ocupa 1 byte, e identifica el tipo de cabecera siguiente.

- Longitud opciones.

Este campo ocupa 1 byte, e indica la longitud de la cabecera, en bytes, sin incluir los ocho primeros.

- Tipo de ruteo.

Este campo ocupa 1 byte, e indica el tipo particular de cabecera de ruteo.

- Nodos restantes.

Este campo ocupa 1 byte, e indica el número de nodos que restan por visitar, siempre sobre los nodos marcados explícitamente.

- Datos.

Este campo tiene una longitud variable, siempre múltiplo de 8 (en bytes), y su formato viene determinado por el tipo de ruteo específico.

Si un nodo encuentra un paquete con un tipo de ruteo desconocido, tomará alguna de estas dos medidas :

- Si el número de nodos restantes es cero, se ignora la cabecera y se pasa a la cabecera siguiente.

- Si el número de nodos restantes NO es cero, se descarta el paquete y se enviará un error ICMP

El Tipo 0 (*tipo de ruteo* = 0) tiene el siguiente formato:

Sig. Cabecera	Long. Cab.	Tipo (= 0)	Nodos rest.	Reservado	Req. vecinos
---------------	------------	------------	-------------	-----------	--------------

(8 bits)	(8 bits)	(8 bits)	(8 bits)	(8 bits)	(24 bits)
Primera Dirección					
"					
[...]					
[...]					
n-ésima Dirección					
"					

Fig. 10.a.13

La longitud de la cabecera se especifica en unidades de 8 bytes, sin incluir los 8 primeros bytes, para el Tipo 0, es igual al doble de direcciones especificadas, y deber ser un número par menor o igual a 46, de igual forma, el máximo número de nodos que pueden especificarse es 23.

Existe un campo marcado como *reservado*, el remitente debe ponerlo a cero, y es ignorado por el receptor.

El campo *requerimiento vecinos* (*Strict / Loose Bit Map*) ocupa 24 bits, y es interpretado bit a bit, de izquierda a derecha, cada bit, indica si la dirección especificada debe corresponder a un nodo vecino del anterior. 1 significa que el nodo debe ser vecino del anterior, 0 significa que no ha de serlo necesariamente.

Las direcciones a visitar se especifican una tras otra, se numeran de 1 a n y pueden aparecer como máximo 23.

En el Tipo 0 no pueden aparecer direcciones *multicast*. Si un paquete IPv6 tiene como destino una dirección *multicast*, no puede contener una cabecera de ruteo de Tipo 0.

Cabecera de fragmento

La *Cabecera de fragmento* es utilizada por el origen del paquete IPv6 para enviar paquetes cuyo tamaño excede el mínimo MTU (*Maximum Transmission Unit*) en el camino del paquete. Al contrario que en IPv4, la fragmentación sólo la lleva a cabo el origen. La cabecera sigue el siguiente formato :

Sig. Cabecera	Reservado 1	Offset fragmento	Reservado 2	flag M	Identificación
8 bits	8 bits	13 bits	2 bits	1 bit	32 bits

Fig. 10.a.14

- Siguiete cabecera.

Este campo ocupa 1 byte, e indica el tipo de la cabecera siguiente.

- Reservado 1.

Este campo ocupa 1 byte. El origen lo pone en cero y es ignorado en destino.

- Offset de fragmento.

Este campo ocupa 13 bits, e indica, en unidades de 8 bytes, el desplazamiento respecto de la parte fragmentable del paquete original.

- Reservado 2.

Este campo ocupa 2 bits. El origen lo pone en cero y es ignorado en destino.

- Flag M.

Este campo ocupa un bit, es el flag de *más fragmentos*. Si 1, indica que quedan más fragmentos, si 0, indica que es el último fragmento.

- Identificación.

Este campo ocupa 32 bits, y sirve para identificar los fragmentos pertenecientes al datagrama original.

Cabecera de opciones en destino

Esta cabecera se usa para contener información que sólo debe ser examinada por el nodo destino. La cabecera tiene el siguiente formato :

Siguiente Cabecera	Longitud cabecera	Opciones
8 bits	8 bits	Long. Variable

Fig. 10.a.15

- Siguiete Cabecera.

Este campo ocupa 1 byte, e indica el tipo de la cabecera siguiente.

- Longitud cabecera.

Este campo ocupa 1 byte, e indica la longitud de la cabecera en unidades de 8 bytes, sin incluir los 8 primeros.

- Opciones.

Este campo tiene una longitud variable, siempre alineada a 8 bytes. Contiene una o más opciones de la estructura descrita en *Opciones TLV*

Hay dos posibles formas de codificar opciones TLV, como una opción en la cabecera *Opciones en destino*, o como una cabecera extendida aparte. Elegir una forma u otra dependerá de cual sea la acción deseada si el destino no entiende la información.

- o Si se quiere que el destino, en caso de no reconocer la opción, descarte el paquete y envíe un error ICMP (sólo si la dirección destino no es *multicast*). Entonces la opción deberá ser codificada en una cabecera extendida aparte.
- o Si se requiere cualquier otra acción, entonces la opción se codificará dentro de una cabecera *Opciones en destino*, y la acción especificada vendrá dada por los dos bit de mayor peso del campo *Tipo de opción*, en la forma descrita en el apartado *Opciones TLV*.

No existencia de más cabeceras.

El valor 59 en el campo *siguiente cabecera* de la cabecera IPv6 o extendida indica que no sigue ninguna cabecera.

2.3. Direcciones de IPv6

Las direcciones de IPv6 son de 128 bits. Las direcciones son asignadas a interfaces individuales sobre nodos, no a los nodos en si mismos. Una simple interfase puede tener muchas direcciones únicas *unicast*. Cualquiera de las direcciones *unicast* asociadas con la interfase de un nodo puede ser utilizada para identificar unívocamente al nodo.

La combinación de grandes direcciones y múltiples direcciones por interfase permite eficiencia de ruteo mejorado sobre IPv4. En IPv4, las direcciones generalmente no tienen una estructura que asiste al ruteo, y de esta manera un router puede necesitar mantener una tabla enorme de caminos de ruteo. Las direcciones largas de Internet permiten agregar direcciones por jerarquías de red, proveedor de acceso, geografía, corporación, etc. Tal agregado se debería hacer para tener tablas de ruteo más pequeñas y tablas de búsqueda más rápidas. Permitir múltiples direcciones por interfase dejaría que un suscriptor que usa proveedores de acceso múltiple a través de la misma interfase, tenga direcciones separadas agregadas bajo cada espacio de dirección del proveedor.

El primer campo de cualquier dirección IPv6 es el prefijo con formato de longitud variable, que identifica distintas categorías de direcciones. La tabla siguiente indica la asignación actual de direcciones basadas en el prefijo con formato.

Asignación de Espacio	Prefijo (binario)	Fracción de espacio de dirección
Reservado	0000 0000	1/256
Reservado para asignación de NSAP	0000 0001	1/256
Reservado para asignación de IPX	0000 001	1/128
No asignado	0000 011	1/128
No asignado	0000 1	1/32
No asignado	0001	1/16
No asignado	001	1/8
Dirección Unicast basada en el proveedor	010	1/8
No asignado	011	1/8
Reservado para direcciones Unicast basadas en geografía	100	1/8
No asignado	101	1/8
No asignado	110	1/8
No asignado	1110	1/16
No asignado	11110	1/32
No asignado	111110	1/64
No asignado	1111110	1/128
No asignado	11111110 0	1/512
Enlace a direcciones de uso local	11111110 10	1/1024
Sitio de direcciones de uso local	11111110 11	1/1024
Direcciones Multicast	11111111	1/256

Fig. 10.a.16

Pv6 permite tres tipos de direcciones :

- Unicast: Un identificador para una simple interfase. Un paquete enviado a una dirección unicast es entregado a la interfase identificada por esa dirección .
- Anycast: Un identificador para un conjunto de interfaces (generalmente pertenecientes a diferentes nodos). Un paquete enviado a una dirección anycast es entregado a alguna de las interfaces identificadas por esa dirección (la más cercana de acuerdo a las medidas de distancia de los protocolos de ruteo).
- Multicast: Un identificador para un conjunto de interfaces (generalmente pertenecientes a nodos diferentes). Un paquete enviado a una dirección Multicast es entregado a todas las interfaces identificadas por esa dirección.

Direcciones Unicast

Las direcciones unicast se pueden estructurar de distintas maneras. Se han identificado las siguientes posibilidades : Global basada en el proveedor, enlace local, sitio local, IPv6 compatible con IPv4, y loopback .

Una dirección unicast global basada en el proveedor provee direccionamiento global a través del universo entero de hosts conectados. La dirección tiene cinco campos después del prefijo de formato :

- ID de Registro: Identifica la autoridad del registro que asigna la porción del proveedor de la dirección.
- ID del Proveedor: Un proveedor específico de servicio de Internet que asigna la porción del suscriptor de la dirección.
- ID de Suscriptor: Distingue entre múltiples suscriptores conectados a la porción del proveedor de la dirección.
- ID de Subred: Un grupo de nodos conectados topológicamente dentro de la red del suscriptor.
- ID de Interfase : Identifica una interfase de nodo simple entre el grupo de interfaces identificadas por el prefijo de la subred.

Las *Direcciones de enlace local* se tienen que usar para direccionamiento en un enlace simple o subred. Ellas no se pueden integrar en el esquema global de direccionamiento. Ejemplos de su uso incluyen configuración de auto dirección, y descubrimiento de entorno.

Las *Direcciones de sitio local* son diseñadas para uso local pero formattedas de tal manera que luego se pueden integrar en el esquema de dirección global. La ventaja de éstas direcciones es que se pueden usar

inmediatamente por una organización que espera la transición para el uso de direcciones globales. La estructura de éstas direcciones es tal que la porción significativa de la dirección está limitada a los bits de bajo nivel no usados para direcciones globales. Los bits restantes constan de un prefijo con formato de uso local (1111 1110 10 o 1111 1110 11) seguidos por un campo en cero. Cuando una organización está lista para la conexión global, estos bits restantes se pueden reemplazar con un prefijo global (por ej. 010+ID de Registro+ID de Proveedor+ID de Suscriptor).

La siguiente figura muestra algunos formatos de dirección IPv6 .

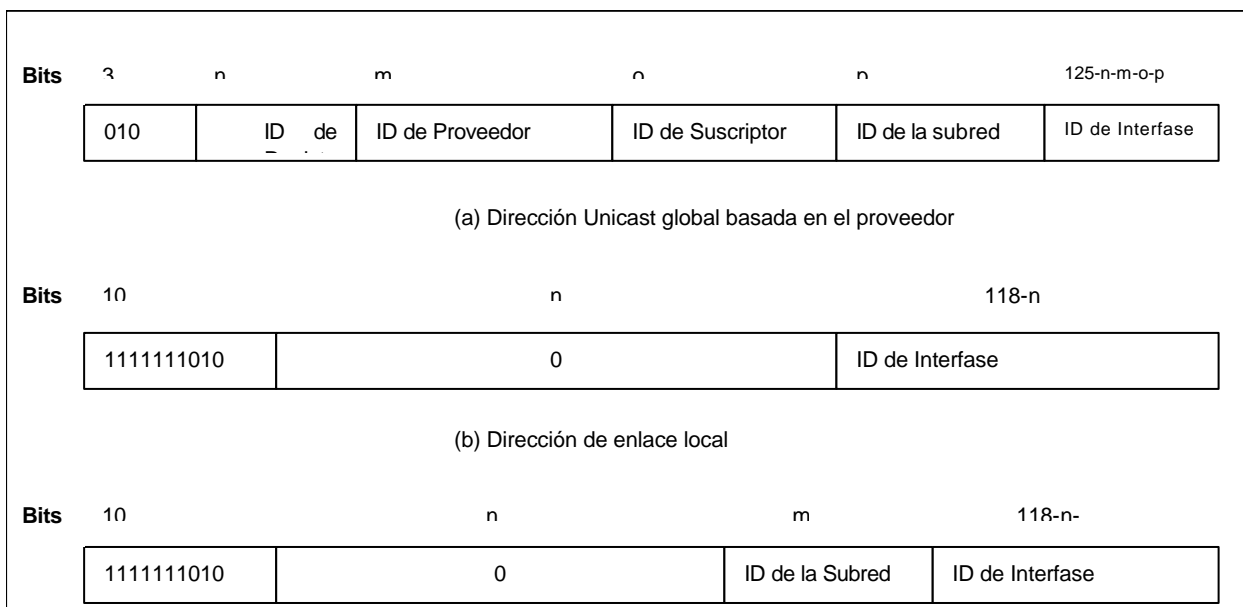


Fig. 10.a.17

En la actualidad, no hay longitud fija asignada a cualquiera de estos campos. Sin embargo, desde el punto de vista del administrador o diseñador responsable por la instalación de una red de un suscriptor, el ID de subred y el ID de interfase son las principales preocupaciones. El suscriptor puede tratar con estos campos de distintas maneras. Una posibilidad para una instalación basada en una LAN es usar una interfase de 48 bits y usar la dirección (MAC) para el control medio de acceso IEEE 802 para ese campo. El resto de los bits disponibles podrían ser el campo para ID de subred, identificando una LAN particular en el sitio del suscriptor.

IPv6 acomoda las direcciones unicast de uso local. Los paquetes con esas direcciones solo pueden ser ruteadas localmente, es decir, dentro de una subred o un conjunto de subredes de un suscriptor dado.

Se han definido dos tipos de direcciones de uso local: enlace local y sitio local.

Un tema clave de IPv6 es la transición de IPv4 a IPv6. No es práctico simplificar el reemplazo de todos los routers de IPv4 en Internet o en una red privada con routers IPv6 y reemplazar todas las direcciones IPv4 con direcciones IPv6. En cambio, habrá un largo período de transición donde IPv4 y IPv6 deban coexistir, por lo cual IPv4 e IPv6 deben seguir existiendo. Las direcciones IPv6 compatibles con IPv4 se ajustan a este período de coexistencia. Esta forma de dirección consta de una dirección IPv4 de 32 bits en los 32 bits de más bajo orden prefijados con 96 ceros. La dirección IPv6 compatible con IPv4 soporta una técnica conocida como *tunneling automático*. El Tunneling se puede emplear para promover el tráfico a través de las topologías de ruteo en IPv4. Esencialmente, un paquete IPv6 está encapsulado en una datagrama IPv4. Con el uso de una dirección IPv6 compatible con IPv4 para el nodo destino, el tunneling puede ser automatizado en el sentido que la dirección de destino IPv4 puede ser obtenida de la dirección IPv6, evitando la necesidad de configurar explícitamente el mapeo de una dirección IPv6 a una dirección IPv4.

Se puede mantener una coexistencia total si todos los nodos IPv6 emplean una dirección compatible con IPv4. Una vez que las direcciones IPv6 más generales se pongan en uso, la coexistencia será más difícil de mantener.

La dirección Unicast 0:0:0:0:0:0:0:1 se llama dirección loopback. Se puede usar por un nodo para enviarse un paquete IPv6 a sí mismo; tales paquetes no se van a enviar fuera de un simple nodo.

Direcciones Anycast

Una dirección anycast le permite a un emisor especificar que se quiere contactar con cualquier nodo de un grupo de nodos via una dirección simple. Un paquete con tal dirección será ruteado a la interfase más cercana del grupo, de acuerdo a las medidas de distancia del router. Un ejemplo del uso de una dirección anycast es dentro de una cabecera de ruteo para especificar una dirección intermedia a través de una ruta. La dirección anycast se podría referir al grupo de routers asociados con un proveedor particular o subred particular, dictando de este modo que el paquete sea ruteado a través de ese proveedor o de Internet de la manera más eficiente.

Las direcciones anycast son asignadas desde el mismo espacio de direcciones que las direcciones unicast. De este modo, los miembros de un grupo se deben configurar para reconocer esa dirección, y los routers se deben configurar para ser capaces de mapear una dirección anycast a un grupo de direcciones de interfase unicast.

Una forma particular de dirección anycast es la dirección anycast subred-router. El campo de prefijo de subred identifica una subred específica. Por ejemplo, en un espacio de dirección global basado en el proveedor, el prefijo de la subred tiene la forma (010+ID de Registro+ID de proveedor+ID de Suscriptor+ID de subred). De este modo, la dirección anycast es idéntica a una dirección unicast para una interfase en esta subred, con el ID de interfase puesto en cero. Cualquier paquete enviado a esta dirección será entregado a un router en la subred; todo lo que se requiere es insertar el ID de interfase correcto en la dirección anycast para formar la dirección destino unicast.

Direcciones Multicast

IPv6 incluye la capacidad para direccionar un grupo predefinido de interfaces con una simple dirección multicast. Un paquete con una dirección multicast se tiene que entregar a todos los miembros del grupo. Una dirección multicast consta de un prefijo con formato de 8 bits todos en uno, un campo con indicadores de 4 bits, un campo de alcance de 4 bits, y un ID de grupo de 112 bits.

En la actualidad, el campo de indicadores consiste en 3 bits en cero seguidos por un bit T con:

- T = 0: Indica una dirección multicast bien conocida o asignada permanentemente, asignada por la autoridad de numeración global de Internet.
- T = 1: Indica una dirección multicast no asignada permanentemente, o transitoria.

El valor de alcance se usa para limitar el alcance del grupo multicast. Los valores son :

0	Reservado
1	Nodo Local
2	Enlace Local
3	No asignado
4	No asignado
5	Sitio local
6	No asignado
7	No asignado
8	Organización local
9	No asignado
10	No asignado
11	No asignado
12	No asignado
13	No asignado
14	Global
15	Reservado

Fig. 10.a.18

El ID de grupo identifica un grupo multicast, en forma permanente o transitoria, dentro del alcance dado. En el caso de una dirección permanente multicast, la dirección misma es independiente del campo de alcance, pero ese campo limita el alcance de direccionamiento de un paquete particular. Por ejemplo, si al "grupo de servidores NTP" se le asigna una dirección multicast permanente con un ID de grupo de 43 (hex), luego :

FF05:0:0:0:0:0:43 significa todos servidores NTP en el mismo sitio que el emisor.

FF0E:0:0:0:0:0:43 significa todos los servidores NTP en Internet.

Las direcciones multicast asignadas en forma no permanente solo tienen sentido dentro de un alcance dado, permitiendo que el mismo ID de grupo sea reutilizado, con interpretaciones diferentes, en sitios distintos.

Multicasting posee una capacidad útil en muchos contextos. Por ejemplo, permite a los hosts y routers enviar mensajes para descubrimiento de entorno solo a aquellas máquinas que estén registradas

para recibirlos, quitando la necesidad de que todas las otras máquinas examinen y descarten paquetes irrelevantes. La mayoría de las LAN proveen una capacidad natural de broadcast. Una dirección multicast se le puede asignar para que tenga un alcance de enlace local con un ID de grupo configurado en todos los nodos de la LAN para ser una dirección broadcast de subred.

3. Seguridad en IPv6

La comunidad de Internet ha desarrollado mecanismos de seguridad específicos en distintas áreas de aplicación, incluyendo correo electrónico (Correo de Privacidad Ampliada, PGP), administración de red (Seguridad SNMPv2), acceso a web (HTTP seguro, Capa de sockets segura), y otros. Sin embargo, los usuarios tienen algunas inquietudes sobre seguridad que se dividen por las capas del protocolo. Por ejemplo, una empresa puede usar una red con TCP/IP en forma privada y segura deshabilitando los enlaces a los sitios no confiables, encriptando los paquetes que dejan las premisas, y autenticando los paquetes que ingresan las premisas. Implementando seguridad en el nivel de IP, una organización puede asegurar una red no solo para las aplicaciones que tienen mecanismos de seguridad sino que también para muchas aplicaciones que ignoran la seguridad.

La seguridad en el nivel de IP abarca dos áreas fundamentales: Autenticidad y Privacidad. El mecanismo de autenticidad asegura que un paquete recibido fue transmitido de hecho por la parte identificada como el origen en la cabecera del paquete. Inclusive, este mecanismo asegura que el paquete no ha sido alterado en tránsito. La facilidad de privacidad permite a los nodos comunicantes encriptar los mensajes para prevenir escuchas secretas por terceras partes.

En Agosto de 1995, el IETF publicó cinco estándares propuestos relacionados a la seguridad que definen una capacidad de seguridad en el nivel de internet. Los documentos incluyen:

- RFC 1825: Resumen de una arquitectura de seguridad.
- RFC 1826: Descripción de la extensión de autenticación del paquete a IP.
- RFC 1828: Un mecanismo específico de autenticación.
- RFC 1827: Descripción de la extensión de encriptado del paquete a IP.
- RFC 1829: Un mecanismo de encriptado específico.

El soporte para estas características es obligatorio para IPv6 y opcional para IPv4. En ambos casos, las características de seguridad se implementan como cabeceras de extensión que siguen a la cabecera principal de IP. La cabecera de extensión para la autenticidad se conoce como *cabecera de autenticación*; y para la privacidad, como *cabecera de opciones de seguridad para la carga (ESP)*.

3.1. Asociaciones de Seguridad

Un concepto clave que aparece en los mecanismos de autenticidad y privacidad para IP es las asociaciones de seguridad. Una asociación es una relación en una sola dirección entre un emisor y un receptor. Si se necesita una relación fija para intercambio seguro en dos direcciones, luego se necesitan dos asociaciones de seguridad.

Una asociación de seguridad es identificada unívocamente por una dirección de destino de internet y un índice de parámetro de seguridad (SPI). Luego, en cualquier paquete IP, la asociación de seguridad es unívocamente identificada por la dirección de destino en la cabecera IPv4 o IPv6 y el SPI en la cabecera de extensión incluida (cabecera de autenticación AH, o cabecera ESP).

Una asociación de seguridad normalmente se define por los siguientes parámetros:

- Algoritmo de autenticación y modo del algoritmo, usado con IP AH (requerido para implementaciones AH).
- Las clave(s) utilizadas con el algoritmo de autenticación en uso con AH (requerido para implementaciones AH).
- Algoritmo de encriptado, modo del algoritmo, y transformación usado con IP ESP (requerido para implementaciones ESP).
- Las clave(s) utilizadas con el algoritmo de encriptado en uso con ESP (requerido para implementaciones ESP).
- Presencia/Ausencia y tamaño de una sincronización criptográfica o el campo del vector de inicialización para el algoritmo de encriptado (requerido para implementaciones ESP).
- Algoritmo de autenticación y el modo usado con la transformación ESP, si alguno hay en uso (recomendado para implementaciones ESP).
- Las clave(s) de autenticación usadas con el algoritmo de autenticación que son parte de la transformación ESP, si hay alguna (recomendado para implementaciones ESP).
- Tiempo de vida para la clave o el momento cuando la clave debería cambiar (recomendado para todas las implementaciones).

- Tiempo de vida para esta asociación de seguridad (recomendado para todas las implementaciones).
- Direcciones origen de la asociación de seguridad; podría ser una dirección si más de un sistema emisor comparten la misma asociación de seguridad con el destino (recomendada para todas las implementaciones).
- Nivel de sensibilidad (por ej. secreto o no clasificado) de los datos protegidos (requeridos para todos los sistemas solicitando que se les provea seguridad multinivel , recomendado para todos los otros sistemas)

El mecanismo de administración de claves usado para distribuir las se une a los mecanismos de privacidad y autenticidad solo por el camino del SPI. Aquí, la autenticidad y privacidad se han especificado independientemente de cualquier mecanismo específico de administración de claves.

3.2. Autenticidad

La cabecera de autenticación provee soporte para la integridad de los datos y la autenticidad de los paquetes IP. La cabecera de autenticación consta de los siguientes campos (ver la siguiente figura) :

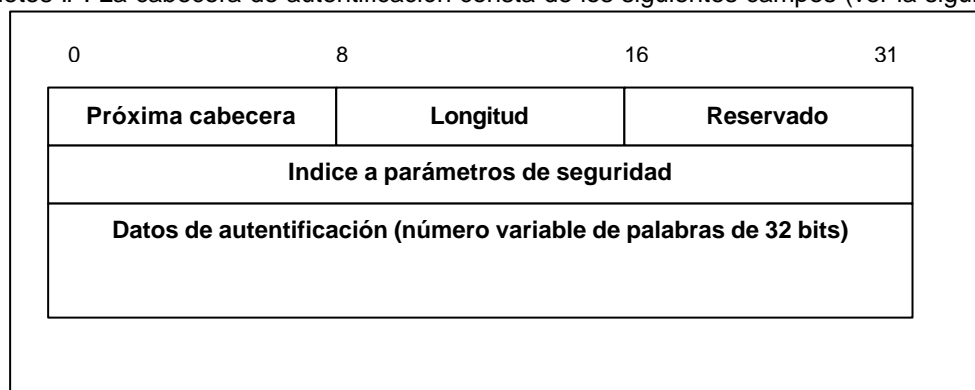


Fig. 10.a.19. **Cabecera de Autenticación**

- Próxima cabecera (8 bits): Identifica el tipo de cabecera que sigue inmediatamente a esta cabecera.
- Longitud (8bits): Longitud del campo de datos de autenticación en palabras de 32 bits.
- Reservado (16 bits): Para uso futuro.
- Índice a parámetros de seguridad (32 bits): Identifica una asociación de seguridad.
- Datos de autenticación (variable): Un número entero de palabras de 32 bits.

Los contenidos del campo de datos de autenticación dependerán del algoritmo especificado de autenticación. En cualquier caso, los datos de autenticación se calculan sobre el paquete entero de IP, excluyendo cualquiera de los campos que pueden cambiar en tránsito. Tales campos se ponen en cero con propósitos de cálculo tanto en el emisor como en el destinatario. El cálculo de autenticación se realiza antes de la fragmentación en el emisor y luego del reensamblaje en el destino; así pues, los campos relacionados con la fragmentación se pueden incluir en el cálculo.

Para IPv4 el tiempo de existencia y los campos de verificación de la cabecera están sujetos a cambio y de esta forma a ponerse en cero para el cálculo de autenticación. Las opciones IPv4 se deben manejar de acuerdo a la regla que cualquier opción cuyo valor puede cambiar durante el tránsito no debe ser incluida en el cálculo.

Para IPv6, el campo límite de salto es el único campo en la cabecera base de IPv6 sujeto a cambio; de esta manera es puesto a cero para el cálculo. Para las cabeceras de opciones de destino y salto a salto , el campo tipo de opción para cada opción contiene un bit que indica si el campo de datos de la opción para esta opción puede cambiar durante el tránsito; si puede, esta opción se excluye del cálculo de autenticidad.

La autenticación utilizando *Keyed MD5 – RFC 1828* especifica el uso de *MD5* para la autenticación. El algoritmo *MD5* es ejecutado sobre el paquete IP más una clave secreta por el emisor y luego insertado en el paquete IP. En el destinatario, se realiza el mismo cálculo en el paquete IP más la clave secreta y se lo compara con el valor recibido. Este procedimiento provee tanto autenticidad como privacidad de los datos.

Específicamente, el cálculo de *MD5* es realizado sobre la siguiente secuencia :

Clave, completar la clave, paquete IP, clave, completar MD5

Donde

- Clave: es la clave secreta para esta asociación de seguridad .

- Completar la clave: relleno para que la clave + la clave completada sea un entero múltiplo de 512 bits.
- Paquete IP: Con los campos apropiados puestos en cero.
- Completar MD5: relleno provisto por MD5 para que el bloque entero sea un entero múltiplo de 512 bits.

El servicio de autenticación de IP se puede usar de distintas maneras. La autenticidad se puede proveer directamente entre las estaciones de trabajo del cliente y el servidor; el cliente puede estar en la misma red que el servidor o en una red externa. Si el cliente y el servidor comparten una clave secreta proegida, el proceso de autenticación es seguro. En forma alternativa, un cliente remoto se autentifica por si mismo al firewall corporativo, ya sea por acceso a la red interna entera o debido a que el servidor solicitado no soporta la característica de autenticidad.

Opciones de seguridad para la carga

El uso de ESP provee soporte para la privacidad e integridad de los datos para los paquetes IP. Dependiendo de los requerimientos del usuario, este mecanismo se puede usar para encriptar un segmento de la capa de transporte, conocido como modo de transporte ESP, o se puede encriptar un paquete entero de IP, conocido como modo tunnel ESP.

La cabecera ESP comienza con un SPI de 32 bits, que identifica una asociación de seguridad. El resto de la cabecera, si hay, puede contener parámetros dependiendo del algoritmo de encriptado que se usa. En general, la primera parte de la cabecera, incluyendo el SPI y posiblemente algunos parámetros, se transmite en forma no encriptada (texto plano), mientras que el resto de la cabecera, si hay, se transmite en forma encriptada (texto cifrado).

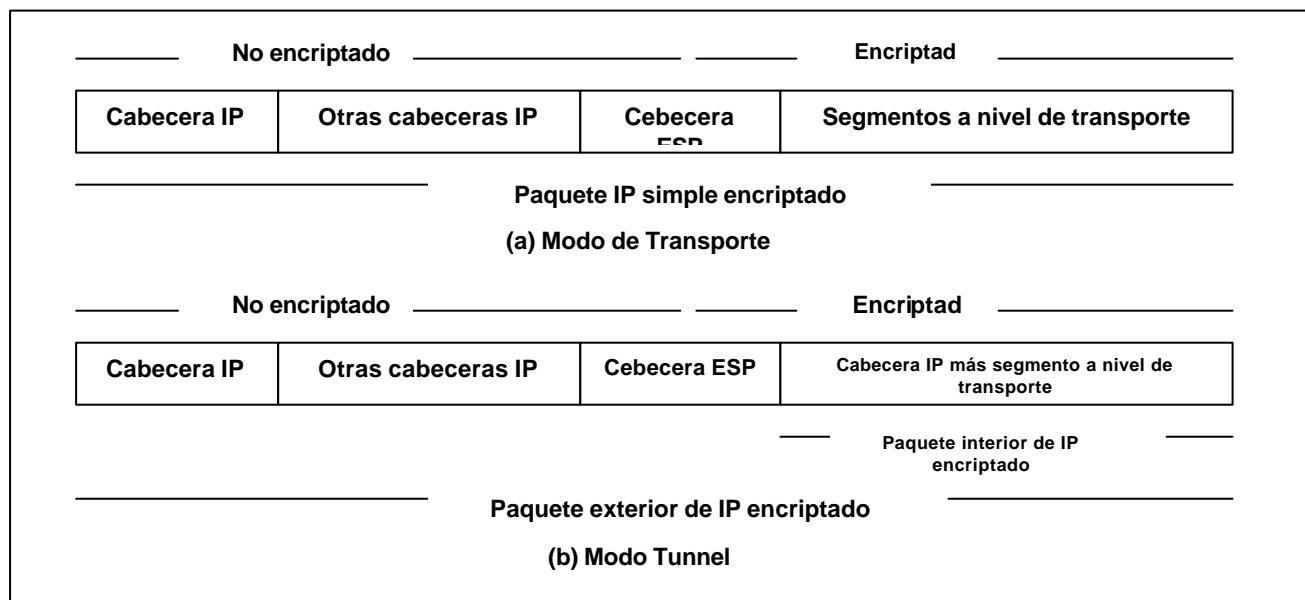


Fig. 10.a.20 Paquete IPv6 o datagrama IPv4

Modo de transporte ESP – Se usa para encriptar los datos transportados por IP. Comúnmente, estos datos están en el segmento de la capa de transporte, tal como el segmento TCP o UDP, los cuales en cambio contienen datos a nivel de aplicación. Para este modo, la cabecera ESP se inserta en el paquete IP inmediatamente antes de la cabecera de capa de transporte (por ej. TCP, UDP, ICMP). En el caso de IPv6, si hay presente una cabecera de opciones de destino, la cabecera ESP se inserta inmediatamente antes de la misma.

La operación *Modo de Transporte* se puede resumir como sigue :

- En el emisor, el bloque de datos que consiste en una parte traída de la cabecera ESP más el segmento de la capa de transporte es encriptado, y el texto plano de este bloque es reemplazado con su texto cifrado para formar el paquete IP para la transmisión.
- Este paquete luego se rutea al destino. Cada router intermedio necesita examinar y procesar la cabecera IP más cualquiera de las cabeceras de extensión IP en texto plano, pero no necesita examinar el texto cifrado.
- El nodo destino examina y procesa la cabecera IP más cualquiera de las cabeceras de extensión IP en texto plano. Luego, en base a SPI en la cabecera ESP, el nodo destino desencripta el resto del paquete para recuperar el segmento de la capa de transporte en texto plano.

La operación en modo de transporte provee privacidad para cualquier aplicación que la use, evitando de este modo la necesidad de implementar privacidad en cada aplicación individual. Este modo de operación también es razonablemente eficiente, agregándole poco a la longitud del paquete IP. Una desventaja de este modo es que es posible hacer análisis de tráfico en los paquetes transmitidos.

Modo tunnel ESP – El modo tunnel se usa para encriptar un paquete entero IP. Para este método, ESP es prefijada al paquete y luego se encripta el paquete más una parte traída de la cabecera ESP. Este método se puede usar para contrariar el análisis de tráfico .

Debido a que la cabecera IP contiene la dirección destino y posiblemente directivas de ruteo de origen e in formación de opción salto a salto, no es posible transmitir en forma simple el paquete IP encriptado prefijado por la cabecera ESP. Las rutas intermedias serían incapaces de procesar tal paquete. De esta manera, es necesario encapsular el bloque entero (la cabecera ESP más el paquete IP encriptado) con una nueva cabecera IP que contendrá información suficiente para el ruteo pero no para el análisis de tráfico.

Mientras que el modo de transporte es adecuado para proteger las conexiones entre los hosts que soportan la característica ESP, el modo tunnel es de utilidad en una configuración que incluye un firewall u otra clase de puerta de seguridad que protegen una red confiable de las redes externas. En este último caso, la encriptación ocurre solo entre un host externo y la puerta de seguridad o entre dos puertas de seguridad. Esto alivia a los hosts en la red interna de la sobrecarga del proceso por encriptado y también simplifica la tarea de distribución de clave reduciendo el número de claves necesarias. Aún más, combate el análisis de tráfico basado en ultimar el destino.

Considerar el caso en que un host externo desea comunicarse con un host en una red interna protegida por un firewall, y en la cual se implementa ESP en el host externo y en los firewalls. Los siguientes pasos ocurren para transferir un segmento a nivel de transporte del host externo al interno :

- El emisor prepara un paquete IP interior con una dirección destino del host interno de destino. El paquete es prefijado con una cabecera ESP; luego se encriptan el paquete y una parte de la cabecera ESP. El bloque resultante se encapsula con una nueva cabecera IP (cabecera base más extensiones opcionales tales como opciones de ruteo o salto a salto) cuya dirección de destino es el firewall ; esto forma el paquete exterior.
- El paquete exterior se rutea al firewall de destino. Cada router intermedio necesita examinar y procesar la cabecera IP exterior y cualquiera de las cabeceras de extensión IP, pero no necesitan examinar el texto cifrado.
- El firewall de destino examina y procesa la cabecera exterior IP y cualquiera de las cabeceras de extensión IP. Luego, en base a SPI en la cabecera ESP, el nodo destino descripta el resto de paquete para recuperar el paquete interior IP en texto plano. Este paquete es transmitido entonces en la red interna.
- El paquete interior es ruteado a través de ninguno o más routers en la red interna al host destino.

3.3. Autenticidad más privacidad

Los dos mecanismos de seguridad de IP se pueden combinar para transmitir un paquete IP que tenga tanto privacidad como autenticidad. Se pueden usar dos alternativas, basadas en el orden en que se aplican los dos servicios.

Encriptado antes de la autenticación – El paquete IP entero que se transmitió es autenticado, incluyendo tanto partes encriptadas como no encriptadas. En este caso, el usuario primero aplica ESP para que los datos esten protegidos, luego escribe la cabecera de autenticación y la(s) cabecera(s) IP de texto plano. Actualmente existen dos posibilidades :

- ESP en modo transporte : La autenticación aplica al paquete entero de IP entregado al último destino, pero solo el segmento de la capa de transporte se protege con el mecanismo de privacidad (encriptado).
- ESP en modo tunnel : La autenticación aplica al paquete entero IP entregado a la dirección exterior de destino IP (por ej. un firewall), y la autenticación es realizada en el destino. El paquete interior entero de IP es protegido con el mecanismo de privacidad, para entregarlo al destino interior de IP.

Encriptación después de la autenticación – Solo es apropiado para ESP en modo tunnel. En este caso, la cabecera de autenticidad es ubicada dentro del paquete interior IP. Este paquete interior IP es tanto autenticado y protegido con el mecanismo de privacidad.

Las funciones de autenticación y encriptado se pueden aplicar en cada orden para ESP en modo tunnel. El uso de la autenticación antes del encriptado podría ser preferible por distintas razones. Primero, puesto que AH es protegido por ESP, es imposible para cualquiera interceptar el mensaje y alterar a AH sin detección. Segundo, puede ser deseable almacenar la información de autenticación con el mensaje y el destino para una posterior referencia. Es más conveniente hacer esto si la información de autenticación se aplica al mensaje no encriptado; de otra forma, el mensaje debería tener que ser reencriptado para verificar la información de autenticación.

3.4. Administración de Clave IPv6

La arquitectura de seguridad de IPv6 y los mecanismos de seguridad relacionados se diseñaron de tal manera que los usuarios de Internet que quieren seguridad, tendrán esos mecanismos, y los usuarios que no necesitan estos mecanismos, no resulten afectados adversamente. En IPv6 el protocolo de administración de clave está unido al resto de los mecanismos de seguridad solo vía la Asociación de Seguridad (SA) implementada con la Cabecera de Autenticación (AH) y la Cabecera de Opciones de Seguridad para la Carga (ESP). La comunidad de Internet aún no ha acordado sobre un protocolo de administración de clave IPv6 y hay distintas propuestas disponibles para la administración de claves.

IPv6 no intenta soportar la administración de clave denominada “en banda”, donde los datos de administración de clave se llevan en una cabecera IPv6 distinta. En cambio, principalmente usará la administración de clave denominada “fuera de banda”, donde los datos de administración de clave se llevarán por un protocolo de capa superior tal como UDP o TCP. Esto permite una clara separación del mecanismo de administración de clave del resto de los mecanismos de seguridad, por lo que es posible reemplazar el método de administración de clave por otro sin tener que modificar las implementaciones de los otros mecanismos de seguridad.

En la administración de clave interesan temas como Generación de claves, Transferencia de claves, Verificación de claves, Uso de las claves, Actualización de las claves, Almacenamiento de las claves, Back-up de las claves, Tiempo de vida de las claves, Destrucción de las claves, etc..

Las propiedades de los protocolos de intercambio de clave se definen para incluir el método de establecimiento de clave, autenticidad, simetría, secretidad de reenvío perfecta, y protección de tráfico de vuelta.

En el establecimiento de la clave, existen dos métodos comunes que usan el encriptado de la clave pública: Transporte de clave y Generación de clave. Un ejemplo de transporte de clave es el uso del algoritmo RSA, para encriptar una clave de sesión generada aleatoriamente con la clave pública de la otra parte. La clave aleatoria encriptada, luego se envía al receptor, quien la desencripta usando su clave privada. En este momento, ambos lados tienen la misma clave de sesión. Un ejemplo de generación de clave es el uso del algoritmo Diffie-Hellman (DH), que genera una clave de sesión basada en información pública y secreta mantenida por ambos usuarios. El algoritmo DH es comenzado por dos partes que intercambian información pública. Cada parte luego combina matemáticamente la información pública del otro con su propia información secreta para calcular un valor secreto compartido. Este valor secreto se puede utilizar como una clave de sesión o una clave para encriptar una clave de sesión generada al azar.

La autenticidad en el intercambio de la clave se puede hacer durante el protocolo o luego del final del protocolo. La autenticación del intercambio de clave durante el protocolo se provee cuando cada parte demuestra tener la clave secreta de sesión antes del fin del protocolo. La prueba se puede dar encriptando los datos conocidos en la clave secreta de sesión durante el intercambio del protocolo. La autenticación luego del protocolo debe ocurrir en las siguientes comunicaciones. La autenticación durante el protocolo se prefiere por que las comunicaciones siguientes no se inician si no se establece la clave secreta de sesión con la parte deseada.

Se logra una simetría deseable en el intercambio de clave si cada parte puede iniciar el intercambio y los mensajes intercambiados pueden viajar sin afectarse la clave ya generada.

La Secreticidad de Reenvío Perfecta es provista por un protocolo de intercambio de clave si el revelado del material de encriptado de clave a largo plazo no compromete a las claves generadas previamente. Las claves de sesión del pasado no estarán obtenibles si la clave a largo plazo es acordada con secreticidad de reenvío perfecta.

La protección del tráfico de vuelta es provisto por la generación independiente de cada clave de tal forma que las claves siguientes no son dependientes de cualquier clave previa. Las claves de sesión pasada no serán obtenibles si la clave de sesión actual es acordada con protección del tráfico de vuelta.

La forma más simple de administración de clave, es la administración de clave manual, donde una persona configura manualmente cada sistema con su propia clave y la de los otros. El método puede ser muy práctico en los medios pequeños y estáticos.

Existen dos alternativas para el manejo de claves en IPv6. En el manejo de claves de host a host, todos los usuarios de un host, comparten la misma clave a ser usada con todos los usuarios del otro de host de comunicación. La otra alternativa es de usuario a usuario, donde cada usuario tiene una única clave de sesión.

Todas las implementaciones IPv6 deben soportar administración de clave manual y deberían soportar un protocolo de Internet standard de administración de clave una vez que se aprueba. Todas las implementaciones IPv6 deben permitir la configuración y el uso de resolución de claves usuario a usuario para el tráfico que se origina en ese sistema y adicionalmente pueden permitir la configuración del manejo de claves host a host para el tráfico que se origina en ese sistema como una característica adicionada para hacer más fácil la distribución de la clave manual y darle al administrador del sistema mayor flexibilidad.

Un dispositivo que encripta o autentica los paquetes IPv6 originados en otros sistemas, por ejemplo un encriptador de IP dedicado o una puerta de encriptado, generalmente no puede proveer resolución de claves usuario a usuario para el tráfico que se origina en otros sistemas. De esta manera, dichos sistemas deben implementar soporte para el manejo de claves usuario a usuario para el tráfico que se origina en los demás sistemas. El método con el cual se configuran las claves en un sistema particular se define en la implementación.

4. Capacidades de Calidad de Servicio

Los campos prioridad y etiqueta de flujo en la cabecera IPv6 se pueden usar por un host para identificar aquellos paquetes para los que se requiere un manejo especial por los routers IPv6, tales como calidad de servicio no por defecto o servicio en “tiempo real”. Esta capacidad es importante para soportar aplicaciones que requieran algún grado de rendimiento consistente y demora. Estos tipos de aplicaciones son conocidos comunmente como aplicaciones “multi-media” o aplicaciones en “tiempo real”.

Etiquetas de flujo

La etiqueta de flujo de 24 bits en la cabecera IPv6 se puede usar por un emisor para etiquetar aquellos paquetes para los que requiere un manejo especial por los routers IPv6, tales como calidad no standard de servicio o servicio en tiempo real.

Los hosts o routers que no soportan las funciones del campo etiqueta de flujo requieren poner el campo en cero cuando generan un paquete, pasar el campo como no modificado cuando reenvían un paquete, e ignorar el campo cuando reciben un paquete.

Un flujo es una secuencia de paquetes enviados de un emisor particular a un destino particular (unicast o multicast) para el cual el emisor desea un manejo especial por los routers intervinientes. La naturaleza de ese manejo especial podría ser llevados a los routers por un protocolo de control, tal como un protocolo de reserva de recurso, o por información dentro de los paquetes de flujo mismos, por ej. en una opción salto a salto.

Podrían existir tanto múltiples flujos activos de un emisor a un destinatario como también tráfico que no se asocia con cualquier flujo. Un flujo es unívocamente identificado por la combinación de una dirección emisora y una etiqueta de flujo distinta de cero. Los paquetes que no pertenecen a un flujo llevan una etiqueta de flujo en cero.

Una etiqueta de flujo es asignada a un flujo por el nodo emisor del flujo. Se deben elegir nuevas etiquetas de flujo (pseudo-) aleatoriamente y uniformemente del rango 1 a FFFFFFF hex. El propósito de la asignación aleatoria es hacer que cualquier conjunto de bits dentro del campo etiqueta de flujo sean adecuados para el uso como una clave de hash por los routers, para buscar el estado asociado con el flujo.

Todos los paquetes que pertenecen al mismo flujo se deben enviar con la misma dirección de origen, la misma dirección destino, y la misma etiqueta de flujo distinta de cero. Si alguno de los paquetes incluye una cabecera de opciones salto a salto, luego todos ellos deben ser generados con los mismos contenidos en la cabecera de opciones salto a salto (excluyendo el campo próxima cabecera de la cabecera de opciones salto a salto). Si alguno de esos paquetes incluyen una cabecera de ruteo, luego todos deben ser generados con los mismos contenidos en todas las cabeceras de extensión, incluyendo la cabecera de ruteo (excluyendo el campo próxima cabecera de la cabecera de ruteo). Se les permite a los routers o destinatarios, pero no se les requiere, verificar que estas condiciones se satisfagan. Si se detecta una violación, se debería reportar al emisor por un mensaje ICMP (Problema de Parámetro), Código 0, apuntando al byte de mayor orden del campo etiqueta de flujo (por ej., desplazamiento 1 dentro del paquete IPv6).

Los routers son libres de establecer “oportunamente” el estado de manejo de flujo para cualquier flujo, aún cuando no se les ha provisto de ninguna información explícita de establecimiento de flujo via un protocolo de control, una opción de salto a salto, u otros. Por ejemplo, hasta recibir el paquete de un emisor particular con una etiqueta de flujo desconocida distinta de cero, un router puede procesar su cabecera IPv6 y cualquier cabecera de extensión necesaria como si la etiqueta de flujo fuera cero. Ese procesamiento incluiría la determinación de la interfase de próximo salto, y posiblemente otras acciones, tales como actualizar una opción salto a salto, avanzar el puntero y direccionar una cabecera de ruteo, o decidir como poner en la cola el paquete basado en su campo Prioridad. El router entonces puede elegir “recordar” los resultados de aquellos pasos de procesamiento y almacenar esa información en un cache , utilizando la dirección de origen más la etiqueta de flujo como la clave para acceder al cache. Los paquetes siguientes con la misma dirección origen y etiqueta de flujo pueden ser manejados refiriéndose a la información del cache en vez de examinar todos aquellos campos que, de acuerdo a los requerimientos del párrafo anterior, se pueden asumir como no modificables desde el primer paquete visto en el flujo.

Prioridad

El campo Prioridad de 4 bits en la cabecera IPv6 le permite a un emisor identificar la prioridad de entrega deseada de sus paquetes, relativa a otros paquetes del mismo origen. Los valores de Prioridad se dividen en dos rangos: Valores entre 0 y 7 se usan para especificar la prioridad o el tráfico para el cual el emisor provee control de congestión, por ej. , tráfico que “vuelve atrás” en respuesta a la congestión, como tráfico TCP. Los valores entre 8 y 15 se usan para especificar la prioridad o tráfico que no vuelve atrás en respuesta a la congestión, por ej. , paquetes enviados en forma constante de “tiempo real”.

Para el tráfico con congestión controlada, los siguientes valores de prioridad se recomiendan para categorías de aplicación particular :

0	Tráfico no caracterizado
1	Tráfico de “relleno”
2	Transferencia de datos no atendidos (por ej. e-mail)
3	(Reservado)
4	Transferencia en volumen atendida (por ej. FTP, HTTP, NFS)
5	(Reservado)
6	Tráfico interactivo (por ej. telnet, X)
7	Tráfico de control Internet (por ej. protocolos de ruteo, SNMP)

Fig. 10.a.21

Para el tráfico sin congestión controlada, el valor de Prioridad más bajo (8) se debería usar para aquellos paquetes que el emisor está más interesado que se descarten bajo condiciones de congestión (por ej. tráfico de video de alta fidelidad), y el valor más alto (15) se debería usar para aquellos paquetes que el emisor está menos interesado que se descarte (por ej. tráfico de audio de baja fidelidad). No existe un orden relativo denotado entre las prioridades con congestión controlada y sin congestión controlada.

5. La Transición a IPv6

Poco se discutiría en la industria con el principio que IPv6 representa un gran salto para Internet y las empresas que transmiten con tecnología internetworking. IPv6 mejora a IPv4 en muchas áreas a corto y largo plazo para los negocios dependientes de redes. Con lo que no existe acuerdo en la industria, sin embargo, es que forma y velocidad tendrá la transición de IPv4 a IPv6. Algunos están apremiados por una comercialización y una rápida adopción de IPv6 en un futuro muy cercano. Otros prefieren dejar esperar para el proyecto IPv6 hasta que se agote el espacio de dirección y otros aspectos fuercen a la conversión. Pero dada la magnitud de una migración que afecta muchos millones de dispositivos de red, está claro que habrá un período extendido donde IPv4 y IPv6 coexistirán en muchos niveles de Internet.

Con la realidad de la aparición de una coexistencia extendida IPv4/IPv6, los diseñadores de protocolos IETF han gastado una cantidad sustancial de esfuerzo para asegurar que los hosts y los routers se pueden mejorar a IPv6 en forma fácil e incremental. Se han obtenido grandes disgustos al asegurar que la transición no implicará la obsolescencia de los nodos IPv4 en gran escala o de las mejoras “fork-lift” para las poblaciones de usuarios en un marco a corto plazo. Los mecanismos de transición se han diseñado para permitir a los administradores de red una gran cantidad de flexibilidad en como y cuando ellos mejoran los hosts y los nodos intermedios. Consecuentemente, IPv6 se puede desplegar primero en los hosts, o en los routers, o alternativamente, en un número limitado de hosts y routers adyacentes o remotos. Los nodos

que son mejorados inicialmente no se tiene que colocar en la misma red de área local o en el mismo terreno.

Otra suposición hecha por los diseñadores de la transición a IPv6 es la probabilidad que muchos hosts y routers mejorados necesiten retener la compatibilidad con los dispositivos IPv4 por un período de tiempo extendido (posiblemente años o aún indefinidamente). También se asumió que los dispositivos mejorados deberían tener la opción de retener su direccionamiento a IPv4. Para lograr estos objetivos, la transición a IPv6 cuenta con distintas funciones especiales que se han construido en el trabajo de estándares para IPv6, incluyendo hosts y routers de capa dual y tunnelling IPv6 via IPv4.

5.1. El método de transición con capa dual

Una vez que unos pocos nodos se han convertido a IPv6, existe una fuerte posibilidad que estos nodos necesiten continua interacción con los nodos existentes IPv4. Esto se logra con la alternativa IPv4/IPv6 en capa dual. Una gran cantidad de hosts y routers en los entornos de red multiplataforma hoy ya soportan múltiples componentes de capas de red. Por ejemplo, la mayoría de los routers en las redes de las empresas son de multiprotocolo variado. Asimismo, muchas estaciones de trabajo ejecutan alguna combinación de IPv4, IPX, AppleTalk, NetBIOS, SNA, DECnet, u otros protocolos. La inclusión de un protocolo adicional (IPv6) en una estación final o en un router es un claro emprendimiento trivial hoy en día. Cuando se ejecuta una capa dual IPv4/IPv6, un host tiene acceso tanto a los recursos de IPv4 como a los de IPv6. Los routers que ejecutan ambos protocolos pueden reenviar el tráfico para los nodos finales de IPv4 e IPv6.

Las máquinas de capa dual pueden usar direcciones IPv4 e IPv6 totalmente independientes, o se pueden configurar con una dirección IPv6 que es compatible con IPv4. Los nodos de capa dual pueden usar los servicios de autoconfiguración convencionales de IPv4 (DHCP) para obtener sus direcciones IPv4. Las direcciones IPv6 se pueden configurar manualmente en las tablas locales del host de 128 bits, u obtenidos por los mecanismos de autoconfiguración de IPv6, cuando esten disponibles. Se espera que los principales servidores ejecuten indefinidamente en modo capa dual, o hasta que todos los nodos activos se conviertan a IPv6.

5.2. Escenarios de Transición

La flexibilidad y utilidad de los mecanismos de transición IPv6 son mejor evaluados a través de los escenarios que direccionan los requerimientos del trabajo en redes del mundo real.

Escenario 1: No existe necesidad de NAT

Toma, por ejemplo, el caso de dos grandes organizaciones dependientes de la red que deben interactuar con operaciones debido a la combinación y adquisición (M&A), o una nueva sociedad de negocios. Ambas empresas en este escenario tienen grandes redes basadas en IPv4 que han crecido desde pequeñas en sus orígenes. Ambas empresas tienen un número sustancial de direcciones privadas IPv4 que no necesariamente son únicas dentro del actual espacio de dirección global de IPv4. Al combinar estos dos espacios de direcciones no únicos se podría requerir una costosa reenumeración y reestructuración de los routers, las direcciones del host, los dominios, las áreas, los protocolos de ruteo exterior, etc.. Este escenario es muy común en el ambiente actual de negocios, no solo para proyectos M&A, sino que también para grandes outsourcing y relaciones de trabajo en redes Cliente / Proveedor, donde muchos de los hosts del padre, outsourcer, proveedor, o parte se deben integrar a la estructura de direcciones existente en la empresa. Independientemente del escenario, IPv6 es una excelente alternativa para este cambio.

La tarea de fusionar lógicamente dos redes de empresas en un simple dominio autónomo es un proyecto potencialmente destructivo y caro. Para evitar el costo y la alteración de la reenumeración asimilativa, las empresas pueden ser seducidas para optar por la solución de un traductor de dirección de red (NAT). En el caso del escenario M&A, un NAT podría permitir que las dos empresas mantengan sus direcciones privadas. Para lograr esto, un NAT debe conducir la traducción de la dirección en tiempo real para todos los paquetes que se mueven entre las dos organizaciones. Desafortunadamente, esta solución trae muchos problemas asociados con el uso de NATs, incluyendo la performance al ser cuellos de botella, falta de mejora gradual, falta de estándares, y falta de conectividad universal entre todos los nodos de la nueva empresa e Internet.

A diferencia de NAT, IPv6 provee una robusta solución "orientada al futuro" para la integración lógica de dos redes físicas. Para facilitar la discusión, las dos empresas originalmente independientes se conocerán como empresa A y B. El primer paso es determinar que hosts necesitan acceder a ambos lados de la nueva organización. Estos hosts están equipados con IPv4/IPv6 de capa dual, que les permiten

mantener conectividad a su red original IPv4 mientras también participan en una nueva red lógica que será creada “por encima” de la infraestructura física existente IPv4.

Es probable que el departamento de contabilidad de las empresas integradas tenga aplicaciones financieras en los servidores que necesitarán ser accedidos por los empleados de contabilidad tanto en la Empresa A como en la B. Tanto los servidores como los clientes tendrán IPv6, pero ellos también retendrán sus componentes de la capa IPv4. Las sesiones IPv6 del departamento de contabilidad viajarán sobre los enlaces existentes locales y remotos como “solo otro protocolo”, sin requerir cambios en la red física. El único requerimiento para la conectividad IPv6 es que los routers que son adyacentes a los usuarios del departamento de contabilidad deben ser mejorados a las capacidades de IPv6. Donde no se puede lograr la conectividad IPv6 end-to-end, se puede emplear alguna de las técnicas de tunnelling IPv4/IPv6.

A medida que continúa la integración, también se les darán hosts IPv4/IPv6 a otros departamentos en las empresas recientemente unificadas. Al adicionarse nuevos departamentos y grupos de trabajos, se les pueden dar hosts de capa dual, o en algunos casos, hosts IPv6. Los hosts que requieren comunicaciones al mundo exterior vía Internet probablemente recibirán capas duales para mantener la compatibilidad con los nodos IPv4 externos a la empresa. Pero en algunos casos, los hosts que solo requieren acceso a servidores internos y a partes específicas externas, pueden ser capaces de lograr conectividad con hosts IPv6. Una migración a IPv6 presenta la oportunidad para un nuevo inicio en términos de asignación de direcciones y estructura de protocolo de ruteo. Los hosts y routers IPv6 inmediatamente pueden tomar ventaja de las características de IPv6 tales como autoconfiguración, encriptado, autenticación, etc. .

Escenario 2: IPv6 desde el borde al núcleo

Para muchísimos usuarios corporativos, los requerimientos de conectividad se centran principalmente en el acceso al e-mail local, a la base de datos, y a los servidores de aplicaciones. En este caso, puede ser mejor actualizar a IPv6 inicialmente solo a los grupos de trabajo y a los departamentos aislados, con actualizaciones en el router fundamental implementadas lentamente. El desarrollo del protocolo IPv6 es más completo para el ruteo de “borde” que para el ruteo fundamental de alto nivel, por lo que esta es una excelente manera para las empresas de hacer la transición fácilmente. Los grupos de trabajo independientes pueden actualizar sus clientes y servidores a hosts de capa dual IPv4/IPv6 o solo a hosts IPv6. Esto crea “islas” de funcionalidad IPv6.

A medida que envejecen los protocolos de ruteo tales como OSPF y BGP de IPv6, se pueden destacar las conexiones fundamentales de IPv6. Luego de instalar los primeros routers de IPv6, puede ser deseable conectar las islas IPv6 junto con túneles router a router. En este caso, se podrían configurar uno o más routers en cada isla como puntos terminales del túnel. Cuando los hosts usan direccionamiento completo de 128 bits, los túneles son configurados manualmente para que los routers que participan en los túneles conozcan los puntos terminales del túnel. Con Direcciones IPv6 compatibles con IPv4, es posible un tunnelling automático, sin configurar. Desde el punto de vista de un protocolo de ruteo, los túneles aparecen como un simple salto IPv6, aún si el tunel incluye muchos saltos IPv4 a través de un número de medios diferentes. En el entorno IPv6, OSPF tendrá la ventaja de las métricas flexibles para los ruteos de tunel, para asegurar que a cada tunel se le dio su propio peso dentro de la topología. En general, los routers toman decisiones de reenvío de paquetes en el entorno tunnelling de la misma manera que toman decisiones en la red IPv6. Las conexiones fundamentales IPv4 son esencialmente transparentes a los protocolos de ruteo IPv6.

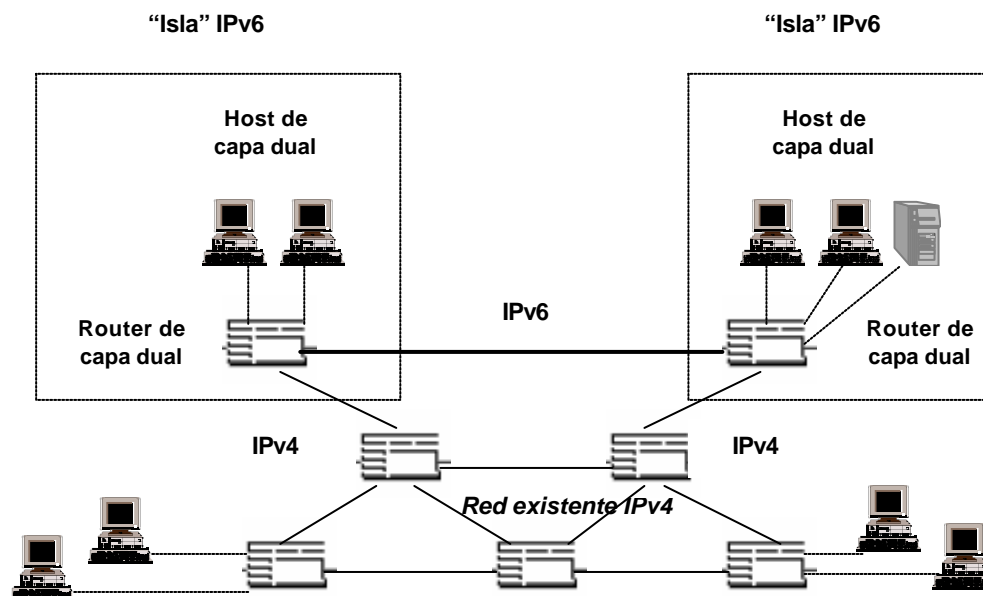


Fig. 10.a.22.

Conclusión

El protocolo de Internet (IP) ha sido la base de Internet y virtualmente de todas las redes privadas internas de trabajo. Este protocolo está alcanzando el final de su vida útil, y se ha definido un nuevo protocolo, conocido como IPv6 (IP versión 6) para reemplazar finalmente a IP.

La motivación conducida para la adopción de una nueva versión de IP fue la limitación impuesta por los campos de dirección de 32 bits en IPv4. Inclusive, IP es un protocolo muy viejo, y se han desarrollado nuevos requerimientos en las áreas de seguridad, flexibilidad de ruteo, y soporte del tráfico. Para satisfacer estas necesidades, se ha definido IPv6, que incluye mejoras funcionales y de formato sobre IPv4. Además, se han emitido un conjunto de especificaciones de seguridad que se pueden usar tanto con IPv4 como con IPv6. Con la mayoría de los detalles técnicos de estas mejoras no aprovechadas, los vendedores pueden comenzar a mover esta capacidad en sus líneas de productos. A medida que IPv6 se despliega en forma gradual, Internet y las redes corporativas se rejuvenecerán, siendo capaces de soportar las aplicaciones del siglo 21.

Bibliografía

- Nueva Generación del Protocolo IP: IPv6 (Fundesco – Dpto. de Redes)
- White Paper: IPv6 (Bay Networks Inc. 1997)
- W. Stallings, *Data and Computer Communications*, 5th Edition, Upper Saddle River, NJ: Prentice Hall, 1996.
- R. Gilligan and R. Callon, "IPv6 Transition Mechanisms Overview," *Connexions*, Oct. 1995.
- R. Hinden, "IP Next Generation Overview," *Connexions*, Mar. 1995.
- S. Bradner and A. Mankin, *IPng: Internet Protocol Next Generation*, Reading, MA: Addison-Wesley, 1996.
- C. Huitema, *IPv6: The New Internet Protocol*, Upper Saddle River, NJ: Prentice Hall, 1996.
- E. Britton, J. Tavs, and R. Bournas, "TCP/IP: The Next Generation," *IBM Sys. J.*, no. 3, 1995.

ANEXO 10.b: Coherencia de caché en sistemas multiprocesadores

En sistemas multiprocesadores es usual tener 1 ó 2 niveles de caché por procesador. Aunque se logra así buena performance, se crea un problema conocido como problema de coherencia. Este consiste en que múltiples copias de los mismos datos pueden existir en diferentes cachés simultáneamente, y si se deja a los procesadores actualizar sus propias copias libremente, puede resultar una vista inconsistente de la memoria.

Existen 2 políticas comunes de escritura:

- **Write-back** (o retroescritura): las operaciones de escritura se hacen solo a caché. La memoria central se actualiza solo si la correspondiente línea de caché se necesita para cargar otros datos.

- **Write-through** (o escritura directa): Todas las operaciones de escritura se hacen tanto a memoria central como a caché, asegurando que la memoria central sea siempre válida.

La política de write-back puede resultar en inconsistencia. Si 2 cachés tienen la misma línea, y esa línea se actualiza en una de ellas, la otra tendrá un valor inválido. Lecturas subsiguientes a la línea inválida darán resultados inválidos. Aún con la política write-through, puede ocurrir inconsistencia a menos que las otras cachés monitoreen el tráfico de memoria o reciban notificación de la actualización.

Para enfrentar el problema de coherencia se verán varias propuestas y luego se tratará la mas usada: el protocolo MESI. Versiones del cual se implementan en Sparc, PowerPC, Pentium II/III, etc.

El objetivo de los protocolos de coherencia de caché, es permitir a variables locales de uso reciente ir hacia la caché apropiada y permanecer allí durante numerosas lecturas y escrituras, mientras se mantiene la consistencia de variables compartidas que podrían estar en múltiples cachés al mismo tiempo.

Las soluciones a coherencia de caché suelen dividirse en soluciones software y hardware. Algunas implementaciones usan una estrategia que combina software y hardware.

Soluciones software

Los esquemas de coherencia por software intentan evitar circuitos y lógica hardware extra, confiando en el sistema operativo y el compilador para enfrentar el problema. Estas soluciones son atractivas pues la sobrecarga para detectar problemas potenciales se transfiere del tiempo de ejecución al tiempo de compilación, y la complejidad de diseño se transfiere de hardware a software. La contrapartida es que en general se hacen decisiones conservativas, llevando a uso ineficiente de caché.

Los mecanismos de coherencia basados en compilador analizan el código para determinar que ítems de datos pueden llegar a ser inseguros para el caché, y los marcan. Entonces el sistema operativo ó hardware apropiado previene que esos ítems se ponga en caché.

La solución mas simple es no cachéar cualquier variable de datos compartida. Esto es muy conservativo pues una estructura de datos compartida puede ser usada exclusivamente durante algunos períodos y puede ser de solo lectura durante otros. Es durante periodos cuando un proceso puede actualizar una variable y otro proceso puede acceder a esa variable que la coherencia de caché es un problema.

Métodos mas eficientes analizan el código para determinar períodos seguros para variables compartidas. El compilador inserta instrucciones en el código generado para forzar coherencia de caché en periodos críticos. Se han desarrollado técnicas para realizar el análisis y hacer cumplir los resultados.

Soluciones hardware

Generalmente se refiere a las soluciones basadas en hardware como protocolos de coherencia de caché.

Estas soluciones proveen reconocimiento dinámico en tiempo de ejecución de condiciones potenciales de inconsistencia. Como el problema solo se trata cuando realmente aparece, el uso de caché es mas efectivo en performance respecto a soluciones software. Además, estas técnicas son transparentes al programador y al compilador, reduciendo la carga de desarrollo de software.

Los esquemas hardware difieren en: donde se mantiene la información sobre líneas de datos, como se organiza la información, donde se hace cumplir la coherencia y los mecanismos empleados para ello. Los esquemas hardware se dividen en 2 categorías: 1) protocolos de directorio y 2) protocolos snoopy.

1) Protocolos de directorio

Los protocolos de directorio reúnen y mantienen información sobre donde residen copias de líneas. Típicamente, hay un controlador centralizado que es parte del controlador de memoria central, y un directorio que se almacena en memoria central. El directorio tiene información de estado global sobre los contenidos de cachés locales. Si un controlador de caché individual hace un pedido, el controlador centralizado chequea y entrega comandos para transferencia de datos entre memoria y cachés o entre cachés. También es responsable de mantener la información actualizada; por lo tanto, toda acción local que puede afectar el estado global de una línea debe ser informado al controlador central.

El controlador mantiene información sobre cuales procesadores tienen una copia de cuales líneas. Antes que un procesador pueda escribir a una copia local de una línea, debe pedir acceso exclusivo a la línea desde el controlador. Antes de otorgar este acceso exclusivo, el controlador envía un mensaje a todos los procesadores con una copia cachéada de esta línea, forzando a cada procesador a invalidar su copia. Después de recibir reconocimiento de cada uno de tales procesadores, el controlador otorga acceso exclusivo al procesador demandante. Cuando otro procesador trata de leer una línea que se otorga exclusivamente a otro procesador, enviará una notificación de falla al controlador. El controlador entonces entrega un comando al procesador que mantiene esa línea que requiere el procesador para hacer una retroescritura a memoria central. La línea puede ahora ser compartida para leer por el procesador original y el procesador demandante.

Los esquemas de directorio sufren las limitaciones de un cuello de botella central y la sobrecarga de

comunicación entre los varios controladores de caché y el controlador central. Sin embargo, son efectivos en sistemas de gran escala que involucran buses múltiples o algún otro esquema de interconexión complejo.

2) Protocolos SNOOPY

los protocolos snoop distribuyen la responsabilidad para mantener coherencia de caché entre todos los controladores de caché en un multiprocesador. Una memoria caché debe reconocer si una línea que tiene es compartida con otros cachés. Cuando una acción de actualización se hace en una línea caché compartida, debe anunciarse a todos los otros cachés por un mecanismo de difusión. Cada controlador de caché es capaz de espiar (snoop) en la red para observar esas notificaciones difundidas y reaccionar adecuadamente.

Los protocolos snoop se adaptan idealmente a un multiprocesador basado en bus, porque el bus compartido provee un medio simple para difundir y espiar. Sin embargo como uno de los objetivos del uso de cachés locales es evitar accesos a bus, se tiene que cuidar que el mayor tráfico de bus por espiar y difundir no cancela las ganancias por el uso de caché locales.

Dos formas del protocolo snoop son: write-invalidate y write-update (o write-broadcast).

Protocolo write-invalidate: puede haber múltiples lectores pero solo un escritor por vez. Inicialmente, una línea puede ser compartida entre varias cachés para lectura. Cuando uno de los cachés quiere hacer una operación de lectura a la línea, primero emite una noticia que invalida esa línea en los otros cachés, haciendo la línea exclusiva al caché escribiendo, entonces el procesador propietario puede hacer escrituras locales hasta que algún otro procesador requiere la misma línea.

Protocolo write-update: pueden haber múltiples escritores y múltiples lectores. Cuando un procesador desea actualizar una línea compartida, la palabra a ser actualizada se distribuye a las otras líneas compartidas, y los cachés conteniendo esa línea pueden actualizarla.

Ninguna de esas dos técnicas es superior a la otra en todas las circunstancias. La performance depende en el número de cachés locales y el patrón de lecturas y escrituras de memoria. Algunos sistemas implementan protocolos adaptativos que usan mecanismos tanto write-invalidate y write-update.

La forma write-invalidate es la mas usada en sistemas comerciales multiprocesadores, tales como el Pentium II o PowerPC. Marca el estado de todas las líneas caché (usando 2 bits extra en el tag de caché) como modificado, exclusivo, compartido o invalido. Por esta razón el protocolo write-invalide se llama MESI. En la siguiente sección se verá su uso entre cachés locales en un multiprocesador.

El protocolo MESI

Se usa el protocolo MESI (Modified / exclusive /shared / invalid)para tener consistencia en la memoria caché de datos. Aunque MESI se diseñó para soportar requerimientos de consistencia en sistemas multi procesador, también es útil en la organización de procesadores únicos.

La memoria caché de datos tiene 2 bits de status por etiqueta, por lo tanto cada línea puede estar en alguno de 4 estados:

- Modificada: la línea en caché fue modificada y esta disponible solo en esta caché
- Exclusiva: la línea en caché es la misma que en memoria central y no esta presente en otra caché
- Compartida: la línea en caché es la misma que en memoria central y puede estar presente en otra caché.
- Inválida: la línea en caché tiene datos inválidos

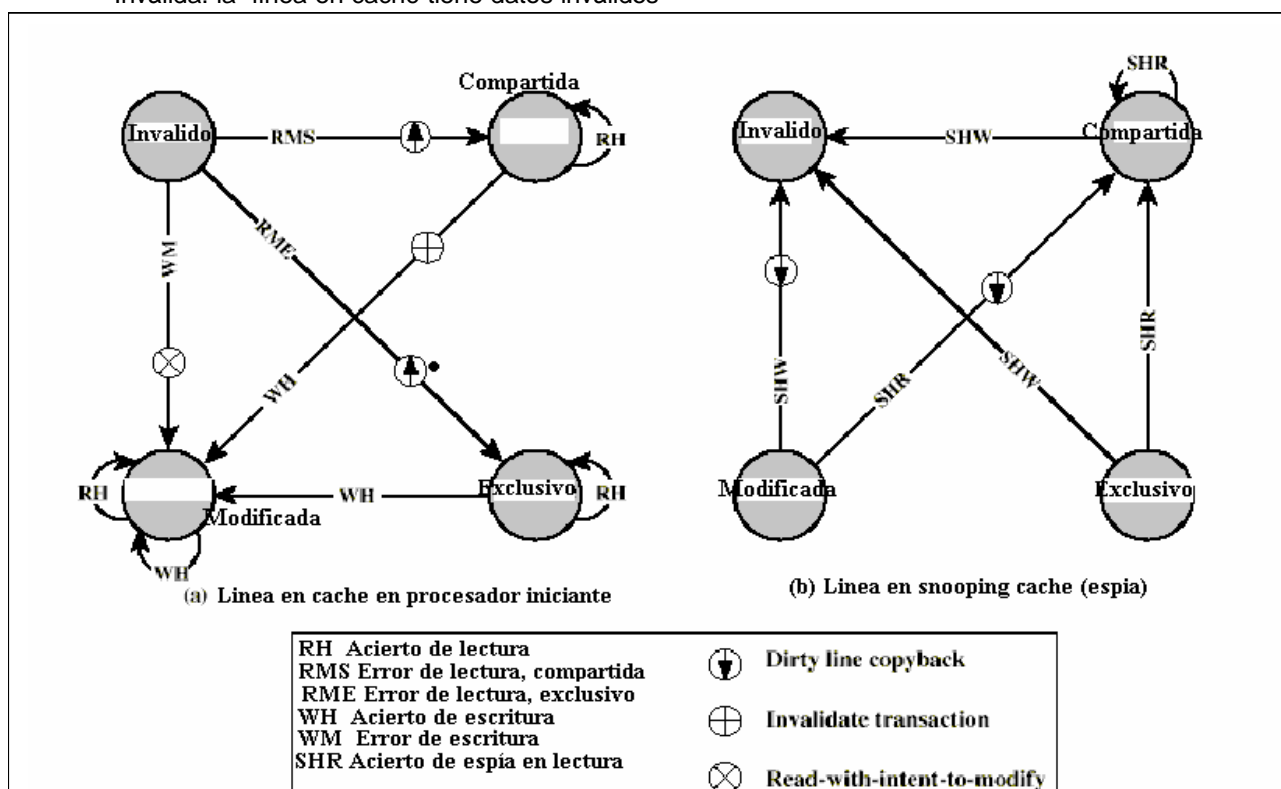


Figura 10.b.1 – Diagrama de transición de estados MESI

La figura 10.b.1 muestra un diagrama de estado para el protocolo MESI. Nótese que cada línea de caché tiene sus propios bits de estado y por lo tanto su propia realización del diagrama de estado. La figura 10.b.1a muestra las transiciones que ocurren debido a acciones iniciadas por el procesador agregado a esta caché. La figura 1b muestra las transiciones que ocurren debido a eventos que son espiados en el bus común. Esta presentación de diagramas de estado, separados para acciones iniciadas por procesador e iniciadas por bus, ayuda a clarificar la lógica del protocolo MESI. Sin embargo en cualquier instante una línea de caché esta en un único estado. Si el próximo evento es desde el procesador, entonces la transición es dictada por la figura 10.b.1a, y si el próximo evento es desde el bus, la transición es dictada por la figura 10.b.1b. Veremos estas transiciones en mas detalle.

ERROR DE LECTURA (miss)

cuando ocurre un error de lectura en caché local, el procesador lee la línea de memoria central conteniendo la dirección faltante, inserta una señal en el bus que alerta a las otras unidades procesador/caché para espiar la transacción. Las posibles salidas son:

Si otra caché tiene una copia inmodificada, desde que se leyó de memoria, de la línea en estado exclusivo, devuelve una señal indicando que comparte esta línea. El procesador transiciona el estado de la copia de exclusiva a compartida. El procesador iniciante lee la línea de memoria central y transiciona la línea en su caché de inválida a compartida.

Si uno o mas cachés tienen una copia limpia de la línea en estado compartido, señalan que comparten la línea. El procesador iniciante lee la línea y transiciona la línea en su caché de inválida a compartida.

Si otro caché tiene una copia modificada de la línea, entonces bloquea la lectura de memoria y provee la línea al caché demandante sobre el bus compartido. El caché respondiente entonces cambia su línea de modificada a compartida.

Si ningún otro caché tiene una copia de la línea (modificada o limpia) Entonces no se retornan señales. El procesador iniciante lee la línea y transiciona la línea en su caché de inválida a exclusiva

Acierto de lectura

cuando ocurre un acierto de lectura en una línea en caché local, el procesador lee el ítem requerido. No hay cambio de estado: permanece modificado, compartido o exclusivo.

Error de escritura

Cuando hay un error de escritura en caché local, el procesador lee la línea desde memoria central conteniendo la dirección faltante y emite una señal en el bus que significa "lectura-con-intento-para-modificar" (RWITM). Cuando se carga la línea, se marca como modificado.

Con respecto a otros cachés, 2 posibles escenarios preceden a la carga de la línea de datos.

Otro caché puede tener una copia modificada de esta línea (estado modificado). En tal caso, el procesador alertado señala al procesador iniciante que tiene una copia modificada de la línea. El procesador iniciante entrega el bus y espera. El otro proceso gana acceso al bus, escribe la línea modificada de caché de vuelta a memoria central, y transiciona el estado de la línea de caché a inválida (porque el procesador iniciante va a modificar esta línea). Luego, el procesador iniciante emitirá de nuevo una señal al bus de RWITM y lee la línea de memoria central, modifica la línea en caché, y marca la línea en el estado modificado.

ningún otro caché tiene una copia modificada de la línea pedida. En este caso, no se retorna ninguna señal y el procesador iniciante procede a leer en la línea y modificarla. Mientras tanto, si uno o mas cachés tienen una copia limpia de la línea en estado compartido, cada caché invalida su copia de la línea, y si un caché tiene una copia limpia de la línea en el estado exclusivo, invalida su copia de la línea.

Acuerdo de escritura

Cuando hay un acuerdo de escritura en una línea en caché local, el efecto depende del estado de esa línea

Compartida: Antes de hacer la actualización, el procesador debe ganar propiedad exclusiva de la línea. El procesador señala su intento en el bus. Cada procesador que tiene una copia compartida de la línea en su caché transiciona el sector desde compartida a invalida. El procesador iniciante entonces realiza la actualización y transiciona su copia de la línea desde compartida a modificada.

Exclusiva: el procesador ya tiene control exclusivo de esta línea, realiza la actualización y transiciona su copia de la línea desde exclusiva a modificada.

Modificada: El procesador ya tiene control exclusivo de esta línea y tiene la línea marcada como codificada, y así simplemente realiza la actualización.

Consistencia de caché L1-L2

Hasta ahora se trataron protocolos de coherencia de caché en términos de actividad cooperativa entre cachés conectados al mismo bus u otra facilidad con interconexión SMP. Típicamente, estas cachés son cachés L2, y cada procesador también tiene un caché L1 que no se conecta directamente al bus y que por lo tanto no puede engancharse en un protocolo snoopy. Así, se necesita algún esquema para mantener integridad de datos en ambos niveles de caché y en todos los cachés en la configuración SMP.

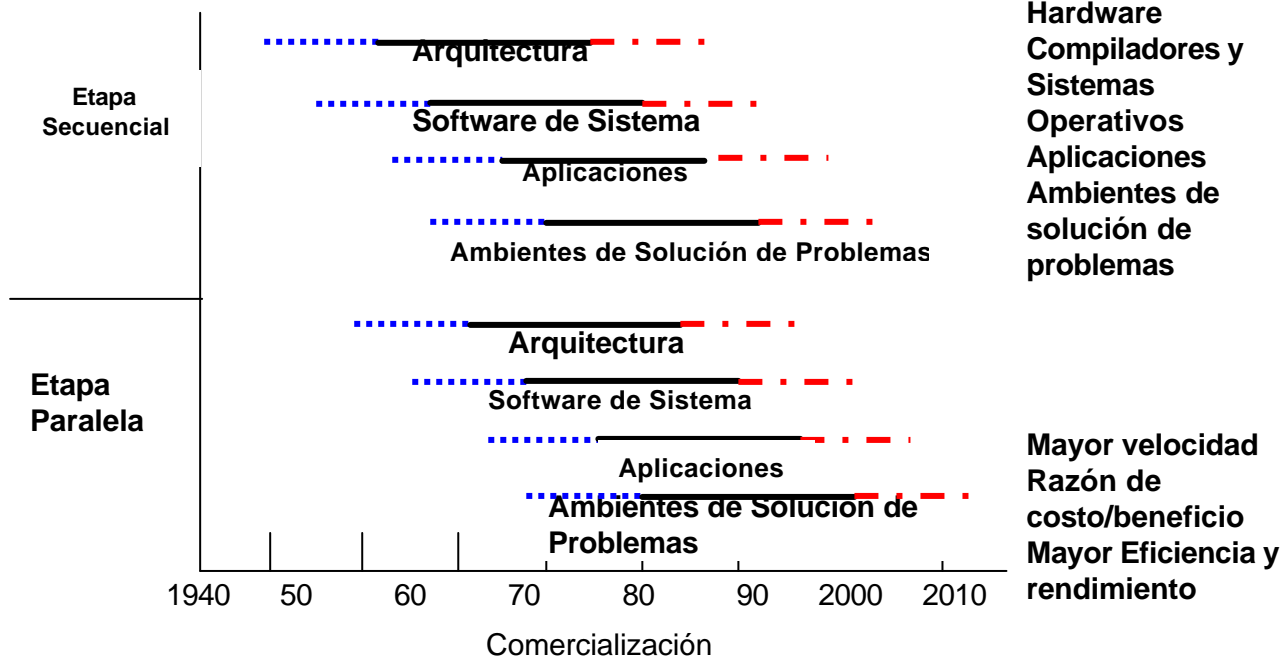
La estrategia es extender el protocolo MESI (u otro protocolo de coherencia de caché) a los cachés L1. Cada línea en la caché L1 incluye bits para indicar el estado, El objetivo es el siguiente: para cualquier línea que esta presente en ambas una caché L2 y su correspondiente caché L1, el estado de la línea L1 debería seguir el estado de la línea L2. Un medio simple de hacer esto es adoptar la política write-through en el caché L1; en este caso el write through es a la caché L2 y no a memoria. La política write-through de L1 fuerza cualquier modificación de una línea L1 a la caché L2 y la hace visible a otros cachés L2. El uso de la política de write-through para L1 requiere que su contenido sea un subconjunto del contenido de L2. Esto sugiere que la asociatividad del caché L2 debería ser igual o mayor que la asociatividad L1. La política write-through de L1 se usa en la IBM S/390 SMP.

Si la caché L1 tiene una política write-back, la relación entre los 2 cachés es mas compleja. Para mantener coherencia hay varias soluciones.

Anexo 9c. : Clustering y Procesamiento paralelo**Cluster, base y evolución.**

Una manera de ver a la computación es como dos etapas prominentes del desarrollo:

- Etapa de la computación Secuencial.
- Etapa de la computación Paralela.



ARQUITECTURAS DE COMPUTACIÓN PARALELAS.

Su taxonomía se basa en como se despliega sus procesadores, memoria e interconexión.

Los sistemas más comunes son:

- Procesadores Paralelos Masivamente (MPP – Massivale Parallel Processors)
- Multiprocesadores Simétricos (SMP – Symmetric Multiprocessors)
- Acceso a Memoria No Uniforme con Coherencia de Cache (CC-NUMA – Cache-Coherent Nonuniform Memory Access)
- Sistemas Distribuidos
- Clusters

Comparación de las diversas arquitecturas paralelas

Caraterísticas	MPP	SMP // CC-NUMA	Cluster	Distribuido
Número de Nodos	100 a 1000	10 a 100	100 o menos	10 a 1000
Complejidad de los nodos.	Grano fino o medio	Grano medio	Grano medio	Amplio Rango
Comunicación entre los nodos	Paso de Mensajes. Variables compartidas p/ memoria distribuida compartida.	Memoria compartida, centralizada y distribuida	Paso de mensajes	Archivos Compartidos. RPC, Paso de mensajes. Comunicaciones entre procesos.
Planificación de Tareas	Cola simple sobre el Host	Cola simple mayormente	Múltiples colas coordinadas	Colas Independientes
Soporte SSI (Single System Image)	Parcialmente	Siempre en SMP y a veces en NUMA	Deseado	NO
SOs de los nodos. Copias y Tipos	N microkernel monolíticos o OSs en Capas	Uno Mono-lítico en SMP y Muchos en NUMA	N Plataformas SOs Homogéneas o de microkernel	N Plataformas de SOs Homogéneas
Espacio de Direcciones	Múltiple-Simple para DMS	Simple	Múltiple o simple	Múltiple
Seguridad entre los nodos	Innecesario	Innecesario	Requerido si se expone	Requerido
Propietario	Una Organización	Una Organización	Una o más Organizaciones	Muchas Organizaciones

CARACTERÍSTICAS DE LOS COMPUTADORES PARALELOS:

Un sistema MPP:

- Es un gran sistema de procesamiento paralelo con una arquitectura que no comparte nada.
- Consiste en cientos de elementos de procesamiento los cuales están interconectados por un Switch o red de alta velocidad.
- Cada nodo puede tener una variedad de componentes de hardware, pero generalmente consisten de una memoria central y de uno o varios procesadores.

Los sistemas SMP:

- Poseen desde 2 a 64 procesadores y pueden ser considerados como una arquitectura que comparte todo.
- En estos sistemas todos los procesadores comparten todos los recursos globales disponibles (bus del sistema, memoria, sistemas de I/O, etc.);
- Una copia sencilla del sistema operativo corre en estos sistemas.

Los sistemas CC-NUMA:

- Es un sistema multiprocesador escalable.
- Como en SMP, cada procesador en un sistema CC-NUMA tiene una vista global de toda la memoria.
- Este tipo de sistema consigue su nombre (NUMA) a partir de los tiempos no uniformes que le toma para acceder ya sea a la parte memoria más cercana, así como a la más remota.

Los sistemas distribuidos:

- Pueden ser considerados redes convencionales de computadores independientes.
- Los mismos tienen múltiples imágenes del sistema, a partir de que cada nodo tiene su propio sistema operativo, y
- Cada máquina individual en un sistema distribuido puede ser, por ejemplo, una combinación de MPPs, SMPs, Clusters y computadoras individuales.

Un cluster es:

- Una colección de estaciones de trabajo o PCs que están conectadas mediante alguna tecnología de red.
- Para fines de computación paralela estas PCs o estaciones de trabajo estarán conectadas mediante una red de muy alta velocidad.
- Un cluster trabaja como una colección integrada de recursos y pueden tener una imagen simple del sistema abarcando todos sus nodos.

Un Cluster y su arquitectura

Un cluster es un tipo de sistema de procesamiento distribuido o paralelo, el cual consiste en una colección de computadores separados trabajando juntos como un único recurso integrado de computación.

Un nodo puede ser un sistema mono o multi procesador (PCs, Workstations o SMPs) con memoria, facilidades de I/O, y un sistema operativo. En general se designa con el término cluster a dos o más computadores (nodos) conectados entre ellos. Los computadores pueden estar dentro de un mismo gabinete o separados conectados mediante alguna LAN. Un cluster interconectado (vía LAN) puede aparecer como una única máquina para usuarios y aplicaciones. En la siguiente figura se muestran un esquema de varios cluster conectados mediante redes de alta velocidad

A continuación se detallan algunos componentes importantes de un cluster:

- Múltiples computadores de alto rendimiento.
- Excelentes Sistemas Operativos
- Redes/Switches de alto rendimiento (tales como Gigabit Ethernet o Redes Myrinet)
- Tarjetas de Interfase de red.
- Protocolos y Servicios de comunicación veloces
- Cluster Middleware (Imagen Unica del Sistema (SSI) e infraestructura de disponibilidad del sistema)
 - Hardware (Canales de memoria, Hardware DSM y técnicas de SMP)
 - Kernel o capa de Sistema Operativo (como Solaris MC y GLUnix)
 - Aplicaciones y subsistemas.
 - ❖ Aplicaciones (como herramientas de administración del sistema y formularios electrónicos)
 - ❖ Sistema de tiempo de ejecución (como software DSM o sistema de archivos paralelo)
 - ❖ Software de planificación y administración de recursos(como LSF (Load Sharing Facility) y CODINE (Computing in Distributed Network Environment)).
- Ambientes y Herramientas de multiprogramación (compiladores, PVM(parallel virtual machine))
- Aplicaciones
 - Secuenciales.
 - Paralelas o Distribuidas.

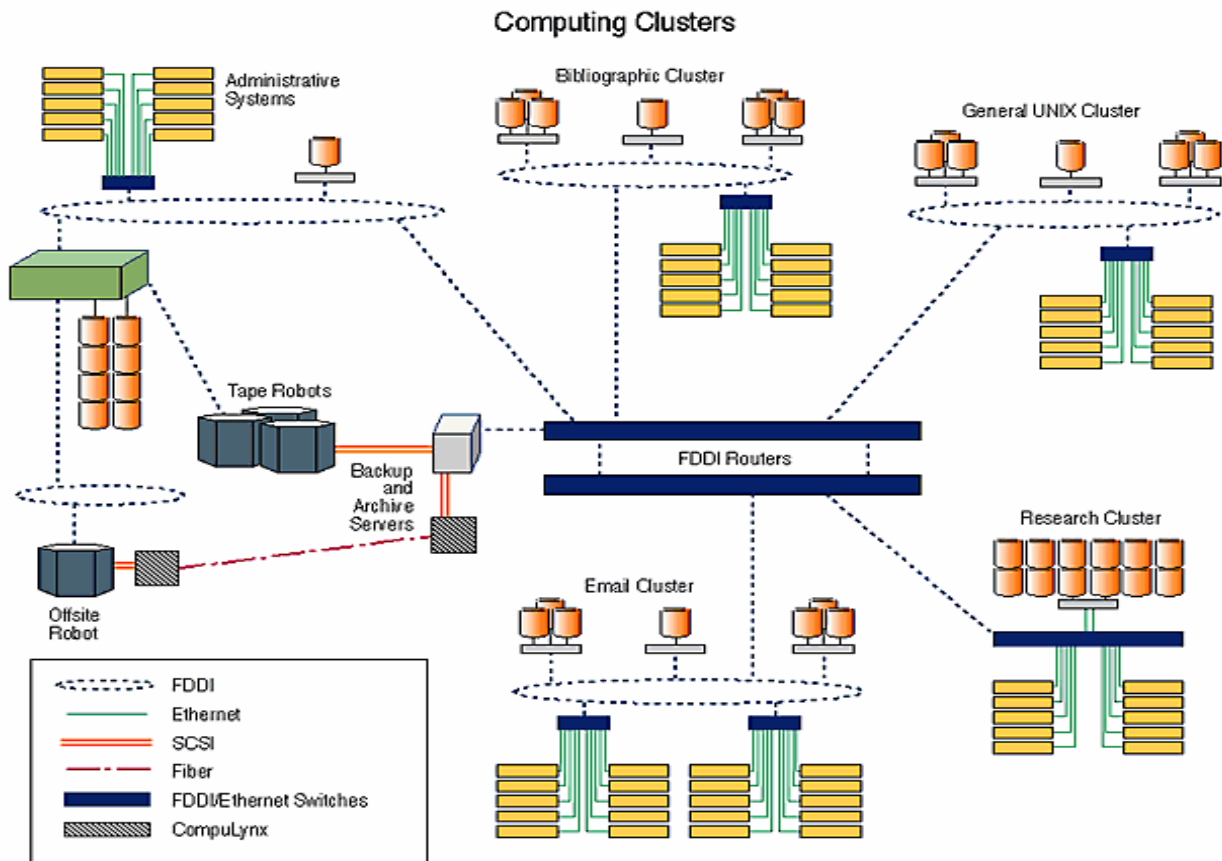


Figura 1.2: Varios tipos de cluster conectados por una red de alta velocidad.

1.4 – Clasificación de los Clusters

Los cluster ofrecen las siguientes características a un costo relativamente bajo tales como:

- Alto rendimiento
- Expansibilidad y escalabilidad.
- Alto respuesta
- Alta Disponibilidad.

La tecnología cluster permite a las organizaciones elevar su poder de procesamiento utilizando tecnología estándar la cual puede ser adquirido a un costo relativamente bajo. Esto provee expansibilidad mientras que preserva la inversión sin incurrir en un montón de gastos extras. El rendimiento de las aplicaciones también se eleva con la utilización de un software escalable apropiado. Otro beneficio del cluster es la capacidad contras fallas en el caso de que alguno de sus componentes falle y deje de estar en servicio.

Los cluster se clasifican en muchas categorías basándose en varios factores como los que se indican a continuación.

1. **Objetivo de la Aplicación** - Ciencia o aplicaciones para misiones críticas.
 - Clusters de Alto Rendimiento (HP – High Performance)
 - Clusters de Alta Disponibilidad (HA – High Availability)
2. **Propiedad del Nodo** – Apropiado por un cluster o como un nodo cluster.
 - Clusters Dedicados
 - Clusters No Dedicados
3. **Hardware del Nodo** – PC, Estación de Trabajo o SMP.
 - Cluster de PCs (CoPs) o Pilas de PCs (PoPs)
 - Clusters de Estaciones de Trabajo (COWs – Cluster or Workstations)
 - Clusters de SMPs (CLUMPs)

4. **Sistema Operativo del Nodo** – Linux, NT, Solaris, AIX, etc.
 - Linux Clusters (v.g. Beowulf)
 - Solaris Clusters (v.g. Berkeley NOW)
 - NT Clusters (v.g. HPVM)
 - AIX Clusters (v.g. IBM SP2)
 - Digital VMS Clusters
 - HP-UX Clusters
 - Microsoft Wolfpack Clusters.
5. **Configuración del Nodo** - Arquitectura del nodo y Sistema Operativos con el que funciona.
 - Cluster Homogéneo: Todos los nodos tiene arquitecturas similares y corren el mismo sistema operativo
 - Cluster Heterogéneo: Todos los nodos tienen diversas arquitecturas y corren diferentes SOs
6. **Niveles de Clustering** – Basado en la ubicación de los nodos y su cantidad
 - Cluster de Grupo (#Nodos: 2-99): Los están conectados por medio de SANs (System Area Networks) como myrnet y los mismos están ubicados dentro de un mismo centro
 - Cluster Departamental (#Nodos: 10-100)
 - Cluster Organizacional (#Nodos: Muchos cientos)
 - Super-Computadores Nacionales (WAN/Internet Based)
 - Super-Computadores Internacionales (Internet Based) (#Nodos: 1000 a muchos millones)

1.5. Servicios de Red / Software de Comunicación.

Las necesidades de comunicación de las aplicaciones distribuidas son muy diversas y varían desde un línea Punto-a-Punto hasta una Comunicación Multicast. La infraestructura de comunicación necesita soportar protocolos para transporte de masivos de información, flujos de datos y comunicaciones de grupos y todos aquello utilizado por los objetos distribuidos.

El servicio de comunicación empleado provee los mecanismos básicos necesarios para el cluster para transporte de información administrativa y del usuario. Estos servicios proveerán al cluster con una gran variedad y calidad de parámetros tales como latencia, ancho de banda, disponibilidad, tolerancia a fallos y control de fluctuaciones de duración de pulsos. Típicamente, los servicios de red están diseñados como una pila jerárquica de protocolos. En tales sistemas por capas cada capa de protocolo en la pila explota los servicios de la capa que se encuentra por debajo de ella en la pila. El clásico ejemplo sería el estándar OSI (Open System Interconnection).

1.6 Administración y Planificación de Recursos.

La Administración y Planificación de Recursos (RMS – Resource Management and Scheduling) es el acto por el cual las aplicaciones distribuidas maximizan sus rendimiento. También permite el efectivo y eficiente aprovechamiento de los recursos del sistema. El software que lleva a cabo es RMS se compone de dos partes: Un administrador de recursos y un planificador de recursos. El Administrador de recursos se encarga básicamente de los problemas, como localizar recursos, asignar recursos computacionales, autenticación, así como también de tareas como creación y migración de procesos. El Planificador de Recursos se encarga, entre otras tareas, de encolar aplicaciones y asignarles recursos a las mismas.

El RMS se utiliza por una serie de razones, incluyendo: Balanceo de carga, optimización de los ciclos del CPU, proveyendo Tolerancia a fallos, acceso gestionado a sistemas potentes. Pero la principal razón de su uso es la habilidad para proveer un elevado y seguro rendimiento de las aplicaciones en los sistemas que el administra.

El RMS provee servicios de Middleware a los usuarios permitiendo una variedad de ambientes como Estaciones de trabajos, SMPs y plataformas paralelas para ser usadas sencilla y eficazmente utilizadas. Los servicios prestados por el RMS pueden incluir, entre otros, los siguientes:

Migración de Procesos - Esto es cuando un proceso debe ser suspendido, movido y reiniciado en otro computador dentro del ambiente del RMS. Generalmente, la migración de procesos ocurre por dos motivos: un recursos se encuentra sumamente cargado y existen otros que están libres o para maximizar la respuesta al usuario.

Puntos de Chequeo – Esto es que se realiza una traza del programa desde algún punto deseado y se puede relanzar el programa desde dicho punto.

Expulsión de Ciclos Ociosos - Es sumamente sabido que en una Estación de Trabajo el 70 o 90% del tiempo se encuentra ociosa. El RMS puede ser puesto a punto para que emplee esos ciclos en los cuales la CPU no trabaja. Esto favorece tanto al usuario como al sistema en general.

Tolerancia a Fallas – Mediante el monitoreo de los trabajos y recursos, un sistema RMS, puede ofrecer varios niveles de Tolerancia a Fallas. En la forma más sencilla esto significa que un proceso que falla será reiniciado hasta que complete su ejecución.

Minimización del Impacto en los Usuarios – La ejecución de trabajos en las terminales de trabajos puede traer aparejado un gran impacto en los usuarios de aplicaciones interactivas. Algunos sistemas de RMS tratan de reducir este impacto mediante la reducción de prioridad en los trabajos planificados, dando prioridad al usuario, o suspendiendo su ejecución.

Balanceo de Carga – Los trabajos pueden ser distribuidos entre todas las plataformas computacionales disponibles dentro de un área. La migración de procesos es una parte importante del balanceo de carga.

Colas de Aplicaciones Múltiples – Las colas de trabajos pueden setearse para ayudar la administración de los recursos de una organización en particular. Cada cola puede ser configurada con ciertos atributos. Por ejemplo, ciertos usuarios tienen prioridad para ejecutar trabajos cortos antes que otros trabajos más largos.

1.7 Tecnologías Clusters del Futuro

Las tecnologías de hardware emergentes junto con los recursos de un software maduro hacen que los sistemas basados en clusters están acortando la brecha de rendimiento con las plataformas de computación dedicadas. Los sistemas de cluster que aprovechan el tiempo ocioso de las estaciones de trabajos y PCs continuarán con el uso de cualquier componente de hardware o software que esté disponible sobre estaciones de trabajo. Los clusters dedicados al alto rendimiento continuarán con evolucionando así como las nuevas y más potentes computadoras e interfaces de redes se encuentren disponibles en el mercado.

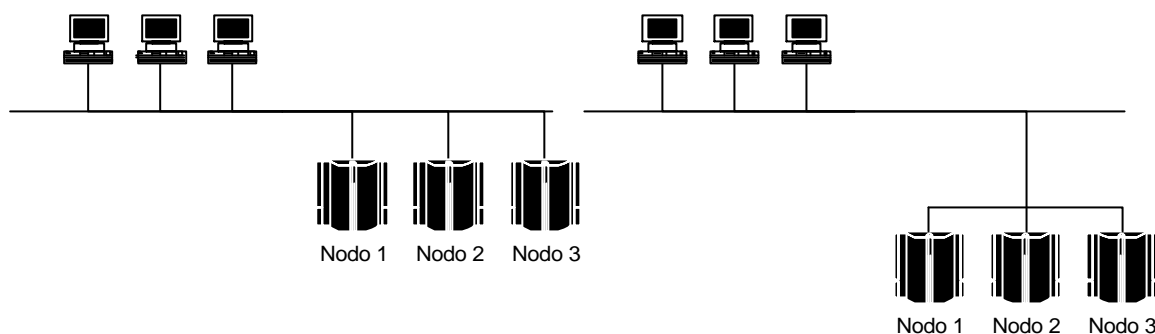
Es probable que los nodos de clusters sean SMPs. Actualmente dos o cuatro procesadores por PCs o Estación de trabajo UNIX se están volviendo comunes. El software que permita usar los nodos SMPs eficiente y eficazmente por aplicaciones paralelas serán desarrollados e incluidos, en un futuro, dentro de los kernel de los S.Os. También es probable que se popularice el Gigabit Ethernet por lo cual se transformará en el estándar para clusters.

La habilidad para proveer un rico conjunto de herramientas y utilidades de desarrollo, así como la provisión de robustos y eficaces servicios determinarán el cambio de los S.Os de los cluster futuros.

2– Hardware

2.1 Expuesto vs. Encerrado

La primera de las distinciones que se puede hacer es decididamente no tradicional. La misma se refiere a las características de seguridad de la comunicación dentro del cluster. La comunicación dentro del cluster puede ser “Expuesta” a la exterior o “Cerrada”, es decir, dentro del mismo cluster.



Esquema Expuesto

Esquema Cerrado

Un Cluster expuesto comparte las facilidades de comunicación con otros computadores que no pertenecen al cluster. Esta forma de compartir tiene varias consecuencias:

- Los nodos del cluster se comunican necesariamente por mensajes, partiendo de que la comunicación pública y estándar se basa en mensajes.
- La comunicación se vuelve excesivamente sobrecargada ya que se tienen que utilizar protocolos de comunicación estándar.
- El canal de comunicación, en sí mismo, no es seguro, por lo que se debe agregar trabajo adicional para asegurar la comunicación dentro de l cluster que lleva información de crítica.
- A partir de que estos canales de comunicación públicos, potencialmente amplios están usados, este tipo de cluster facilita la recolección de basura de los ciclos libres de las estaciones de trabajo dentro del cluster
- Es muy sencillo de construir.

Un cluster encerrado, por el contrario, tiene sus propias facilidades de comunicación. Esas facilidades pueden ser estándares como puede ser el emplear una red Ethernet aparte o una Token-Ring. Pero lo realmente importante es que es privado. Sin embargo las posibilidades son mas amplias que con las técnicas estándar de comunicación:

- La comunicación puede realizarse por varios medios: Disco compartido, Memoria compartida, Mensajes o cualquier otro medio.
- La comunicación tiene la posibilidad de tener muy poca sobrecarga ya que los diseñadores no necesitan estar limitados a los mecanismos estándares. Es decir, estos mecanismos no tienen que estar dentro de las normas de algún comité internacional, por lo que pueden ser implementados más rápidos.
- La seguridad del medio de comunicación es implícita, entonces la información puede ser transferida entre los nodos con una seguridad simple como la utilizada en la comunicación entre las secciones del kernel de un sistema operativo.

Si bien las distinciones antes expuestas son interesantes y muy usadas, las mismas no son la mayor razón para hacer esta distinción. Esto es: A nivel de mercado es mucho más fácil conseguir software para habilitar un cluster sobre un Cluster Cerrado.

La comunicación puede ser más rápida y barata lo cual siempre ayuda mucho. La seguridad no es dejada al margen lo que ayuda tremendamente. Es altamente improbable, si no es intrínsecamente imposible, que accidentalmente el cluster se fraccione en un número de partes individualmente "vivas", las cuales luego haya que unir. Esto es increíblemente una gran ayuda. En suma, es posible hacer uso de un almacenamiento compartido para mantener, lo que de otro modo debería ser información replicada distribuida. Esto es una reducción de la dificultad de implementación. En otras palabras, es más fácil controlar un bloque de almacenamiento que todo un algoritmo asincrónico que permite la replicación de estructuras de datos con múltiples fuentes de actualización.

Poniendo todo esto en otras palabras: La programación de un cluster cerrado no tiene que luchar contra todos los problemas tradicionales de los sistemas distribuidos para proveer al cluster de la funcionalidad deseada. Desgraciadamente, para tomar ventaja de esta relativa simplicidad, a menudo los ingenieros de hardware deben convencer a los ingenieros de software de realizar implementaciones no estándares. Esto es a menudo es una encrucijada ya que el software está constantemente agregando funciones sobre sus "estándares" en la lucha contra la competencia. Esta es una razón por la cual la comunicación rápida y *estándar* es una pieza extremadamente importante en el rompecabezas de los clusters.

Hablando de estándares rápidos, al menos en el área del hardware: Facilidades como el Estándar de Fibre-Channel o ATM requieren un particular cuidado cuando se trata el tema de Expuesto

o Cerrado. Es posible instalar switches de comunicación los cuales provean seguridad permitiendo que solo ciertos puntos finales se conecten a otros. Esto permitiría que en una misma red determinadas máquinas sean inexistentes para algunas y totalmente visibles para otras. Con esto estaríamos logrando Cluster Cerrados cada uno con su propia red de comunicación.

2.2 Clusters de “Casa de Cristal” vs “Campo Abierto”

Al momento de armar un cluster surge la contradicción de si el mismo debe estar enteramente compuesto por computadores que se encuentren todos en un mismo sitio aislado de los usuarios del mismo; o ,si por le contrario, cada estación de trabajo de un usuario será un nodo del cluster. Esta distinción es lo que se conoce como Casa de Cristal en el primero de los casos (Glass-House); mientras que el segundo se conoce como Campo Abierto (Wide-Campus).

Los Clusters de Campo Abierto son también conocidos como sistemas NOW –Network Of Workstations – Los mismos basan su potencial en utilizar la CPU de aquellos usuarios que están trabajando con una menor carga de trabajo..

Dos factores distinguen los clusters Campo Abierto de la Caja de Cristal: Los nodos en el primero de ellos operan en un ambiente menor controlado; y pueden rápida y totalmente abandonar el uso de un nodo por un usuario.

El Campo Abierto es menos controlado de diversas maneras. Es menos disponible en la medida de que los cables de electricidad, que están a la vista, son desenchufados. Ya sea por error o por deseo. Las máquinas pueden tener café tirado sobre ellas así como otros restos. También es común que el usuario experimente con archivos de configuración del sistema. Es también obvio que el ambiente físico es mucho menos seguro. Es por todo esto que los nodos de un cluster de este tipo, pueden no estar siempre disponibles.

Mejorar estos controles implica cierta pérdida de autonomía.

El segundo de los factores que diferencian estos esquemas es la necesidad de abandonar el uso total de uno de los nodos del cluster cuando un usuario decide retirarse. Este inconveniente no se resuelve simplemente con el manejo de un esquema de prioridades que mande sino que se plantean políticas de No Presencia.

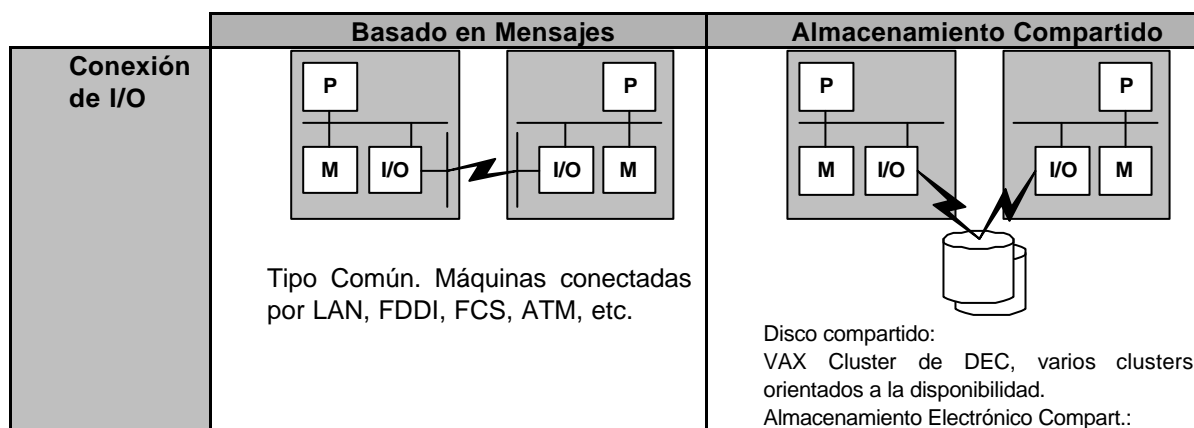
Finalmente, se puede agregar que es posible que muchas técnicas de programación, se pueden implementar de una manera mucho más fácil, en un salones conjuntos que en lugares dispersos.

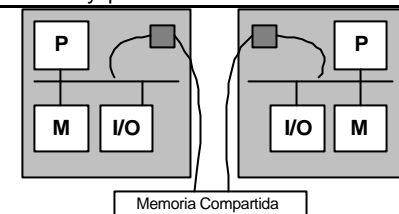
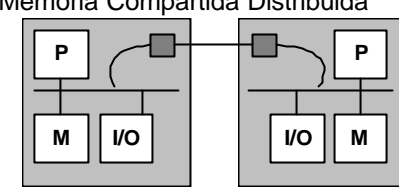
2.3 Estructuras de Hardware de Clusters.

Existen muchas maneras de diferenciar las estructuras de hardware que se pueden emplear en un cluster. Acá se empleará una organización basada en dos características ortogonales muy relacionadas.: El método de conexión a todas los computadores involucrados; y si el medio de comunicación emplea o no alguna forma de almacenamiento compartid.

A - Las cuatro categorías

Los clusters se componen en su gran mayoría de computadores convencionales organizados. Estas maquinas tienen un número limitado de lugares en los cuales la conexión con otra computadora puede ser llevada a cabo. En realidad, estos lugares son exactamente dos: En el subsistema de I/O y en el subsistema de Memoria.



Conexión de Memoria	<p>Convenciones de Software en el Uso de memoria compartida (Ver figura de abajo a la derecha)</p> <p>u</p> <p>Optimización de conexión de I/O Basada en Mensajes.</p>	<p>Parallel Sysplex de IBM.</p>  <p>Parallel Sysplex de IBM con Facilidad de Acoplamiento.</p> <p>Memoria Compartida Distribuida</p>  <p>Sistemas similares a NUMA.</p>
----------------------------	--	---

En el caso de conexión por I/O, la comunicación es realizada usando operaciones de Entrada/Salida que son controladas por el sistema operativo. En el caso de conexión por memoria, procesador-nativo, es el resultado de carga y almacenamiento en memoria de la información entre los nodos. La S/E por memoria mapeada puede aparecer para embarrar esta distinción, pero realmente no lo afecta. La carga y almacenamiento de las E/S de Memoria mapeada no afectan el subsistema real de hardware de memoria.

Las propiedades del medio de comunicación en si mismas son ampliamente independientes de donde están conectadas. Eso forma la segunda distinción: El medio de comunicación contiene o no explícitamente de algún modo uso de almacenamiento compartido.

Si no hay uso explícito de almacenamiento compartido, este sistema se conoce como *Basado en Mensajes* o de *Paso de Mensajes*. En estos sistemas la comunicación finaliza cuando la información ha sido recibida por el nodo destino. Un sistema basado en mensajes puede contener almacenamiento incidental consecuencia del uso de algunos registros para mantener datos mientras se transfieren al otro nodo. Por el contrario, en los sistemas de almacenamiento compartido, la comunicación desde un nodo es finalizada cuando la información es depositada por completo en el almacenamiento compartido.; la ubicación en el almacenamiento es explícitamente utilizada por el procesador como el destino de la información.

La categoría de almacenamiento compartido soporta unas pocas subcategorías. El almacenamiento compartido puede ser, por ejemplo, electrónico (rápido, memoria) o puede ser electromecánico (lento, con disco). Esta diferencia, si bien es prácticamente significativa, no afecta la estructura del sistema, solamente los promedios de información. Sin embargo hay otra distinción, esta es estructural: donde se localiza el almacenamiento compartido. Si se distingue de los almacenamientos propios de cada nodo o si forma parte de ellos.

B – Conexión de I/O Basada en Mensajes.

Los sistemas de conexión de I/O basados en Mensajes son los más comunes y los más básicos. En esta categoría se encuentran todos los cluster que se hallan conectados por LAN, FDDI, ATM y FCS, así como aquellos cluster conectados por sistemas de switching propietarios. Son básicos porque todas las demás combinaciones también incluyen alguna forma de esta conexión, al menos en el nivel de un sencilla interrupción de nodo que puede realizar otro nodo para informar cuando cierta información está esperando por él en algún lado. Es la forma más sencilla puesto que se puede utilizar hardware abierto (no propietario).

C – Conexión de I/O con Almacenamiento Compartido.

La manifestación más común de este modelo es el sistema de disco compartido. En este caso una controladora de disco es capaz de aceptar comandos de 2 (o más) computadores, cada uno de los cuales lee información del disco que ha sido escrita por el otro. El cluster *Open VMS* de DEC, es un claro ejemplo de esta arquitectura. También se destacan el *Sysplex* de IBM y el *SPARCcluster x000 PDB* de Sun Microsystems. En base de datos podemos mencionar el *Parallel Server System* de Oracle que también trabaja con este estándar.

Los sistemas conectados por almacenamiento compartido como estos sistemas, requieren de alguna manera, la forma de apropiarse de segmentos del almacenamiento para no pisar la información de

otro. Esta “arbitrariedad” es realizada en alguna forma por algún control de coherencia basado en software, ya que alguna máquina puede tener en memoria algún bloque de información que actualizó y aún no la ha vuelto a escribir al disco; particularmente cuando se utilizan almacenamiento por semiconductores, pero igualmente existe asistencia proveniente del hardware.

D – Conexión de Memoria con Almacenamiento Compartido.

Los sistemas de conexión de memoria con almacenamiento compartido vienen en 2 variedades: Aquellos con al menos un bloque físico común de almacenamiento compartido separado del almacenamiento propio de cada nodo; y aquellos los que el almacenamiento, contenido en cada nodo, es compartido. Los sistemas conectados por Interconexión Coherente Escalable (SCI – Scalable Coherent Interconnect) comúnmente serán del segundo tipo, lo que no impide que sistemas que emplean SCI accedan a un modulo común de almacenamiento. El sistema IBM POWER /4 RPQ fue un ejemplo de esto último permitiendo acceso al un bloque común de discos compartidos, sin accesibilidad entre los nodos de la memoria local de cada uno de ellos.

El almacenamiento compartido puede ser “lógicamente compartido” como el caso de la utilidad de DEC para Canal de Memoria conocido como Encore's Reflective Memory. Es este caso, al escribir en ciertas zonas del almacenamiento compartido reflexivo resulta, en algún punto, de que todos lo nodos están viendo la información. – Pero en realidad el almacenamiento es replicado en cada nodo. En efecto al almacenar en la memoria reflexiva la información se “refleja” en los demás nodos.

E – Conexión de Memoria Basada en Mensajes.

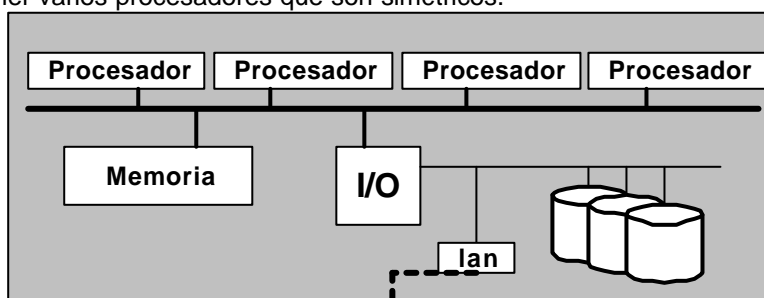
La cuarta categoría de hardware de cluster, conexión de memoria basada en mensajes, no contiene ejemplos actuales. Sin embargo el software puede obviamente emplear la memoria bajo condiciones que se limiten al paso de mensajes.; esto es común es SMPs y perfectamente posible con otros sistemas basado en memoria, pero la base hardware en esos caso es sin embargo de almacenamiento compartido.

Hay excepciones para esto último. DEC aconseja a los programadores en muchos casos para emplear su Canal de Memoria como un dispositivo de paso de mensajes.

2.4 – SMP, el brazo fuerte de los clusters.

Introducción: Multiprocesadores simétricos (SMP's), Sistemas NUMA y los cluster son las formas primarias del procesamiento paralelo. Si bien son tres arquitecturas bien diferenciadas, las mismas a menudo se unen para formar un gran esquema que haga frente a las necesidades actuales de procesamiento de la información.

¿Qué es un SMP? – Un multiprocesador simétrico es una variante de un computador que se caracteriza por tener varios procesadores que son simétricos.



- Más detalladamente, decimos que solamente tiene múltiples procesadores. No tiene múltiples sistemas de I/O, no tiene múltiples memorias. Puede estar en un gabinete o en varios pero eso es transparente para las aplicaciones, subsistemas, sistema operativo, o lo que sea. Es decir, la forma en que se ordenen los procesadores no afecta su arquitectura o función. El hecho de que haya varios procesadores no significa que es invisible para todos los programas, algunos de ellos detectan esta ventaja. Entre ellos está el sistema operativo que entre sus funciones está la de ocultar a ciertas aplicaciones la existencia de varios procesadores. El SO debe ser totalmente consiente de la existencia de varios procesadores pues él el que acelerará los tiempos de las aplicaciones empleando el poder de procesamiento de todos los procesadores que posee.

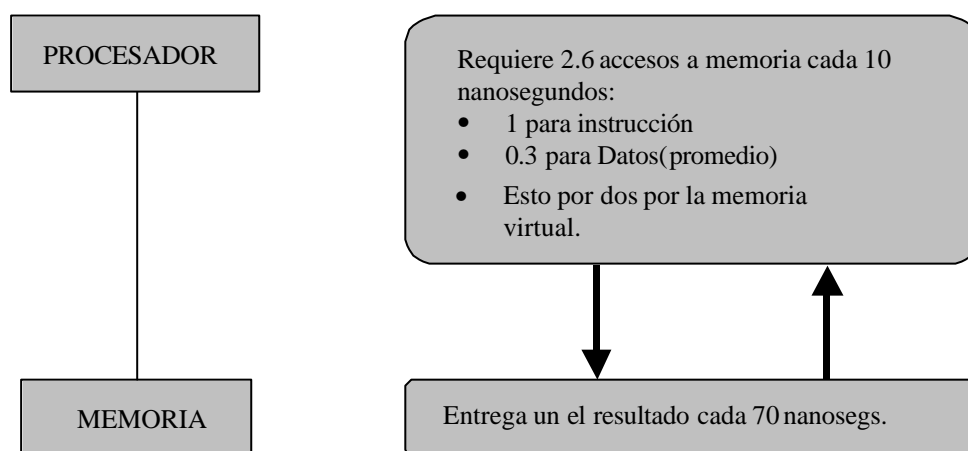
- Los procesadores son simétricos. Esto significa que cada procesador tiene las mismas capacidades. Esto significa que cada procesador puede hacer todo. Tienen acceso a las mismas áreas de memoria, pueden controlar igualmente cualquier dispositivo de control I/O, saltar cualquier interrupción como un simple límite, etc. El resto de la máquina luce igual para cada procesador. Es por todo esto que son *simétricos*. Existe muchos sistemas con Múltiples Procesadores donde esto no es cierto.(ver figura).

Los SMPs son a menudo referenciados como **Multiprocesadores Estrechamente Acoplados** para diferenciarlos de los *Sistemas Pobremente Acoplados*. Estos últimos pueden ser la categoría en la que mejor se pueden encajar a los dusters. Los SMPs y los clusters difieren enormemente en muchas cosas, pero principalmente en como son programados.

Es importante que los programas, en especial los SO, soporten este modelo de programación. Optimizar los tiempos de ejecución y procesamiento es lo más importante en los sistemas actuales.

¿Qué es un Caché, por qué es necesario? Las *memorias de acceso* y las *memorias caché* (o solo cache) la parte central de la utilidad que puede arrojar un SMP.

A continuación se muestra un breve ejemplo del por que de la invención del caché. Existe un conflicto de velocidades entre la velocidad del procesador y la velocidad de memoria.



La memoria central está construida comúnmente en chips grandes, baratos y comprimidos llamados DRAM – Dynamic Random Access Memory. Cuando se consulta algún valor en estos chips el mismo tarda de 40 a 100 nanosegundos en devolver el dato solicitado. Tomemos un promedio de 70 nanosegundos. Aunque este promedio es ficticio porque en el caso de acceder ha varios bancos de memorias este tiempo se eleva.

Los procesadores, por otra parte, corren con un reloj que varía entre 50 MHz hasta varios cientos de MHz. Esto significa que la velocidad interna de un ciclo varía entre 50 nanosegundos y 2 nanosegundos. Nuevamente para simplificar las diferencias tomemos una estimación de 10 nanosegundos, lo que corresponde a un reloj de 100MHz.

Un procesador RISC moderno trabaja muy duro y a veces puede llegar a ejecutar varias instrucción en uno de los ciclos. Algunas máquinas CICS con VLIW (Very Large Instruction Word) llegan a ejecutar diez instrucciones en un ciclo, pero en general nunca se llegan a obtener estas instrucciones ejecutadas simultáneamente. Lo que se ha obtenido depende en gran medida del procesador, compilador y de varios factores del hardware, pero fundamentalmente del programa que se está ejecutando. Ejecutando sobre un RISC se podría obtener desde 3 ciclos/instrucción hasta 0,2 ciclos/instrucción. Otra vez, para simplificar tomamos 1 ciclo/instrucción.

Ahora: Cada instrucción (10 nanosegundos) necesita al menos una, cuando no varias referencias a memoria (70 nanosegundos). Acá surge el principal problema. ¿Por qué todas las referencias se hacen a memoria?

Bueno, primero se selecciona la siguiente instrucción. Eso es una referencia a memoria. Además puede que tengamos que hacer otra referencia más en el caso que la instrucción implique una lectura desde memoria.

Esta diferencia da a simple vista un motivo para buscar algún método más rápido para evitar esta sobrecarga. Es acá donde aparece la Memoria Caché....o simplemente caché.

Referencias en caché. Existe una manera relativamente más barata de solucionar las diferencias de velocidades que no sea la compra de equipo más rápido y memoria más cara (aunque esta compra igualmente no resuelve el problema del cuello de botella existente) y es el empleo de caches.

Teniendo en cuenta, el principio de cercanía existente en los programas, por el cual es posible que una instrucción sea nuevamente referenciada nuevamente lo que se trata es de mantener esas instrucciones recientemente ejecutadas para que, en el caso de volver a necesitarlas, no tenga que acceder a memoria sino que pueda accederlas donde las tenga guardadas.

El objetivo es ubicar una pequeña memoria de muy alta velocidad entre el procesador y la memoria central. Esta pequeña memoria es la denominada **caché**. El costo de estas memorias es muy elevado por lo cual no pueden tenerse toda la memoria central con esta arquitectura.

Ahora el cómo se accede al caché en lugar de la memoria es todo un tema de hardware. Al momento de solicitarse una instrucción a memoria el procesador examinará primero la caché. De encontrarse allí la utilizará. Pero en el caso de no hallarse irá a la memoria central. Debido a que al no encontrar el dato buscado en el caché se tiene que ir a memoria se está generando un tiempo extra, entra acá un concepto conocido como **Porcentaje de Acierto del Cache**. El mismo se determina por la cantidad de veces que buscó en la caché y encontró contra todas las veces que tuvo que ir a buscar datos. A mayor porcentaje mayor velocidad general de proceso.

Una manera de acelerar más aún esto es mantener un doble caché. El primero con las instrucciones recientemente ejecutadas y otro con datos recientemente requeridos. El la primera de estos caché tendríamos asociado un TLB (Translation Lookaside Buffer) en el cual se pondrían muy pocas de las instrucciones recientemente ejecutadas. El procesador entonces primero examinaría el TLB, en caso de encontrar lo que esta buscando lo utiliza, caso contrario va a caché y en caso de no hallar nada, va a la memoria central.

Caché de Nivel 2 Estas memorias caché no muy grandes y rápidas son extremadamente caras por lo que surge la posibilidad de incorporar una memoria un poco más lenta pero más grande (siempre más rápida que la memoria central) en la cual se pueden guardar más cantidad de información y accederse desde los caché a la misma.

Esta nueva memoria se conoce como caché de nivel 2.

Líneas de Caché. Otro aspecto importante en el desarrollo de los SMPs es cuanta información se trae desde la memoria por vez.

El caché no se trae solo lo que le pide la CPU, lo que podría llegar a ser tan chico como un byte. Tiene que manejar variables fijas, pedazos de información.

En lugar de traer solo lo pedido se traen un tamaño *conveniente* de información. Se entiende por conveniente alguna potencia de dos que puede variar de 4 bytes a 256 bytes. Entonces *Línea de Cache* es el término para indicar la porción de información desplazada entre la memoria y el caché.

3 – Redes

3.1 Redes de Alta Velocidad

La red es la parte más crítica de un cluster. Sus aptitudes y performance influyen directamente en la aplicabilidad del sistema principal de High Performance Computing (HPC). Después de describir algunos puntos de diseño en general para redes de alta velocidad se presentarán en detalle varios clusters muy conocidos que se conectan entre sí. Su arquitectura y propiedades principales son descriptas y evaluadas en el orden en que la redes fueron evolucionando a lo largo de los años, comenzando por las Local/Wide Area Network (LAN/WAN) como las Fast Ethernet y ATM, a las System Area Network (SAN) como las Myrinet y Memory Chanel.

3.2 Elección de Redes de Alta Velocidad

Para efectivizar el trabajo en un ambiente distribuido, una red rápida es de gran importancia. La importancia creciente de las LAN's actualmente y la creciente complejidad de las aplicaciones de escritorio están incentivando la necesidad de utilizar redes de alta velocidad. El ancho de banda proporcionado por una conexión Ethernet de 10 Mb. no es adecuado para un sistema distribuido que lleva una carga de alta velocidad para cálculos de estudios científicos o la carga de trabajo de una industria en una red corporativa o eventualmente la carga típica de las aplicaciones de escritorio.

Una aproximación más especializada en redes y a un bajo costo, una alta performance de interconexión optimizada especialmente para incrementar tanto la performance en paralelo como la alta disponibilidad de los aspectos del cluster.

Fast Ethernet es un término genérico empleado para una familia de LANs de alta velocidad corriendo a 100Mbit/s sobre cable UTP o fibra óptica. Emplea el protocolo CSMA/CD (Carrier Sense Multiple Access with Collision Detection).

Interfase Paralela de Alto Rendimiento (HiPPI – High Performance Parallel Interface)

HiPPI es un standard de comunicación de datos basado en cobre capaz de transferir 888Mbit/s sobre 32 líneas paralelas o 1.6 Gbit/s sobre 64 líneas paralelas. HiPPI es un canal punto-a-punto que no soporta configuraciones con bajadas de líneas múltiples.

Modo de Tránsito Asíncronico. (ATM)

ATM es un conmutación de paquetes orientados a la conexión. Es una tecnología de largo fijo (53 bytes), altamente adecuado para áreas extensas. ATM puede manejar cualquier tipo de información, por ejemplo voz, datos, imágenes, texto, video de una manera integrada. La tecnología ATM jugará un rol central en la evolución de los clusters ya que dará importantes ventajas sobre las tecnologías LAN y WAN existentes, incluyendo la promesa de anchos de bandas escalables a un precio y rendimiento sin precedentes y una calidad de servicio garantizada, lo que facilitará nuevas clases de aplicaciones.

Interfase Coherente Escalable (SCI)

SCI es un estándar de comunicaciones reciente para interconexión de clusters. SCI eleva el direccionamiento para permitir mayores recursos y poder computacional, aliviando a los programadores de las técnicas de paso de mensajes. Esta interfase de programación fácil de usar y el hecho de que SCI es un estándar de IEEE, se ha vuelto muy popular para los recientes clusters de PC.

ServerNet

ServerNet es una SAN cuyos procesadores clusters y otros dispositivos proveen alto ancho de banda, escalabilidad e integridad requeridas por las aplicaciones HPC. La segunda generación de ServerNet provee un ancho de banda de 125 Mbytes/s entre dos nodos en un sistema cluster.

Myrinet

Myrinet es una red de costo/efectividad y Gigabits/Segundos basada en las tecnologías empleada en los súper computadores. A gusto con los ambientes intracomputacionales, las latencias de comunicación entre los nodos están ampliamente reducidas. Una de las principales razones para su gran aceptación es el microcontrolador programable adjunto que permite a los investigadores ajustar la conexión a sus necesidades específicas.

Canal de Memoria (Memory Channel)

La red de Memoria Canal es una interconexión dedicada de cluster que provee memoria virtual compartida entre los nodos mediante mapeo de espacio de direcciones entre nodos. La conexión implementa mensajería directa a nivel de usuario garantizando el orden estricto de los mensajes bajo todas las condiciones, incluyendo transmisión de errores. Provee un ancho de banda físico de 100 Mbytes/s

SynFinity

SynFinity soporta paso de mensajes así como memoria compartida a muy altos promedios de transmisión de datos (arriba de 1.6 Gbyte/s). Además las operaciones de envío/recepción normales, las transacciones remotas de memoria y el soporte de hardware para sincronización ofrecen la capacidad de implementar capas de protocolos optimizadas.

3.3 Evolución de las tendencias de Interconexión.

Históricamente, la capacidad computacional de los servidores comunes UNIX de alto rendimiento ha medido la habilidad de las redes para conocer el ancho de banda requerido para grandes grupos de datos distribuidos. Las tecnologías de redes de alta velocidad fueron restringidas a los centros de súper computadores. La industria de redes ha crecido exponencialmente y está tomando un rol preponderante en la creación de tecnologías disponibles para las masas.

Ethernet ha sido una tecnología popular de red gracias a su simplicidad y al hecho de que las redes existentes necesitaban no ser modificadas para soportar una nueva tecnología. Pero Ethernet transmite un frame y espera una respuesta. A medida que la distancia entre los nodos se agranda la espera de confirmar empeoran. IEEE 802.5, el cual es el estándar de redes Token Ring, mantiene el token mientras el frame va alrededor de la red. FDDI conserva la ruta completa cuando solamente emplea una parte de ella. Estas tecnologías son características de la vieja generación donde la alta velocidad no era un factor y las latencias ofrecidas eran muy altas (100 – 1000s microsegundos), ambas con un ancho de banda relativamente bajo (1/10 Mbit/s). Retransmisión selectiva, corrección adelantada de errores, anticipación, búsqueda anticipada y acuerdo implícito son algunas de las características requeridas en la nueva generación de redes de alta velocidad. Otra limitación de las tecnologías de redes estándar es que existe un techo en el total de ancho de banda disponible para la red. Esto significa que sobre un segmento de red de 100 Mbit/s con ocho sistemas, cualquier sistema tratando de comunicarse con la red tendría solamente 12,5 Mbit/s de rendimiento disponible para su uso.

Para las necesidades actuales de interconexión, se requieren redes de muy alta velocidad, y las tecnologías de redes de alta velocidad se han transformado en una atractiva solución de interconexión para los mercados comunes. Las tecnologías de redes de emergentes prometen capacidades de rendimiento en el rango de uno a varios Gbit/s. Adicionalmente, muchas tecnologías de red nuevas poseen capacidades de distancia mejoradas para medios ópticos. Las viejas tecnologías de redes de alto rendimiento eran, a menudo propietarias limitando su conectividad. La reciente transición de Ethernet a LAN switcheadas, así como Ethernet switchheada y ATM han incrementado los anchos de banda mientras que sus latencias continúan en el rango de los cientos de microsegundos.

3.4 Cuestiones de Diseño

A continuación se detallan los principales puntos en el diseño para la interconexión de hardware. El adaptador de red (Tarjetas de I/O, chips de sistema) se detallan como **Interfase de Red** (NI = Network Interface), mientras que la conexión entre dos nodos (adaptadores o switches) se detallan como *enlace*.

3.4.1 Aciertos

Varias decisiones deben ser hechas cuando se diseña la interconexión de cluster. La más importante es, innegablemente, la relación Precio/Rendimiento.

Precio vs Rendimiento.

En los últimos años, los clusters de PCs han ganado creciente popularidad gracias a los precios extremadamente bajos de las PCs estándar. Tradicionalmente la tecnología de súper computadores es reemplazada por PCs interconectadas muy compactamente. En el mercado de interconexión existe una gran franja entre una interconexión de un ancho de banda moderado como Fast Ethernet a un bajo precio (u\$s50-100 por un adaptador de red); y una red de alto rendimiento como Myrinet o ServerNet (u\$s1000 o más). Por supuesto esto también es consecuencia de los bajos volúmenes de producción. Pero otros factores como RAM on-board, o costosas capas de hardware como la Fibra Canal, pueden elevar el costo considerablemente.

Escalabilidad

La escalabilidad es otro aspecto importante. Se refiere a la posibilidad de la red de crecer también linealmente con el número de nodos del cluster. La interconexión en súper computadores tradicionales tiene una topología de red fija (malla, hipercubo) y el hardware o software descansan en esta topología fija. Pero los clusters son más dinámicos. A menudo se ajusta un pequeño sistema para test para ver si encaja con las necesidades de cierta aplicación. Con la creciente demanda de poder, se agregan más y más nodos al sistema. La red debe soportar la creciente carga y poder entregar un ancho de banda y latencia similar para un cluster pequeño (8-32 nodos) así como para uno grande (cientos de nodos).

Disponibilidad.

Las aplicaciones de computación paralela pueden ser duramente divididas en dos grandes clases principales, científicas o de negocios. Especialmente en el campo de los negocios, la corrupción o pérdida de información no pueden ser toleradas. Para garantizar la entrega de información, se emplean software de protocolos de las tradicionales redes LAN/WAN como CRCs, buffer de datos, mensajes de reconocimiento y retransmisión de datos corruptos. Esta capa de protocolo, ha sido identificada como la principales razones de la pobre latencia de estas redes. Para los clusters que necesitan una baja latencia, y capas delgadas de protocolos, estas sobrecargas deben ser minimizadas.

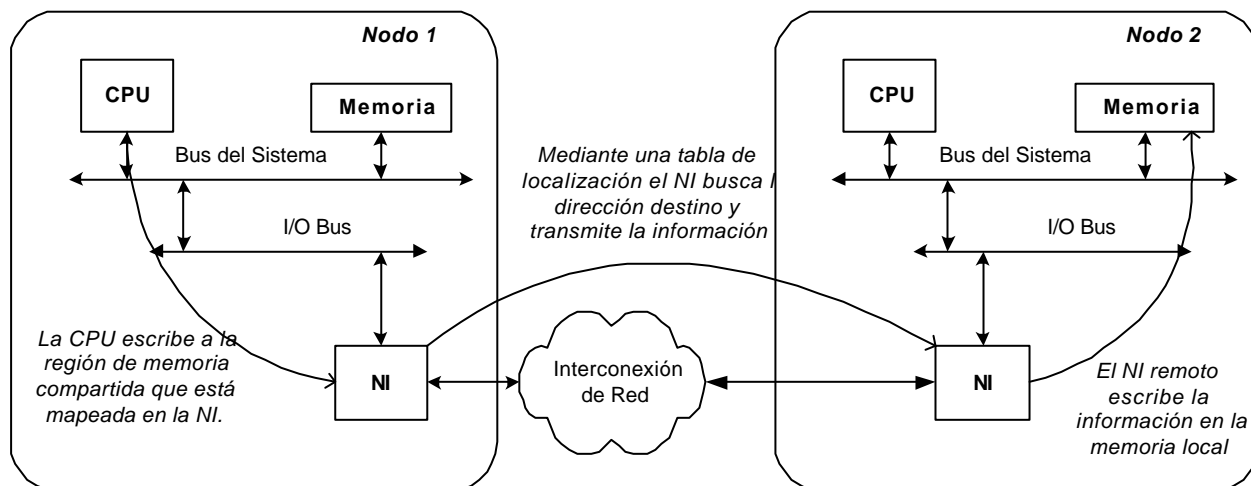
3.4.2 Arquitectura General

Una decisión general de diseño debe ser hecha entre una NI muda, la cual es controlada por un procesador, y una NI inteligente y autónoma realizando la mayor parte del trabajo por si misma. La primera solución tiene la ventaja de un bajo esfuerzo de diseño pudiendo con esto lograr un mejor tiempo de puesta en marcha y menor costo de rediseño. En la otra mano, permitiendo que la NI haga los trabajos, como transferencias de información o controlando el ID del receptor que es la dirección/path de la red liberando al procesador de esta tarea los que generará una menor latencia en el inicio de un transferencia de mensajes.

Memoria compartida vs Memoria Distribuida.

La primera decisión de un diseñador en la interconexión de un cluster es el modelo de memoria (programación) a ser soportado. El modelo de memoria compartida permite a los procesos ver la red del cluster transparente a través de todo el espacio global de direcciones. El hardware y software de administración de memoria virtual se emplea para mapear direcciones virtuales a las direcciones físicas locales o remotas. A partir de que la sobrecarga de aplicar de este modelo a todo el espacio de direcciones es bastante costoso, la interconexión soportando memoria compartida ofrece al habilidad de páginas de memoria remota en el espacio de direcciones local de las aplicaciones, como el Memory Channel de DEC.

En el modelo de memoria distribuida, el software de paso de mensajes hace que la red se torne visible a las aplicaciones. La información puede ser enviada a otros nodos mediante llamadas de envío/recepción a la API. Comparado con el modelo de memoria compartida, el usuario debe realizar llamados explícitos a las rutinas de comunicación hacia o desde la red.



Operación de Escritura a Memoria Remota.

Ubicación del NI

La ubicación del NI dentro de un sistema tiene un gran impacto en lo que respecta a rendimiento y reusabilidad. En general, cuanto más cerca está del procesador un mayor ancho de banda está disponible.

NI-1: Una solución interesante es soporte para comunicaciones a nivel del conjunto de instrucciones dentro del procesador. Mediante el movimiento de información dentro de registros, la misma se transfiere dentro de la red a la velocidad del procesador. Esta técnica se ha empleado varias veces en

el pasado en algunas arquitecturas; el más famoso de estos ejemplos es el sistema Transputer de INMOS. El mismo funcionaba con cuatro links sobre chips a velocidad total del procesador.

NI-2: Asumiendo un diseño del bus del sistema de alto rendimiento, esta ubicación es un lugar ideal para una interfase de red. Los sistemas de bus de hoy en día ofrecen muy altos anchos de banda del rango de los Gbytes/s. Los mecanismos de Coherencia de Cache pueden ser empleados para observar eficientemente el estado del NI. El procesador puede escrutar los registros del NI para Coherencia del Caché y así obtener información sin ocupar ancho de banda.

NI-3: Las interconexiones más comunes poseen interfaces con el Bus de I/O, principalmente PCI. La principal razón en la gran aceptación del PCI como un estándar de bus de I/O. Los NIs basados en PCI pueden ser conectados a cualquier PC o estación de trabajo, aún formando clusters heterogéneos.

3.4.3 Detalles de Diseño.

A continuación se realiza un breve detalle de algunos detalles específicos de implementación. Pequeñas modificaciones en el hardware pueden tener un gran impacto en el rendimiento del NI.

Protocolo de Enlace.

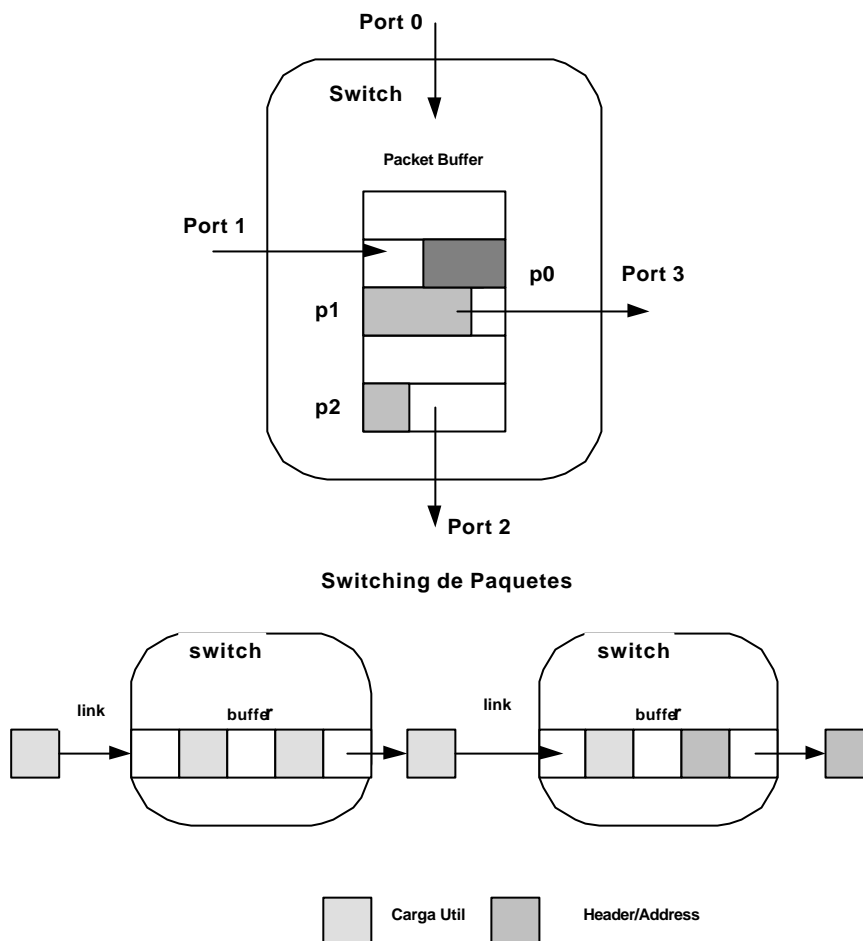
El término *protocolo de enlace* es empleado para el esquema de mensajes, los cuales son transmitidos sobre una capa física y los puntos finales del canal de comunicaciones. Típicamente, la información de un mensaje está encerrada por palabras de control especiales, las cuales se emplean para detectar el inicio y fin del mensaje y para señalar eventos del protocolo de enlace (el receptor no puede recibir más información, pedido de transmisión, etc.).

Capa Física.

Seleccionar el medio físico adecuado para el canal es un acuerdo entre el promedio de datos a procesar y el costo del cable. Los medios seriales ofrecen un ancho de banda moderado, pero pueden tener que depender de una nivel de enlace de la capa de transporte estándar como Gigabit Ethernet o Fiber Channel. Se pueden construir switches de red de tamaño moderado, mientras el promedio de datos a procesar no exceda en demasía el del medio serie. A partir de que los largos de las transmisiones eléctricas están muy restringidas, las capas ópticas podrán reemplazar en un corto plazo a los cables de cobre en su totalidad. Pero el costo de hoy en día (\$300 por cada interfase serie óptica) evita el total uso de componentes ópticos.

Switching.

La conexión entre los puertos de entrada y salida y la transferencia de información de entre ellos es conocida como switching. Existen dos técnicas principales empleadas en las redes de hoy en día, *Paquete* o *Agujero de Gusano*. El primero almacena un mensaje completo en un paquete de red antes de ser enviado a la próxima etapa. Este mecanismo de almacén y envío necesita un límite superior para el tamaño de unidad (MTU – Maximum Transfer Unit) y algún espacio de buffer para almacenar uno o varios paquetes temporalmente. Las más tradicionales redes LAN/WAN emplean paquetes (Fast Ethernet, ATM).

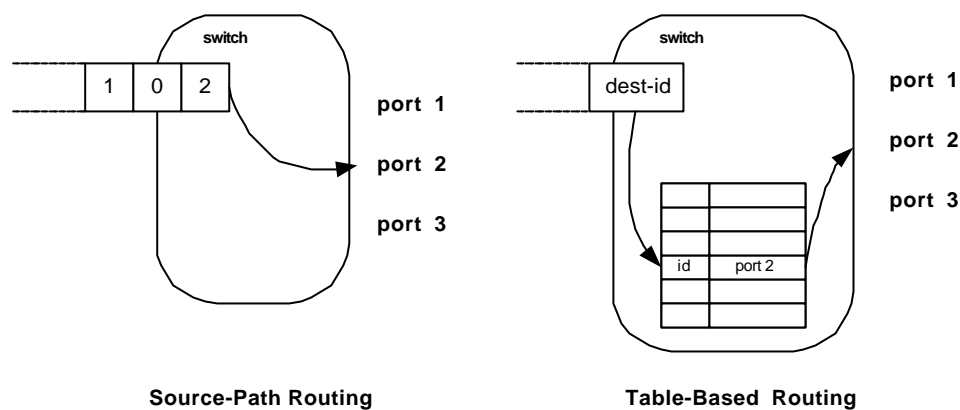


Switching de Agujero de Gusano

Las SANs más nuevas, como Myrinet, emplean switching del tipo Agujero de Gusano, donde la información es enviada automáticamente a la siguiente etapa tan rápido como se decodifique la dirección de destino de la cabecera. La baja latencia y el poco espacio de buffer requerido son parte de esta técnica. El largo del mensaje puede ser definido como variable, pero el diseñador debe tener en mente que según el largo de mensaje puede o no ocupar varios bloques que deberán pasar por varias etapas. También se debe aclarar que con esta técnica la corrección de errores se vuelve más difícil. El switching de paquetes puede primero chequear un paquete entero antes de ser reenviado a la etapa siguiente. Empleando la técnica de Gusano la información corrupta puede ser reenviada antes de que sea reconocida como errónea.

Routing

La cabecera de dirección en un mensaje lleva consigo la información necesaria para el hardware rutéo dentro de un switch para determinar el canal de salida correcto. Los esquemas de routing adaptables tratan de encontrar, dinámicamente, camino alternativos entre ellos y el destino de la información a través de toda la red, en caso de congestión o en caso de imposibilidad de usar un enlace en particular. Para realizar el routing se emplean dos mecanismos: source-path y table-based routing.



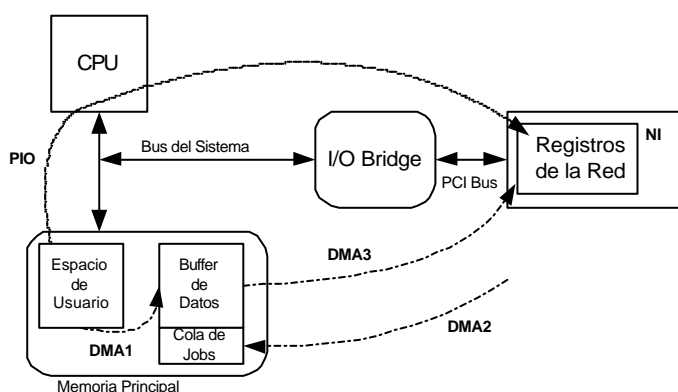
Control de Flujo.

El control de flujo se emplea para evitar que el buffer se sature dentro del switch, lo que podría resultar en la pérdida de información. Antes de que el origen pueda empezar a transmitir la información, el receptor debe informar acerca de la posibilidad de recibirla o no. Una solución posible es la esquema basado en crédito, donde cada transmisor tiene una serie de créditos del receptor. Cada vez que el transmisor envía un paquete se consume un crédito, esto finaliza cuando se le consumen todos los créditos al transmisor. Luego de liberar algo de espacio en el buffer en receptor puede continuar con la transmisión enviándole al transmisor créditos para que este último esté en condiciones de enviar nuevos paquetes. O también puede que el transmisor envíe una señal al receptor para ver si este está en condiciones de recibir nuevos paquetes. En ambos casos el control de flujo se realiza en sentido contrario de la información.

Transferencia de información.

La transferencia efectiva de mensajes entre los nodos de la memoria central y el NI es un factor crítico al que hay que llevar al margen del ancho de banda físico en las aplicaciones de usuario reales. Para poder salir exitoso en este proceso, los protocolos modernos de NI involucran solamente al SO, solamente cuando se abren o cierran canales de comunicaciones con las aplicaciones de usuarios. La transferencia normal de información es realizada íntegramente en modo usuario por rutinas de librerías para evitar el costo de llamadas al sistema operativo. La clave de la transferencia es la *cero copia* de datos, donde la información es transmitida directamente entre el espacio de usuario en la memoria central y la red.

I/O Programada (PIO) vs Acceso Directo a Memoria (DMA).



Transferencia PIO vs DMA

La información de los mensajes puede ser transferida de dos maneras: Entrada/Salida programada (PIO), donde el procesador copia la información entre la memoria y el NI.; y el Acceso Directo a Memoria (DMA – Direct Memory Access), donde el dispositivo de red inicia la transferencia por sí mismo. A continuación se muestran ambos mecanismos desde el punto de vista del transmisor.

PIO requiere que algunos de los registros estén mapeados dentro del espacio de direcciones del usuario. El procesador entonces es capaz de copiar información desde cualquier dirección virtual al NI o viceversa. PIO ofrece muy bajas veces de inicio, pero se vuelve ineficiente con el crecimiento del tamaño de los

mensajes, a partir de que parte del tiempo del procesador es empleada solamente para rutinas de copia de información. DMA necesita un bit más de soporte a partir de que el controlador de DMA dentro del NI necesita direcciones físicas para transmitir la información correcta. La mayoría de las interconexiones que ofrecen transferencia por DMA requieren que las páginas estén obligatoriamente en memoria puesto que no pueden acceder a disco.

Varios factores influyen en el rendimiento de ambos mecanismos. La implementación de PIO más sencilla escribe información del mensaje secuencialmente dentro de un registro del NI, el cual reside en el espacio de I/O. Esto normalmente lleva varios ciclos del bus con un pobre ancho de banda. Para obtener un adecuado ancho de banda, el procesador debe ser capaz de poder ganar ciclos del bus. Esto puede ser hecho mediante la elección de una pequeña área de direcciones como objetivo, la cual es tratada como memoria. Escribiendo en estas direcciones consecutivas se permite al procesador o al bridge aplicar técnicas como la combinación de escritura, donde se realizan varias escrituras sobre un buffer y luego se escribe todo en una única ráfaga transaccional. Este mecanismo se puede encontrar en arquitecturas como la Alpha de DEC o en Intel. Debido a que el bus PCI implementa transacciones por ráfagas de largo variable, el controlador de DMA dentro del NI podría tratar de leer/escribir bloque grande de información en un solo ciclo.

Para resumir se puede decir que PIO es superior a DMA pero solo para mensajes cortos arriba de una cierta medida en la cual la sobrecarga originada por la copia desaprovecha al procesador. Está altamente comprobado que el 90% del tráfico de una red lo originan mensajes de longitud corta. Pero es obvio que los diseñadores deben implementar soportes para ambas tecnologías.

Sondeo o escrutinio (polling) vs Interrupción.

Si se emplea DMA, otra elección crítica de diseño que se debe hacer es la manera de avisar al procesador de la finalización de la recepción total de un mensaje (también conocido como control de transferencia). En modo de sondeo, el procesador lee continuamente un registro de estado del NI, donde el NI setea un bit de estado cuando finaliza una transacción. Si el NI reside sobre el bus de I/O, esto puede desperdiciar mucho ancho de banda. Como mejora, el NI puede reflejar su estado en memoria central. Otra solución es interrumpir al procesador, pero esto genera un cambio de contexto a modo kernel, lo que es una operación costosa. Una híbrida sería posibilitar al NI de interrumpir al procesador cuando la información de la transferencia está presente.

4 – Software

4.1 Software RAID

Una de las principales ventajas de los clusters de workstations es la gran cantidad de recursos disponibles en el sistema (discos, memoria, unidades de cintas, etc.). Tradicionalmente, todos estos recursos no estaban accesibles desde todos los nodos de una red. Solamente los recursos que estaban en la maquina con el dispositivo deseado podían acceder a él. Y, si había manera de acceder a los recursos, las mismas no eran ni simples ni transparente.

Si se necesita llevar a cabo un filesystem de alto rendimiento, primero debemos examinar las características de un cluster de workstations y tratar de construir un filesystem mejor.

La primer ventaja, y la más obvia, es la gran cantidad de recursos disponibles en un cluster de workstations. Estos sistemas tienen varios discos que pueden ser tratados paralelamente para incrementar el ancho de banda.

La segunda ventaja encontrada en esta clase de ambiente, son las redes de alta velocidad que están disponibles conectando todos los nodos.

Existen dos problemas comunes en el diseño de un cluster de workstations. El primero se refiere a la visibilidad, por parte del usuario, de la información. El segundo problema consiste en realizar un sistema de I/O de alto rendimiento. En este último problema es importante tener en cuenta que existen ciertas demoras que no son fáciles de reducir como las ocasionadas por los movimientos de las partes mecánicas del disco (Movimiento del cabezal, rotación del disco, etc.). Más aun, no se espera que estos movimientos mecánicos se optimicen. Por esta razón, la única forma de acelerar el rendimiento del disco, es lograr que la información se guarde de tal manera que las partes mecánicas tengan un leve efecto en el rendimiento global del disco.

Para el primer problema planteado acerca de la visibilidad de la información, el mismo se puede solucionar si se lleva a cabo un buen uso de los recursos. Para solucionar esto, varios sistemas distribuidos como Sun NFS (Network File System), CODA (Constant Data Availability), Sprite Filesystem, etc. han usado el concepto de UNIX de *montar* (mount) para incrementar la disponibilidad del sistema.

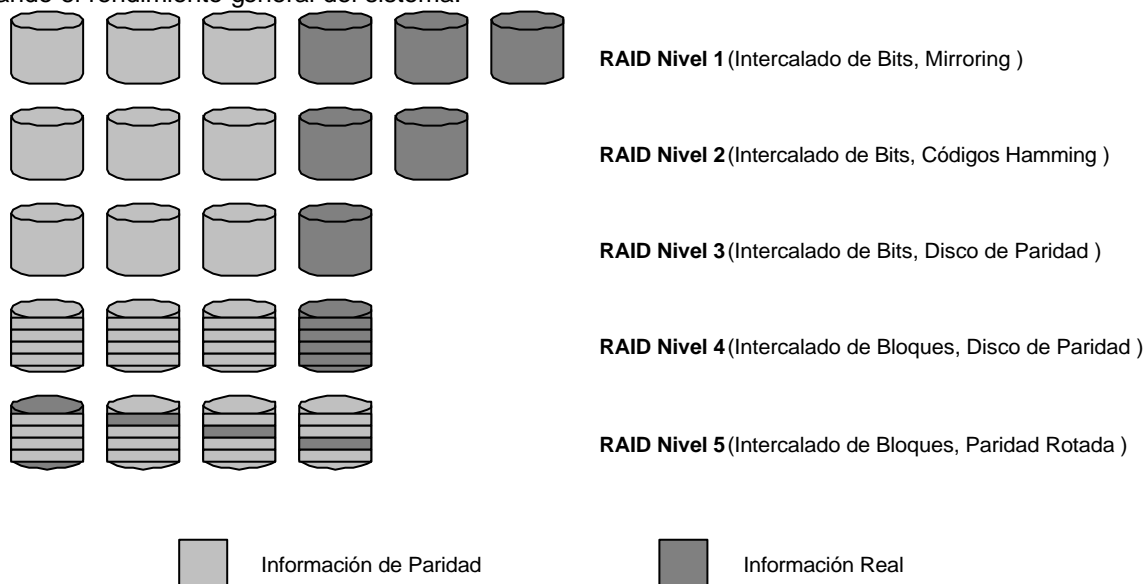
La idea usada en estos sistemas consiste en administrar sistema de archivo remoto como si los mismos fueran locales. El administrador del sistema puede montar un filesystem remoto en cualquiera de los directorios ya accesibles de la estructura actual del sistema. Cuando se realiza un pedido de información a uno de estos directorios, la petición es reenviada al sistema dueño del nodo. Este sistema remoto ejecuta la operación y envía el resultado al nodo que lo requirió. Toda esta operación es realizada de una manera transparente al usuario, el cual tiene la impresión de un único filesystem.

En cuanto al segundo inconveniente, se ha mencionado que los clusters de estaciones de trabajo pueden tener uno o varios discos conectados a ella. La idea es distribuir la información entre los distintos discos para que pueda ser recuperada lo más paralelamente posible. Esto incrementará el ancho de banda de transferencia de la información tantas veces como discos sean empleados en esta paralelización.

4.2 RAID (Hardware)

La primera vez que se implementó esta idea no fue en un cluster de estaciones de trabajo pero sí en conseguir un *único disco* con gran ancho de banda. Esta idea fue conectar varios discos a una única controladora y dar la impresión de que el disco tenía un ancho de banda de transferencias mayor. Esta clase de disco son conocidos en la actualidad como RAID (Redundant Array of Inexpensive Disks).

El alto rendimiento de los RAIDs se debe principalmente a tres razones. La primera es que la información de cada disco puede ser traída al mismo tiempo aumentando con ello el ancho de banda. La segunda es que todos los discos pueden ejecutar la búsqueda en las pistas en paralelo reduciendo el tiempo total. Finalmente, en algunos tipos de RAID, se pueden manejar más de un pedido en paralelo, mejorando el rendimiento general del sistema.



Representación Gráfica de los cinco niveles de RAID

Como se desea acceder a la mayor cantidad de discos en paralelo como sea posible, la información debe estar distribuida sobre los discos de una manera adecuada. Intercalar la información entre los discos parece la manera correcta de hacerlo. Empleando esta distribución, si el pedido es lo suficientemente grande, cada disco contendrá al menos un bloque del pedido, y la información de los discos será traída en paralelo, obteniendo el mayor ancho de banda posible. Esta forma de intercalar información puede caracterizarse como de grano fino o grano grueso.

Los arrays de discos de grano fino conceptualmente intercalan la información en unidades relativamente pequeñas y con ello todos los pedidos de información, indistintamente del tamaño, acceden a todos los discos en el array de discos. Si bien esto es rápido, una desventaja es que cualquier solicitud de I/O lógica en cualquier momento es atendida por todos los discos desperdiciando tiempo.

Los arrays de grano grueso, intercalan la información en unidades de gran tamaño con lo que un pedido de información se puede alojar en pocos discos y un pedido grande puede estar en todos los discos. Esto tiene como ventaja que se pueden atender varios pedidos pequeños de una sola vez. Más aún, si muchos pedidos pequeños se hacen en paralelo es tiempo de búsqueda también se consume en paralelo

con lo que el tiempo general disminuye, mientras que en los arrays de grano fino este tiempo sería consecutivamente.

Otro importante factor de diseño es obtener cierto grado de tolerancia a fallas. Como se emplean muchos discos, las probabilidades de fallas bastante alta. Esto significa que el RAID necesita cierto mecanismo para poder llegar a tener fallas sin perder información. La manera en que esta redundancia es lograda basándose en la granularidad del corte la información es lo distingue los cinco niveles de RAID.

Un RAID de nivel 1 emplea la mitad de sus discos para copiar la mitad restante. Cuando se copia alguna información a un disco también es copiada al disco redundante. Cuando se tiene que obtener información se toma de aquel disco que tiene el menor tiempo de búsqueda. El problema de este esquema es que se desperdicia la mitad de la disponibilidad de disco existente.

Para resolver este problema se han propuestos nuevos esquemas de redundancia. Por ejemplo, en el RAID de nivel 2 la redundancia se ha implementado empleando códigos Hamming. Estos códigos contienen información de distintos conjuntos superpuestos. Partiendo de que el número de discos redundantes es proporcional al total del número de discos, la eficiencia del almacenamiento se incrementa como crece el número de discos de información. Si un componente único falla, varios de los componentes de la paridad tendrán valores inconsistentes, y el componente con fallas es la parte común con cada subconjunto incorrecto.

En un RAID de nivel 3 se emplea paridad por intercalado de bits. En este mecanismo de paridad, la información es conceptualmente intercalada a nivel de bit entre los discos de datos, y se emplea un único disco para tolerar cualquier falla de un disco. Este esquema presenta un par de problemas. Primero, solo se puede atender de a un pedido por vez por tratarse de un granulado fino. El segundo problema es que el disco de paridad nunca se emplea en operaciones de lecturas, limitando el rendimiento de lectura.

El RAID de nivel 4 fue diseñado para evitar el primer problema del nivel anterior de RAID. En esta versión la información es intercalada en bloque de tamaño variable entre los discos de datos en lugar de bits. El tamaño del bloque se conoce como unidad de striping. Empleando esta solución se logra poder atender a varios pedidos a la vez, aunque persiste el problema de la degradación de lectura, puesto que aún hay un disco que no se emplea para la lectura.

Finalmente, el RAID de nivel 5 es como el anterior salvo que elimina el problema de cuello de botella en el disco de paridad, mediante la distribución de la paridad sobre todos los discos. Con esta modificación todos los discos son empleados en las operaciones de lectura. Esta versión tiene el mejor rendimiento de pequeña lectura, gran lectura y gran escritura de todos los niveles de RAID. Por otro lado las pequeñas escrituras son algo ineficiente comparado con otros esquemas de redundancia.

4.3 - RAIDs Lógicos.

La misma idea ha sido usada para discos que no se encuentran conectados a una misma controladora. Podemos partir la información entre los discos pertenecientes a una red. La única diferencia es que ahora el filesystem es responsable de ambas tareas, la distribución de la información y mantenimiento del nivel deseado de tolerancia. Esta clase de distribución de información se conoce como RAID lógico o RAID de software. Estos RAIDs se comportan en general como RAID de nivel 5, y tienen las mismas ventajas y problemas que la versión de hardware. La diferencia más importante es que las soluciones de este enfoque pueden seguir diferentes enfoques.

4.4 Planificando Trabajos Paralelos sobre Clusters.

Los están siendo a ser empleados en aplicaciones de alto poder de computación (HPC- High Performance Computing) gracias al elevado costo de las arquitecturas MPP por un lado, y por la amplia disponibilidad de estaciones de trabajos y PCs para red. La pregunta es como poner la carga de trabajo HPC sobre el cluster proveyendo un alto rendimiento para estas aplicaciones, pero no degradando el rendimiento de las aplicaciones originales.

Los puntos que deben ser considerados para poder integrar el soporte de las aplicaciones HPC son:

- La adquisición de recursos. v.g. Como distinguir entre estaciones de trabajos que están activas y aquellas que tienen recursos disponibles.
- Los requerimientos para dar prioridad a los dueños de las estaciones de trabajos, y no causar una notable degradación de su trabajo.
- Los requerimientos para soportar diversos estilos de programas paralelos que pueden generar diferentes restricciones es la planificación de sus procesos.
- El posible uso de control de admisión y políticas de planificación para regular la carga de trabajo de HPC adicional.

Estos puntos son interdependientes entre sí, y la preferencia de uno puede afectar la el rendimiento de los otros.

4.4.1 Background

Primero definiremos posibles estructuras de aplicaciones paralelas.

Modos de empleo del Cluster.

Los clusters son típicamente usados en dos modos principales de operación.

➤ **NOW (Network of Workstations)**

Este modo se basa en ocupar los ciclos ociosos de recursos existentes, particularmente estaciones de trabajo y PCs. Cada una de estas máquinas tiene un dueño, el cual tiene prioridad en utilizar su máquina, pero cuando el usuario está inactivo, la misma se vuelve disponible para el propósito general. Los ejemplos incluyen el proyecto NOW de la Univ de Berkeley, Condor y MOSIX.

➤ **PMMPP ("poor man's" MPP)**

Este modo involucra un cluster dedicado para correr aplicaciones paralelas de alto rendimiento. Acá existen menos restricciones entre la carga normal y la generada por aplicaciones HPC.

Tipos de Trabajos y requerimientos.

Los trabajos paralelos incluyen múltiples procesos interactivos. La estructura del trabajo y los tipos de interacciones ocupan varios requerimientos sobre la planificación del sistema. Los 3 tipos más comunes son:

- Trabajos rígidos con estrecho acoplamiento. Tales trabajos son clásicos de ambientes MPP. Contienen un número fijo de procesos, los cuales se comunican y sincronizan a un alto promedio. Es por ello, que se requiere ejecutar estos procesos en distintos procesadores pero para asegurar la correcta
- Trabajos rígidos con procesos balanceados e interacciones perdidas. Estos trabajos no necesitan que los procesos corran simultáneamente, porque no interactúan frecuentemente.
- Trabajos estructurados como pila de trabajos de tareas independientes. Estos trabajos son ejecutados por un número de procesos trabajadores que toman procesos de la cola y los ejecutan, posiblemente creando nuevos procesos en la tarea.

Trabajos Rígidos con Migración de Procesos

En un ambiente dinámico como NOW, opuesto a lo que es MPP, el subsistema responsable por las aplicaciones HPC no tiene control sobre todo el sistema. Es por ello, que necesita ser capaz de poder responder a cambios en el ambiente.

La migración involucra remapear procesos a los procesadores durante la ejecución. Las razones para migrar incluyen la necesidad de entregar la estación de trabajo a su dueño nuevamente y el deseo de lograr balancear la carga de trabajo entre las demás estaciones. Métricas para la calidad de migración son las sobrecargas que involucra, y el grado para el cual se pasa al proceso desde su localización anterior (en algunas implementaciones, puede tener que dejar algo de su estado detrás)

Los aspectos de los algoritmos incluyen la selección de cual proceso migrar y hacia donde migrarlo. Estas decisiones, por vez, dependerán de la información disponible acerca del nodo local y de los demás nodos. De este modo, los puntos de medidas de carga y diseminación de la información también son importantes.

4.4.2 Trabajos maleables con Paralelismo Dinámico.

Otro enfoque de la dinámica de los clusters de estaciones de trabajo: Las estaciones de trabajo se transforman en disponibles y luego son reclamadas por su dueño en cantidades de veces incontables, y por ello, los trabajos paralelos deben ajustarse a tal variación de recursos.

Los trabajos ejecutándose en paralelo sobre un cluster de estaciones de trabajo, son a menudo pensados como actividad de background. Cada estación de trabajo está preparada para soportar el trabajo de su dueño, y solamente los ciclos inactivos está disponible para los trabajos en paralelos siempre que no interfiera con su trabajo normal.

La mejor manera de asegurarse de que el dueño de una estación de trabajo no está alienada es utilizar solamente aquellas PCs que están inactivas. Si bien esto es un enfoque conservativo, el mismo plantea una carga de trabajo paralela pareja con plena funcionalidad de los recursos.

Emplear las estaciones de trabajo inactivas para correr trabajos paralelos requiere dos cosas:

- La capacidad de identificar las estaciones ociosas. Esto es relativamente fácil si nos basamos en el control de actividad del teclado y mouse. Típicamente, las estaciones que no muestran esta actividad por unos minutos se consideran ociosas.
- La habilidad de migrar desde las estaciones que reclaman sus dueños. Esto es posible si la aplicación está estructurada como un conjunto de tareas independientes ejecutadas por procesos trabajadores. Cuando se tiene que devolver una terminal se mata al trabajador y su tareas se reasignan a otro trabajador.

5 – Caso Práctico - Proceso paralelo sobre clusters de máquinas Linux

5.1 Conceptos Generales

Las exigencias de cálculo de las aplicaciones comerciales y científicas crecen día a día. Para hacer frente a esta demanda en precio aumentar la capacidad de la unidad de proceso, o bien hacer que varias colaboren para la resolución conjunta de una tarea. La primera alternativa es fácil de llevar a la práctica, basta con aumentar la memoria o sustituir el procesador por otro de tecnología superior y / o de mayor velocidad de reloj. Pero esto a menudo es insuficiente; ¿qué CPU utilizaríamos si la necesidad de potencia de cálculo se multiplicase por cien?. La segunda opción, el procesamiento en paralelo, no es de implantación tan inmediata, pero escala mucho mejor con las necesidades del problema. El procesamiento en paralelo consiste en acelerar la ejecución de un programa mediante su descomposición en fragmentos que puedan ejecutarse en forma paralela, cada uno en su propia unidad de proceso. La mayoría de los procesadores modernos incorporan en su diseño unidades que trabajan de forma paralela, cada uno en su propia unidad de proceso. La mayoría de los procesadores modernos incorporan en su diseño unidades de que trabajan de forma paralela, como los que emplean la tecnología de *pipeline* (operar a la vez sobre las distintas fases que requiere la ejecución de una instrucción del procesador, como en una cadena de montaje), o el proceso superescalar (la instrucción del paralelo de instrucciones independientes, como operaciones sobre enteros, sobre números de coma flotante o accesos a memoria; el procesador Alpha 21264 es capaz de ejecutar hasta cuatro instrucciones por ciclo de reloj). Los denominados procesadores vectoriales que forman parte de los supercomputadores tradicionales, como el Cray C90, son capaces de operar a la vez sobre varios elementos de un vector. Más recientemente toma cuerpo el concepto de paralelismo sobre un registro, como el que implementan las extensiones multimedia de Intel. Aunque el procesamiento paralelo no se inventó ayer (*Parallel Computers* de Hockney Jesshope se publicó en 1891), sin embargo es todavía una técnica muy poco familiar para la mayoría de los usuarios de computadores. Primero se introducirá las distintas arquitecturas paralelas destacando su aplicación en el mundo de Linux y definiendo términos según van apareciendo. Luego se describirá cuándo y cómo utilizar un cluster de máquinas Linux.

5.2 TAXONOMIA DE LAS ARQUITECTURAS PARALELAS

Un Procesador serial es capaz de ejecutar cada vez una única instrucción sobre un único dato. Las arquitecturas paralelas operan simultáneamente sobre distintos datos pero cada CPU puede ejecutar la misma instrucción que las demás o bien instrucciones distintas. Así, se definen los dos tipos clásicos de procesamiento en paralelo: SIMD (*Single Instruction, Multiple Data*; instrucción única, datos múltiples) y MIMD (*Multiple Instruction, Multiple Data*; instrucciones múltiples, datos múltiples). En el modelo SIMD cada procesador ejecuta en todo momento la misma operación que los demás, pero sobre sus propios datos; los diversos procesadores están siempre sincronizados. La memoria no se comparte. Si un procesador precisa conocer el resultado de una operación efectuada algún otro, debe haber una comunicación explícita. Los procesadores de las máquinas SIMD suelen ser poco potentes pero están presentes en gran número, del orden de miles.

Cada procesador dispone de una pequeña cantidad de memoria local y canales de comunicación en una disposición tal que el número de procesadores conectados directamente sea el máximo, pero sin complicar excesivamente el cableado, es habitual usar la topología de hipercubo. El ejemplo más clásico de máquina SIMD es la Connection Machine, fabricada por Thinking Machines Corporation, constituida por 65536 procesadores (un hipercubo de dimensión 16) cada uno con 4 Kbytes de memoria. La inflexibilidad del modelo SIMD ha provocado que en los últimos años se encuentre relegado a entornos muy especiales o de investigación. Sin embargo, una máquina SIMD siempre requiere una estación de trabajo que actúe de controlador (*host*) , y aquí Linux tiene mucho que decir por su carácter abierto, la disponibilidad de controladores para todo tipo de *hardware* y su planificación casi en tiempo real. Se dice que la máquina de SIMD más grande del planeta opera cada mañana cuando empiezan los programas televisivos de *aeróbic*.

La *controladora* recita sus instrucciones: arriba, abajo, vuelta... mientras miles de esforzados espectadores le siguen como autómatas.

En el modelo MIMD cada procesador actúa de forma esencialmente independiente. Uno o varios procesadores pueden, por ejemplo, estar transformando datos mediante un algoritmo complejo, mientras otro genera una salida gráfica de los resultados. Esto es lo que se denomina descomposición de control, donde cada procesador puede estar ejecutando un programa distinto. Mucho más habitual es la descomposición de datos en la que los datos del problema se reparte entre los distintos procesadores y todos ejecutan el mismo programa sobre su fragmento. A esta versión restringida del modelo MIMD se la denomina SPMD (*Simple Program, Multiple Data*, programa única, múltiples datos). Nótese que contrariamente a lo que ocurre en el modelo SIMD, cada procesador que ejecuta en un programa SPMD puede fluir a través de ramas de código diferentes. Este modelo es, por tanto, mucho más flexible. Los procesadores de una arquitectura MIMD suelen ser potentes y con una memoria del orden de decenas o centenas de Mbytes. El modelo MIMD es equivalente a la división de un proyecto en grupos de trabajo.

Cada grupo se ocupa únicamente de su tarea, pero precisan reunirse de vez en cuando para intercambiar conclusiones y coordinar el trabajo posterior. En el ámbito que nos ocupa, los datos se intercambian entre procesadores bien mediante memoria compartida bien por medio del paso de mensajes. Los procesadores de los sistemas con memoria compartida físicamente la memoria, es decir, todos acceden al mismo espacio de direcciones. Un valor escrito en memoria por un procesador puede ser accedido directamente por cualquier otro. En estas arquitecturas la latencia (tiempo requerido para transmitir una unidad de información incluyendo los períodos de envío y recepción) es baja, mientras que el ancho de banda (máxima cantidad de información que se puede transmitir por unidad de tiempo toda vez que la transmisión ya ha comenzado) es alto ... siempre que varios procesadores no traten de acceder al bus de datos simultáneamente. Por este y otros problemas relacionados, el número de procesadores en este tipo de sistemas suele ser pequeño, raras veces superior a 16.

A la combinación de MIMD y memoria compartida se la denomina SMP (*symetric multi-processing*, multiprocesamiento simétrico). Linux soporta multiproceso simétrico para hardware compatible con la especificación Intel MPS (*multi-processor specification*, especificación multiprocesador) o máquina multiprocesador Sparc de la serie 4 m. Basta con recompilar el núcleo especificado SMP=1 en el *makefile* (aunque como dice el propio Hank Dietz, autor del documento *Linux Parallel Processing HOWTO*, hacer que SMP valga la unidad sea irónico.). El sistema así configurado será capaz de ejecutar procesos en las distintas CPU's sin intervención del usuario.

Si lo que se quiere es SPMD, hay que recurrir a la programación de hebras (*threads*) con alguno de los excelentes paquetes disponibles para Linux (por ejemplo, LinuxThreads), o bien utilizar algún otro esquema de programación de memoria compartida. Estas técnicas no se describen en este documento. Por otra parte, la comunicación por paso de mensajes tiene lugar en sistemas con memoria distribuida, sistemas en los que cada procesador dispone de su propia memoria independiente de la del resto. Como en la máquinas SIMD, para que un dato que reside en la memoria de un procesador pase a la del otro, el primero debe construir un mensaje por software, enviarlo de una red de interconexión y el segundo debe recibirlo; es un mecanismo más complejo que con memoria compartida. La latencia puede ser alta, pero un cuidadoso reparto de los datos sobre los procesadores puede disminuir la granularidad de las comunicaciones (minimizar el número de mensajes y maximizar su tamaño), contribuyendo a un uso eficiente del ancho de banda disponible.

Existen herramientas software que permiten programar una arquitectura de memoria compartida, sin preocuparte (mejor dicho, sin preocuparse *mucho*), de en qué procesador reside cada dato. A este modelo se le denomina de memoria virtual. Los sistemas con memoria distribuida pueden, a su vez, ser un único computador con múltiples CPU's comunicaciones por un bus de datos o bien múltiples computadores, cada uno con su procesador, enlazados por una red más o menos rápida. En el primer caso se habla de procesadores enormemente paralelos (MPP, *masive parallel processor*), como los FUJITSU, VPP, IBM SP2 o SGI T3E. Estas máquinas extraordinariamente costosas, que copan los primeros cientos de puestos en la lista de los 500 supercomputadores más rápidos del mundo, suelen utilizar sistemas operativos más o menos específicos. El único ejemplo disponible de una máquina MPP que utiliza Linux es la descrita en el proceso Linux/AP+. Se trata de un computador FUJITSU AP1000+ con 64 procesadores SuperSparc (¡aunque soporta hasta 1024!) y sistemas de red y disco específicos.

Más interés tienen los sistemas constituidos por varios computadores conectados entre sí, lo que de forma genérica se conoce como *cluster* (traducido como grupo en la mayoría de los diccionarios) de estaciones de trabajo (algunas veces se usa la palabra Farm, granja). Los Cluster ofrecen importantes ventajas sobre otros tipos de arquitecturas paralelos:

- Cada una de las máquinas de un cluster puede ser un sistema completo utilizable para otros propósitos. Se puede, por ejemplo, montar un aula de informática con algunas estaciones Linux que den servicio a los alumnos durante el día y disponer del conjunto durante la noche para

realizar cálculos complejos. Incluso es posible usar la parte de la CPU que los alumnos no precisen (los ciclos muertos) para continuar con los cálculos durante el día.

- Los elementos del proceso de un cluster son computadores normales y por tanto, baratos. El hardware de red, incluso para redes rápidas, es también cada vez más barato. Si se quiere economizar aún más es posible tener un único monitor, tarjeta de video y teclado para todo el cluster. Avalon, un cluster de 70 computadores Alpha bajo Linux, da aproximadamente la misma potencia de cálculo que un SGI Origin 2000, con 64 procesadores y de arquitectura MPP por un coste 10 veces inferior. Stone Super Computer es un cluster de 126 nodos que no a costado nada, ya que los computadores han sido donados. La mayoría son antiguos, Intel 486 con 16 Mbytes de memoria, pero la relación entre la potencia del cálculo y el precio es insuperable.
- Los cluster escalan bien hasta sistemas muy grandes. Mientras que es difícil encontrar máquinas Linux SMP con más de 4 procesadores, la mayor parte de los cluster cuentan con 16 computadores, y no es difícil construirlos con cientos y hasta miles de ellos.
- Reemplazar un computador defectuoso de cluster, es trivial si se compara con el trabajo necesario para separar un máquina MP. Lo que permite obtener sistemas de muy alta disponibilidad. Incluso es posible diseñar un cluster de tal forma que si un nodo falla el resto continua trabajando.
- Existe muchos soportes software para la programación de clusters. Además, con el nivel de estandarización actual, existe la garantía de que los programas escritos para un cluster funcionaran en cualquier otro con independencia del tipo de procesador en cada nodo.

Pero no todo el monte es orégano. También los clusters tienen su problema:

- Las redes ordinarias no están diseñadas para el procesamiento paralelo, la latencia es alta y el ancho de banda relativamente bajo se comparan con un sistemas SMP. Si el cluster no está aislado del resto de la red, la situación es aún peor.
- Existe muy poco soporte software para tratar un cluster, como un sistema único. Por ejemplo el comando ps solo lista los procesos de un sistema Linux, no los del todo el cluster. Pero esto está cambiando. Se haga hacho pública del paquete procps, versión 1.2.7 que soporte cluster de máquinas Linux.

A su vez, los clusters pueden ser de dos tipos dependiendo de si cada computador del cluster está o no exclusivamente dedicado a el. Si es así se habla de un cluster de clase Beowulf, y si no de una simple red de estaciones de trabajo (NOW, Network of Workstations)..Los beowulfs tiene algunas ventajas sobre las redes de estaciones:

- Al estar todas las CPU's al servicio del cluster es más favor mantener el balance de carga (conseguir que todos los procesadores tengan un grado de operación un grado de ocupación semejante, no que algunos estén saturados de trabajo, mientras otros no tienen nada que hacer). Además el único uso de la red es debido a la aplicación paralela que se está ejecutando, lo que permite sincronizar el trafico y disminuir así la latencia. Al ser un entorno más restrictivo, los programas diseñado para NOW's no tienen problemas para ejecutarse en BEOWULFS, pero lo contrario puede no ser cierto.
- Ni un beowulf es posible modificar el núcleo para manejar el procesamiento en paralelo. En Linux, es habitual utilizarse un módulo que permite la asignación a cada proceso de un identificador normal a todo el cluster.
- Un computador de una red ordinaria debe estar configurado para una respuesta interactiva adecuada al usuario, pero en un BEOWULF se pueden ajustar los parámetros del sistema de acuerdo a la granularidad de la aplicación paralela que se vaya a ejecutar, lo que contribuye a mejorar su rendimiento.

Beowulf era un guerrero escandinavo del ciclo VI cuyas aventuras se relatan en el primer texto conocido en lengua inglesa (la semejanza con nuestro cantar, del Mio CID son obvias). El término lo utilizó la NASA para dar nombre a un proyecto que pretendía construir un computador capaz de alcanzar el *gigaflops* (1000.000.000 de instrucciones sobre números de coma flotante por segundo), a partir de componentes asequibles. Y lo consiguieron. El Beowulf original de 1994 con 16 procesadores, 486 DX4 ejecutando Linux, fue capaz de alcanzar 1,25 Gigaflops. El éxito del proyecto BEOWULF, a originado un enorme interés en este tipo de arquitectura. La Web esta atestada de clusters Linux, con nombres que hacen referencia al proyecto original: Loki, Hyglac, Avalon, Grendel, Drexel, Hrothgar, Naegling... Este último ha mostrado que es capaz de dar 11 GIGAFLOPS, sostenidos con su Pentium Pro 140. Sin duda una de las claves de esta proliferación de Beowulf es el propio Linux. Y es que nuestro sistema operativo favorito es idóneo para esta tarea por una serie de razones:

- 1- Su carácter abierto,
- 2- Su diseño modular,

- 3- La existencia de controladores para todo tipo de dispositivos de mas y su extraordinario rendimiento.
- 4- Su disponibilidad en múltiples arquitecturas.
- 5- Su inmejorable soporte técnico la comunidad de usuarios Linux y el hecho de que se han portado todas las herramientas software relevante.

Además es muy barato, lo que contribuye a ajustar el precio del sistema completo.

5.3 – Como y cuando Utilizar un Cluster.

La razón de ser del procesamiento en paralelo es acelerar la resolución de un problema. La aceleración (*speedup*) que puede alcanzarse depende tanto del problema en sí como de la arquitectura del computador paralelo. Las aplicaciones que se benefician de una aceleración más significativa son aquellas que describen procesos intrínsecamente paralelos. Las simulaciones de modelos moleculares, climáticos o económicos tienen todas una amplia componente paralela, como los sistemas que representan (los átomos no establecen turnos a la hora de interactuar, todo lo hacen a la vez). El *hardware* de la máquina entra en juego ya que es preciso maximizar la relación entre el tiempo de cálculo útil y el perdido en el paso de mensajes, parámetros que dependen de la capacidad de proceso de las CPU's y de la velocidad de la red de comunicaciones, respectivamente.

La clave es descomponer el problema de tal forma que cada procesador pueda operar el mayor tiempo posible sobre su fragmento de los datos, sin tener que recurrir a los de los demás procesadores. Este es el problema de la localidad de los datos. En los siguientes párrafos nos concentraremos en la descomposición de datos. Si se está interesado en descomposición de control es más adecuado a utilizar un entorno de objetos distribuidos como COBRA. El cálculo del área bajo una curva por integración numérica es un ejemplo de aplicación completamente paralelizable. Basta con dividir el intervalo de integración entre todos los procesadores disponibles y que cada uno resuelva su fragmento sin preocuparse que hacen los demás. Al final, los resultados parciales se recolectan y se suman convenientemente. Con n procesadores es posible resolver el problema n veces más rápido que haciendo uso de uno solo (salvo por el mínimo retraso que supone el reparto de trabajo inicial y la recolección de datos final), consiguiendo una aceleración lineal con el número de procesadores.

Si las condiciones son muy favorables es incluso posible alcanzar la soñada aceleración *superlineal*, consistente en que el programa ejecuta más rápido que en régimen lineal. La aparente paradoja se entiende recordando que cada procesador cuenta con su propia memoria ordinaria y caché, que pueden ser usadas de forma más eficiente con un subconjunto de los datos. De hecho, es posible que el problema no se pueda resolver en un único procesador pero sí sobre un *cluster*, simplemente por cuestión de tamaño de los datos. El extremo opuesto encontramos los problemas que paralelizan muy mal, que necesitan estar continuamente compartiendo datos entre los procesadores. Las comunicaciones determinan el avance del programa y es posible encontrar que ¡su ejecución en un *cluster* es más lenta que en un único computador!. Los problemas que se ejecutan sobre las arquitecturas paralelas consisten, a menudo en un algoritmo básico que se repite muchas veces y donde entre paso y paso los procesadores intercambian información con sus vecinos. Por ejemplo, en el clásico juego de la vida en dos dimensiones cada celda sobrevive en la próxima generación si en la actual tiene exactamente dos vecinos vivos.

Si se programa el juego en un *cluster* y se decide que cada procesador se encargue de una única celda, cada computador en cada paso precisa conocer el estado de sus ocho celdas vecinas, lo que supone un intercambio de mensajes con los ocho computadores correspondientes. Otros problemas requieren que cada procesador conozca, tras cada paso el estado del sistema completo, con lo que todos los datos deben propagarse a todos los computadores. Es el caso de las simulaciones que involucran interacciones de largo alcance, como la gravitatoria (colisión entre dos galaxias) o la coulombica (medios cargados, como una disolución de electrolitos). Todas estas aplicaciones ofrecen un grado de paralelismo intermedio entre el régimen lineal y el de deceleración. La aceleración es prácticamente lineal cuando el número de computadores en el *cluster* es pequeño, pero al ir añadiendo nodos la distribución de los datos requiere más tiempo y la ejecución no se acelera tanto. A veces es preciso utilizar ocho nodos, como por ejemplo, para hacer que un programa se ejecute cinco veces más rápido; esta es una cifra habitual en *clusters* reales. Por ello, merece la pena utilizar un *cluster* si la aplicación es suficientemente paralela y o bien ha sido ya paralelizada (reescrita para hacer uso del procesamiento en paralelo) o se está dispuesto a hacerlo uno mismo. Cada vez es posible encontrar más paquetes que han sido adaptados para su ejecución en un *cluster*. Red Hat y NASA han organizado una distribución de Linux orientada a la operación de *cluster* llamada Extreme Linux que contiene un conjunto de tales aplicaciones paralelizadas, como el generador de imágenes Persistence Of Vision (POV). Pero a menudo, uno está interesado en adaptar su

propia aplicación de cálculo intensivo para sacar provecho de un *cluster*, y para ello no hay más remedio que recurrir a su paralelización. Existen dos opciones para hacer que la aplicación sea capaz de usar los computadores de un *cluster*, programar el paso de mensajes o utilizar herramientas que lo hagan por uno. El segundo caso abarca las herramientas de paralelización automática y los lenguajes de programación paralelos. Desde luego, lo más sencillo es dar el código fuente aun programa paralelizador y dejar que éste produzca código paralelo.

El problema es que estos paralelizadores suelen ser muy caros y, lo que es peor, funcionan realmente mal. Aunque en estos momentos muchos grupos de investigación están dedicando un gran esfuerzo al desarrollo y mejora de herramientas de paralelización automática, lo cierto es que por el momento no son muy útiles. En los años 80 se diseñaron un sinnúmero de lenguajes explícitamente paralelos, pero últimamente los desarrollos se han dado cuenta de que los usuarios no van a usar su nuevo lenguaje, pese a lo bueno que sea, si no es compatible con Fortran o C. Por ello, últimamente los esfuerzos se dirigen al desarrollo de extensiones paralelas a estos lenguajes. El ejemplo más notable es HPF (*High Performance Fortran*, Fortran de alto rendimiento). HPF es fundamentalmente Fortran 90 estándar al que se han añadido una serie de directivas (que se introducen como comentarios en el código fuente), para informar al compilador, por ejemplo, cómo distribuir los datos entre los procesadores o qué bucles puede ejecutar en paralelo son preocuparse por las dependencias de las expresiones. La última revisión del lenguaje Fortran, llamada Fortran 95 (a pesar de que el estándar es de diciembre de 1997), incorpora la mayoría de las características de HPF para la programación paralela. Pero habitualmente se obtiene mayor rendimiento programado explícitamente el paso de mensajes entre procesadores. Uno puede arañar hasta el último microsegundo utilizando la red a bajo nivel, usando *sockets* o el dispositivo de red directamente, pero al precio de una programación compleja, susceptible a errores y, lo que es peor, no portable.

El mejor compromiso entre eficiencia, facilidad de uso y portabilidad lo dan las librerías de paso de mensajes, en las que le programador construye explícitamente el mensaje entre dos procesadores, o las de función colectiva, modelo intrínsecamente paralelo en el que un grupo de procesadores se comunica a la vez. Entre las primeras destacan PVM (***Parallel Virtual Machine***, máquina paralela virtual) y MPI (***Message Passing Interface***, interfase para el paso de emnsajes) y el ejemplo más significativo de las segundas es AFAPI (***aggregate Function API***, API de función colectiva). Todas estas librerías pueden ser utilizadas en cualquier arquitectura paralela a la que se hallan portado, no solamente en *clusters* de computadores. Todas están disponibles para Linux. PVM y MPI son las librerías de programación por excelencia, cada una con sus ventajas y sus inconvenientes. Por su parte, AFAPI es el corazón del proyecto PAPERS (*Purdue's Adapter for Parallel Execution and Rapid Synchronization*, adaptador para ejecución paralela y sincronización rápida de la Universidad de Purdue). El objetivo de AFAPI es suministrar comunicaciones con muy alta latencia. Al ser una librería de función colectiva tiene un API aún más sencillo que MPI. Existen versiones diseñadas para que las comunicaciones tengan lugar por el puerto paralelo de los computadores.

CONCLUSIONES SOBRE LINUX

La idoneidad de Linux para la operación de clusters junto a la potencia y bajo costo de las PC's de última generación ha hecho que el procesamiento paralelo en clusters de máquinas Linux sea una posibilidad muy atractiva, no solo para grupos pequeños con bajo presupuesto sino incluso para grandes proyectos. En la programación para procesamiento paralelo hay que preocuparse de asuntos como la granularidad, el balance de carga o la localidad de los datos, lo que la hace más compleja que la programación tradicional. Por otra parte, las aplicaciones tienen un grado de paralelismo intrínseco que las hace más o menos susceptibles de ser aceleradas por un entorno paralelo. Pero salvados este par de escollos, el potencial de los clusters de máquinas de Linux es enorme. Y montar uno no es ya tarea de expertos. Algunas compañías están empezando a distribuir clusters preconfigurados con Linux como sistema operativo.

La distribución Extreme Linux está diseñada específicamente para la operación de clusters. Algunos fabricantes de compiladores como The Portland y Kuck and Associates han adquirido el compromiso de desarrollar herramientas para Extreme de Linux. Además, la experiencia acumulada sobre clusters de máquina de linux es muy considerada, por lo que siempre se encontrará a alguien que nos de una mano. Con Linux y su presupuesto ajustado hoy es más fácil que nunca disponer de un supercomputador. Pero un supercomputador de verdad. Avalon ha alcanzado a partir del año 1999, el puesto 315 entre los computadores más potentes del mundo.

6 – Conclusión

Hoy en día las aplicaciones requieren más poder de procesamiento. Para esto existen dos posibilidades: Adquirir enormes computadores u optar por armar un cluster. Esta tecnología distribuyen los requerimientos de procesamiento entre todas la máquinas disponibles mejorando el tiempo respuesta de todo el sistema.

Como puntos fundamental se tiene un costo bajo de armado, permite poder utilizar PCs de muy buen rendimiento y bajo precio para su estructura.

Por las tendencias actuales de tamaños de los programas y de los recursos de sistema que consumen es muy fácil de decir que el cluster será una de las herramientas fundamentales de los centros de cómputos del nuevo milenio.

Además a medida que disminuyan los costos de los componentes de las redes de alta velocidad se podrá lograr mejores resultados a precios más bajos.

AUTOEVALUACIÓN DEL MODULO 10:

Preguntas:

Múltiple Choice:

1.-

2.-

3.-

4.-

5.-

6.-

7.-

8.-

9.-

10.-

11.-

12.-

13.-

14.-

15.-

16.-

17.-

18.-

19.-

20.-

Respuestas a las preguntas

1. ¿

Respuestas del múltiple choice.