o

CAPÍTULO 1
Introducción a los Sistemas Distribuidos

Patricia Bazán

Los sistemas distribuidos pueden concebirse como aquellos cuya funcionalidad se encuentra fraccionada en componentes que al trabajar sincronizada y coordinadamente otorgan la visión de un sistema único, siendo la distribución, transparente para quien hace uso del sistema.

En términos computacionales, se dice que un sistema distribuido es aquel cuyos componentes, de hardware o de software, se alojan en nodos de una red comunicando y coordinando sus acciones a través del envío de mensajes.

El concepto de componente como pieza funcional autónoma y con interfaces bien definidas alcanza al área del desarrollo de software pero también puede concebirse dentro de otras disciplinas como la idea de una pieza que, cuando se combina con los demás componentes, forma parte de un todo.

En términos estrictamente informáticos, un componente es cada parte modular de un sistema de software.

Entre las motivaciones para construir sistemas distribuidos, sin lugar a dudas está el hecho de compartir recursos, pero también favorece el diseño de software modular donde cada componente se ejecuta en la plataforma más adecuada y brinda un servicio más eficiente debido a la especificidad de su construcción. Por ejemplo, concebir un sistema de software donde uno de los componentes es la gestión y manipulación de los objetos de información (datos), le da un papel importante a los Sistemas de Gestión de Bases de Datos - en inglés DataBase Management Systems o DBMS - que logran dar un servicio de acceso a los datos eficiente, correcto y consistente.

En este capítulo presentamos un conjunto de definiciones básicas que ilustran la idea de sistemas distribuidos, sus complejidades y desafíos, así como la manera que se administran y comunican.

El capítulo se organiza de la siguiente manera: en la Sección 1 se presenta la motivación y las principales definiciones de sistemas distribuidos. La Sección 2 muestra ejemplos de sistemas distribuidos de la vida real y desde un enfoque funcional. La Sección 3 analiza los desafíos a los que se enfrentan los sistemas distribuidos y los objetivos que persiguen. Finalmente, en las Secciones 4 y 5 se detallan los modelos arquitectónicos y los mecanismos de comunicación existentes en los sistemas distribuidos con un enfoque conceptual. En la Sección 6 se arriban a algunas conclusiones.

1.1. Motivaciones y Definiciones

Como hemos mencionado, la principal motivación para construir y utilizar sistemas distribuidos se sustenta en la idea de compartir recursos. La idea de recursos abarca desde un elemento de hardware - impresoras, unidades de disco, memoria -, hasta entidades de software - archivos, bases de datos, servicios, objetos, elementos multimedia -.

En este sentido una definición de sistema distribuido aportada por Colouris en [Colouris, 2000] es:

Un sistema en el cual las componentes de hardware y software se ubican en una red de computadoras y comunican y coordinan sus acciones solo por envío mensajes.

Otra definición que se puede formular es:

Un sistema que consiste de una colección de dos o más computadoras independientes que coordinan su procesamiento a través del intercambio sincrónico o asincrónico de mensajes.

Por su parte, Tanenmbaum en (Tanenbaum, 2007) afirma:

Un sistema distribuido es una colección de computadoras independientes que se muestran al usuario del sistema como un sistema único.

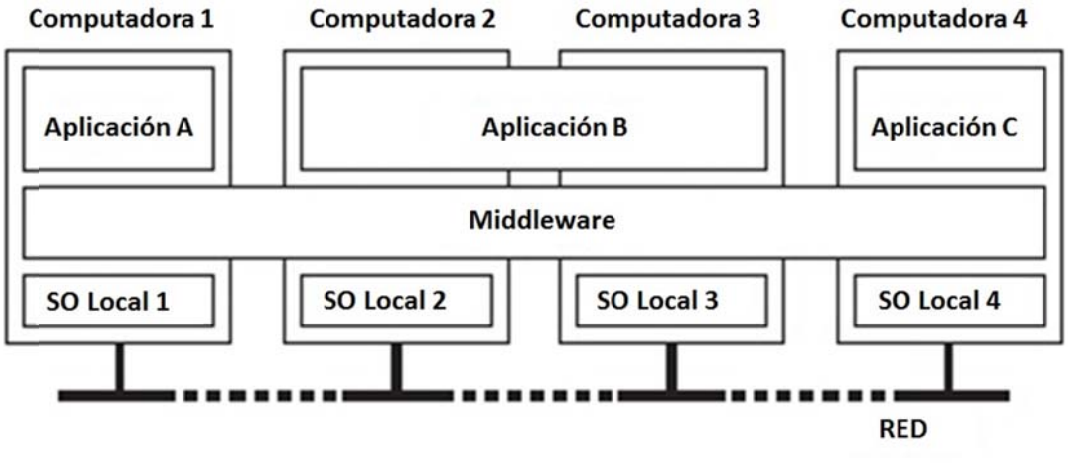

Fig. 1. Componentes de un Sistema Distribuido

En la Figura 1 muestra que no necesariamente debe existir una correspondencia unívoca entre computadoras y aplicaciones. También se observa que la visión del sistema único es lograda por la capa middleware que oculta los detalles del entorno de ejecución local de cada computadora.

Finalmente, también se puede asegurar que:

> Un sistema distribuido es una colección de computadoras autónomas enlazadas a través de una red con software diseñado para producir facilidades de cómputo integradas.

A la luz de estas definiciones se advierte la coexistencia de dos conceptos complementarios: sistema distribuido y red de computadoras. Se puede decir que mientras una red de computadoras son un conjunto de computadoras físicamente visibles y que debe ser explícitamente direccionada, un sistema distribuido es aquel donde las múltiples computadoras autónomas son transparentes pero se basa en ellas.

Sin embargo existen problemáticas comunes a ambos conceptos: las redes son a su vez sistemas distribuidos si pensamos en su servicio de nombres y además todo sistema distribuido se basa sobre los servicios provistos por las redes de computadoras.

Otro elemento conceptual es la noción de middleware software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, o paquetes de programas, redes, hardware y/o sistemas operativos. Se profundizará sobre este concepto en próximos capítulos.

Se revisan a continuación las principales razones para distribuir sistemas:

Distribución funcional

Este tipo de distribución radica en que cada computadora posee capacidades funcionales diferentes pero coordinan sus acciones con un objetivo común. Por ejemplo, cliente/servidor, existe un componente que brinda servicios y otro que necesita consumirlos. Otro caso es la relación host/terminal, un equipo posee toda la funcionalidad (host) y existe otro que se limita a exponer dicha funcionalidad (terminal). Asimismo la concurrencia de datos/procesamiento de datos, es otro caso de sistema distribuido donde el recurso a compartir es el dato y las funcionalidades están vinculadas a su manipulación y procesamiento como es el caso de los sistemas de gestión de bases de datos - en inglés DataBase Management Systems o DBMS.

Distribución inherente al dominio de aplicación

Este tipo de distribución se refiere a la que se realiza dentro de un mismo dominio de aplicación y cuyas componentes se identifican y modelan dentro de este dominio. Ejemplos de tal tipo de distribución lo constituyen un sistema de cajas de un supermercado y el sistema de inventario respectivo o el trabajo colaborativo soportado por computadoras.

Balanceo de carga y distribución

Este tipo de distribución se realiza para poder asignar tareas a distintos procesadores a fin de mejorar el rendimiento general del sistema.

Replicación del poder de cómputo. Varios computadores sumados pueden alcanzar una velocidad de cómputo que nunca se lograría con un super computador.

Razones económicas. Un conjunto de microprocesadores ofrecen una mejor relación precio/rendimiento que un gran mainframe[1] que es 10 veces más rápido pero 1000 veces más caro.

Entre los interrogantes que se plantean podemos encontrar:

¿Por qué usar sistemas distribuidos y no hardware o software aislado? para compartir recursos, para incrementar la comunicación entre personas, para potenciar la capacidad de cada componente (de hardware y de software) y otorgar flexibilidad al sistema

¿Qué problemas aparecen en los sistemas distribuidos y conectados? el diseño de software se complejiza porque intervienen más componentes, pueden surgir dependencias de la infraestructura de ejecución subyacente, la facilidad de acceso a datos compartidos puede afectar la seguridad.

Consecuencias

Los sistemas distribuidos son concurrentes. Esto implica que cada componente es autónomo (lo que implica diferenciar proceso de programa) y ejecuta tareas concurrentes. De este modo debe haber una sincronización y coordinación de dichas tareas y a su vez, aparecen los problemas de la concurrencia como deadlocks[2] y comunicación no confiable.

Ausencia de reloj local. Debida a la comunicación por medio de mensajes, se dificulta la precisión con la que cada componente debe sincronizar su reloj, apareciendo el concepto de latencia[3].

Ausencia de contexto global: Debido a la concurrencia y al mecanismo de comunicación por mensajes, no existe un proceso único que conozca o albergue el estado global del sistema. Por este motivo se dice que los sistemas distribuidos son sistemas “sin estado”.

Nuevos modos de falla. Los procesos son autónomos en su ejecución, de modo que las fallas individuales pueden no ser detectadas y a su vez desconocen el impacto en el sistema completo.

1 Una computadora central (en inglés mainframe) es una computadora grande, potente y costosa, usada principalmente por una gran compañía para el procesamiento de una gran cantidad de dato

2 El bloqueo mutuo (también conocido como interbloqueo, traba mortal, deadlock, abrazo mortal) es el bloqueo permanente de un conjunto de procesos o hilos de ejecución en un sistema concurrente que compiten por recursos del sistema

3 Latencia a la suma de retardos temporales dentro de una red. Un retardo es producido por la demora en la propagación y transmisión de paquetes dentro de la red.

1.2. Ejemplos de Sistemas Distribuidos

En esta sección se describen ejemplos típicos de sistemas distribuidos. En cada uno de ellos se evidencian los distintos componentes y se pone de manifiesto lo enunciado anteriormente respecto a que el sistema funciona como un todo donde cada componente se comunica con otro para poder dar respuesta al sistema de carácter global.

Ejemplo 1. La Internet

La Internet es una red que aglutina computadoras y aplicaciones heterogéneas implementadas bajo el protocolo de pilas análogo al modelo OSI para redes de computadoras.

Este protocolo se basa en una comunicación horizontal capa a capa donde las capas de nivel superior invocan servicios de niveles inferiores mediante interfaces de servicio. Esta visión por capas como proveedoras y consumidoras de servicios es un método de abstracción para aislar a los niveles superiores de detalles de la transmisión física. En la Figura 2 se puede observar el protocolo de pilas usado por Internet donde se visualiza cómo los niveles superiores tienen una visión más abstracta respecto de lo que sucede a nivel de comunicación en la capa enlace.

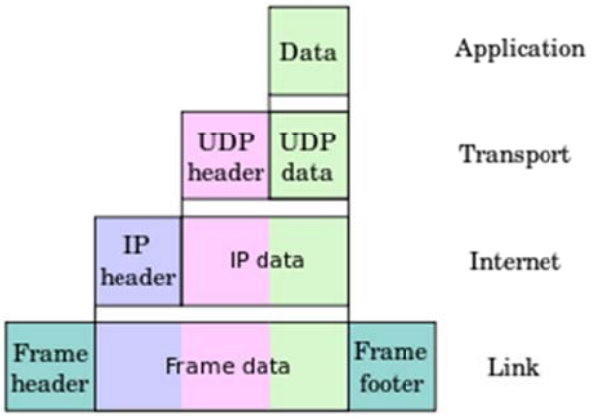

Fig. 2. Encapsulamiento de datos en la capa de aplicación descendiendo a través de los niveles

En la Figura 3 se observa cómo se lleva a cabo la comunicación entre capas (protocolos) y cuál es el flujo de datos a través de cada capa.

La capa de aplicación

Es el ámbito donde las aplicaciones crean datos de usuario y se comunican con otras aplicaciones del mismo equipo o de otro. Las aplicaciones, o procesos, hacen uso de los servicios provistos por los niveles inferiores, especialmente con la capa de transporte que provee un medio confiable o no, con otros procesos.

La capa de transporte

Es la capa que realiza la comunicación computadora-computadora sobre el mismo o diferente equipo de la red local o de redes remotas separadas por routers. Asegura la comunicación punta a punta mediante un servicio de datagramas

La capa de red

Es la capa que intercambia datagramas a través de la red y provee una interfaz uniforme que oculta la topología subyacente de conexión. El la capa que realiza lo que se denomina "internetworking" (interconexión) que define Internet. Este nivel define las direcciones y las estructuras de ruteo usadas por el protocolo TCP/IP.

La capa de enlace

Es la capa que define los métodos de conexión con alcance en la red local en la que las computadoras se comunican sin ruteo. Esta capa incluye los protocolos usados para describir una topología de red local y las interfaces necesarias para efectivizar la trasmisión de los datagramas de la capa de red, hacia sus vecinos.

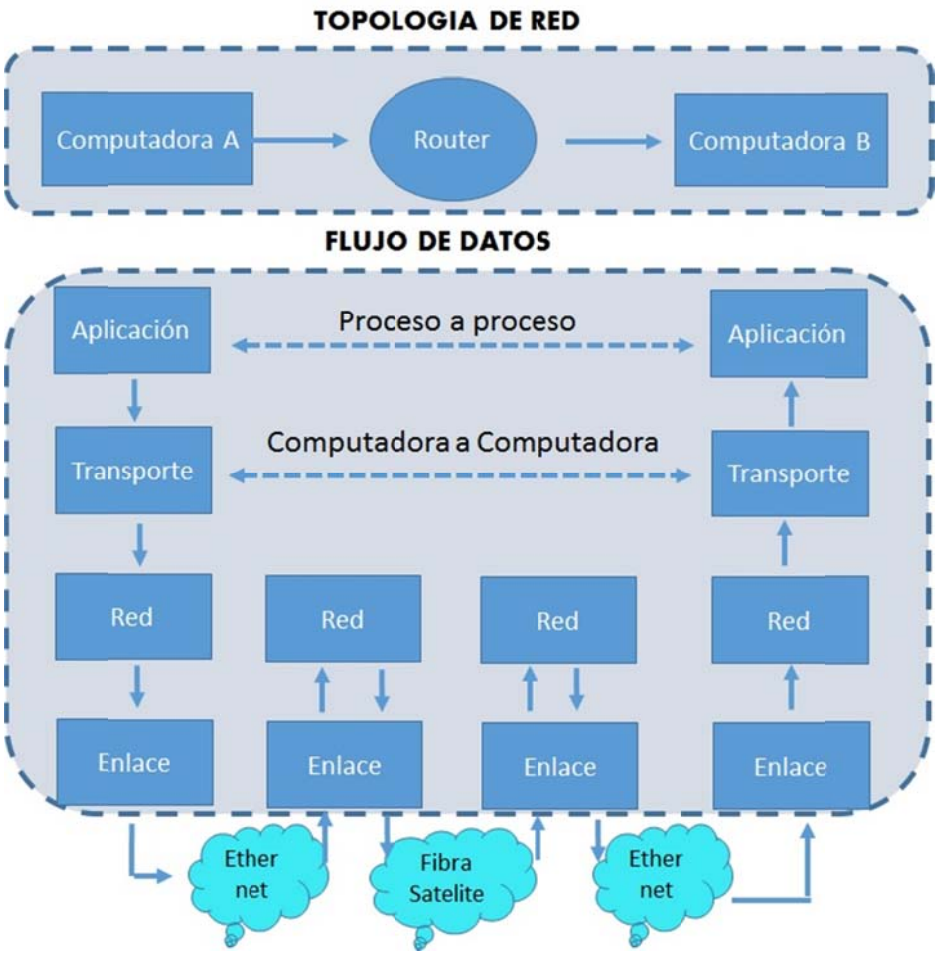

Fig. 3. Topología de red y flujo de datos entre capas

La Figura 4 muestra una configuración típica de Internet como ejemplo de sistema distribuido.

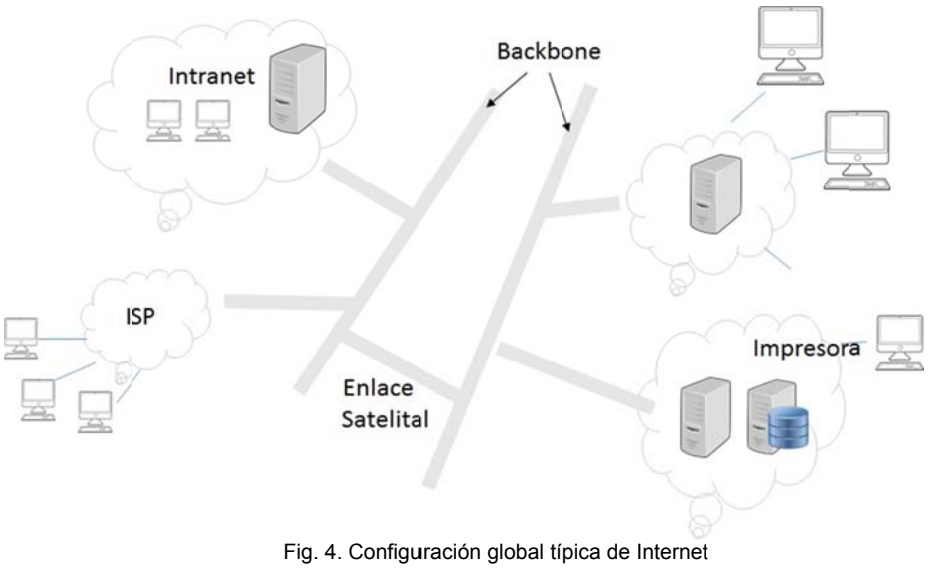

Fig. 4. Configuración global típica de Internet

Ejemplo 2. Intranets

Las intranets son redes administradas localmente, usualmente propietarias (la red del campus de una universidad, la red de una organización o empresa) y se comunican a Internet ubicando firewalls[4].

La Figura 5, muestra una configuración posible de una intranet.

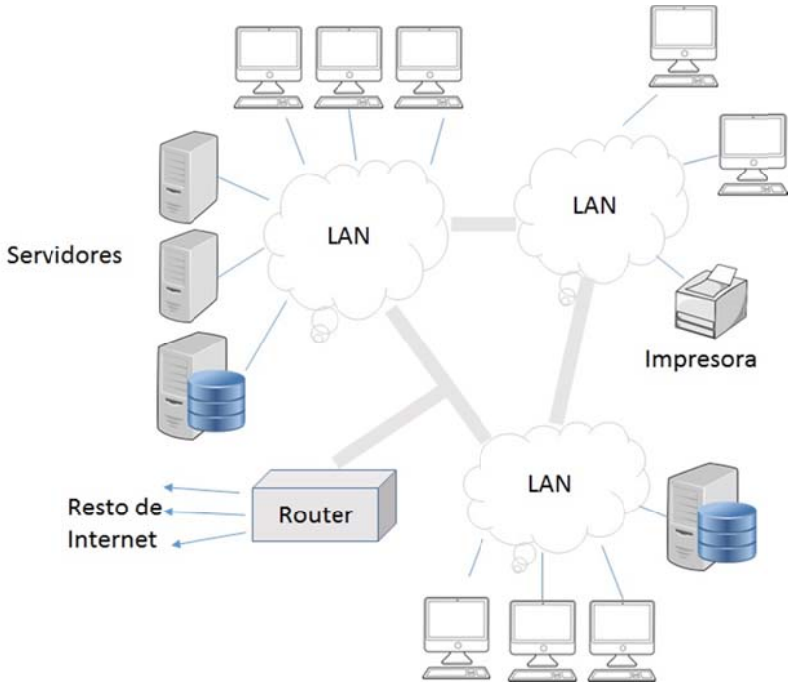

Fig. 5. Configuración de una intranet

4 Un cortafuegos (firewall) es una parte de un sistema o una red que está diseñada para bloquear el acceso no autorizado, permitiendo al mismo tiempo comunicaciones autorizadas

Ejemplo 3. Sistemas de cómputo móviles y ubicuos

Este tipo de sistemas distribuidos se caracterizan porque sus componentes son móviles -dispositivos pequeños que pueden portarse y ser utilizados durante su transporte -, y ubicuos -dispositivos que admiten estar conectados todo el tiempo sin importar el lugar.

En la Figura 6 se observa una configuración posible para un sistema distribuido móvil y ubicuo donde los dispositivos pueden ser cámaras fotográficas, impresoras, teléfonos celulares, laptop e incluso PDAs y tabletas.

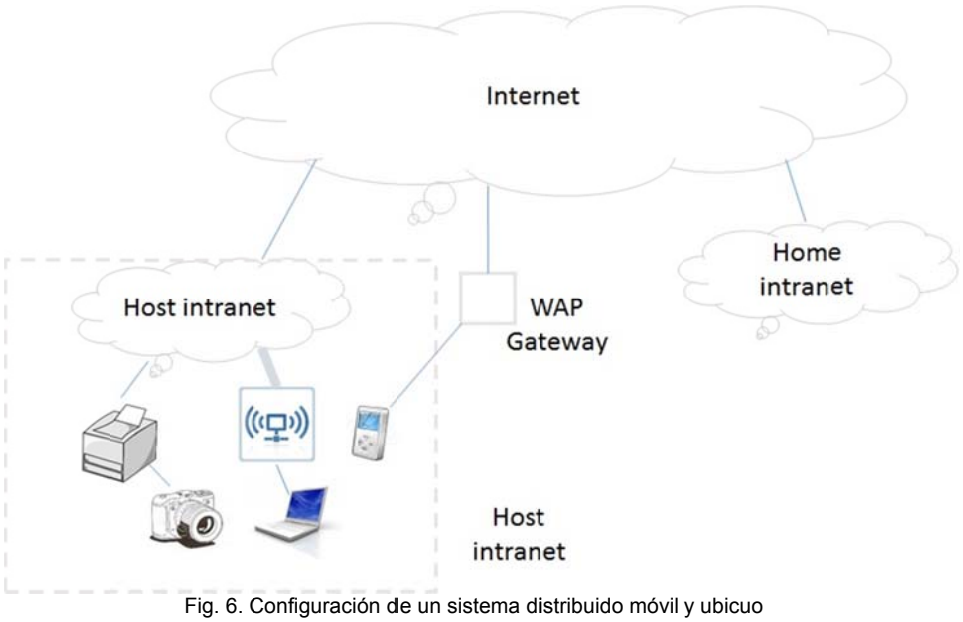

Fig. 6. Configuración de un sistema distribuido móvil y ubicuo

1.3. Desafíos y Objetivos

En esta sección se plantean los principales desafíos que enfrenta un sistema distribuido, así como los objetivos que persigue. Estos últimos son la apertura, escalabilidad, accesibilidad y transparencia, mientras que los desafíos se puede enumerar como la concurrencia, la seguridad y la heterogeneidad, algunos de los cuales son consecuencia de los objetivos.

Apertura

Un sistema distribuido es abierto cuando ofrece extensibilidad y mantenibilidad mediante la adhesión a estándares que describen la sintaxis y semántica de los servicios que ofrece.

Escalabilidad

Un sistema distribuido es escalable cuando ofrece capacidad de crecimiento en términos de cantidad y tipo de recursos que administra y cantidad usuarios que lo acceden.

Accesibilidad

Un sistema distribuido es accesible cuando garantiza el acceso a los recursos distribuidos que lo componen a todos los usuarios y en todo momento.

Transparencia

Un sistema distribuido es transparente en la medida que oculta su heterogeneidad y su distribución a los usuarios, quienes conservan la visión de un sistema único.

Como consecuencia de perseguir estos objetivos, los sistemas distribuidos se enfrentan a los siguientes desafíos:

Concurrencia

Los recursos de un sistema distribuido se caracterizan por estar expuestos al acceso concurrente de uno o más usuarios, debiendo ser capaz de planificar los accesos evitando deadlocks.

Seguridad

Además de garantizar la seguridad a nivel de transporte de datos, debido al alto grado de promiscuidad de la red como medio de comunicación, un sistema distribuido debe abordar la seguridad en términos de autenticación de usuarios, permisos de acceso sobre los recursos y auditoría del uso de los mismos.

Heterogeneidad

Un sistema distribuido es heterogéneo en cuanto es capaz de permitir que los usuarios accedan a servicios y ejecuten aplicaciones bajo cualquier plataforma (esto es diferentes redes, hardware, sistemas operativos y lenguajes de programación)

1.4. Modelos de Distribución

Los modelos de distribución pueden responder a aspectos arquitectónicos como a aspectos fundamentales o semánticos [Colouris, 2000].

El modelo arquitectónico de un sistema distribuido determina su estructura en función de cada una de sus componentes por separado.

El modelo fundamental o semántico revela los problemas claves a los que se enfrenta un desarrollador en un sistema distribuido y que condiciona la metodología de trabajo para alcanzar sistemas confiables en términos de seguridad, correctitud y fiabilidad.

Arquitecturas y Modelo Arquitectónico

La arquitectura de un sistema distribuido define la estructura completa de un sistema según sus componentes específicos, determinado el rol y funciones que cumple cada componente, la ubicación dentro de la red y la interrelación entre las componentes lo que determina su rol o interfaz de comunicación.

En cuanto a las funciones de las componentes, se definen como procesos servidores (atienden requerimientos solicitados por otros procesos), procesos clientes (inician la comunicación, solicitan el requerimiento y esperan la respuesta) y procesos pares (procesos que cooperan y comunican simétricamente para realizar una tarea).

Esta visión de componentes amerita considerar una estructura de software expresada por niveles de servicio y que facilitan la interacción en una misma computadora o en varias de ellas.

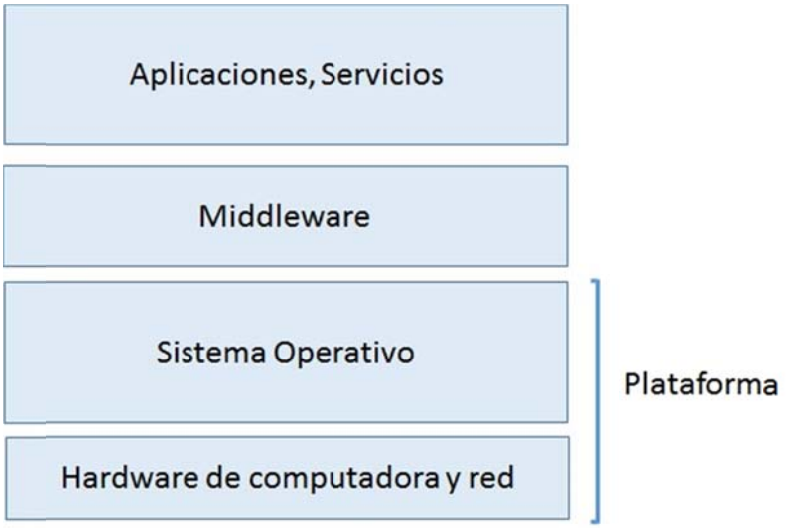

Fig. 7. Niveles de Servicio en una Arquitectura Distribuida

La plataforma está conformada por el software y hardware de bajo nivel que establece el entorno de ejecución de cada componente, sin dicho entorno, no es viable ejecutar aplicaciones siendo esto válido tanto para componentes distribuidas como para componentes que se ejecutan en un sistema único.

El middleware es el nivel de software que otorga abstracción al programador y a las aplicaciones así como también enmascara la heterogeneidad de la plataforma subyacente. El lenguaje RPC, las invocaciones basadas en SQL y los protocolos para invocación de Web Services, constituyen ejemplos de middleware que retomaremos en próximos capítulos.

El modelo arquitectónico en un sistema distribuido, simplifica y abstrae las funciones de cada componente dándoles una ubicación y una interrelación entre ellas.

Modelo Cliente/Servidor

El modelo cliente/servidor cuenta con procesos que ofrecen servicios (servidores) y procesos que utilizan o consumen dichos servicios (clientes). Se valen de un mecanismo de comunicación basado en pregunta-respuesta, siendo siempre el cliente quien inicia dicha comunicación.

Los servicios pueden ser implementados por varios procesos en diferentes computadoras que los alojan (host). Esto significa que tanto servidores como clientes pueden residir en diferentes máquinas, siendo el mecanismo de comunicación lo que pone en evidencia la distribución.

Este mecanismo de comunicación conforma una tercer componente o middleware, que establece la manera en que los procesos (tanto clientes como servidores) van a interactuar, determinado si la comunicación será sincrónica o asincrónica. Se profundizará sobre este concepto en el capítulo 4.

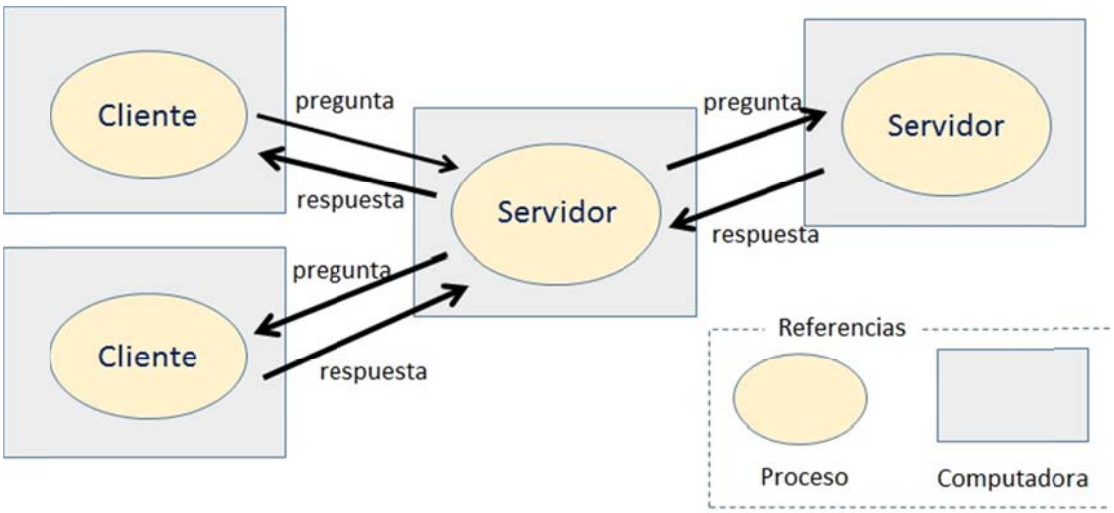

Fig. 8. Modelo Cliente/Servidor

Modelo peer-to-peer

En este modelo todos los procesos juegan roles similares considerándose pares (o peers) sin distinción entre clientes y servidores. El modelo adhiere claramente a un esquema descentralizado pero el mecanismo de comunicación no establece jerarquía, por lo que el código del proceso en cada nodo debe mantener la consistencia de los recursos y sincronizar las acciones de la aplicación.

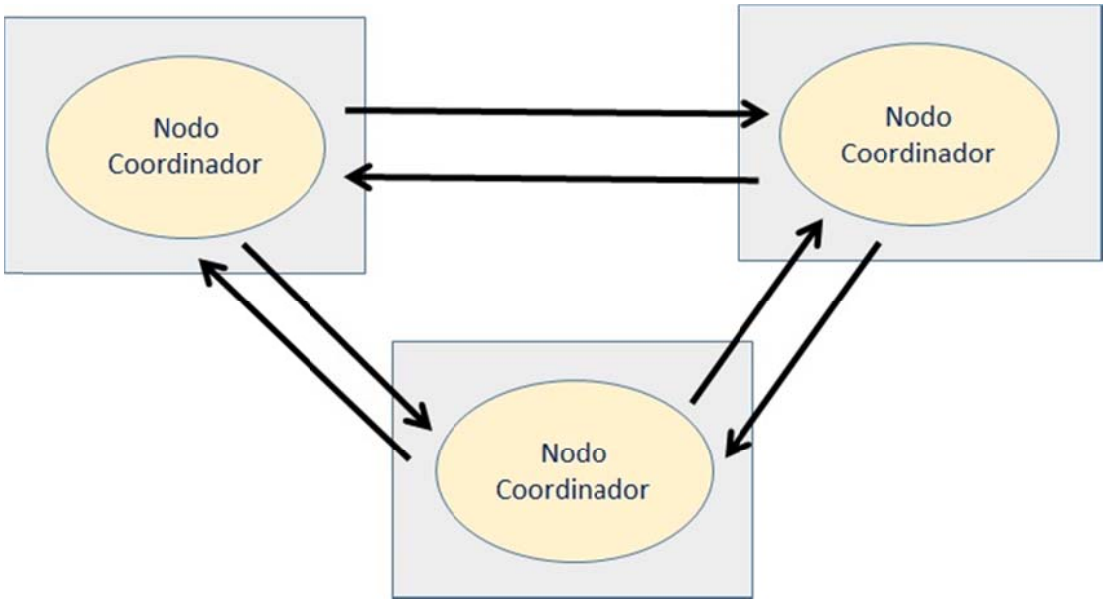

Fig. 9. Modelo Peer To Peer

Arquitecturas de software

Los diferentes modelos arquitectónicos se fundamentan en la arquitectura de software subyacente que determina la organización lógica (no física) de los componentes entre varias máquinas.

En la Figura 10 se muestran una organización en componentes lógicas diferentes, y distribución de las componentes entre varias máquinas. Estas arquitecturas pensadas como:

a) Modelo en capas usado por sistemas cliente-servidor

b) Modelo basado en objetos usado en sistemas de objetos distribuidos

Establecen un mecanismo de comunicación sincrónico siendo el sistema distribuido altamente acoplado en el tiempo.

Esto significa que las componentes del sistema distribuido deben estar presentes y activas para que el sistema pueda funcionar.

En el caso del modelo de capas determina un flujo de comunicación dentro de cada componente que deberá recorrer el camino inverso cuando llegue a destino y además el destino debe estar disponible.

El modelo de objetos distribuidos, por su parte, sigue el paradigma de la orientación a objetos donde cada objeto posee un estado interno y un comportamiento que será puesto en ejecución ante la llegada de un mensaje que lo requiera. En este caso no hay niveles o capas pero sí se requiere la disponibilidad del objeto que recibe el mensaje y de la computadora que lo contiene.

Por otra parte, la Figura 11 presenta dos modelos desacoplados en espacio y tiempo como lo son:

a) Publicar/Suscribir

b) Espacio de datos compartido

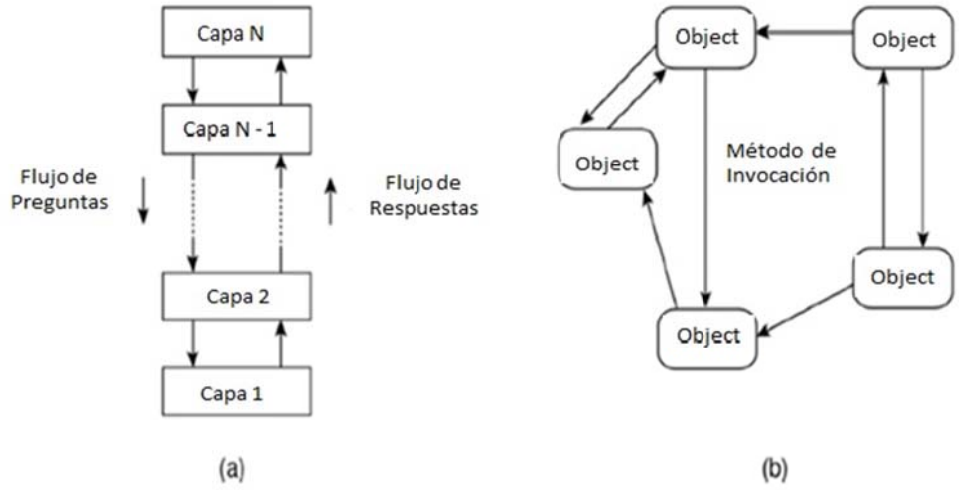

Fig. 10. Modelos en Capas usados por sistemas Cliente/Servidor

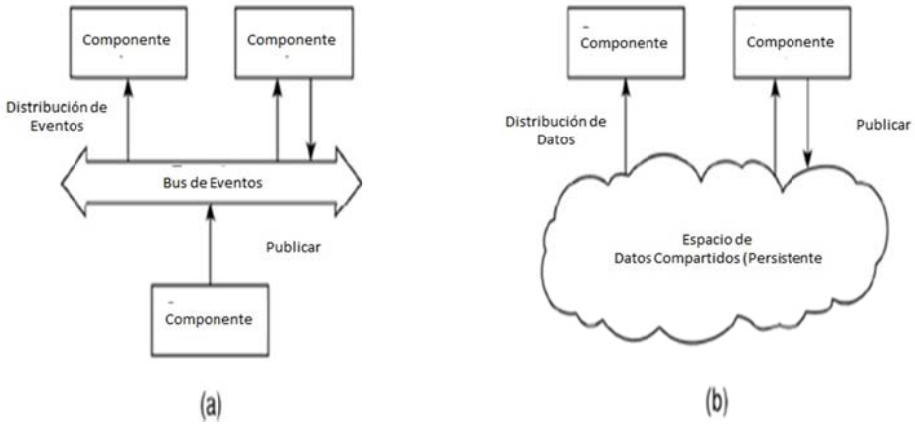

Fig. 11. Modelos de Objetos usados en Sistemas de Objetos Distribuidos

El modelo de publicación/suscripción no requiere sincronismo (coincidencia en el tiempo) debido a que las componentes publican sus funcionalidades en un *bus* de eventos que se ocupa de dar respuesta a las componentes que se suscriben a cada una de las funcionalidades publicadas. Tampoco se requiere acoplamiento en espacio, dado que las componentes como el bus de eventos mismo pueden estar distribuidos en la red.

Por su parte, el modelo de espacio de datos compartido permite que la comunicación entre las componentes se encuentre absolutamente desacoplada en el tiempo y favorece el acceso a los datos usando una descripción más que una referencia explícita.

1.5. Mecanismos de Comunicación

Los mecanismos de comunicación en los sistemas distribuidos son aquellos que definen la manera en que intercambian información los componentes del sistema de manera tal que el funcionamiento global se perciba como un sistema único. Estos mecanismos de comunicación son un componente más del sistema y se lo denomina middleware.

Tal como sucede con los otros componentes del sistema distribuido, el middleware cubre las necesidades de comunicación tanto a bajo nivel (comunicación a nivel física y de red) como a alto nivel (comunicación entre aplicaciones o procesos distribuidos).

El middleware a nivel del sistema operativo de red define el mecanismo de comunicación entre componentes y será desarrollado en el Capítulo 3.

Los mecanismos de comunicación entre procesos distribuidos son aquellos que se definen para dar respuestas a las siguientes preguntas:

¿Cómo conversan los componentes?

¿Cómo se sincronizan los requerimientos con las respuestas?

¿Cómo se obtiene independencia de la representación física de datos?

¿Qué sucede cuando algunos de los extremos no está disponible?

Estos mecanismos de comunicación o middleware se representan por debajo de la capa de aplicaciones y por sobre el sistema operativo tal como indica la Figura 12.

Fig. 12. Nivel de Comunicación o Middleware

Los mecanismos para comunicar procesos entre diferentes computadoras pueden clasificarse como sincrónicos o asincrónicos. Esta clasificación determina si las preguntas y respuestas entre componentes se sincronizan o no en el tiempo y a su vez esto determina si se requiere o no disponibilidad de los componentes para poder establecer la comunicación:

Entre las variantes sincrónicas encontramos sockets y RPC/RMI, mientras que MOM (Message Oriented Middleware) es un caso de mecanismo de comunicación asincrónico.

Mecanismo de Comunicación por Sockets

El mecanismo de comunicación por *sockets* establece un canal de comunicación entre dos computadoras con una conexión ya existente a nivel de transporte. Generalmente se resuelve sobre el protocolo TPC pero existen mecanismos similares sobre otros protocolos.

Una vez establecido el canal de comunicación, el socket se define como un par que contiene la dirección IP de cada computador y un número de puerto. A partir de allí, el programador debe resolver, mediante la escritura de líneas de código, todas las funcionalidades para que la comunicación a nivel de programa de aplicación pueda realizarse.

Para usar sockets es necesario:

- Establecer la dirección del socket (IP, Nro. de puerto)
- Conocer la dirección de destino
- Usar las operaciones del socket

La Figura 13 muestra un ejemplo:

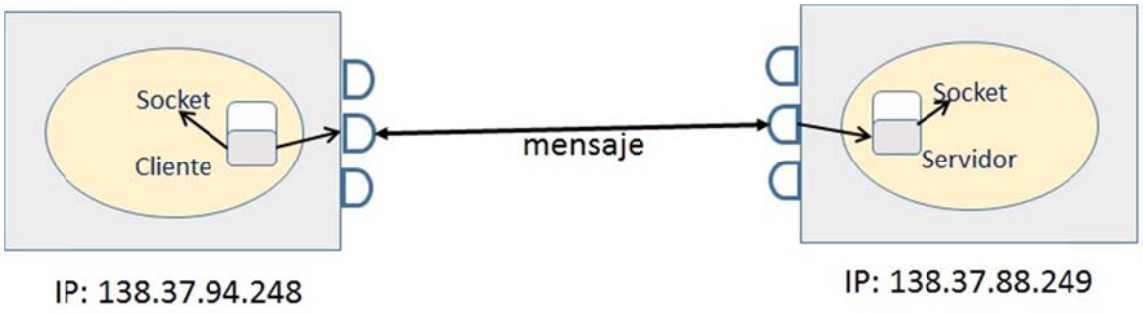

Fig. 13. Mecanismo de Comunicación por Socket

Las operaciones de los socket son:

1- Se crea un socket con la función socket
2- Los clientes deben conectarse con connect
3- Los servidores aceptan conexiones de clientes usando la función accept
4- Clientes y servidores intercambian datos con read y write
5- Las conexiones se cierran con close

El mecanismo de comunicación por socket carece de un entorno que facilite las tareas del programador, sólo dispone de un conjunto de operaciones básicas, siendo responsabilidad del programador resolver:

Diferencias en la representación de datos: cada computadora posee su propio hardware y sistema operativo, con lo cual es posible que se trate de arquitecturas diferentes y que los datos no sean codificados de la misma manera. En ese caso, el programador del socket debe tener en cuenta estas diferencias.

Manejo de errores: si se producen errores en la ejecución de los programas remotos, el programador deberá tener en cuenta los mismos y escribir el código necesario para manejar los mismos.

Asociación de cada componente distribuida: como se ha observado, se debe conocer de antemano las direcciones de las computadoras que participarán en la comunicación.

Mecanismo de comunicación por RPC/RMI

RPC (Remote Procedure Call) es un mecanismo de comunicación entre computadoras definido en la década del 60 y que se transformó más tarde en uno de los primeros lenguajes para programación distribuida.

Se trata de un mecanismo de comunicación sincrónico que se caracteriza por incluir las siguientes características:

- Lanzamiento y localización de funciones del servidor. El código compilado en RPC cuenta con un componente servidor y un componente cliente y esto determina que el componente servidor pueda lanzarse y permanezca escuchando en una determinada localización.
- Pasaje y definición de parámetros entre cliente y servidor. Si bien RPC intenta simular el modelo call-return de la programación clásica (no distribuida), este modelo ejecutado en componentes distribuidas se tropieza con el hecho de que no existe el concepto de memoria compartida. RPC garantiza que la pila de ejecución del proceso remoto que se invoca reciba los parámetros y retorna los resultados. Para ello utiliza el modelo de pasaje de parámetros por entrada/salida en lugar del modelo por referencia, que sería inviable dada la falta de memoria compartida.
- Manejo de fallas y caídas. Las fallas de ejecución de programas en entornos distribuidos, conllevan a revisar diversas fuentes de error, en contraposición con los errores en entornos únicos, donde la cancelación por error de un programa es absoluta responsabilidad del único programa que se está ejecutando.

En entornos distribuidos, la falla puede producirse por caída del medio de comunicación (red) o por caída de los extremos (computadores), debiendo considerarse una semántica para manejar dichos errores.

1. Semántica idempotente: las operaciones se pueden realizar varias veces y siempre se obtiene el mismo resultado que si se realiza una vez.
2. Semántica *exactamente una vez*: la operación se lleva a cabo una sola vez sin reintentos. En caso de falla, la operación queda sin ejecutarse.
3. Mecanismo *a lo sumo una vez*: la operación se lleva a cabo una vez y se aplican tantos intentos como sean necesarios hasta garantizar que se ejecutó, pero solo una vez.

La Figura 14 resume los tres modelos semánticos detallando su comportamiento en cada caso: retrasmisión del requerimiento, filtro de duplicados y retransmisión del mensaje.

			Semántica de Invocación
Retrasmite Mensaje	Filtra Duplicados	Re-ejecuta Proceso	
NO	No Aplica	No Aplica	Idempotente
SI	NO	Re-Ejecuta	A lo sumo una vez
SI	SI	Re-Trasmite	Al menos una vez

Fig. 14. Comportamiento en los Distintos Modelos Semánticos

El lenguaje RPC posee un compilador que genera por cada código fuente, cuatro archivos: componente servidor, componente cliente, componente para resolver la representación de datos (mediante mecanismo de serialización o marshaling) y el código ejecutable propiamente dicho.

Una evolución del RPC se hizo presente ante la aparición del paradigma de programación orientado a objetos. Surge así RMI (Remote Method Invocation) que es una herramienta similar al RPC pero para Java y que es análoga al RPC pero aplicado a objetos distribuidos.

Mecanismo de Comunicación Orientado a Mensajes (MOM)

El mecanismo de comunicación orientado a mensajes se basa en el concepto que los componentes distribuidos se comunican enviando mensajes de cualquier tipo y que quedan encolados hasta que son leídos por el destinatario.

Este mecanismo es asincrónico y oculta totalmente los detalles del medio de comunicación (red) y de la arquitectura de los componentes, eliminando algunos de los inconvenientes de los métodos sincrónicos.

Sin embargo, aparece el inconveniente de decir si las colas de mensaje serán persistentes o no y, en caso que lo sean, como asegurar esa persistencia. Es decir, que pasa con los mensa-

jes no persistidos si el componente que los ha recibido y no los ha procesado, deja de funcionar o cae en una condición de error.

Existen numerosos productos de software que otorgan distintas variantes de solución e implementación para los mecanismos orientados a mensajes.

La Figura 15 resume algunas de las características de los mecanismos de comunicación y cómo se comportan en el caso de sincronismo o en el caso de asincronismo.

Característica	**MOM**	**RPC**
Metáfora	Como oficina postal	Como teléfono
Tiempo de relación C/S	Asincrónico. Clientes y servidores operan en distinto tiempo.	Sincrónico. Clientes y servidores corren a la vez y se esperan
Secuenciamiento C/S	Sin secuencia fija	Los servidores se lanzan antes que los clientes.
Estilo	Cola	Call/Return
Datos persistentes	SI	NO
Compañero disponible	NO	SI
Balanceo de carga	Una cola simple FIFO o con prioridades	Requiere un monitor de TP adicional
Soporte transaccional	SI (algunos productos)	NO (requiere RPC transaccional)
Filtrado de mensajes	SI	NO
Performance	Baja	Alta
Procesamiento asincrónico	SI (se requiere colas y triggers)	Limitado (requiere hilos y trucos de programación

Fig. 15. Comparación de Mecanismos de Comunicación Sincrónicos y Asincrónicos

1.6. Conclusiones

El capítulo presenta algunas definiciones clásicas de los sistemas distribuidos y también marcó los desafíos que dichos sistemas deben afrontar así como los objetivos que persiguen. Los ejemplos que se plantean permiten conocer diversos tipos de sistemas distribuidos clasificados por las funciones y prestaciones que brindan.

El análisis arquitectónico permite comprender gráficamente la manera que se ubican los componentes y también cómo interactúan.

Los mecanismos de comunicación se constituyen en la componente fundamental para aglutinar cada parte de un sistema distribuido dando la visión de un sistema único, definiendo así el concepto de middleware.

El próximo capítulo presenta la evolución de los sistemas distribuidos a través del tiempo y a través de las diferentes tecnologías que fueron surgiendo como soporte de dichos sistemas.

CAPÍTULO 2
Evolución de los Sistemas Distribuidos

Patricia Bazán

Los sistemas distribuidos concebidos como aquellos que funcionan como si se tratara de un sistema único, no hacen ninguna presuposición acerca de la variedad funcional de sus componentes ni de cómo estos componentes se comunican. De esta manera las variantes que existen han dado origen a sistemas distribuidos de distinta índole y que van de la no-distribución (sistemas monolíticos) hacia la distribución completa (servicios orquestados por procesos de negocio).

En este capítulo se analiza esta evolución tanto desde el punto de vista tecnológico que sin dudas se ve afectado por la evolución de las comunicaciones, plataformas y hardware.

El capítulo se organiza de la siguiente manera: la Sección 1 presenta el concepto de sistema monolítico y sus mejoras ante la aparición de distintos componentes funcionales. En la Sección 2 se describen los sistemas Cliente/Servidor y su evolución hacia sistemas de n-capas. En la sección 3 se analizan los sistemas distribuidos con objetos como antecesores a las arquitecturas orientadas a servicios presentadas en la Sección 5. Mientras tanto, en la Sección 4 se describen las distintas alternativas de integración de aplicaciones. En la Sección 6 se analizan a los procesos de negocio como consumidores de Servicios. Finalmente en la Sección 7 se arriban a algunas conclusiones.

2.1. Sistemas Monolíticos

Un sistema de información monolítico es aquel que se concibe como un único elemento funcional donde sus prestaciones se ofrecen a través de una sola pieza de código que resuelve tanto la presentación al usuario (interface), como el acceso a los datos (trata con operaciones de entrada/salida) y a su vez resuelve la lógica algorítmica del problema que aborda.

Estos sistemas de información se construían en un único lenguaje de programación, sobre un único computador y con un único sistema operativo como plataforma. La comunicación con el usuario y también con el programador, era a través de terminales de caracteres que responden únicamente a comandos lanzados por consola. Su ubicación en el tiempo se remonta a los años ‘70.

Una versión mejorada de los sistemas de información así concebidos, pudo aportarse con las técnicas de programación estructurada, los lenguajes de programación y las metodologías