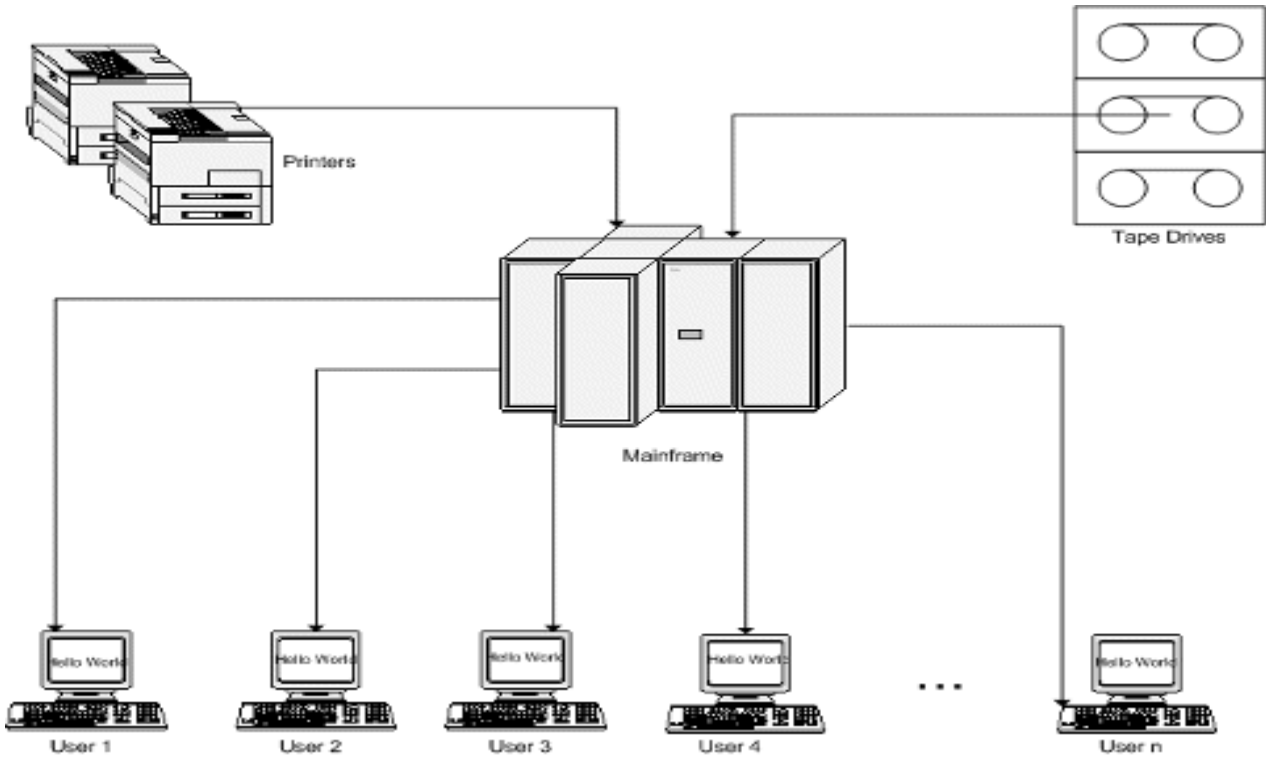
Desarrollo de Aplicaciones N-Tier

Lic. Guillermo Cherencio. Versión 1.0 – Febrero 2009/15

Ambiente Mainframe

La primera forma de automatización de negocios tomó la forma de una gran computadora central, llamada "Mainframe" o “Host” , la cual era un enorme computador central que prestaba servicio a toda la empresa a través de terminales. Todo el proceso se producía dentro de un costoso computador central (Mainframe), todos los recursos estaban conectados directamente a este computador central (cintas, discos, impresoras, etc.). Las terminales no tenían recursos propios[1], sólo tenían un simple teclado y una pantalla (por lo general, verde y negra) para realizar todas las peticiones al computador central. Esta arquitectura también se denominó “single tier”. Es una arquitectura simple, eficiente, segura, cara.

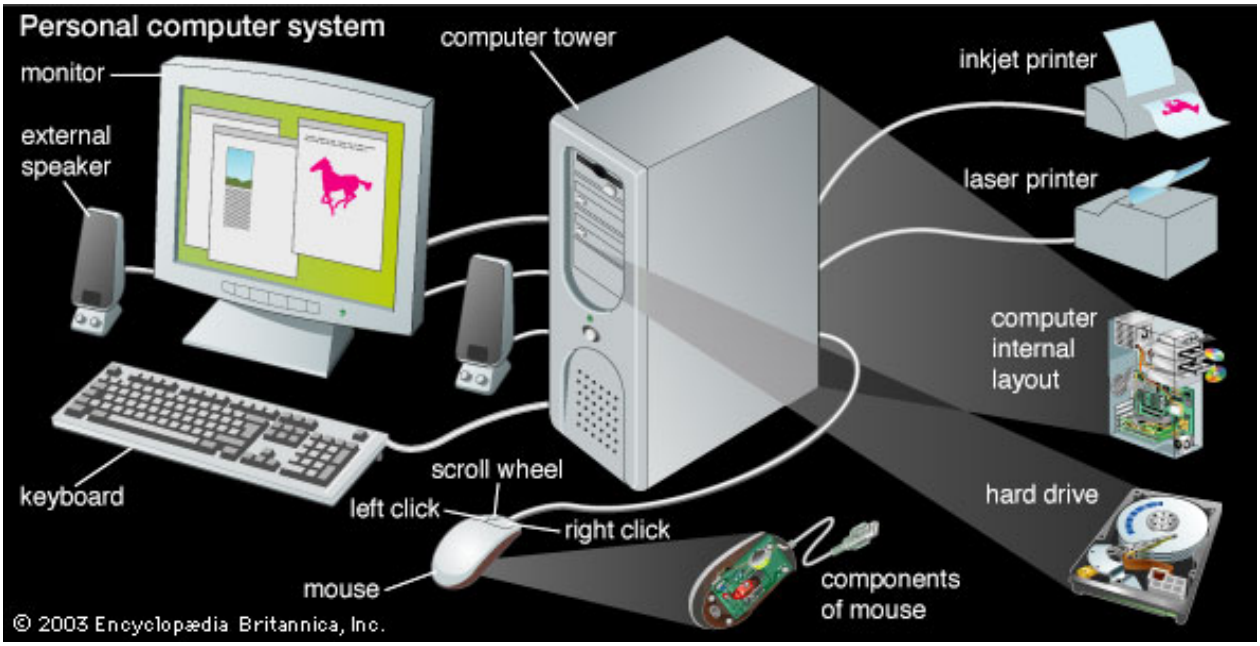


Ambiente Monousuario

A partir de 1980, se produce una revolución en los negocios de computación: “La Computadora Personal (PC)” (algo inimaginable hasta entonces) que

[1] También eran llamadas “terminales bobas” puesto que no contaban con una CPU (unidad central de proceso) propia, sólo tenían un microprocesador para gestionar la entrada (por teclado) y salida (por pantalla).

rápidamente se convirtió en el equipo standard sobre los escritorios de todo el mundo. Se produjo una gran demanda de "software personal ", estas computadoras solían contar con sistema operativo MS-DOS[2] que luego fue reemplazado por las primeras versiones de MS-Windows[3]. Este modelo se masificó y permitió el acceso de muchas personas al mundo informático. Una de las principales desventajas de este modelo era la compartición de información y recursos entre distintos usuarios, en especial en un ambiente de negocios. Otra desventaja era la escasa seguridad.



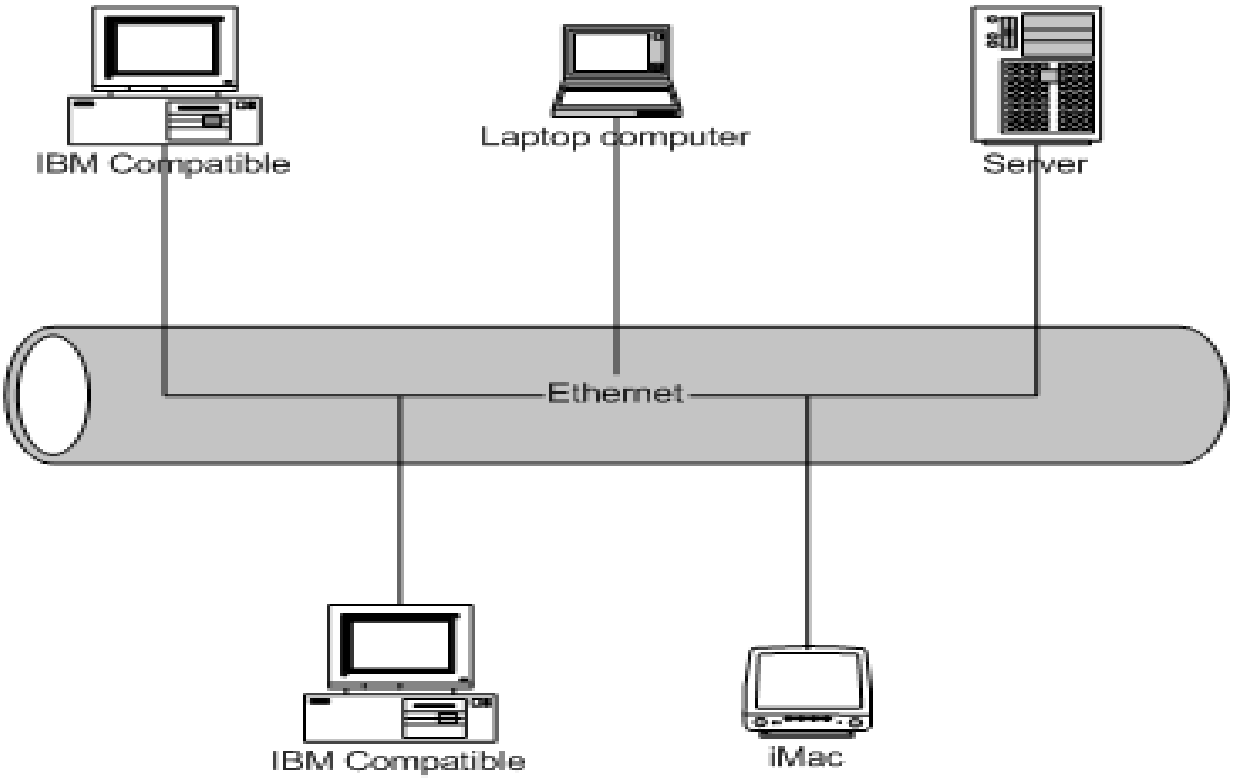
Ambiente Multiusuario – LAN – WAN

La arquitectura de hardware de los computadores personales permitía que otros fabricantes de hardware diseñaran dispositivos que podían acoplarse a estas nuevas máquinas a través de unas ranuras o "slots" que permitían un acceso directo al interior de la máquina. Para suplir las desventajas de un ambiente monousuario, surgieron las placas de red o NIC's (network interface cards) que se acoplaban al computador personal y permitían –a través de cables- conectar a un computador personal a una red que interconectaba otros

[2] Sistema operativo con una interfaz de usuario orientada a carácter o consola. El usuario tipea los comandos (en la sintaxis apropiada) que el computador debe ejecutar.

[3] Sistema operativo con una interfaz de usuario gráfica. El usuario ya no necesita acordarse la sintaxis de todos los posibles comandos que el computador puede ejecutar, sino que interactúa con ella a través de una metáfora gráfica: iconos, imágenes, botones, listas, etc.. Esto fue posible gracias al invento del mouse como dispositivo apuntador para "hacer click" sobre regiones muy específicas de la pantalla del computador y que estas acciones sean interpretadas por el computador como determinados comandos que éste debía ejecutar.

computadores personales , que ahora dejaban de ser personales y pasaban a ser multiusuarios, denominándose estaciones de trabajo o “workstations “. Esto también fue posible gracias al desarrollo de protocolos de comunicaciones[4] que permitían el envío y recepción de “mensajes” a través de la red de computadores y que éstos puedan interpretar cada uno de esos mensajes sin importar las características propias de hardware y software de cada computador conectado a la red. Para esta época, no sólo existía MS-Windows como sistema operativo de un computador personal, sino que también había otros, tales como: OS2 de IBM, Solaris de Sun Microsystems, MAC-OS, UNIX, etc. Esto permitió la compartición de recursos, intercambio de información entre plataformas de hardware y software muy distintas. Surgieron distintas redes, las más populares fueron la Ethernet y Token-Ring.

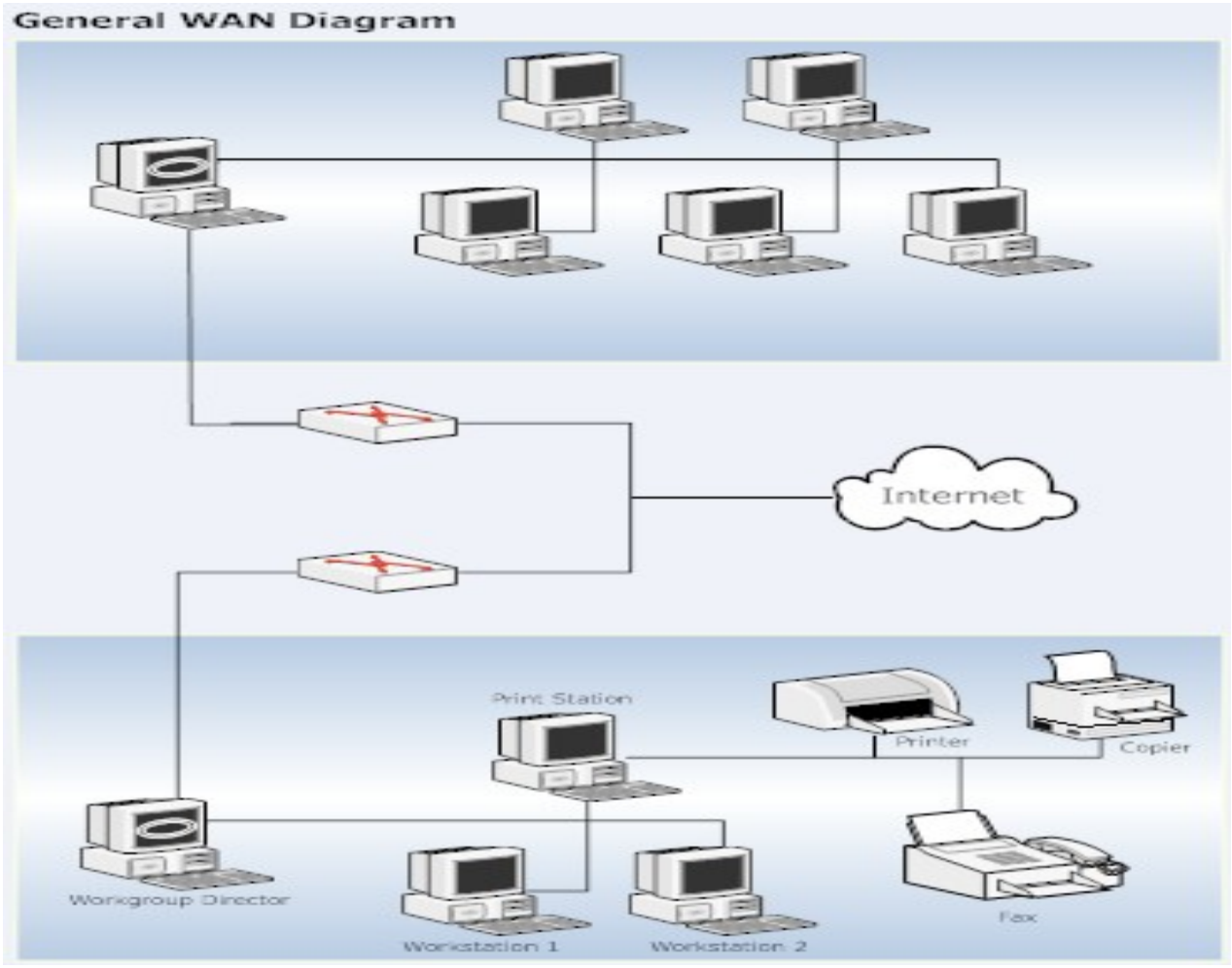


Las redes fueron al comienzo de alcance local, es decir, pequeñas redes de negocios situadas en un mismo edificio o edificios muy próximos que permitían interconectar todos los computadores de la empresa, esas redes se las denominó redes de area local o LAN (local area network).

[4] Software que permite la comunicación entre computadores, independientemente de las características propias de cada una de ellas, establece una especie de “idioma artificial” entre los computadores para el intercambio de mensajes dentro de una red. Uno de los protocolos de comunicaciones más populares es TCP/IP que es el utilizado en internet.

Luego surgió la necesidad de interconectar computadores a mayor distancia, allí muchas veces hubo que suplir el medio físico del cable por otros medios, tales como las comunicaciones satelitales y así se permitió la interconexión de computadores personales muy distantes entre sí, a estas redes se las denominó WAN (wide area network).

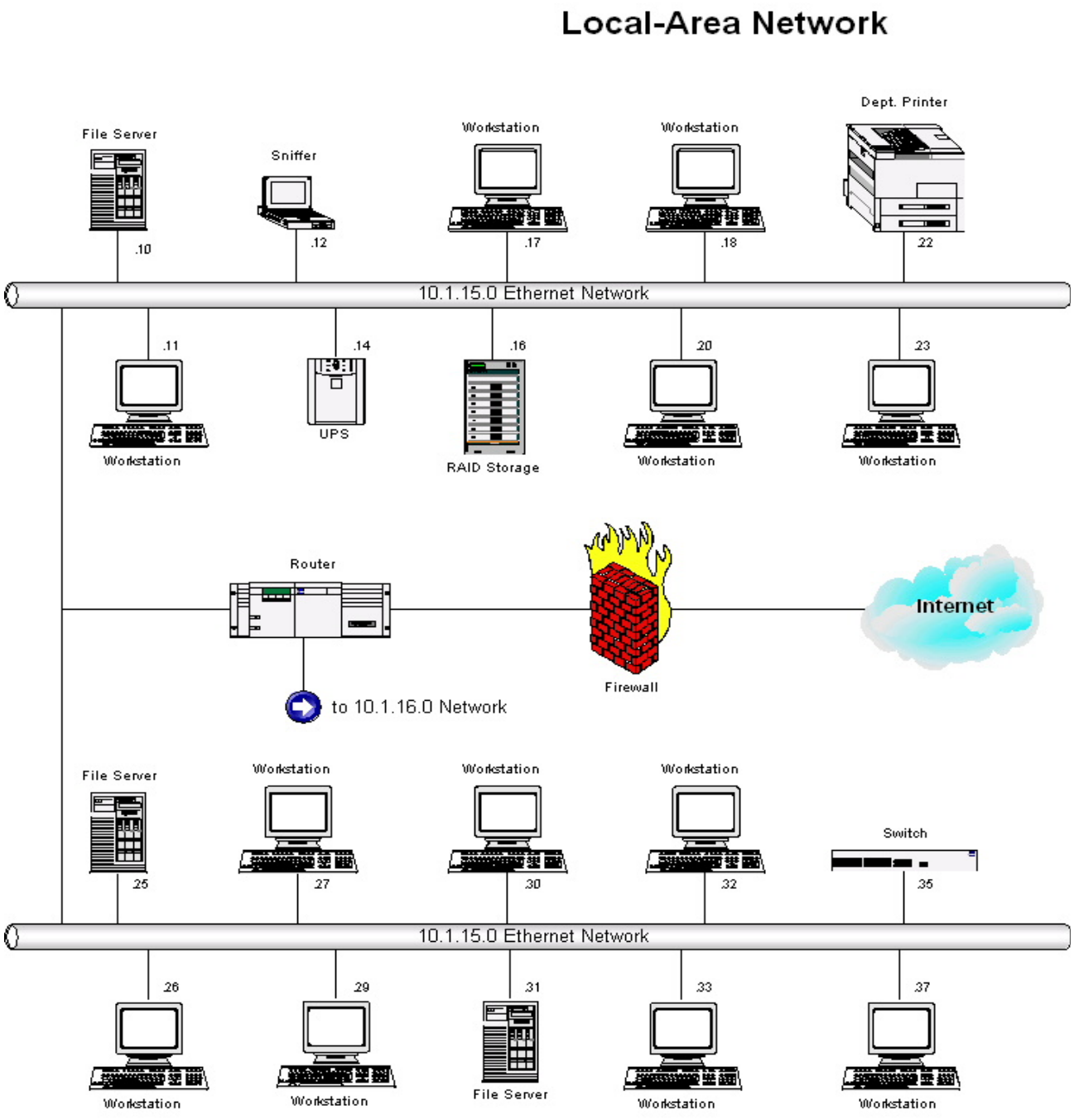
Imaginemos una empresa con sucursales muy distantes, en cada sucursal podría haber una red de area local para interconectar todas las computadoras y recursos. A su vez, para interconectar toda la empresa, se necesitaría interconectar a cada una de estas redes de area local de cada una de las sucursales de la empresa. Esto debía hacerse a través de enlaces rentados a empresas de comunicaciones privadas o enlaces satelitales propios. Hoy día podría usarse la propia internet como medio de enlace de redes locales a través de una red virtual privada o VPN (virtual private network).



Ambientes y Desarrollo de Aplicaciones

Arquitectura File – Server. Modelo 1-Tier.

Volviendo al ejemplo de la empresa con sucursales distantes interconectadas a través de una WAN, podemos imaginarnos que una forma de compartir información entre todos o un grupos de usuarios sería publicarla en una máquina determinada de la red y que todos los usuarios autorizados puedan accederla fácilmente. En este tipo de redes, se denomina Servidor de Archivos o “File Server” a una máquina designada a tal efecto. Por ejemplo, podríamos “publicar” este documento en un Servidor de Archivos y todas las estaciones de trabajo de la red podrían abrirlo. Esto implica que este documento -si bien reside en el Servidor de Archivos- al ser solicitado por una estación de trabajo, el mismo debe “viajar” por toda la red hasta llegar a la estación de trabajo que la solicitó. Así funcionaría nuestro acceso a este documento. También implica que todos los usuarios deberían tener acceso directo al documento para poder “transferirlo” a través de toda red hasta su ubicación. Si muchos usuarios, al mismo tiempo, solicitan el mismo documento, se produce un gran “tráfico” en la red y una gran carga de trabajo para cada una de las estaciones de trabajo que solicitaron el documento. Otro problema es la seguridad: ¿Cómo hacemos para asegurarnos que nadie nos modifique el documento? ¿Qué sucede si el documento debe ser actualizado al mismo tiempo por muchos usuarios?.



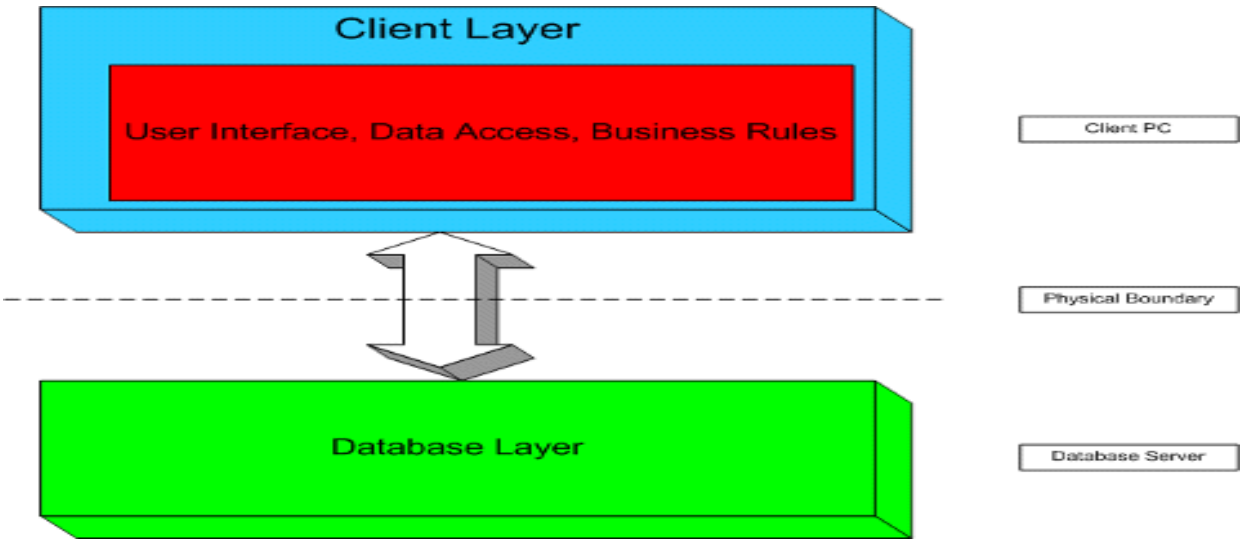
Arquitectura Client – Server. Modelo 2-Tier.

Como una forma de superación del modelo anterior surge una nueva arquitectura que afecta al desarrollo de software: el modelo Cliente- Servidor o "Client-Server". La idea es bastante simple, comparando con el modelo File-

Server el programa que se ejecuta sobre la estación de trabajo necesita “de un acceso directo” sobre el archivo con el cual pretende trabajar. En una arquitectura Cliente – Servidor intervienen dos programas: un programa que se ejecuta en la estación de trabajo, el cual se denomina cliente y otro programa que se ejecuta en otra máquina de la red[5] al cual se denomina server. El cliente realiza una “petición” al server, por ejemplo: “necesito modificar el recurso X con la información Y”; el programa server recibe la petición del cliente y la procesa. Esto implica que el programa cliente no tiene acceso directo al recurso X, sino que el acceso es indirecto, a través del programa server. Este simple esquema tiene enormes implicaciones en el desarrollo de software y en la superación del modelo anterior: ahora las aplicaciones deben pensarse con una lógica diferente: no existe un único programa sino dos, uno cliente y otro servidor. ¿Qué cosas debe hacer el cliente y qué cosas debe hacer el servidor?. La seguridad puede ser implementada por el servidor, controlando quién puede modificar un recurso y de qué forma. Se evita el acceso directo al recurso. Se mejora el tráfico en la red: sólo viajan peticiones y respuestas, en vez del recurso mismo. Un server atiende a n clientes. Servidor y cliente son independientes de la plataforma de software y hardware. Distintos clientes, de distintos fabricantes, corriendo en distintos tipos de máquinas pueden “dialogar” con un mismo servidor, sólo se requiere que “hablen el mismo idioma”.

Este es un modelo dual, de dos capas o “2-Tier”. Imaginemos que la empresa necesita un sistema de facturación, entonces tenemos, por un lado, la interacción con el usuario (User Interface), comunicación con el servidor para acceder a los datos (Data Access) , validaciones, reglas de negocio (Business Rules) (Client Layer) y por otro lado tenemos el repositorio de datos o base de datos en donde los datos de la facturación serán almacenados (Database Layer), el esquema conceptual sería el siguiente:

[5] Generalmente se trata de una máquina con mayores recursos de hardware que una estación de trabajo, al cual se denomina Servidor o “Server”, que se encarga de “atender” a todos los clientes que le solicitan recursos.



Puede observarse que la mayoría de los componentes de software se encuentran en cliente. Si bien es una superación de la Arquitectura File-Server, aún persisten algunos problemas con este modelo:

- Las conexiones tienen un costo elevado: llevan tiempo de conexión y demandan mucha memoria en el servidor, debido a esto, muchas aplicaciones se conectan cuando arrancan, mantienen todo el tiempo la conexión abierta con el servidor y se desconectan cuando la aplicación termina. Si la aplicación se cuelga, deja abierta la conexión sobre el servidor.

- Se pueden conectar un número limitado de usuarios al servidor, si se aumenta el número, el servidor terminará gastando más tiempo en administrar las conexiones que en procesar las peticiones de los clientes.

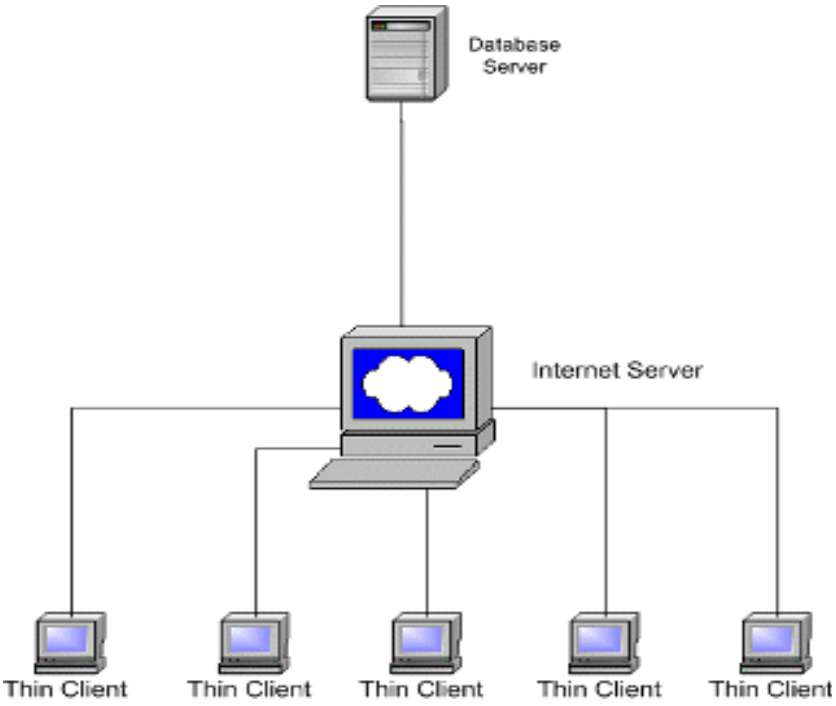
- Las conexiones no son muy eficientes, pues la mayoría de los usuarios sólo la están usando efectivamente entre el 2 y 3% del tiempo. El resto del tiempo están sólo gastando memoria y recursos dentro del servidor para administrar la conexión.

Arquitectura Internet. Modelo 3-Tier.

Con el advenimiento de internet y la separación de las aplicaciones en distintas capas, surge una alternativa al modelo anterior: el modelo 3-Tier o modelo en 3 capas. La idea básica de este modelo se trata de tener una capa cliente (Client Layer, ver figura anterior) más liviana, más delgada (thin) y hasta incluso, standard![6]. Toda la arquitectura, en realidad, son dos arquitecturas

[6] Puesto que los denominados clientes delgados (Thin Client) se refieren a clientes que sólo interpretan código HTML, como es el caso de los navegadores de internet, tales como: MS-Internet Explorer, Mozilla Firefox, Opera, etc. Todos ellos son Thin Client's.

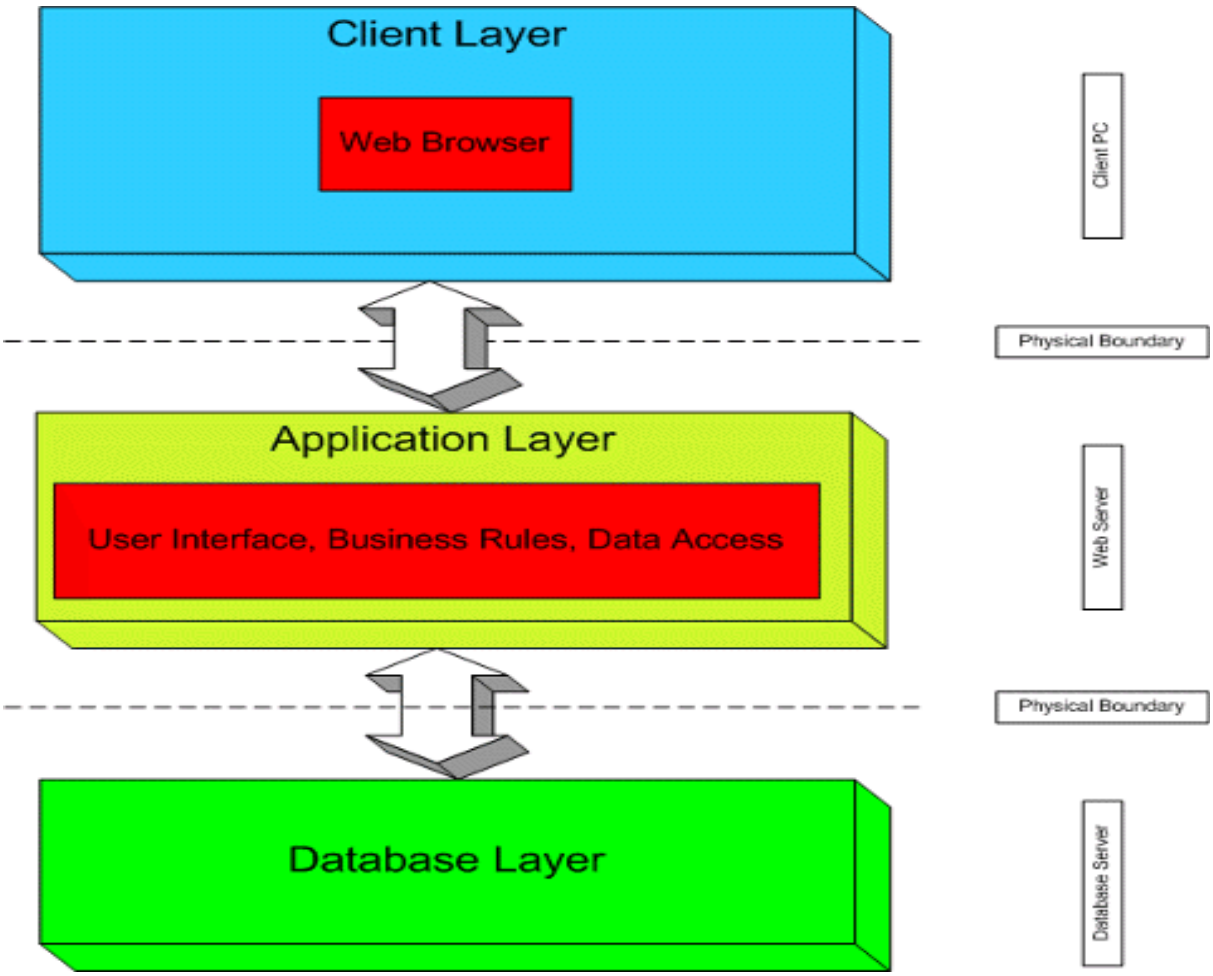
client-server enlazadas: client – server/client – server en donde una máquina intermedia (Internet Server) hace al mismo tiempo de server y cliente:



La capa cliente sólo muestra la interfase con el usuario (pantallas para ingresar y mostrar datos), pero no participa en la formación de los datos. Esta arquitectura muestra una ventaja con respecto a la anterior: un web server (Internet Server)[7] bien implementado puede manejar un conjunto de conexiones con la base de datos[8] al mismo tiempo que corre las aplicaciones. Una desventaja es que el web server puede ser saturado de peticiones clientes y por lo tanto, se requiera agregar más web servers (para atender a más clientes al mismo tiempo) o poner hardware más potente. Ahora conceptualmente, nuestra aplicación podría tener este aspecto:

[7] Uno de los más populares en Internet es Apache Web Server.

[8] Esto también se denomina pool de conexiones (connection pool), la idea es simple, por ejemplo, un servidor podría mantener -al mismo tiempo- sólo 10 conexiones permanentes con el servidor de base de datos para dar servicio a 200 usuarios, de esta forma, las conexiones serían más eficientes, pues se supone que estarían más ocupadas, elevando su utilización muy por encima del 2 o el 3%.



Modelo N-Tier.

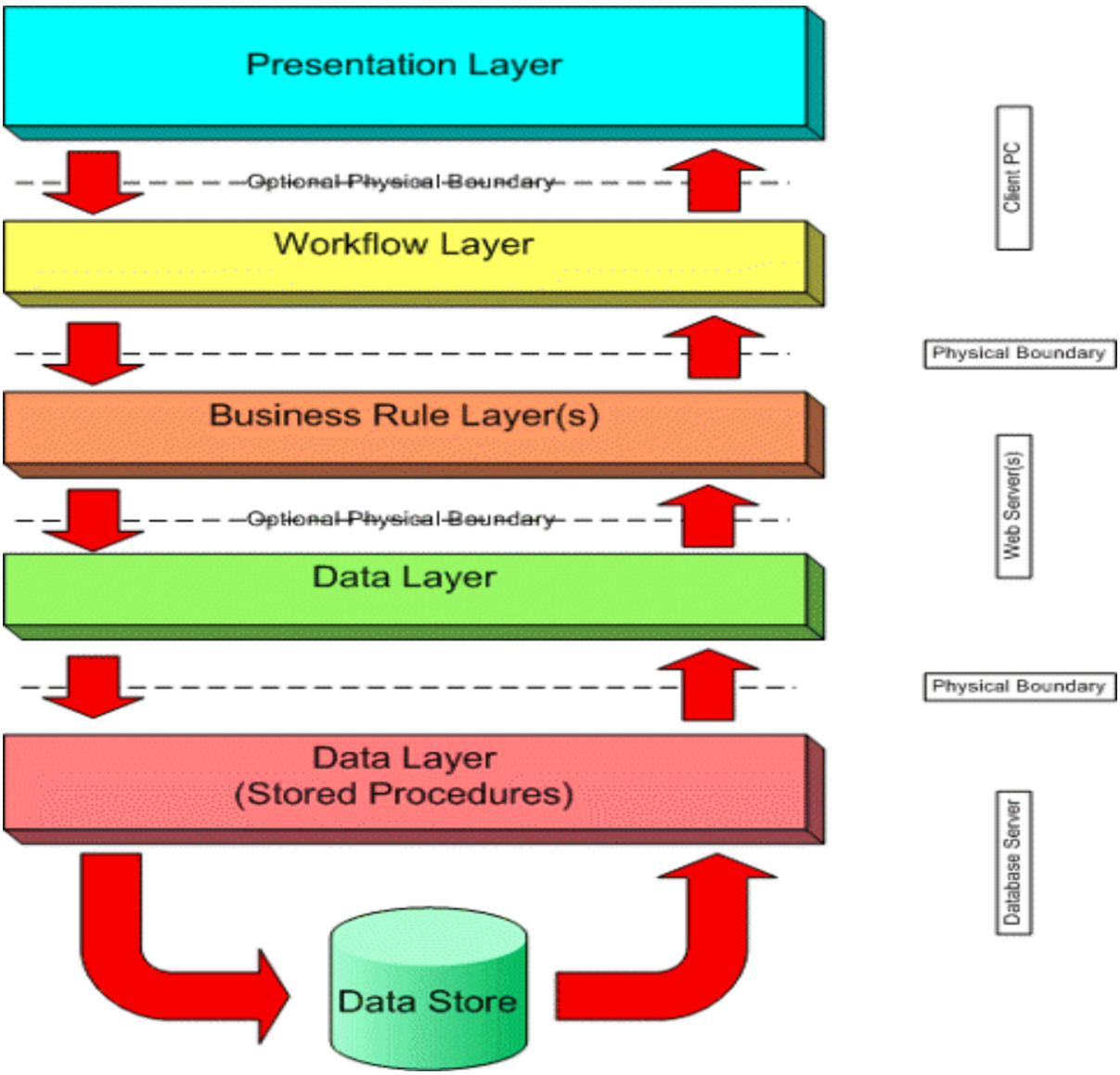
La capa de datos (Database Layer) suele dividirse en dos capas: una capa implementa una serie de procedimientos almacenados (stored procedures) en la base de datos misma (Data Layer, Stored Procedures) que permiten interactuar directamente con los datos que residen en la base de datos (Data Store). Por encima de la capa de procedimientos almacenados, está la capa de datos (Data Layer) que son una serie de clases simples[9] las cuales ejecutarán los procedimientos almacenados (Data Layer, Stored Procedures) y manejarán una o más tablas de datos simples relacionadas, agregando control sobre concurrencia y un mayor nivel de abstracción sobre los datos del sistema. Por encima de la capa de datos (Data Layer) se encuentra la capa de reglas de negocio (Business Rule Layers) la cual implementa toda la funcionalidad de

[9] Se refiere al paradigma de programación orientada a objetos, en donde el mundo se representa a través de objetos que interactúan entre sí para resolver un problema determinado. Las clases son diseños de objetos. Estos objetos vinculan a datos que deben actualizarse o recuperarse de la base de datos.

negocios, no se interesa en el acceso a datos (a lo sumo, los accede a través de la capa de datos) o la presentación de datos, su interés se focaliza en la complejidad del negocio, trata de aislar esta funcionalidad y puede tratarse de varias capas. Por encima de la(s) capa de reglas de negocios, suele existir una capa que se denomina capa de flujo de trabajo o WorkFlow (WorkFlow Layer), la cual suele manejar todos los mensajes que suelen enviarse y recibirse dentro del servidor de aplicaciones y fuera del mismo. Supongamos que su sistema requiere usar un servicio de otro sistema[10], por ejemplo, recuperar datos meteorológicos; el manejo de esta comunicación estará implementado en esta capa. Por último, ya en la máquina del cliente, se encuentra la capa de presentación (Presentation Layer), ésta incluye todas las pantallas, controles gráficos y clases necesarias para la interacción con el usuario y presentación de datos.

Las nuevas herramientas de desarrollo de aplicaciones distribuidas tales como la plataforma .NET de Microsoft o la plataforma Java de Oracle (antes Sun Microsystems), facilitan el desarrollo de aplicaciones N-Tier. Aquí podemos ver el modelo conceptual de una aplicación bajo este moderno enfoque:

[10] Se refiere a sistemas modernos que se han desarrollado pensando en brindar servicios a otros sistemas, ello se logra implementando un protocolo que se denomina SOA (Service Oriented Applications) o aplicaciones orientadas a servicios. Esto permite que cada sistema se focalice en el problema puntual a resolver y delegue en otros SOA's la resolución de otras cuestiones que no son centrales para él. Muchos de estos SOA's son públicos y gratuitos, están "vivos" en Internet y disponibles para quien desee utilizarlos.



Preguntas de Revisión

1. ¿Qué es más barato: agregar más puestos de trabajo en ambiente Mainframe o en un ambiente multiusuario?
2. En el ambiente Mainframe, ¿En dónde se encuentra la mayor carga de trabajo?
3. En la Arquitectura File-Server, ¿En dónde se encuentra la mayor carga de trabajo?
4. En la Arquitectura Client-Server, ¿En dónde se encuentra la mayor carga de trabajo?

5. Teniendo en cuenta la respuesta de la pregunta 2, si los usuarios se quejan de que sus aplicaciones funcionan muy lentas, ¿Qué se debería hacer?

6. Teniendo en cuenta la respuesta de la pregunta 3, si los usuarios se quejan de que sus aplicaciones funcionan muy lentas, ¿Qué se debería hacer?

7. Teniendo en cuenta la respuesta de la pregunta 4, si los usuarios se quejan de que sus aplicaciones funcionan muy lentas, ¿Qué se debería hacer?

8. Si los clientes de una arquitectura 3-Tier se quejan de que los datos que solicitan tardan mucho en ser desplegados en sus pantallas, ¿Qué cree Ud. que se podría hacer para mejorar su rendimiento?

9. ¿Por qué cree Ud. Que el desarrollo de aplicaciones modernas requiera de tantas capas? ¿Qué ventajas cree Ud. que trae el hecho de diseñar las aplicaciones de esta forma?

10. Supongamos que Ud. desarrolló un Sistema de Facturación en una arquitectura File-Server, y resulta que se encuentra con el siguiente problema: "la factura (ya impresa y cobrada) Nº 10231 es por un total de $1.500; pero si consultamos el sistema, éste nos dice que la factura Nº 10231 es por un valor de $500". Ud. Busca desesperadamente el error en el programa y no logra encontrar ningún error en el programa de facturación, ¿Qué cree Ud. que podría haber provocado esta situación?

Bibliografía

Traducciones y adaptaciones de G. Cherencio sobre texto original de:

Hyatt, Karim, *"N-Tier Application Development with Microsoft.NET. Part 1 : What is N-Tier Architecture?"*, Microsoft Developer Network, http://www.microsoft.com/belux/msdn/nl/community/columns/hyatt/ntier1.mspx