UDS

LIBRO

SISTEMAS OPERATIVOS DISTRIBUIDOS

INGENIERÍA EN SISTEMAS COMPUTACIONALES
6TO CUATRIMESTRE

Marco Estratégico de Referencia

ANTECEDENTES HISTORICOS

Nuestra Universidad tiene sus antecedentes de formación en el año de 1979 con el inicio de actividades de la normal de educadoras "Edgar Robledo Santiago", que en su momento marcó un nuevo rumbo para la educación de Comitán y del estado de Chiapas. Nuestra escuela fue fundada por el Profesor de Primaria Manuel Albores Salazar con la idea de traer Educación a Comitán, ya que esto representaba una forma de apoyar a muchas familias de la región para que siguieran estudiando.

En el año 1984 inicia actividades el CBTiS Moctezuma Ilhuicamina, que fue el primer bachillerato tecnológico particular del estado de Chiapas, manteniendo con esto la visión en grande de traer Educación a nuestro municipio, esta institución fue creada para que la gente que trabajaba por la mañana tuviera la opción de estudiar por las tarde.

La Maestra Martha Ruth Alcázar Mellanes es la madre de los tres integrantes de la familia Albores Alcázar que se fueron integrando poco a poco a la escuela formada por su padre, el Profesor Manuel Albores Salazar; Víctor Manuel Albores Alcázar en septiembre de 1996 como chofer de transporte escolar, Karla Fabiola Albores Alcázar se integró como Profesora en 1998, Martha Patricia Albores Alcázar en el departamento de finanzas en 1999.

En el año 2002, Víctor Manuel Albores Alcázar formó el Grupo Educativo Albores Alcázar S.C. para darle un nuevo rumbo y sentido empresarial al negocio familiar y en el año 2004 funda la Universidad Del Sureste.

La formación de nuestra Universidad se da principalmente porque en Comitán y en toda la región no existía una verdadera oferta Educativa, por lo que se veía urgente la creación de una institución de Educación superior, pero que estuviera a la altura de las exigencias de los jóvenes que tenían intención de seguir estudiando o de los profesionistas para seguir preparándose a través de estudios de posgrado.

Nuestra Universidad inició sus actividades el 18 de agosto del 2004 en las instalaciones de la 4ª avenida oriente sur no. 24, con la licenciatura en Puericultura, contando con dos grupos de cuarenta alumnos cada uno. En el año 2005 nos trasladamos a nuestras propias instalaciones en la carretera Comitán – Tzimol km. 57 donde actualmente se encuentra el campus Comitán y el Corporativo UDS, este último, es el encargado de estandarizar y controlar todos los procesos operativos y Educativos de los diferentes Campus, Sedes y Centros de Enlace Educativo, así como de crear los diferentes planes estratégicos de expansión de la marca a nivel nacional e internacional.

Nuestra Universidad inició sus actividades el 18 de agosto del 2004 en las instalaciones de la 4ª avenida oriente sur no. 24, con la licenciatura en Puericultura, contando con dos grupos de cuarenta alumnos cada uno. En el año 2005 nos trasladamos a nuestras propias instalaciones en la carretera Comitán – Tzimol km. 57 donde actualmente se encuentra el campus Comitán y el corporativo UDS, este último, es el encargado de estandarizar y controlar todos los procesos operativos y educativos de los diferentes campus, así como de crear los diferentes planes estratégicos de expansión de la marca.

MISIÓN

Satisfacer la necesidad de Educación que promueva el espíritu emprendedor, aplicando altos estándares de calidad Académica, que propicien el desarrollo de nuestros alumnos, Profesores, colaboradores y la sociedad, a través de la incorporación de tecnologías en el proceso de enseñanza-aprendizaje.

VISIÓN

Ser la mejor oferta académica en cada región de influencia, y a través de nuestra Plataforma Virtual tener una cobertura Global, con un crecimiento sostenible y las ofertas académicas innovadoras con pertinencia para la sociedad.

VALORES

- Disciplina
- Honestidad
- Equidad
- Libertad

ESCUDO

El escudo de la UDS, está constituido por tres líneas curvas que nacen de izquierda a derecha formando los escalones al éxito. En la parte superior está situado un cuadro motivo de la abstracción de la forma de un libro abierto.

ESLOGAN

"Mi Universidad"

ALBORES

Es nuestra mascota, un Jaguar. Su piel es negra y se distingue por ser líder, trabaja en equipo y obtiene lo que desea. El ímpetu, extremo valor y fortaleza son los rasgos que distinguen.

Sistemas operativos distribuidos

Objetivo de la materia: Proveer a los alumnos los conceptos básicos asociados con los Sistemas Operativos Distribuidos y Sistemas Distribuidos. Que los alumnos conozcan y comprendan los mecanismos que entran en juego cuando el Sistema Distribuido tiene que manejar procesos residentes en equipos en red.

1. **UNIDAD I INTRODUCCIÓN A LOS SISTEMAS DISTRIBUIDOS**
   - 1.1. Introducción.
   - 1.2. Ejemplos de sistemas distribuidos
   - 1.3. Desafíos.
   - 1.4. Modelos de sistema
   - 1.5. Qué es un Modelo de Sistema Distribuido.
   - 1.6. Servidor
   - 1.7. Cliente
   - 1.8. Internet
   - 1.9. Modelo Cliente Servidor.
   - 1.10. Apache e IIS
   - 1.11. Páginas web
   - 1.12. Modelos fundamentales.

2. **UNIDAD IICOMUNICACIONES EN LOS SISTEMAS DISTRIBUIDOS**
   - 2.1. Las redes y los sistemas distribuidos.
   - 2.2. Fundamentos de redes
   - 2.3. La conmutación de Circuitos virtuales
   - 2.4. ATM
   - 2.5. Metro Ethernet.
   - 2.6. Comunicación entre procesos
   - 2.7. Introducción.
   - 2.8. API para los protocolos de Internet.
   - 2.9. Representación externa de datos y empaquetado.
   - 2.10. Comunicación en grupo
   - 2.11. Conexiones por IP

3. **UNIDAD III OBJETOS DISTRIBUIDOS E INVOCACION DE METODOS**
   - 3.1. Introducción.
   - 3.2. RPC.
   - 3.3. Ejecucion RPC en c#
   - 3.4. RMI
   - 3.5. Ejecucion en Java de RMI.
   - 3.6. Diferencia entre RPC y RMI

3.7. Soporte del sistema operativo.
3.8. Sistemas de archivos distribuidos.
3.9. Características de los sistemas de archivos.
3.10. Arquitectura del servicio de archivo
3.11. Procesos e hilos
3.12. Los hilos y el sistema operativo
3.13. Patrones de trabajo con hilos
3.14. Concurrencia
3.15. Mecanismos de sincronización

4. **UNIDAD IV SINCRONIZACION Y ESTADOS GLOBALES**
4.1. Relojes eventos y estados de proceso.
4.2. Estados globales.
4.3. Depuración distribuida.
4.4. Coordinación y acuerdo
4.5. Suposiciones sobre fallos y detectores de fallos.
4.6. Exclusión mutua distribuida.
4.7. Comunicación por multidifusión (Multicast).
4.8. El consenso y sus problemas relacionados
4.9. Transacciones y control de concurrencia.
4.10. Modelo de fallos para transacciones.
4.11. Transacciones anidadas

Índice

UNIDAD I INTRODUCCIÓN A LOS SISTEMAS DISTRIBUIDOS

1.1 INTRODUCCIÓN.

La computación desde sus inicios ha sufrido muchos cambios, desde los grandes ordenadores que permitían realizar tareas en forma limitada y de uso un tanto exclusivo de organizaciones muy selectas, hasta los actuales ordenadores ya sean personales o portátiles que tienen las mismas e incluso mayores capacidades que los primeros y que están cada vez más introducidos en el quehacer cotidiano de una persona.

Los mayores cambios se atribuyen principalmente a dos causas, que se dieron desde las décadas de los setenta:

1. El desarrollo de los microprocesadores, que permitieron reducir en tamaño y costo a los ordenadores y aumentar en gran medida las capacidades de los mismos y su acceso a más personas.
2. El desarrollo de las redes de área local y de las comunicaciones que permitieron conectar ordenadores con posibilidad de transferencia de datos a alta velocidad.

Es en este contexto que aparece el concepto de "Sistemas Distribuidos" que se ha popularizado tanto en la actualidad y que tiene como ámbito de estudio las redes como, por ejemplo: Internet, redes de teléfonos móviles, redes corporativas, redes de empresas, etc.

SISTEMAS DISTRIBUIDOS (definición).

"Sistemas cuyos componentes hardware y software, que están en computadoras conectadas en red, se comunican y coordinan sus acciones mediante el paso de mensajes, para el logro de un objetivo. Se establece la comunicación mediante un protocolo preestablecido[II].

CARACTERÍSTICAS.

• Concurrencia. - Esta característica de los sistemas distribuidos permite que los recursos disponibles en la red puedan ser utilizados simultáneamente por los usuarios y/o agentes que interactúan en la red.

• Carencia de reloj global. - Las coordinaciones para la transferencia de mensajes entre los diferentes componentes para la realización de una tarea, no tienen una temporización general, está más bien distribuida en los componentes.

• Fallos independientes de los componentes. - Cada componente del sistema pudiera fallar de manera independientemente, y los demás continuar ejecutando sus acciones. Esto permite el logro de las tareas con mayor efectividad, pues el sistema en su conjunto continúa trabajando.

EVOLUCIÓN.

Procesamiento central (Host).- Refiere a uno de los primeros modelos de computadoras interconectadas, llamados centralizados, donde todo el procesamiento de la organización se llevaba a cabo en una sola computadora, normalmente un Mainframe, y los usuarios empleaban sencillas computadoras personales.

Algunos problemas de este modelo son:

• Cuando la carga de procesamiento aumentaba se tenía que cambiar el hardware del Mainframe, lo cual es más costoso que añadir más computadores personales clientes o servidores que aumenten las capacidades.

• El otro problema que surgió son las modernas interfases gráficas de usuario, las cuales podían conllevar a un gran aumento de tráfico en los medios de comunicación y por consiguiente podían colapsar a los sistemas.

Grupo de Servidores. - Otro modelo que entró a competir con el anterior, también un tanto centralizado, son un grupo de computadoras actuando como servidores, normalmente de archivos o de impresión, poco inteligentes para un número de minicomputadores que hacen el procesamiento conectados a una red de área local.

Algunos problemas de este modelo son:

Podría generarse una saturación de los medios de comunicación entre los servidores poco inteligentes y los minicomputadores, por ejemplo, cuando se solicitan archivos grandes por varios clientes a la vez, podían disminuir en gran medida la velocidad de transmisión de información.

La Computación Cliente Servidor. - Este modelo, que predomina en la actualidad, permite descentralizar el procesamiento y recursos, sobre todo, de cada uno de los servicios y de la visualización de la Interfaz Gráfica de Usuario. Esto hace que ciertos servidores estén dedicados sólo a una aplicación determinada y por lo tanto ejecutarla en forma eficiente.

1.2 EJEMPLOS DE SISTEMAS DISTRIBUIDOS

- Una red de estaciones de trabajo en un departamento de una universidad o compañía, donde además de cada estación personal, podría existir una pila de procesadores en el cuarto de máquinas, que no estén asignados a usuarios específicos, sino que se utilicen de manera dinámica cuando sea necesario.
- Una fábrica de robots, donde los robots actúan como dispositivos periféricos unidos a la misma computadora central.
- Un banco con muchas sucursales por el mundo, cada oficina tiene una computadora maestra para guardar las cuentas locales y el manejo de las transacciones locales, la cual se puede comunicar con cualquier computadora de la red. Las transacciones hechas se realizan sin importar dónde se encuentre la cuenta o el cliente.

1.3 DESAFÍOS.

Heterogeneidad

La heterogeneidad se aplica en los siguientes elementos:

- Redes
- Hardware de computadores
- Sistemas operativos
- Lenguajes de programación
- Implementaciones de diferentes
- Desarrolladores

Middleware: es el estrato de software que provee una abstracción de programación, así como un enmascaramiento de la heterogeneidad subyacente de las redes, hardware, sistemas operativos y lenguajes de programación. Ejem: Corba, Java RMI

Heterogeneidad y código móvil. Código que puede enviarse desde un computador a otro y ejecutarse en este último. El concepto de máquina virtual ofrece un modo de crear código ejecutable sobre cualquier hardware.

Extensibilidad

Es la característica que determina si el sistema puede extenderse de varias maneras. Un sistema puede ser abierto o cerrado con respecto a extensiones de hardware o de software. Para lograr la extensibilidad es imprescindible que las interfaces clave sean publicadas.

Los Sistemas Distribuidos Abiertos pueden extenderse a nivel de hardware mediante la inclusión de computadoras a la red y a nivel de software por la introducción de nuevos servicios y la re implementación de los antiguos. Otro beneficio de los sistemas abiertos es su independencia de proveedores concretos.

Seguridad

La seguridad tiene tres componentes: Confidencialidad: protección contra individuos no autorizados, Integridad: protección contra la alteración o corrupción, Disponibilidad: protección contra la interferencia que impide el acceso a los recursos.

Escalabilidad

Se dice que un sistema es escalable si conserva su efectividad cuando ocurre un incremento significativo en el número de recursos y en el número de usuarios.

Tratamiento de Fallos

Detección de fallos: Ejem. Se pueden utilizar sumas de comprobación (checksums) para detectar datos corruptos en un mensaje.

Enmascaramiento de fallos: Ejem: Los mensajes pueden retransmitirse, Replicar los datos.
Tolerancia de fallos: los programas clientes de los servicios pueden diseñarse para tolerar ciertos fallos. Esto implica que los usuarios tendrán también que tolerarlos.

Recuperación de fallos: implica el diseño de software en el que, tras una caída del servidor, el estado de los datos puede reponerse o retractarse (rollback) a una situación anterior.
Redundancia: emplear componentes redundantes.

Concurrencia

Existe la posibilidad de acceso concurrente a un mismo recurso. La concurrencia en los servidores se puede lograr a través de threads. Cada objeto que represente un recurso compartido debe responsabilizarse de garantizar que opera correctamente en un entorno concurrente. Para que un objeto sea seguro en un entorno concurrente, sus operaciones deben sincronizarse de forma que sus datos permanezcan consistentes.

Transparencia

- Transparencia de acceso: permite acceder a los recursos locales y remotos empleando operaciones idénticas.
- Transparencia de ubicación: permite acceder a los recursos sin conocer su localización. Transparencia de concurrencia: permite que varios procesos operen concurrentemente sobre recursos compartidos sin interferencia mutua.
- Transparencia de replicación: permite replicar los recursos sin que los usuarios y los programadores necesiten su conocimiento.
- Transparencia frente a fallos: permite ocultar fallos.
- Transparencia de movilidad: permite la reubicación de recursos y clientes en un sistema sin afectar la operación de los usuarios y los programas.
- Transparencia de rendimiento: permite reconfigurar el sistema para mejorar el desempeño según varíe su carga.
- Transparencia al escalado: permite al sistema y a las aplicaciones expandirse en tamaño sin cambiar la estructura del sistema o los algoritmos de aplicación.

1.4 MODELOS DE SISTEMA

Los sistemas distribuidos (SD) son los sistemas software más complejos. Con los modelos arquitectónicos tratamos de simplificar estos sistemas viendo la colocación de las partes y las relaciones entre ellas. Incluye también el control global de estructura, los protocolos para comunicación, sincronización, y acceso a datos, la asignación de funcionalidad, distribución física, escalamiento y desempeño, dimensiones de evolución y selección de alternativas de diseño.

Los diferentes modelos arquitectónicos

son: Capas de software.

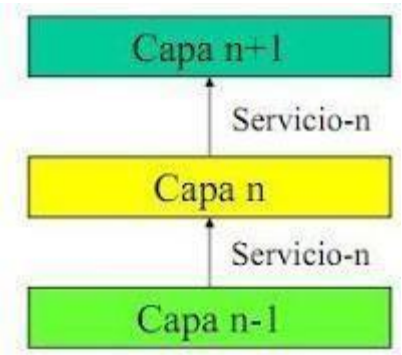
Arquitectura de

Sistemas. Interface y

Objetos.

Capa de software:

La idea básica es desmembrar la complejidad de sistema mediante el diseño en capas y servicios. El término arquitectura de software se refería inicialmente a la estructuración del software como capas en un único computador. Más recientemente las capas son uno o varios procesos, localizados en el mismo o diferente computador, que ofrecen y solicitan servicios.



Capas: grupo de funcionalidades fuertemente y altamente

coherentes. Servicios: funcionalidades proporcionales a capas

superiores.

La estructura típica en capas de un SD es la siguiente:

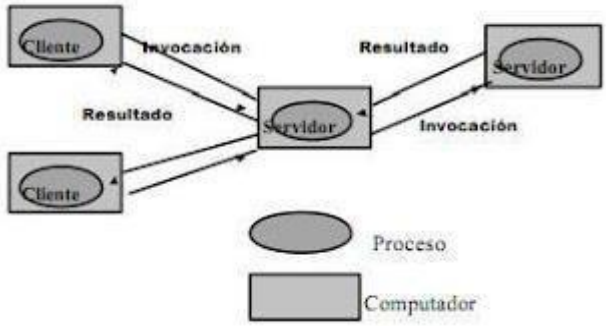
Plataforma: Hadware y sistema operativo. Estas capas más bajas proporcionan servicios a las superiores y su implementación depende de cada computador.

Middleware: Es una capa software que logra transparencia en la heterogeneidad en el nivel de plataforma. Logra comunicación y compartición de recursos. El middleware se ocupa de proporcionar bloques útiles para la construcción de componentes de software que puedan trabajar con otros en un sistema distribuido. En particular mejorar el nivel de las actividades de comunicación en los procesos de aplicación, soportando abstracciones como: llamadas a procedimientos remotos, comunicación entre grupos de procesos, etc.

Ejemplos: Sun RPC(llamadas a procedimientos remotos), CORBA(middleware orientado a objeto), JAVA RMI(Invocación de objetos remotos en JAVA), DCOM(Modelo común de objetos distribuidos de Microsoft).

El middlewasre también puede proporcionar otros servicios, aparte de la comunicación, para su uso en programas de aplicación. Por ejemplo: gestión de nombres, seguridad, almacenamiento persistente, etc.

Modelo Básico: Cliente Servidor.



Cliente: el proceso requiere acceder datos, utilizar recursos o ejecutar operaciones en una computadora diferente.

Servidor: proceso maneja datos y otros recursos compartidos, permite al cliente acceder a recursos y ejecutar cómputos.

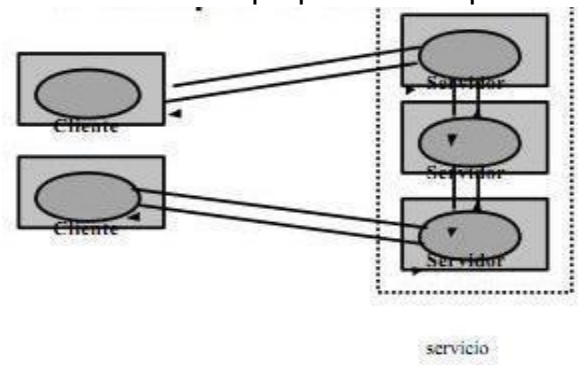
Iteración: invocación/ par de mensajes resultantes

Ejemplo: servidor http: cliente (navegador) página solicitada, servidor entrega página.

Servidor de caching(servidores proxy): Caching de páginas Web frecuentemente utilizadas.

Procesos pares (no cliente-servidor: peer-to-peer): procesos que tienen una gran parte de similitudes de funcionalidad.

Variante: Servicios proporcionados por múltiples servidores.



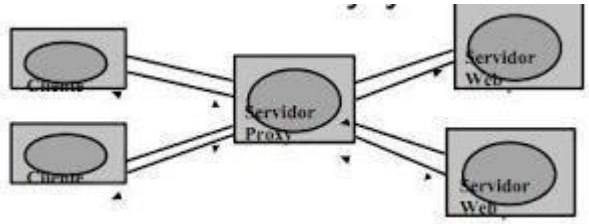
Ejemplos: muchos servidores de comercio Web están implementados en diferentes servidores.

Motivación: desempeño y confiabilidad.

Los servidores mantienen bases de datos replicadas o distribuidas.

Variantes: Servidores proxy y caché

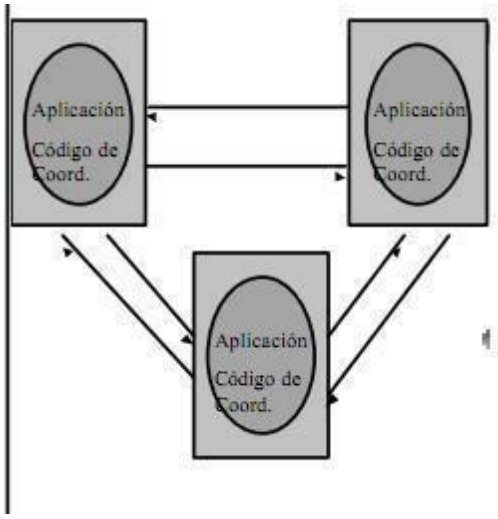
Suministra replicación/ distribución transparente.



Un caché es un almacén de objetos da datos utilizados recientemente. Los cachés pueden estar ubicados en los clientes o en servidores Proxy que se puede compartir desde varios clientes.

El propósito de los servidores proxy es incrementar la disponibilidad y las prestaciones del servicio, reduciendo la carga en las redes de área amplia y servidores WEB.

Caching: los servidores proxy mantienen caches, como almacenes de recursos solicitados recientemente. Son utilizados frecuentemente en motores de búsqueda. Variantes: Procesos Peer-to-Peer



Todos los procesos desempeñan tareas semejantes, interactuando cooperativamente como iguales para realizar una actividad distribuida o cómputo sin distinción entre cliente y servidores.

Los procesos pares mantienen la consistencia de los recursos y sincronizan las acciones a nivel de aplicación.

Interface y Objetos

Una interface de un proceso es la especificación del conjunto de funciones que se pueden invocar sobre él.

En lenguajes orientados a objetos, los procesos distribuidos pueden ser construidos de una forma más orientada al objeto. Las referencias a estos objetos se pasan a otros procesos para que se pueda acceder a sus métodos de forma remota. Esta es la aproximación adoptada por CORBA y JAVA RMI.

Variaciones del modelo cliente-servidor: Código móvil.

Códigos enviados a un proceso cliente para realizar una tarea específica

Ejemplos: Applets, mensajes Activos.

Variaciones del modelo cliente-servidor: Agentes móviles

Programa que se traslada a la red, de un computador a otro, realizando una tarea para alguien.

Programa ejecutado (códigos+datos), migración entre procesos, realizando una tarea autónoma, frecuentemente en representación de otros procesos. Ventajas: flexibilidad, ahorro en costo de comunicación. Variaciones del modelo cliente-servidor: Clientes Delgados

En el cliente sólo se ejecuta una interfaz basada en ventajas, mientras que la aplicación si se ejecuta en un servidor remoto, usualmente muy potente.

1.5 QUÉ ES UN MODELO DE SISTEMA DISTRIBUIDO.

Se denomina cómputo distribuido a un proceso de cómputo realizado entre computadoras independientes, o, más formalmente, entre procesadores que no comparten memoria (almacenamiento primario). Puede verse que un equipo de diseño NUMA está a medio camino entre una computadora multiprocesada y el cómputo distribuido.

Hay diferentes modelos para implementar el cómputo distribuido, siempre basados en la transmisión de datos sobre una red. Éstos son principalmente

(clusters) Computadoras conectadas por una red local (de alta velocidad), ejecutando cada una su propia instancia de sistema operativo.
Pueden estar orientadas al alto rendimiento, alta disponibilidad o al balanceo de cargas (o a una combinación de éstas). Típicamente son equipos homogéneos, y dedicados a la tarea en cuestión.

Mallas (Grids) Computadoras distribuidas geográficamente y conectadas mediante una red de comunicaciones. Las computadoras participantes pueden ser heterogéneas (en capacidades y hasta en arquitectura); la comunicación tiene que adecuarse a enlaces de mucha menor velocidad que en el caso de un cluster, e incluso presentar la elasticidad para permitir las conexiones y desconexiones de nodos en el transcurso del cómputo.

Cómputo en la nube Un caso específico de cómputo distribuido con partición de recursos (al estilo del modelo cliente-servidor); este modelo de servicio está fuertemente orientado a la tercerización de servicios específicos. A diferencia del modelo cliente-servidor tradicional, en un entorno de cómputo en la nube lo más común es que tanto el cliente como el servidor sean procesos que van integrando la información, posiblemente por muchos pasos, y que sólo eventualmente llegarán a un usuario final. La
implementación de cada uno de los servicios empleados deja de ser relevante, para volverse un servicio opaco. Algunos conceptos relacionados son:

Servicios Web Mecanismo de descripción de funcionalidad, así como de solicitud y recepción de resultados, basado en el estándar HTTP y contenido XML.

Software como servicio El proveedor ofrece una aplicación completa y cerrada sobre la red, exponiendo únicamente su interfaz (API) de consultas.

Plataforma como servicio El proveedor ofrece la abstracción de un entorno específico de desarrollo de modo que un equipo de programadores pueda desplegar una aplicación desarrollada sobre dicha plataforma tecnológica. Puede ser visto como un conjunto de piezas de infraestructura sobre un servidor administrado centralmente.

1.6 SERVIDOR

Un servidor es un sistema que proporciona recursos, datos, servicios o programas a otros ordenadores, conocidos como clientes, a través de una red. En teoría, se consideran servidores aquellos ordenadores que comparten recursos con máquinas cliente. Existen muchos tipos de servidores, como los servidores web, los servidores de correo y los servidores virtuales.

Un sistema individual puede, al mismo tiempo, proporcionar recursos y usar los de otro sistema. Esto significa que todo dispositivo podría ser a la vez servidor y cliente.

Los primeros servidores eran mainframes o microcomputadoras, que se denominan así por ser mucho más pequeñas que los equipos de mainframe. Sin embargo, conforme progresaba la tecnología, terminaron superando en tamaño a los ordenadores de sobremesa, por lo que el término microcomputadora resultaba un tanto inapropiado.

Inicialmente, dichos servidores estaban conectados a clientes que no realizaban ninguna computación real, y se les conocía como terminales. Estos terminales (también llamados dumb terminals), existían simplemente para aceptar entradas a través de un teclado o lector de tarjetas y devolver los resultados de cualquier cálculo a una pantalla o impresora. La computación real se efectuaba en el servidor.

Más tarde, los servidores pasaron a ser sistemas individuales de gran potencia que se conectaban a un conjunto de ordenadores cliente menos potentes a través de una red. A esta arquitectura

de red se la conoce como el modelo cliente-servidor, en el que tanto el ordenador cliente como el servidor poseen potencia computacional pero determinadas tareas se delegan a los servidores. En anteriores modelos informáticos, como el modelo mainframe-terminal, el primero sí actuaba como un servidor con todas las de la ley, a pesar de que no se mencionaba con dicho nombre.

La definición del concepto de servidor ha ido evolucionando con el avance de la tecnología. Hoy en día, un servidor puede no ser más que un software que se ejecuta en uno o más dispositivos informáticos físicos. A tales servidores se les suele adjetivar como virtuales. Originalmente, los servidores virtuales se usaban para incrementar el número de características que un servidor individual de hardware podía efectuar. Actualmente, los servidores virtuales se suelen ejecutar en la nube, es decir, dentro de un hardware que pertenece a un tercero y al que se puede acceder a través de internet.

Un servidor puede estar diseñado para realizar una sola tarea, como un servidor de correo, que acepta y almacena mensajes de correo electrónico y, luego, se los proporciona a un cliente que los solicita. Los servidores también pueden realizar más de una tarea, como un servidor de archivos e impresión que almacena archivos y acepta trabajos de impresión de los clientes, para luego enviarlos a una impresora conectada a la red.

Cómo funciona un servidor

Para que un dispositivo trabaje como un servidor, debe estar configurado para escuchar las solicitudes de los clientes en un entorno de red. Esta funcionalidad puede existir como parte del sistema operativo: en forma de aplicación instalada, un rol o una combinación de ambos.

Por ejemplo, el sistema operativo Windows Server de Microsoft proporciona las características necesarias para escuchar y responder a las solicitudes de los clientes. Además, los roles o servicios instalados incrementan el número de tipos de solicitudes del cliente a los que puede responder el servidor. En otro ejemplo, un servidor web Apache responde a las solicitudes del navegador de internet del cliente a través de una aplicación adicional, Apache, que se instala en la capa superior del sistema operativo.

Cuando un cliente pide datos o una funcionalidad de un servidor, lo hace enviando una solicitud a través de la red. El servidor recibe dicha solicitud y responde con la información correspondiente. Este es el modelo de solicitud y respuesta de la conexión cliente-servidor, lo que también se conoce como el modelo de llamada y respuesta.

A menudo, un servidor realizará numerosas tareas adicionales como parte de una sola solicitud y respuesta, como verificar la identidad del solicitante, asegurarse de que el cliente tenga permiso

para acceder a los datos o recursos solicitados y formatear o devolver adecuadamente la respuesta requerida de la forma esperada

Servidores de archivos

Los servidores de archivos almacenan y distribuyen ficheros que varios clientes o usuarios pueden compartir. Además, el almacenamiento centralizado de archivos ofrece soluciones de copia de seguridad o tolerancia a fallos de forma más sencilla que tratar de proporcionar seguridad e integridad a los archivos en todos y cada uno de los dispositivos de la organización. Se puede diseñar el hardware del servidor de archivos de modo que potencie las velocidades de lectura y escritura para mejorar el rendimiento.

Servidores de impresión

Los servidores de impresión permiten la gestión y distribución de la funcionalidad de imprimir documentos. Para no tener que conectar una impresora a cada estación de trabajo, podemos tener un único servidor de impresión para responder a las solicitudes de impresión de numerosos clientes. Hoy en día, algunas impresoras de alta gama y gran tamaño vienen con su propio servidor de impresión incorporado, ahorrando la necesidad de instalar uno en un equipo separado. Este servidor de impresión interno hace que la impresora responda también a las solicitudes de impresión de los clientes conectados.

Servidores de aplicaciones

Este tipo de servidores sirve para ejecutar aplicaciones de forma remota, en lugar de que los equipos cliente lo hagan localmente. Los servidores de aplicaciones a menudo ejecutan software que hace un uso intensivo de los recursos, y lo comparten para una gran cantidad de usuarios. Al hacerlo, por un lado, solo tenemos que instalar y mantener el software en una única máquina, y evitamos la necesidad de que cada cliente disponga de suficientes recursos de forma local.

Servidores DNS

Los servidores del sistema de nombres de dominio (DNS) son servidores de aplicaciones que proporcionan funcionalidades de resolución de nombres a los equipos cliente. La resolución de

nombres consiste en convertir nombres fácilmente comprensibles por los humanos en direcciones IP legibles por las máquinas. El sistema DNS es una base de datos ampliamente distribuida de nombres y otros servidores DNS a los que se puede consultar para obtener un nombre de equipo desconocido. Cuando un cliente necesita la dirección de un sistema, envía una solicitud con el nombre del recurso deseado a un servidor de DNS, que le responde con la dirección IP correspondiente de su tabla de nombres.

Servidores de correo

Los servidores de correo son un tipo muy común de servidor de aplicaciones. Los servidores de correo reciben los mensajes de correo electrónico que se remiten a un usuario y los almacenan hasta que un cliente los solicite en nombre de dicho usuario. Disponer de un servidor de correo electrónico nos permite tener una sola máquina configurada y conectada correctamente a la red en todo momento, lista para enviar y recibir mensajes en lugar de esperar que cada cliente tenga su propio subsistema de correo electrónico ejecutándose de forma continua.

Servidores web

Uno de los tipos de servidores más abundantes en el mercado actual son los servidores web. Un servidor web es un tipo especial de servidor de aplicaciones que aloja programas y datos solicitados por los usuarios a través de internet o en una intranet. Los servidores web responden a las solicitudes de páginas web u otros servicios basados en la web que llegan de los navegadores que se ejecutan en los ordenadores cliente. Entre los servidores web que podemos encontrar más frecuentemente tenemos servidores Apache, Microsoft Internet Information Services (IIS) y Nginx.

Servidor de base de datos

La cantidad de datos utilizados por empresas, usuarios y otros servicios es sobrecogedora. Gran parte de ellos se almacena en bases de datos. Estas bases de datos deben poder ser accesibles por parte de múltiples clientes en cualquier momento y, generalmente, exigen cantidades extraordinarias de espacio de almacenamiento. Ambas necesidades son la excusa perfecta para ubicar dichas bases de datos en un servidor. Los servidores de bases de datos ejecutan aplicaciones de bases de datos y responden a numerosas solicitudes de clientes. Los servidores de bases de datos más frecuentes son Oracle, Microsoft SQL Server, DB2 e Informix.

Servidores virtuales

Los servidores virtuales están arrasando en el mundo de los servidores. A diferencia de los servidores tradicionales, que se instalan como una dupla de sistema operativo y máquina de hardware, los servidores virtuales solo existen según los parámetros establecidos en un software especializado denominado hipervisor. Cada hipervisor puede ejecutar cientos o incluso miles de servidores virtuales a la vez. El hipervisor presenta el hardware virtual al servidor como si de una máquina física se tratase. El servidor virtual usa el hardware virtual como de costumbre, y el hipervisor traslada las necesidades reales de computación y almacenamiento al hardware real subyacente que se comparte entre todos los demás servidores virtuales.

Servidores *proxy*

Un servidor *proxy* actúa como intermediario entre un cliente y un servidor. A menudo se emplean para aislar a clientes o servidores por motivos de seguridad. Un servidor *proxy* toma la solicitud del cliente pero, en lugar de responderle directamente, traslada la solicitud a otro servidor o proceso. El servidor *proxy* recibe la respuesta del segundo servidor y, luego, responde al cliente original como si lo hiciera por sí mismo. De este modo, ni el cliente ni el servidor que se comunican realmente se conectan entre sí.

Servidores de supervisión y administración

Algunos servidores tienen la finalidad de supervisar o gestionar otros sistemas y clientes. Hay muchos tipos de servidores de supervisión. Algunos de ellos escuchan la red, recibiendo cada solicitud del cliente y cada respuesta del servidor, pero otros ni solicitan ni responden a los datos por sí mismos. De este modo, el servidor de supervisión puede realizar un seguimiento de todo el tráfico en la red, así como de las solicitudes y respuestas de otros servidores y clientes sin interferir con tales operaciones. Un servidor de supervisión responderá a las solicitudes de monitorización de los clientes, como aquellas que ejecutan los administradores de red cuando vigilan el estado de la red.

Estructuras de servidor

El concepto del servidor es casi tan antiguo como el de la red. Después de todo, el objetivo de una red es permitir que un equipo se comunique con otro de forma que se distribuya el trabajo o los recursos. La informática ha evolucionado mucho desde entonces, dando como resultado diferentes tipos de estructuras de servidor y hardware.

Mainframe o minicomputadora (AS/400)

Se podría decir que los servidores originales, los equipos de *mainframe* y, más tarde, las minicomputadoras, se encargaban de gestionar casi todas las tareas operativas exceptuando la interacción con el usuario a través de una pantalla y un teclado, que se dejaba a los sistemas cliente.

Servidor de hardware

La siguiente gran ola de servidores comprendía servidores basados en ordenadores. En muchos aspectos, estos no eran más que ordenadores de escritorio, pero más grandes y potentes. Generalmente, también eran más caros y tenían mucha más memoria y espacio en disco que la mayoría de los equipos cliente. Cada servidor seguía siendo una unidad autónoma con su propia placa base, procesador, memoria, unidades de disco y fuente de alimentación. A menudo, este tipo de servidores se almacenaba en estancias con aire acondicionado denominadas salas de servidores y, luego, se atornillaban en bastidores o racks para un mejor almacenamiento y accesibilidad.

Servidores Blade

Los servidores de hardware originales ocupaban mucho espacio y se almacenaban en bastidores que podían pesar toneladas. Sin embargo, con el tiempo, la aparición de medios de transferencia más rápidos en el hardware provocó la extracción de partes de esos servidores autónomos. Gracias a la posibilidad de eliminar los discos duros, quitar el sistema de refrigeración interna y miniaturizar de forma continua los componentes computacionales, los servidores se acabaron reduciendo a un único "servidor delgado" conocido como servidor Blade. Aunque se siguen almacenando en bastidores dentro de las salas de servidores, los Blade son más pequeños y se pueden reemplazar más fácilmente.

Combinar servidores

Ya se buscaba separar los servidores del modelo estándar de máquina de hardware individual con sistema operativo propio antes de que la virtualización entrara en escena. Los avances tecnológicos, como la funcionalidad de almacenamiento conectado a la red, eliminaron la necesidad de que un servidor tuviera su propio sistema de almacenamiento. Otras tecnologías, como el *mirroring* y los clústeres, permitieron combinar partes de hardware en servidores más grandes y potentes que podían estar compuestos de diferentes *blades*, varios dispositivos de almacenamiento conectados y una fuente de alimentación externa, y cada uno de estos módulos podía intercambiarse por otro mientras el servidor seguía en funcionamiento.

Servidores virtuales

Los servidores virtuales siguen necesitando de una capa de hardware, pero dicha capa ejecuta ahora un proceso diferente conocido como hipervisor. En algunos casos, como Hyper-V de Microsoft, tenemos un sistema operativo completo que se ejecuta sobre el propio hardware. En otros casos, los llamados "hipervisores *bare-metal*" se pueden instalar directamente en el hardware del servidor. En ambos casos, el hardware como tal suele extenderse a través de múltiples servidores *blade*, sistemas de almacenamiento en red y fuentes de alimentación, lo que produce un entorno en el cual es imposible saber dónde termina un servidor individual y dónde comienza otro.

Ejemplos de sistemas operativos de servidor

Microsoft Windows Server

Podría decirse que Windows para trabajo en grupo (Windows for Workgroups) fue el primer sistema operativo de servidores de Microsoft. En esa versión, ciertos ordenadores podían configurarse para compartir recursos y responder a las solicitudes de los clientes, lo que los convertía en servidores, por definición. El primer sistema operativo de servidores de verdad de Microsoft fue Windows NT. Sus versiones 3.5 y 3.51 podían encontrarse en muchas redes comerciales hasta que Microsoft lanzó su línea Windows Server, que continúa existiendo a día de hoy. La versión más actual es Windows Server 2016, que admite numerosas aplicaciones y bases de datos, así como un hipervisor que permite servidores virtuales.

Servidores Linux/Unix

El otro actor principal de los sistemas operativos de servidores es Linux/Unix. Podemos encontrarlo en múltiples versiones y de todos los colores, como Red Hat Enterprise Linux, Debian y CentOS como algunos de los más famosos. Al ser un sistema operativo de código abierto, Linux es muy popular como servidor web, generalmente con una instalación del servidor web Apache.

NetWare

Aunque ya ha pasado a mejor vida, NetWare fue un sistema importante en el espacio del software para servidores conforme se extendía la adopción del paradigma cliente-servidor. Finalmente, NetWare trasladó su sistema operativo de servidor a un *kernel* basado en Linux y lo bautizó como Novell Open Enterprise Server (OES).

Servidores en la nube

Aquellos servidores virtuales alojados en una infraestructura de terceros en una red abierta, como internet, se denominan servidores en la nube. En la actualidad, existen numerosos proveedores de servidores en la nube como Google Cloud Platform, Microsoft Azure e IBM Cloud. Sin embargo, el principal pionero en el campo de la computación en la nube corporativa fue la plataforma AWS de Amazon. Comenzó usando la capacidad libre de los propios servidores y redes de Amazon, y actualmente AWS ahora permite a sus clientes crear un servidor virtual casi instantáneamente y, luego, ajustar la cantidad de recursos que dicho servidor puede utilizar sobre la marcha.

Hoy en día, un servidor puede no ser más que un conjunto de datos del hardware físico, compuesto por multitud de procesadores, unidades de disco, memoria y conexiones de red. No obstante, en el fondo, un servidor sigue siendo simplemente un sistema que responde a una solicitud de un cliente.

1.7 CLIENTE

Un sistema que utiliza servicios remotos de un servidor. Algunos clientes tienen una capacidad de almacenamiento en disco limitada o, tal vez, nula. Estos clientes dependen de sistemas de archivos remotos de un servidor para funcionar. Los sistemas sin disco y los sistemas de dispositivos son ejemplos de este tipo de clientes.

Otros clientes quizás utilicen los servicios remotos (como el software de instalación) de un servidor. Sin embargo, no dependen de un servidor para funcionar. Un sistema independiente es un buen ejemplo de este tipo de clientes.

El concepto de cliente servidor**,** **o** cliente-servidor**,** refiere por lo tanto a un modelo de comunicación que vincula a varios dispositivos informáticos a través de una red El cliente, en este marco, realiza peticiones de servicios al servidor, que se encarga de satisfacer dichos requerimientos.

Con esta arquitectura, las tareas se distribuyen entre los servidores (que proveen los servicios) y los clientes (que demandan dichos servicios). Dicho de otro modo: el cliente le pide un recurso al servidor, que brinda una respuesta.

Este tipo de modelos permite repartir de la capacidad de procesamiento. El servidor puede ejecutarse sobre más de un equipo y ser más de un programa. De acuerdo a los servicios que brinda, se lo puede llamar servidor web, servidor de correo o de otro modo.

En las redes estructuradas bajo el modelo cliente servidor, los clientes centralizan diferentes aplicaciones y recursos en el servidor. El servidor, a su vez, se encarga de que estos recursos estén disponibles cada vez que un cliente los requiere.

Es importante mencionar que gran parte de los servicios de Internet obedecen a la arquitectura cliente servidor. El servidor web pone a disposición del cliente los sitios web, a los cuales el cliente accede a través de su navegador. El servidor, de esta manera, aloja los datos que el cliente solicita mediante el navegador instalado en su computadora.

Uno de los ejemplos más "antiguos" en este contexto es el correo electrónico, que demuestra a cada segundo de su funcionamiento los principios del modelo cliente servidor. En este caso, el cliente envía y recibe mensajes que "viajan" a través de redes de comunicación, y éstos se alojan en "buzones" cuyo nombre técnico es servidores de correo.

A diferencia de lo que ocurre fuera de la informática, en el ámbito del correo tradicional, el usuario no abre un buzón material para mirar dentro de él si hay cartas o paquetes, sino que debe

solicitar a un servidor (normalmente remoto) que verifique la presencia de nuevos mensajes y le envíe una respuesta con el resultado. Cada acción se apoya en esta relación, de manera que el equipo del cliente no es autosuficiente.

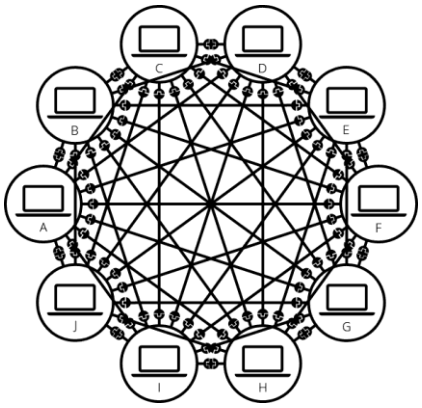
Una de las ventajas menos aparentes de la organización en servidores y clientes es que la capacidad de procesamiento y memoria de estos últimos no debe ser tan grande como la de los primeros, lo cual beneficia al consumidor final permitiéndole usar un equipo relativamente antiguo para disfrutar de servicios generalmente muy avanzados.

Por ejemplo, a pesar de que el correo electrónico parezca una "aplicación" muy liviana y sencilla, los servidores deben almacenar volúmenes colosales de datos para satisfacer a todos sus clientes, y, por consiguiente, realizar búsquedas y consultas muy demandantes para responder a todas sus solicitudes. Cuando buscamos un término en nuestra casilla para dar con un mensaje en particular, el servidor debe revisar cientos o miles de archivos, y lo hace en una fracción de segundo, algo que sería imposible en nuestros hogares.

Los sistemas de streaming de videojuegos para usarlos a distancia son otro ejemplo, en este caso mucho más exigente que el correo electrónico, ya que el cliente puede disfrutar de un programa de última generación en tiempo real con un ordenador que simplemente le permita recibir el vídeo de forma fluida y enviar los eventos de su mando, teclado y ratón.

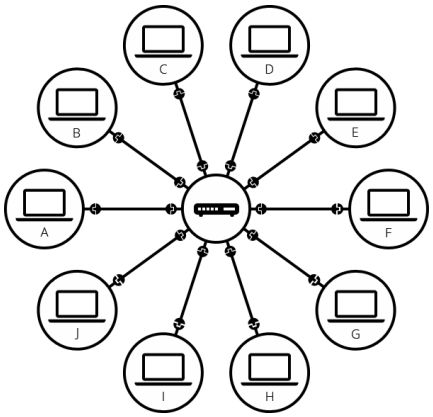
1.8. INTERNET

La red no se limita a dos ordenadores, se pueden conectar tantos como se deseen aunque siendo más complicado cada vez. Por ejemplo, para conectar diez ordenadores, se necesitarían 45 cables y unos nueve conectores por ordenador!



Para resolver este problema, cada ordenador en una red está conectado a una pequeña computadora especial llamada enrutador o router (en inglés). Este enrutador cumple una función: tal como hace un señalizador en una estación de tren, el router se encarga de asegurar que el mensaje enviado desde un ordenador emisor llegue al destino correcto. Para que el ordenador B reciba un mensaje desde el ordenador A, este debe enviarlo primero al router, quien a su vez lo remite al ordenador B asegurándose que dicho mensaje no sea entregado a otro ordenador C.

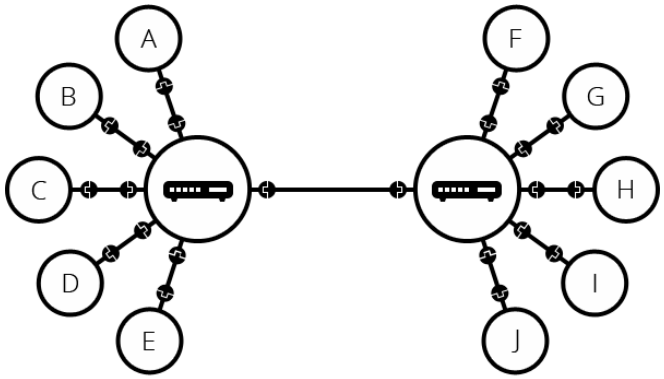
Una vez que agregamos un enrutador al sistema, nuestra red de 10 ordenadores solo requiere 10 cables: un enchufe para cada ordenador y un enrutador con 10 enchufes.



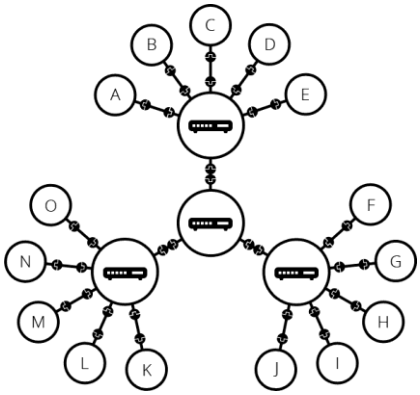
Una red de redes

Hasta aquí todo es más o menos simple, pero ¿qué sucede al conectar cientos, miles, millones de ordenadores entre sí?. Por supuesto un solo *enrutador* no puede escalar tanto, pero, si lees

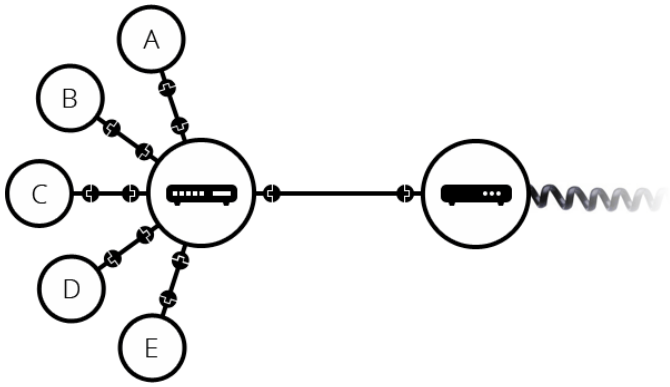
cuidadosamente, dijimos que un *enrutador* es como un pequeño ordenador, entonces ¿qué nos impide conectar dos *enrutadores* a la vez?. Nada: hagámoslo.



Conectando ordenadores a enrutadores y luego enrutadores entre sí, podemos escalar infinitamente.



Una red así se acerca mucho a lo que llamamos Internet, pero hay algo que nos falta. Construimos esa red para nuestros propios propósitos. Hay otras redes ahí fuera: tus amigos, tus vecinos, cualquiera puede tener su propia red de ordenadores. Pero no es posible instalar cables entre tu casa y el resto del mundo, así que ¿cómo puedes manejar esto? Bueno, ya hay cables conectados a tu casa, por ejemplo, la energía eléctrica y el teléfono. La infraestructura telefónica ya conecta tu casa con cualquier persona en el mundo, así que es el cable perfecto que necesitamos. Para conectar nuestra red a la infraestructura telefónica, necesitamos un equipo especial llamado *modem*. Este *modem* convierte la información de nuestra red en información manejable por la infraestructura telefónica y viceversa.

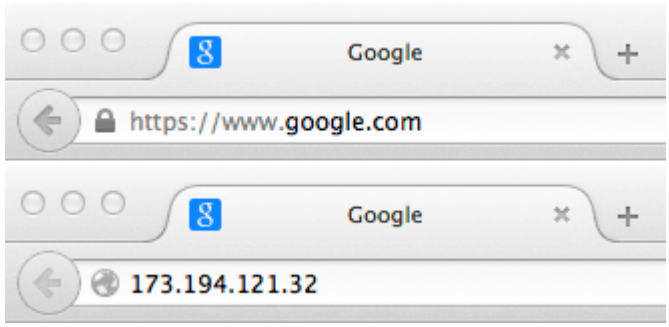


Entonces estamos conectados a la infraestructura telefónica. El siguiente paso es enviar el mensaje desde nuestra red a la red que queremos llegar. Para lograr eso, conectaremos nuestra red a un proveedor de servicios de internet (ISP de sus siglas en inglés Internet Service Provider). Un ISP es una empresa que gestiona algunos enrutadores especiales interconectados, que también pueden acceder a enrutadores de otros ISP. Así, el mensaje de nuestra red es llevada a través de la red de redes de ISP, hasta la red de destino. Internet consiste en toda esta infraestructura de redes.

Encontrando ordenadores

Si deseas enviar un mensaje a una computadora, debes especificar a cuál. Es por ello que todo ordenador conectado a una red cuenta con una dirección única que lo identifica, llamada "dirección IP" o Protocolo de Internet(IP de sus siglas en inglés *Internet Protocol*). Esta dirección está compuesta por una serie de cuatro números separados por puntos, por ejemplo: 192.168.2.10.

Para los ordenadores es un identificador simple, pero los humanos tienen mayor dificultad a la hora de recordar y memorizar este tipo de dirección. Con el propósito de convertir esta serie numérica en algo que podamos asociar con mayor facilidad a la dirección IP se utiliza lo que conocemos como *nombre de dominio*. Por ejemplo, google.com es el nombre de dominio utilizado para sustituir la dirección IP 173.194.121.32. Así, usar un nombre de dominio es la manera más fácil para nosotros de identificar un ordenador a través de Internet.



Internet y la web

Como puedes notar, cuando navegamos por la web con un navegador, normalmente utilizamos el nombre de dominio para llegar a un sitio web. ¿Significa eso que Internet y la Web son la misma cosa? No es tan simple. Como vimos, Internet es una infraestructura técnica que permite que miles de millones de ordenadores estén conectadas entre sí. Algunos de estos ordenadores, llamados *servidores web* son capaces de enviar mensajes inteligibles a los navegadores. Por tanto *Internet* es una infraestructura, mientras que la *Web* es un servicio construido sobre dicha infraestructura. Cabe señalar que existen otros servicios soportados por Internet, como es el correo electrónico e IRC.

1.9 MODELO CLIENTE SERVIDOR.

TCP es un protocolo orientado a conexión. No hay relaciones maestro/esclavo. Las aplicaciones, sin embargo, utilizan un modelo cliente/servidor en las comunicaciones.

Un servidor es una aplicación que ofrece un servicio a usuarios de Internet; un cliente es el que pide ese servicio. Una aplicación consta de una parte de servidor y una de cliente, que se pueden ejecutar en el mismo o en diferentes sistemas.

Los usuarios invocan la parte cliente de la aplicación, que construye una solicitud para ese servicio y se la envía al servidor de la aplicación que usa TCP/IP como transporte.

El servidor es un programa que recibe una solicitud, realiza el servicio requerido y devuelve los resultados en forma de una respuesta. Generalmente un servidor puede tratar múltiples peticiones (múltiples clientes) al mismo tiempo.

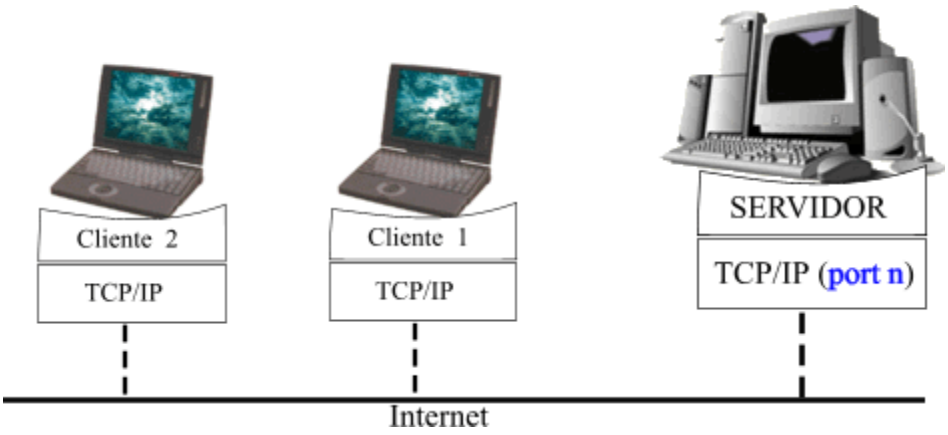


Figura: El modelo de aplicación cliente/servidor

Algunos servidores esperan las solicitudes en puertos bien conocidos de modo que sus clientes saben a qué zócalo IP deben dirigir sus peticiones. El cliente emplea un puerto arbitrario para comunicarse. Los clientes que se quieren comunicar con un servidor que no usa un puerto bien conocido tienen otro mecanismo para saber a qué puerto dirigirse. Este mecanismo podría usar un servicio de registro como Portmap, que utiliza un puerto bien conocido.

1.10 APACHE E IIS

Dos de los servidores web mas conocidos son internet information server y apache, por asi decirlo, cabe mencionar que nos son los unicos, pero si dos de los mas conocidos y comerciales, es por ello que debemos hablar de ellos, algo que debe quedar muy en claro es que ambos trabajan bajo el entorno cliente servidor, y el ejemplo mas claro de su funcionamiento, ya que una pagina web es alojada en el servidor (pc1) y diferentes equipos conectados en la red podras acceder a la pagina a traves del protocolo http:

IIS.

Internet Information Services/Server (IIS): son servicios para los ordenadores que funcionan con Windows.
Inicialmente formó parte del Option Pack para Windows NT. Luego se integra en otros sistemas

operativos de Microsoft destinados a brindar servicios, como Windows 2000 o Windows Server 2003. Windows XP Profesional incluye una versión limitada de Internet Information Server. Brinda servicios como FTP, SMTP, NNTP y HTTP/HTTPS. Además procesa páginas de ASP y ASP.NET; y puede incluir también PHP o Perl. Una de las desventajas que presenta este servidor web es que solo se puede utilizar en sistemas de Window

SEGURIDAD

- Autenticación de texto implícita avanzada: la autenticación de texto implícita avanzada es compacta, proporciona a los usuarios una autenticación segura y eficaz a través de servidores proxy y servidores de seguridad, no es necesario utilizar software cliente adicional e impide que se pasen a través de Internet el nombre de usuario y la contraseña en texto sin cifrar.
- Comunicaciones seguras: Secure Sockets Layer (SSL) 3.0 y Seguridad de capa de transporte (TLS) facilitan una forma segura para el intercambio de información entre clientes y servidores. Permiten además al servidor confirmar quien es el cliente antes de que el usuario inicie una sesión en el servidor.
- Cifrado canalizado por servidor (SGC): es una extensión de SSL que permite utilizar un cifrado de alto nivel de 128 bits a instituciones financieras con versiones de exportación de IIS.
- Asistentes para seguridad: los asistentes para seguridad reducen las tareas de administración del servidor.
- Restricciones de dominio de Internet e IP: permite asignar o rechazar accesos Web a equipos individuales, grupos de equipos o dominios enteros.
- Almacenamiento de certificados: facilita un único punto de entrada que le permite almacenar, realizar copias de seguridad y configurar certificados de servidor.

Capacidades De programación

- Páginas Active Server (ASP): provee una alternativa fácil de manejar a CGI e ISAPI que admite a los programadores de contenido incrustar cualquier lenguaje de secuencias de comandos o componente del servidor en las páginas HTML. ASP provee acceso a todas las peticiones HTTP y secuencias de respuesta, así como conectividad con bases de datos basada en estándares y la capacidad para personalizar el contenido para diferentes exploradores.

- Nuevas características de ASP: las páginas Active Server poseen nuevas y mejoradas características para incrementar el rendimiento y reducir las secuencias de comandos del servidor.
- Protección de aplicaciones: ofrece mayor protección e incrementa la confiabilidad de las aplicaciones Web. De forma predeterminada, IIS ejecutará todas las aplicaciones en un proceso común o agrupado que está separado de los procesos básicos de IIS.

APACHE

Apache es un software especializado en ofrecer servicios de servidor web. Es versátil, ligero y muy útil, además de ser completamente gratuito y de código abierto. Su popularidad es tal que, actualmente, cerca del 50% de las páginas web de todo el mundo se ejecutan en un servidor de este tipo.

Aunque se le conoce así, su nombre completo es Apache HTTP Server, y sus responsables tienen también un nombre similar: Apache Software Foundation. Esta es la firma responsable de todo el código que da forma a este software para servers que cualquiera puede utilizar sin necesidad de pagar, como también modificar a su total antojo al ser completamente abierto.

Lleva en activo desde el año 1995, tiempo más que suficiente para erigirse como el estándar que es en la actualidad. Fiable, robusto y muy flexible, permite al dueño de cualquier web publicar el contenido que desea en esta, como también gestionar todos sus ficheros de forma fácil y sencilla. Es una comunidad de usuarios la que sigue ofreciendo soporte para él y mejorándolo en todo lo posible. Actualmente se utiliza en plataformas Unix, Windows y Macintosh, de ahí que esté presente en la gran mayoría de páginas web de todo el mundo. Podemos hablar del software Apache como el nombre que aparece con más frecuencia en Internet, como también del responsable de que podamos entrar a la mayoría de webs en la que lo hacemos.

Pros:

- De código abierto y gratuito, incluso para uso comercial.
- Software confiable y estable.
- Parches de seguridad regulares y actualizados con frecuencia.
- Flexible debido a su estructura basada en módulos.
- Fácil de configurar para principiantes.
- Multiplataforma (funciona tanto en servidores Unix como en Windows).
- Viene listo para trabajar con sitios de WordPress.
- Enorme comunidad y soporte fácilmente disponible en caso de cualquier problema.

Contras:

- Problemas de rendimiento en sitios web con demasiado tráfico.
- Demasiadas opciones de configuración pueden generar vulnerabilidades de seguridad.

Ejercicios recomendados:

- ***Instalación de servidor apache***
- ***Crear una página html o php***
- ***Acceder a página web a través de una dirección ip***

Tiempo estimado dos horas, para demostrar el funcionamiento del entorno cliente servidor se recomienda la instalación de un servidor web apache, puede ser XAMP, APPserver o WAMP, con la finalidad de configurar una red local, actualmente se puede utilizar un teléfono celular como modem para darle IP a todos los equipos e intentar accesar a la página web seleccionada, dependiendo del nivel de los alumnos se deberá de programar una página php (HTML) la intención es acceder de manera remota al menos tres clientes a la vez a la misma página y partir de una explicación.

1.11 PÁGINAS WEB

Se conoce como página Web, página electrónica o página digital a un documento digital de carácter multimediático (es decir, capaz de incluir audio, video, texto y sus combinaciones), adaptado a los estándares de la World Wide Web (WWW) y a la que se puede acceder a través de un navegador Web y una conexión activa a Internet. Se trata del formato básico de contenidos en la red.

En Internet existen más de mil millones de páginas Web de diversa índole y diverso contenido, provenientes del mundo entero y en los principales idiomas hablados. Esto representa el principal archivo de información de la humanidad que existe actualmente, almacenado a lo largo de miles de servidores a lo largo del planeta, a los que es posible acceder velozmente gracias a un sistema de protocolos de comunicación (HTTP).

En muchos casos, el acceso a una página Web o a sus contenidos puntuales puede estar sometido a prohibiciones, pagos comerciales u otro tipo de métodos de identificación (como el registro on-line).

El contenido de esta inmensa biblioteca virtual no está del todo supervisado, además, y su regulación representa un reto y un debate para las instituciones tradicionales de la humanidad, como la familia, la escuela o incluso las leyes de los países.

Las páginas Web se encuentran programadas en un formato HTML o XHTML, y se caracterizan por su relación entre unas y otras a través de hipervínculos: enlaces hacia contenidos diversos que permiten una lectura compleja, simultánea y diversa, muy distinta a la que podemos hallar en los libros y revistas.

Por último, no es lo mismo hablar de página Web (Webpage) y de sitio Web (Website), ya que estos últimos contienen un número variable de las primeras.

Un navegador Web es un software de aplicación que sirve para abrir páginas Web tanto en una ruta local (como el disco rígido) o provenientes de la Internet.

Se les conoce como "navegadores" o "exploradores", a partir de la metáfora de que la Red es un lugar vasto y abarrotado, para el que se necesita de una plataforma.

En ese sentido, los navegadores Web nos permiten "entrar" a Internet y visualizar distintos contenidos a partir del ingreso de direcciones URL o del empleo de servicios online de búsqueda de datos (conocidos como Buscadores Web).

Ejemplo de php guardar como index.php o default.php.

```
<html>
 <head>
  <title>Prueba de PHP</title>
 </head>
 <body>
 <?php echo '<p>Hola Mundo</p>'; ?>
 </body>
</html>
```

1.12 MODELOS FUNDAMENTALES.

Interacción: el cómputo ocurre en los procesos; los procesos interaccionan por paso de mensajes, lo que deviene en comunicación y coordinación (sincronización y ordenamiento de actividades) entre procesos.

Fallo: la correcta operación de un sistema distribuido se ve amenazada allá donde aparezca un fallo en cualquiera de los computadores sobre el que se ejecuta (incluyendo fallos de software) o en la red que los conecta.

Seguridad: la naturaleza modular de los sistemas distribuidos y su extensibilidad los expone a ataques tanto de agentes externos como internos.

Modelo de Interacción

Prestaciones de los canales de comunicaciones: las características de prestaciones de la comunicación sobre una red de computadores incluyen la latencia, el ancho de banda y las fluctuaciones:

El retardo entre el envío de un mensaje por un proceso y su recepción por otro se denomina latencia.

El ancho de banda de una red de computadores es la cantidad total de información que puede transmitirse en un intervalo de tiempo dado.

La fluctuación (jitter) es la variación en el tiempo invertido en completar el reparto de una serie de mensajes.

Relojes de computadores y eventos de temporización: Cada computador de un sistema distribuido tiene su propio reloj interno; los procesos locales obtienen de él, el valor del tiempo actual.

Esta es la forma en que dos procesos en ejecución sobre dos computadores diferentes asocian marcas (o sellos) temporales a sus eventos. Sin embargo, incluso si dos procesos leen sus relojes a la vez, sus relojes locales proporcionarán valores de tiempo diferentes.

Dos variantes del modelo de interacción. En un sistema distribuido es difícil establecer cotas sobre el tiempo que debe tomar la ejecución de un proceso, el reparto de un mensaje o

la deriva del reloj. Tenemos dos modelos simples que parten de posiciones extremas y opuestas: el primero tiene en cuenta una fuerte restricción sobre el tiempo; en el segundo no se hace ninguna presuposición.

Sistemas distribuidos síncronos: Hadzilacos y Toueg [1994] definen un sistema distribuido síncrono como aquel en el que se establecen los siguientes límites:

El tiempo de ejecución de cada etapa de un proceso tiene ciertos límites inferior y superior conocidos.

Cada mensaje transmitido sobre un canal se recibe en un tiempo limitado conocido.

Cada proceso tiene un reloj local cuya tasa de deriva sobre el tiempo real tiene un límite conocido.

Sistemas distribuidos asíncronos: muchos sistemas distribuidos, por ejemplo, Internet, son de gran utilidad aun sin ser sistemas síncronos. En consecuencia, necesitamos un modelo alternativo: un sistema distribuido asíncrono es aquel en que no existen limitaciones sobre:

La velocidad de procesamiento (por ejemplo, un paso de un proceso puede requerir tan sólo un pico segundo y otro un siglo; lo único que podemos decir es que cada paso se tomará un tiempo arbitrariamente largo).

Los retardos de transmisión de mensaje (por ejemplo, al repartir un mensaje de un proceso A otro B puede tardar un tiempo cero o bien varios años. En otras palabras, un mensaje puede recibirse tras un tiempo arbitrariamente largo).

Las tasas de deriva de reloj (de nuevo, la tasa de deriva de reloj es arbitraria). El modelo asíncrono no presupone nada sobre los intervalos de tiempo involucrados en cualquier ejecución. Éste es exactamente el modelo de Internet, en él no hay límite intrínseco en la carga del servidor o la red (en consecuencia, no se sabe cuánto tomará, por ejemplo, transferir un archivo usando ftp). A veces un mensaje de correo puede tomarse varios días en llegar.

Ordenamiento de eventos. En muchos casos, nos interesa saber si un evento (enviar o recibir un mensaje) en un proceso ocurrió antes, después o concurrentemente con otro evento en algún otro proceso. La ejecución de un sistema puede describirse en términos de los eventos y su ordenación aun careciendo de relojes precisos.

Modelo de fallo.

Fallos por omisión. Las faltas clasificadas como fallos por omisión se refieren a casos en los que los procesos o los canales de comunicación no consiguen realizar acciones que se suponen que pueden hacer.

Fallos por omisión de procesos: El principal fallo por omisión de un proceso es el fracaso o ruptura accidentada del procesamiento (crash). Cuando decimos que un proceso se rompe queremos decir que ha parado y no ejecutará ningún paso de programa más**.**

Fallos por omisión de comunicaciones: El canal de comunicación produce un fallo de omisión si no transporta un mensaje desde el búfer de mensajes salientes de p al búfer de mensajes entrantes de q. A esto se denomina «perder mensajes» y su causa suele ser la falta de espacio en el búfer de recepción de alguna pasarela de entre medias, o por un error de red, detectable por una suma de chequeo de los datos del mensaje.

Fallos arbitrarios. Un fallo arbitrario en un proceso es aquel en el que se omiten pasos deseables para el procesamiento o se realizan pasos no intencionados de procesamiento. En consecuencia, los fallos arbitrarios en los procesos no pueden detectarse observando si el proceso responde a las invocaciones, dado que podría omitir arbitrariamente la respuesta.

Fallos de temporización. Los fallos de temporización se aplican en los sistemas distribuidos síncronos donde se establecen límites en el tiempo de ejecución de un proceso, en el tiempo de reparto de un mensaje y en la tasa de deriva de reloj.

Fiabilidad y comunicación uno a uno. El término comunicación fiable se define en términos de validez e integridad, definidos como sigue:

Validez: Cualquier mensaje en el búfer de mensajes salientes será hecho llegar eventualmente al búfer de mensajes entrantes.

Integridad: El mensaje recibido es idéntico al enviado, y no se reparten mensajes por duplicado.

UNIDAD II COMUNICACIONES EN LOS SISTEMAS DISTRIBUIDOS

2.1.- LAS REDES Y LOS SISTEMAS DISTRIBUIDOS.

Para entender los sistemas distribuidos hay que comentar que se trata de un sistema de –tolerancia a fallosII. ¿Qué queremos decir con esto? Pues que, al ser una única red, pero con muchas computadoras si alguno de los elementos falla, los otros podrán seguir realizando la función correctamente, por lo que los errores se complementan y evitan rápidamente. Por este motivo los sistemas distribuidos suelen otorgar bastante confianza a la hora de trabajar con ellos, ya que es muy raro que falle el sistema por completo.

También hay que tener en cuenta que esta confianza hace que el sistema sea muy seguro, puesto que las tareas no radican solo en un aparato, sino en varios equipos. Esto además facilita que se hagan varias copias de seguridad, existiendo normalmente una por cada ordenador. En ocasiones incluso estos dispositivos pueden trabajar con sistemas operativos diferentes, lo que no evita que siempre vayan a poder ofrecer a los usuarios los mismos servicios. Por este motivo todos los dispositivos que están conectados son compatibles entre ellos. Se evita así obtener errores a la hora de realizar las labores pertinentes y se consigue que el ambiente de trabajo sea mucho más cómodo, ya que todos pueden realizar sus tareas con los mismos servicios y programas. En este sentido el diseño del software es otro punto fundamental, puesto que este es también compatible con todos los usuarios y sistemas que se presentan en cada computadora.

Otra de las características principales es que los sistemas integrados ofrecen la posibilidad de la interacción entre todos los equipos, pudiendo conectarse el usuario desde cualquier ordenador a otros. Es mucho más rápido el acceso a la información, además de otorgar transparencia al sistema.

¿Cuántos sistemas distribuidos existen?

Aunque hemos comentado que los sistemas distribuidos se centran en una única red conectada a varias computadoras, en cuanto a la función de los mismos la red de mallas distribuidas puede clasificarse en dos: las computacionales y las de datos.

A través de ambas **se pueden compartir los recursos de un dispositivo a otro**, dependiendo de en qué se basen, es decir, si en la información en sí o en los dispositivos que la albergan. En este sentido el estado principal de esta clasificación es el conocido como XML. Son los llamados servicios web, a través de los cuales **podemos acceder a un gran número de aplicaciones** y programas dentro de una red de mallas, como las existentes en los sistemas distribuidos. Teniendo en cuenta todo esto cabe destacar que este tipo de sistemas se pusieron en funcionamiento a partir de 1970 con un objetivo principal: resolver grandes problemas de los supe sistemas informáticos y a la vez **poder seguir trabajando desde pequeños dispositivos**.

Hoy en día estos sistemas se diferencian bastante de los que se presentaron en un comienzo, aunque siguen siendo una herramienta de gran utilidad para organizar todos los recursos que las nuevas tecnologías de la información nos ofrecen. Por este motivo en la actualidad los sistemas distribuidos son los más utilizados en cuanto a las redes se refiere, teniendo sus redes varios tamaños, las locales, las metropolitanas y la más grande entre las grandes, Internet, que da soporte a millones de usuarios al día.

2.2.- FUNDAMENTOS DE REDES.

Red de datos

Una red de computadoras, también llamada red de telecomunicaciones, es un conjunto de equipos de informática y software que se encuentran conectados entre ellos de la mano de dispositivos de tipo físico que envían y reciben impulsos eléctricos u ondas constantemente, o

en todo caso cualquier otro medio para el transporte de datos, con la finalidad de compartir información, recursos informáticos y ofrecer servicios para el beneficio del usuario.

Finalidad de una red de datos

Es unir o conectar usuarios entre ciertas distancias, que pueden ser pequeñas o considerablemente grandes, dándoles así la posibilidad de realizar un intercambio de información preciso y confiable mediante una red que es común entre ellos, es decir, que conecta a dicho usuario con el otro. A través de éstas es posible el intercambio de información y de recursos importantes que son de uso común en ciertas áreas y lugares, como serían las impresoras y un disco duro en un área de oficina o en un edificio comercial.

Red de área local

Una red de área local (LAN, por sus siglas en inglés) es una red de computación que está diseñada para interconectar computadores en un área limitada, como sería un colegio, un hogar, un laboratorio de informática o un edificio de oficina usando medios de comunicación de redes. Las características que definen las redes de área local, en contraste con otras redes, incluyen sus usualmente altas tasas de transferencia de datos y que esas redes cubren áreas geográficas más pequeñas. Estas redes se pueden lograr con conexiones de red de tipo inalámbrica y cables de red trenzados, que son ideales para anular interferencias que perturban la experiencia de usuario.

Red de área amplia

Una red de área amplia (WAN, por sus siglas en inglés) es una red que cubre un espacio amplio, por ejemplo; cualquier red de telecomunicación que vincule metrópolis, conurbaciones o áreas de carácter regional o nacional haciendo uso de redes de transporte de datos, que pueden ser públicas o privadas. Los comercios, las empresas y las entidades públicas hacen uso de las redes

de área amplia para distribuir información importante entre sus trabajadores, clientes, compradores y proveedores de diversas ubicaciones geográficas. En esencia, este tipo de telecomunicación le permite a un comercio el llevar a cabo las funciones del día a día eficazmente independientemente de la ubicación, es decir, es posible para un empresario gestionar correctamente los datos relacionados a su aun cuando no se está ahí.

Red de igual a igual

Una red de igual a igual o peer-to-peer (P2P, por sus siglas en inglés) es una red informática en la que todos o algunos aspectos de ella funcionan sin clientes ni servidores de carácter fijos, sino una serie de puntos de intersección en los que coinciden distintos ordenadores y dispositivos en general que se interconectan en un mismo punto, llamándole a estas intersecciones nodos que han de comportarse como iguales entre sí. Dichos dispositivos actúan al mismo tiempo como servidores y clientes a los demás nodos de la red. Las redes de igual a igual permiten el intercambio directo de información en cualquier formato o extensión entre los ordenadores o dispositivos interconectados.

Redes cliente - servidor

La red cliente/servidor es la red de comunicaciones en la cual todos los clientes están conectados a un servidor, que puede ser cualquier computadora en el que se centralizan los diversos recursos y aplicaciones; y que los pone a disposición de los clientes cada vez que estos son solicitados. Es importante resaltar que los servidores cumplen un papel de proveedores, los cuales responden a las peticiones que realizan los clientes; quienes por su parte demandan los contenidos, recursos y servicios que poseen los servidores. Este tipo de red es muy utilizada en las empresas que manejan grandes cantidades de datos por varios factores, como serían el bajo costo; ya que se usa un solo ordenador para la distribución de la información y la facilidad de mantener los datos privados a salvo gracias a la centralización de los mismos por medio de un servidor.

Es un periférico que permite la comunicación de aparatos conectados entre sí, al igual que compartir recursos entre dos o más computadoras, discos duros, CD-ROM, impresoras o cualquier otro sistema, incluyendo la preparación y control de datos en la red. Las tarjetas de

red presentan configuraciones que pueden modificarse. Algunas de estas son: los interruptores de hardware (IRQ) la dirección de E/S y la dirección de memoria (DMA).

Protocolo de red

Un protocolo de red, se define como un conjunto de normas a seguir utilizadas para regular la comunicación entre distintos componentes existentes en una red de informática o red de ordenadores. Hay dos tipos de protocolos: de nivel bajo y protocolos de red.

Los protocolos de bajo nivel mantienen el control de las señales que se transmiten por el cable o por el medio físico. Los de red organizan la información para llevar a cabo su transmisión por el medio físico a través de los protocolos de nivel bajo.

Topología de red

Definida como una familia de comunicación que es usada por las computadoras que son parte de una red para intercambiar datos. Las topologías de red indican de qué manera están organizados los dispositivos de una red. Dichas topologías son arquitecturas lógicas, esto significa que señalan la dirección en la que las señales van entre los dispositivos que forman parte de la red, pero, los cables que han de conectar estos dispositivos pueden no estar conectados de la misma manera como la señalada por la topología, por ejemplo; las topologías de redes en bus y en anillo son comúnmente organizadas físicamente como una red en estrella.

Red en estrella

Un dispositivo que va en el centro de la "estrella" se conecta con otros dispositivos. La única manera en la que los dispositivos que se encuentran a los extremos de la estrella puedan comunicarse con otros de otro extremo es mediante el dispositivo que se encuentra en el medio. Un conmutador es un ejemplo común de una red en estrella, los computadores en la red deben de "pasar" por dicho dispositivo para hacer comunicación o intercambio de datos entre ellos. Como ventaja se puede mencionar un mejor desempeño a nivel del intercambio de datos y el aislamiento de dispositivos que previene las desconexiones de un dispositivo a otro. Entre sus desventajas destaca la alta dependencia del sistema de la red sobre el eje de la misma ya que una falla por parte de este se traduciría en una red inutilizable.

Red en estrella extendida

Es donde un hub o eje central se conecta con otros ejes que dependen de él. Se generan otros nodos que dependen del eje central de la red, que a su vez tienen otros dispositivos, es decir, son los centros de otras estrellas y operan como repetidoras.

Red en malla

En esta topología cualquier dispositivo puede realizar una comunicación con cualquier otro que forme parte de la red y no se creará interferencia alguna entre ellos. Un ejemplo bastante representativo de una red en malla es una red inalámbrica, donde los dispositivos que la conforman o están en ella usan la multiplexación o el uso de distintas frecuencias para evitar interferir entre sí. La ventaja que destaca en este tipo de red es la inexistencia de interferencia alguna entre los dispositivos presentes en la red, y su mayor desventaja es que mientras más extensa se quiera hacer este tipo de red hay que hacer una mayor inversión económica, más que todo debido a los precios de los enrutadores inalámbricos.

Ésta consiste en que un dispositivo se comunica con otros dos presentes en la red, y así todos los dispositivos que la forman se comunican en círculo. La información viaja de nodo a nodo, y cada uno de estos a lo largo del "anillo" maneja cada paquete de datos. Algunos círculos envían información en una sola dirección o sentido, mientras que existen otros que comunican en ambos sentidos, derecha e izquierda. Su ventaja principal es que no requiere un nodo central para manejar la información, pero este aspecto es también su mayor desventaja, ya que si uno de los dispositivos que forma parte de la red llegase a dañarse, la red entera se ve afectada por la imposibilidad de continuar el manejo de la información.

Red en bus

Esta topología significa que la señal es puesta en el medio y todos los dispositivos en el bus reciben dicha señal. Si más de un dispositivo intenta enviar una señal al mismo tiempo, pueden interferir entre ellos. Tiene entre sus ventajas la facilidad de implementación e instalación y como gran desventaja el límite de equipos dependiendo de la calidad de la señal de la red.

Medios de red

Cable coaxial

Es un cable utilizado para transportar señales eléctricas de alta frecuencia que posee dos conductores concéntricos, uno central, llamado vivo, encargado de llevar la información, y uno exterior, de aspecto tubular, llamado malla o blindaje, que sirve como referencia de tierra y retorno de las corrientes. Entre ambos se encuentra una capa aislante llamada dieléctrico, de cuyas características dependerá principalmente la calidad del cable. Todo el conjunto suele estar protegido por una cubierta aislante.

Consiste en dos alambres de cobre aislados que se trenzan de forma helicoidal, igual que una molécula de ADN. De esta forma el par trenzado constituye un circuito que puede transmitir datos. Esto se hace porque dos alambres paralelos constituyen una antena simple. Cuando se trenzan los alambres, las ondas de diferentes vueltas se cancelan, por lo que la radiación del cable es menos efectiva. Así la forma trenzada permite reducir la interferencia eléctrica tanto exterior como de pares cercanos. Un cable de par trenzado está formado por un grupo de pares trenzados, normalmente cuatro, recubiertos por un material aislante. Cada uno de estos pares se identifica mediante un color.

Cable STP

Twisted pair (STP) o par trenzado blindado: se trata de cables de cobre aislados dentro de una cubierta protectora, con un número específico de trenzas por pie. STP se refiere a la cantidad de aislamiento alrededor de un conjunto de cables y, por lo tanto, a su inmunidad al ruido. Se utiliza en redes de ordenadores como Ethernet o Token Ring. Es más caro que la versión sin blindaje y su impedancia es de 150 Ohmios.

Cable UTP

El cable de par trenzado no blindado (UTP, siglas de unshielded twisted pair) es un tipo de cable de par trenzado que se utiliza más que todo para las telecomunicaciones. Son muy utilizados para realizar las conexiones de telecomunicaciones en la actualidad tanto en interiores; como por ejemplo los cables Ethernet que se conectan del módem al computador como también en el exterior; por ejemplo, el extenso cableado telefónico en los postes.

La fibra óptica es un medio de transmisión empleado habitualmente en redes de datos; un hilo muy fino de material transparente, vidrio o materiales plásticos, por el que se envían pulsos de luz que representan los datos a transmitir. El haz de luz queda completamente confinado y se propaga por el interior de la fibra con un ángulo de reflexión por encima del ángulo límite de reflexión total, en función de la ley de Snell. La fuente de luz puede ser láser o un LED.

Medios de transmisión inalámbrica

Los medios inalámbricos transmiten y reciben señales electromagnéticas sin un conductor óptico o eléctrico, técnicamente, la atmósfera de tierra provee el camino físico de datos para la mayoría de las transmisiones inalámbricas, sin embargo, varias formas de ondas electromagnéticas se usan para transportar señales, las ondas electromagnéticas son comúnmente referidas como medio; dichos medios inalámbricos son los siguientes:

- Infrarrojo: se aplica al tipo de radiación que es emitida por una fuente de calor y no es visible por el ojo humano por tener una longitud de onda mayor que la que corresponde a la luz visible.
- Radiofrecuencias: cada una de las frecuencias de las ondas electromagnéticas empleadas en la radiocomunicación.
- Microondas: ondas electromagnéticas cuya longitud está comprendida en el intervalo del milímetro al metro y cuya propagación puede realizarse por el espacio y por el interior de tubos metálicos.

Medios de radiofrecuencia

- Infrarrojo.
- Banda angosta: transmite y recibe en una radiofrecuencia específica. Mantiene la frecuencia de la señal de radio tan angostamente posible para hacer posible el poder pasar la información.

Debe evitar que los canales se crucen, así que tiene que coordinar diferentes usuarios en diferentes canales de frecuencia para evitar los choques y las interferencias.

El radio receptor filtra todas aquellas frecuencias que no son de su competencia o que no debería manejar.

Usa una amplia gama de frecuencias, una para cada usuario, lo cual resulta bastante impráctico si se tienen muchos.

- Banda ancha: intercambia eficiencia y productividad eficaz en ancho de banda por confiabilidad, integridad y seguridad.
- Reduce la interferencia entre la señal procesada y otras señales que resultan ajenas al sistema que recibe la radiofrecuencia.

- Otras tecnologías:

- Bluetooth: es una tecnología que permite interconectar teléfonos móviles, agendas electrónicas, ordenadores, etc., ya sea en el hogar, la oficina o en el automóvil, con una conexión inalámbrica que consta de un corto alcance.

Internet por microondas

Una red por microondas es un tipo de red inalámbrica que utiliza microondas como medio de transmisión. El protocolo más frecuente es el IEEE 802.11b y transmite a 2.4 GHz, alcanzando velocidades de 11 Mbps (Megabits por segundo). Otras redes utilizan el rango de 5,4 a 5,7 GHz para el protocolo IEEE 802.11ª.

Router

Un router es un dispositivo hardware o software de interconexión de redes de computadoras que opera en la capa tres (nivel de red) del modelo OSI. Este dispositivo interconecta segmentos de red o redes enteras. Hace pasar paquetes de datos entre redes tomando como base la información de la capa de red.

Switch

Un conmutador o switch es un dispositivo digital lógico de interconexión de redes de computadoras que opera en la capa de enlace de datos del modelo OSI. Su función es interconectar dos o más segmentos de red, de manera similar a los puentes de red, pasando datos de un segmento a otro de acuerdo con la dirección MAC de destino de las tramas en la red.

Módem

Un módem es un dispositivo que sirve para enviar señales moduladoras mediante otra señal llamada portadora. Se usan distintos tipos de módems, principalmente para que la transmisión directa de las señales electrónicas inteligibles, a largas distancias, sean más eficaces. Es habitual encontrar en muchos módems de red conmutada la facilidad de respuesta y marcación automática, que les permiten conectarse cuando reciben una llamada de la RTPC (Red Telefónica Pública Conmutada) y así proceder a la marcación de cualquier número previamente grabado por el usuario. Gracias a estas funciones se pueden realizar automáticamente todas las operaciones de establecimiento de la comunicación.

Servidor

Un servidor es un nodo que forma parte de una red, provee servicios a otros nodos denominados clientes. Una aplicación informática o programa que realiza algunas tareas en beneficio de otras aplicaciones llamadas clientes. Algunos servicios habituales son los servicios de archivos, que permiten a los usuarios almacenar y acceder a los archivos de una computadora y los servicios de aplicaciones, que realizan tareas en beneficio directo del usuario final. Este es el significado original del término. Es posible que un ordenador cumpla simultáneamente las funciones de cliente y de servidor.

Firewall

Es una parte de un sistema de red diseñado para bloquear el acceso no autorizado, permitiendo a su vez comunicaciones autorizadas y evitando ataques de un pirata informático. Se trata de un dispositivo que limita, cifra, descifra, el tráfico entre los ámbitos de la base de un conjunto de normas y otros criterios.

Hub

Es un dispositivo utilizado para redes de área local que concentra computadoras y repite señales que recibe de sus distintos puertos. Al igual que es un servidor por capas de gestionar los recursos compartidos de una red. Es la base de las redes tipo estrella.

2.3.- LA CONMUTACIÓN DE CIRCUITOS VIRTUALES

Un circuito virtual (VC por sus siglas en inglés) es un sistema de comunicación por el cual los datos de un usuario origen pueden ser transmitidos a otro usuario destino a través de más de un circuito de comunicaciones real durante un cierto periodo de tiempo, pero en el que la conmutación es transparente para el usuario. Un ejemplo de protocolo de circuito virtual es el ampliamente utilizado TCP (Protocolo de Control de Transmisión).

Es una forma de comunicación mediante conmutación de paquetes en la cual la información o datos son empaquetados en bloques que tienen un tamaño variable a los que se les denomina paquetes. El tamaño de los bloques lo estipula la red.

Los paquetes suelen incluir cabeceras con información de control. Estos se transmiten a la red, la cual se encarga de su encaminamiento hasta el destino final. Cuando un paquete se encuentra con un nodo intermedio, el nodo almacena temporalmente la información y encamina los paquetes a otro nodo según las cabeceras de control.

Es importante saber que en este caso los nodos no necesitan tomar decisiones de encaminamiento, ya que la dirección a seguir viene especificada en el propio paquete.

Las dos formas de encamación de paquetes son:

- Datagramas y
- Circuitos Virtuales.

Otras características en:

>**Conmutación de circuitos:**

>- Servicio transparente y velocidad constante.

- Sufren retardo debido al establecimiento de llamada.

- Los datos Analógicos o Digitales van desde el origen hasta el destino.

Conmutación de Paquetes:

>- Los datos deben ser convertidos de Analógicos a Digitales por medio de un circuito virtual antes de la transmisión.

- Tienen bits suplementarios relativos.

- Existe retardo previo a la transmisión.

> **Datagramas**:

>- Su llegada es en orden diferente.

- No existe establecimiento de llamada(son rápidos para mensajes cortos).

2.4. ATM

Modo de Transferencia Asíncrona

Las nuevas necesidades de comunicaciones aparecidas en la década de los 80 orientaron las comunicaciones hacia la conmutación de paquetes en alta velocidad para contar simultáneamente con las ventajas de las redes de circuitos y las redes de paquetes. La nueva tecnología debería ser capaz de proporcionar anchos de banda variables, ser transparente a los protocolos utilizados y soportar una gama amplia de servicios con soluciones específicas de velocidad, sincronización

y latencia. Con estas especificaciones aparecieron dos tecnologías de acceso en la interface usuario/red: Frame Relay y Cell Relay, la primera para transmitir datos especialmente y la segunda para transmitir cualquier tipo de tráfico. Las dos reclaman para sí lo mejor de ambos mundos, esto es la predictibilidad de las redes de circuitos y la flexibilidad de las redes de paquetes.

Frame Relay (FRL)

Envía unos paquetes de tamaño variable, de 4Kbs a 8Kbs, denominados tramas. Esta tecnología garantiza un uso eficiente del ancho de banda disponible y es apta para transmitir datos o imágenes estáticas. Sin embargo, resulta inapropiado para datos asincrónicos debido a que el tamaño grande y variable de sus tramas no permite garantizar un retardo de entrega constante. El frame relay se presenta como un sólido interfaz de usuario al optimizar los recursos disponibles, aunque no debería contemplarse como una arquitectura de red.

Cell Relay (ATM)

Envía unos paquetes de 53 bytes denominados células. El pequeño tamaño de los paquetes garantiza un mínimo retardo, aunque supone un incremento del overhead: cuanto más pequeño es el paquete, más proporción hay de cabeceras y más pérdida de ancho de banda. Las ventajas obtenidas es una baja latencia que permite transportar datos isocrónicos y una eficiente conmutación hardware gracias al tamaño constante de los paquetes.

ATM: Una Introducción

Breve historia de ATM

La primera referencia del ATM (Asynchronous Transfer Mode) tiene lugar en los años 60 cuando un norteamericano de origen oriental perteneciente a los laboratorios Bell describió y patentó un modo de transferencia no síncrono. Sin embargo, el ATM no se hizo popular hasta

1988 cuando el CCITT decidió que sería la tecnología de conmutación de las futuras red ISDN en banda ancha. En aquella histórica fecha los valedores del ATM tuvieron primero que persuadir a algunos representantes de las redes de comunicaciones que hubieran preferido una simple ampliación de las capacidades de la ISDN en banda estrecha. Conseguido este primer objetivo y desechando los esquemas de transmisión síncronos, se empezaron a discutir aspectos tales como el tamaño de las células. Por un lado, los representantes de EEUU y algún otro país proponían un

tamaño de células grande de unos 128 bytes: `cuanto mayor es el tamaño de las células menor es el overhead parámetro muy importante cuando se desean transmitir datos' era su argumento. Sin embargo, los representantes de los países europeos el tamaño ideal de las células era de 16 bytes, y señalaron que un tamaño de célula de 128 bytes provocaría retardos inaceptables de hasta 85 ms. Este retardo no permitiría la transmisión de voz con cierto nivel de calidad a la vez que obligaba a instalar canceladores de eco.

Después de muchas discusiones ambas partes habían hecho una concesión: el lobby norteamericano proponía 64 bytes y el lobby europeo 32 bytes que básicamente coincidían con los representantes de las redes de datos y las redes de voz respectivamente. Ante la falta de acuerdo en la reunión del CCITT celebrada en Ginebra en junio de 1989 se tomó una decisión salomónica: –Ni para unos ni para otros. 48 bytes será el tamaño de la célula**II**. Para la cabecera hubo posicionamientos similares, y el definitivo tamaño de 5 bytes también fue un compromiso.

Un extraño número primo 53 (48+5) sería el tamaño definitivo, en octetos, de las células ATM. Un número que tuvo la virtud de no satisfacer a nadie pero que suponía un compromiso de todos grupos de interés y evitaba una ruptura de consecuencias imprevisibles.

Arquitectura de un nodo ATM

El ATM puede ser considerado como una tecnología de conmutación de paquetes en alta velocidad con unas características particulares:

- Los paquetes son de pequeño y constante tamaño (53 bytes).
- Es una tecnología de naturaleza conmutada y orientada a la conexión.
- Los nodos que componen la red no tienen mecanismos para el control de errores o control de flujo.

El header de las células tiene una funcionalidad limitada.

Simplificando al máximo podemos ver que una red ATM está compuesta por nodos de conmutación, elementos de transmisión y equipos terminales de usuarios. Los nodos son capaces de encaminar la información empaquetada en células a través de unos caminos conocidos como Conexiones de Canal Virtual. El routing, en los nodos conmutadores de células, es un proceso hardware mientras que el establecimiento de conexiones y el empaquetamiento/ desempaquetamiento de las células son procesos software.

La Capa de Adaptación a ATM adapta y segmenta el flujo de tráfico en celdas de 48 bytes. La capa ATM añade los 5 bytes de overhead, y los pasa a la Capa Física, que convierte las celdas en señales eléctricas u ópticas.

Jerarquía de transmisión

Bajo un punto de vista basado exclusivamente en la transmisión, el ATM se puede dividir en tres niveles que se combinan de forma jerárquica de modo que cada capa superior puede tener uno o varios de los elementos inferiores.

Canal Virtual (VC)

Así llamada a la conexión unidireccional entre usuarios. Importante resaltar la unidireccionalidad: si dos usuarios quisieran estar conectados en Full Duplex deberán utilizar dos canales. Los VC, además de transportar datos entre usuarios, también son utilizados para transportar la señalización y la gestión de la red.

Trayecto Virtual (VP)

Se entiende al conjunto de canales virtuales que atraviesan multiplexadamente un tramo de la red ATM. Los VP facilitan la conmutación de los canales virtuales, pues conectan tramos enteros de la red ATM. De no existir por cada conexión entre usuarios obligaría a reelaborar todas las tablas de routing de los nodos atravesados lo cual supondría un incremento del tiempo necesario para establecer una conexión.

Sección Física (PS)

Conecta y proporciona continuidad digital entre los diferentes elementos que componen la red controlando el flujo de bits. Debe mantener en óptimas condiciones las señales físicas, eléctricas u ópticas regenerándolas cuando resultan afectadas por atenuaciones, ruido o distorsiones.

Modelo de referencia ATM

Bajo una perspectiva arquitectónica el ATM se divide en tres niveles que ocupan las capas 1 y parte de la 2 del modelo de referencia OSI:

Nivel de adaptación ATM (AAL)

Se encarga de las relaciones con el mundo externo. Acepta todo tipo de información heterogénea y la segmenta en paquetes de 48 bytes a la velocidad que fue generada por los usuarios. Sólo se encuentra en los puntos terminales de la red. Según el modelo OSI maneja, en el nivel 2, las conexiones entre la red ATM y los recursos no ATM pertenecientes a los usuarios finales.

Nivel Modo de Transferencia Asíncrona (ATM)

Encargado de construir las cabeceras de las células ATM, responsable del routing y el multiplexado de las células a través de los Canales y Rutas Virtuales. También es misión suya el control del flujo de datos y la detección de errores ocurridos en la cabecera, aunque no en los datos.

Nivel físico (PL)

Es el nivel inferior encargado de controlar las señales físicas, ya sean ópticas o eléctricas, e independizarlas de los niveles superiores de protocolo adaptándolas al medio de transmisión y codificación utilizado. Puede soportar diversas configuraciones punto-a-punto y punto-a-multipunto. En una red ATM se distinguen dos tipos de nodos: los terminales que proporcionan los puntos de acceso a los usuarios finales y los nodos de conmutación responsables dentro de la red del routing de las células.

Nivel de Adaptación ATM (AAL)

Responsable de las relaciones con el mundo externo, por esta razón el nivel AAL sólo se encuentra en los nodos terminales de la red. Su misión es la de aceptar la información adaptando los niveles superiores de comunicación noATM a los formatos ATM. Son funciones del nivel AAL:

- adaptación a la velocidad de los usuarios,
- segmentación de los datos en células de 48 bytes (sin cabecera ATM)
- detección células erróneas y perdidas,
- mantenimiento del sincronismo entre terminales.

El Nivel de Adaptación ATM adapta cada tráfico a su velocidad inicial, segmenta/reensambla la información en trozos de 48 bits, detecta celdas erróneas o perdidas, y mantiene el sincronismo entre los usuarios conectados.

Estructura de la Capa AAL

Internamente el AAL se divide en dos partes:

1.El subnivel de Convergencia (CS)

Es capa más externa y ejecuta funciones como la detección y demultiplexión de datos, detección de células perdidas y mantenimiento del sincronismo de la conexión.

2.El subnivel Segmentación y Reensamblado (SAR)

Esta capa segmenta los datos en células y las envía al nivel ATM para que les ponga la cabecera. El proceso inverso se verifica al lado opuesto cuando recibe células y reconstruye la información original.

Calidad de servicio (QoS).

La información que llega a un nodo terminal ATM es captada, segmentada y dispuesta en células con las cabeceras adecuadas para cada tipo de tráfico. Este servicio proporcionado por el nivel AAL se denomina QoS que queda definido por tres parámetros:

Caudal, define el volumen de información que puede ser enviada en un período de tiempo. Si el tráfico es constante, el parámetro es único: velocidad pico; pero si el tráfico es a ráfagas, está expresado por tres parámetros de conexión: velocidad pico, velocidad media y duración de la ráfaga. retardo, definido por su media y su varianza que relaciona el retardo global medio de toda la transmisión y la variación entre los retardos individuales que afectan a cada célula. nivel de seguridad, se refiere a la tolerancia de un determinado tipo de tráfico a la pérdida de células que puede ocurrir durante períodos de congestión.

La Capa ATM

Este nivel es el auténtico núcleo sobre el que se vértebra la tecnología del cell relay. Sus funciones, fundamentales y comunes a cualquier nodo, se encargan de la manipulación de células ejecutándose los siguientes procesos:

1. construcción/extracción de cabeceras
2. routing entre los nodos
3. multiplexión y de multiplexión de células

Formato de las Células ATM

Son estructuras de datos de 53 bytes compuestas por dos campos principales:

- 1.- Header, sus 5 bytes tienen tres funciones principales: identificación del canal, información para la detección de errores y si la célula es o no utilizada. Eventualmente puede contener también corrección de errores, número de secuencia...

- 2.- Payload, tiene 48 bytes fundamentalmente con datos del usuario y protocolos AAL que también son considerados como datos del usuario.

Dos de los conceptos más significativos del ATM, Canales Virtuales y Rutas Virtuales, están materializados en dos identificadores en el header de cada célula (VCI y VPI) ambos determinan el routing entre nodos. Existen dos formatos de células: la UNI (User Network Interface) utilizado en el interfaz red/usuario y la NNI Network Interface) cuando circulan por la red.

Conexiones y Routing

Los conmutadores de VP modifican los identificadores VPI para redirigir las rutas de entrada hacia una salida específica. Un conmutador de VP no analiza ni modifica el campo VCI, ya que al operar en un nivel inferior conmuta todos los Canales asociados a dicha Ruta. Los conmutadores de VC aplican un mayor nivel de complejidad ya que manejan atributos como nivel de errores, calidad servicio, ancho de banda o servicios relacionados con la tarificación. Las tablas de routing de cada nodo pueden estar ya predefinidas, o bien deben construirse dinámicamente en el tiempo del establecimiento de las conexiones realizadas mediante el protocolo Q.2931 similar al Q.931 utilizado en el ISDN para banda estrecha.

Una Ruta Virtual puede ser Permanente (PVP) o Conmutada (SVP). Si es conmutada, es decir si se ha establecido explícitamente para una comunicación, todos sus Canales Virtuales (VC) asociados son dirigidos a través de ese camino y no será necesario conmutarlos. Si el VP es permanente es probable que sólo conecte troncales de la red por lo que los VC deberán ser conmutadas en algún nodo de la red. El routing de Canales y Rutas Virtuales es realizado mediante etiquetas, nunca con direcciones explícitas. Por ejemplo, un nodo de conmutación debe leer el identificador VCI = i de cada célula que entra por el puerto K y de acuerdo con su tabla de routing, la envía por el puerto Q modificando el header al escribir VCI = j.

El nivel físico

El nivel físico realiza dos funciones fundamentales: el transporte de células válidas y la entrega de la información de sincronismo

Estructura del nivel Físico Se divide en dos

capas: El subnivel Convergencia de la

Transmisión (TC)

Encargado de adaptar la velocidad y de crear el datastream para su posterior transmisión al medio físico. El proceso inverso se realiza en el otro extremo de la red donde el TC destino debe extraer las células del datastream recibido, comprobar su corrección y entregarlas finalmente al nivel superior ATM. Las células incorrectas o vacías se desechan.

El subnivel Medio Físico (PM)

Es el encargado de la transmisión de bits y de la sincronización de
señales. Dos velocidades estandarizadas por el ITU son 155,52 Mbit/s y
622,08
Mbit/s; mientras que el ATM Forum ha estandarizado interfaces
con velocidades a 25 Mbit/s, 44,736 Mbit/s, 100 Mbit/s y 155,52
Mbit/s.

Datastream del medio de transmisión

El servicio portador de la red encargado de transportar la información hasta los usuarios puede ser de dos modelos:

1. Basado en células, es la forma nativa utilizado en redes locales. Consiste en la transmisión directa de la secuencia de células ATM sobre el medio de transmisión que puede ser fibra y cable de diversas categorías. Dependiendo del estándar utilizado deben ser insertadas señales de delineación, sincronismo de las células.

2. Basados en tramas plesiócronas o PDH, las células se agrupan en una trama plesiócrona que incluye funciones de mantenimiento. El estándar utilizado se deriva del IEEE 802.6 utilizado por el DQDB en redes metropolitanas.

3. Basados en tramas síncronas o SDH, en este caso las células son empaquetadas en frames síncronos denominados STM transmitidos a velocidades ópticas múltiplos de 155,52 Mbit/s. Estas estructuras transportan también información de sincronismo y el overhead necesario para el transporte. La ventaja de los frames STM es que ofrecen un mecanismo estandarizado para realizar la multiplexión de los canales a medida que los enlaces aumentan o disminuyen su capacidad de transporte.

El ITU-T seleccionó la SDH como una de las bases para el B-ISDN para el transporte y multiplexión de señales a través de una red óptica. Es importante señalar que el SDH no es en sí mismo una red de comunicaciones, ni forma parte del ATM, sino el más bajo nivel de transporte de la red también utilizable por otras redes de transmisión como Frame Relay o SMDS.

B-ISDN Y ATM

Existe cierta confusión entre ATM y B-ISDN, y a menudo se usa incorrectamente un término por el otro. La diferencia es clara:

1.- El ATM es una tecnología para la conmutación de células en alta velocidad utilizable en múltiples entornos, LAN, MAN y WAN.

2.- El B-ISDN es una red de área extensa (WAN) que utiliza el N-ISDN como modelo de referencia y señalización; el ATM como tecnología de conmutación y el SDH como estándar de transporte dentro de la red.

Es decir, otros tipos de redes como por ejemplo una LAN puede también utilizar la tecnología ATM pero no han de utilizar necesariamente ni el SDH y ni el modelo de referencia ISDN.

2.5. METRO ETHERNET.

Una **Red Metro Ethernet**, es una arquitectura tecnológica destinada a suministrar servicios de conectividad de datos en una Red de área metropolitana (MAN) de capa 2 en el modelo OSI, a través de interfaces (UNIs) Ethernet. Estas redes denominadas "multiservicio", soportan una amplia gama de servicios, aplicaciones, y cuentan con mecanismos donde se incluye soporte a tráfico "RTP" (tiempo real), para aplicaciones como Telefonía IP y Video IP, aun cuando este tipo de tráfico es especialmente sensible al retardo y al jitter(Fluctuación).

Las redes Metro Ethernet pueden utilizar líneas de cobre (MAN BUCLE), lo que garantiza la posibilidad de despliegue en cualquier punto del casco urbano, soportando el 100% de los servicios demandados por los proyectos de Smart City.

Las redes Metro Ethernet suelen utilizar principalmente medios de transmisión guiados, como son el cobre (MAN BUCLE) y la fibra óptica, existiendo también soluciones de radio licenciada, los caudales proporcionados son de 10 Mbit/s, 20 Mbit/s, 34 Mbit/s, 100 Mbit/s, 1 Gbit/s y 10 Gbit/s.

La tecnología de agregación de múltiples pares de cobre, (MAN BUCLE), permite la entrega de entre 10 Mbit/s, 20 Mbit/s, 34 Mbit/s y 100 Mbit/s, mediante la transmisión simultánea de múltiples líneas de cobre, además esta técnica cuenta con muy alta disponibilidad ya que es casi imposible la rotura de todas las líneas de cobre y en caso de rotura parcial el enlace sigue transmitiendo y reduce el ancho de banda de forma proporcional.

La fibra óptica y el cobre, se complementan de forma ideal en el ámbito metropolitano, ofreciendo cobertura total a cualquier servicio a desplegar.

Los beneficios que Metro Ethernet ofrece son:

- Presencia y capilaridad prácticamente "universal" en el ámbito metropolitano. En especial gracias a la disponibilidad de las líneas de cobre, con cobertura universal en el ámbito urbano.
- Muy alta fiabilidad, ya que los enlaces de cobre certificados Metro Ethernet, están constituidos por múltiples pares de en líneas de cobre (MAN BUCLE) y los enlaces de Fibra Óptica, se configuran mediante Spanning tree (activo-pasivo) o LACP (caudal Agregado).

• Fácil uso: Interconectando con Ethernet se simplifica las operaciones de red, administración, manejo y actualización.

• Economía: los servicios Ethernet reducen el capital de suscripción y operación de tres formas:

• Amplio uso: Se emplean interfaces Ethernet que son la más difundidas para las soluciones de Networking

• Bajo costo: Los servicios Ethernet ofrecen un bajo costo en la administración, operación y funcionamiento de la red.

• Ancho de banda: Los servicios Ethernet permiten a los usuarios acceder a conexiones de banda ancha a menor costo.

• Flexibilidad: Las redes de conectividad mediante Ethernet permiten modificar y manipular de una manera más dinámica, versátil y eficiente, el ancho de banda y la cantidad de usuarios en corto tiempo.

El modelo básico de los servicios Metro Ethernet, está compuesto por una Red switcheada MEN (Metro Ethernet Network), ofrecida por el proveedor de servicios; los usuarios acceden a la red mediante CEs (Customer Equipment), CE puede ser un router; Bridge IEEE 802.1Q (switch) que se conectan a través de UNIs (User Network Interface) a velocidades de 10 Mbit/s, 20 Mbit/s, 34 Mbit/s, 100 Mbit/s, 1 Gbit/s y 10 Gbit/s.

Los organismos de estandarización (IEEE, IETF, ITU) y los acuerdos entre fabricantes, están jugando un papel determinante en su evolución. Incluso se ha creado el MEF (Metro Ethernet Forum), organismo dedicado únicamente a definir Ethernet como servicio metropolitano.

Para Metro Ethernet se tienen en cuenta los siguientes parámetros:

• **CIR** (Committed Information Rate): es la cantidad promedio de información que se ha transmitido, teniendo en cuenta los retardos, pérdidas, etc.

• **CBS** (Committed Burst Size): es el tamaño de la información utilizado para obtener el CIR respectivo.

• **EIR** (Excess Information Rate): especifica la cantidad de información mayor o igual que el CIR, hasta el cual las tramas son transmitidas sin pérdidas.

• **EBS** (Excess Burst Size): es el tamaño de información que se necesita para obtener el EIR determinado.

2.6 COMUNICACIÓN ENTRE PROCESOS.

La comunicación entre procesos (comúnmente IPC, del inglés *Inter-Process Communication*) es una función básica de los sistemas operativos. Los procesos pueden comunicarse entre sí a través de compartir espacios de memoria, ya sean variables compartidas o buffers, o a través de las herramientas provistas por las rutinas de IPC. La IPC provee un mecanismo que permite a los procesos comunicarse y sincronizarse entre sí, normalmente a través de un sistema de bajo nivel de paso de mensajes que ofrece la redsubyacente.

La comunicación se establece siguiendo una serie de reglas (protocolos de comunicación). Los protocolos desarrollados para internet son los mayormente usados: IP (capa de red), protocolo de control de transmisión (capa de transporte) y protocolo de transferencia de archivos, protocolo de transferencia de hipertexto (capa de aplicación).

Los procesos pueden estar ejecutándose en una o más computadoras conectadas a una red. Las técnicas de IPC están divididas dentro de métodos para: paso de mensajes, sincronización, memoria compartida y llamadas de procedimientos remotos (RPC). El método de IPC usado puede variar dependiendo del ancho de banda y latencia (el tiempo desde el pedido de información y el comienzo del envío de la misma) de la comunicación entre procesos, y del tipo de datos que están siendo comunicados.

La comunicación puede ser:

- Síncrona o asíncrona
- Persistente (persistent) o momentánea (transient)
- Directa o indirecta
- Simétrica o asimétrica
- Con uso de buffers explícito o automático
- Envío por copia del mensaje o por referencia
- Mensajes de tamaño fijo o variable

Síncrona

Quien envía permanece bloqueado esperando a que llegue una respuesta del receptor antes de realizar cualquier otro ejercicio.

Asíncrona

Quien envía continúa con su ejecución inmediatamente después de enviar el mensaje al receptor.

Persistente

El receptor no tiene que estar operativo al mismo tiempo que se realiza la comunicación, el mensaje se almacena tanto tiempo como sea necesario para poder ser entregado (Ej.: e-Mail).

Momentánea (transient)

El mensaje se descarta si el receptor no está operativo al tiempo que se realiza la comunicación. Por lo tanto no será entregado.

Directa

Los primitivos enviar y recibir explicitan el nombre del proceso con el que se comunican. Ejemplo enviar (*mensaje, A*) envía un *mensaje* al proceso *A*
Es decir se debe especificar cual va a ser el proceso fuente y cual va a ser el proceso Destino. Las operaciones básicas *Send* y *Receive* se definen de la siguiente manera: Send (P, mensaje);

envía un mensaje al proceso P (P es el proceso destino). Receive (Q, mensaje); espera la recepción de un mensaje por parte del proceso Q (Q es el proceso fuente).

Nota: Receive puede esperar de un proceso cualquiera, un mensaje, pero el Send sí debe especificar a quién va dirigido y cuál es el mensaje.

Indirecta

La comunicación Indirecta: Es aquella donde la comunicación está basada en una herramienta o instrumento ya que el emisor y el receptor están a distancia.

Simétrica

Todos los procesos pueden enviar o recibir. También llamada bidireccional para el caso de dos procesos. Es una comunicación equilibrada donde tanto emisor como receptor reciben la misma información.

Asimétrica

Un proceso puede enviar, los demás procesos solo reciben. También llamada unidireccional. Suele usarse para hospedar servidores en Internet.

Uso de buffers automático

El transmisor se bloquea hasta que el receptor recibe el mensaje (capacidad cero).

2.7. API PARA LOS PROTOCOLOS DE INTERNET.

Una API es una llave de acceso a funciones que podemos utilizar de un servicio web provisto por un tercero, dentro de una página web empresarial, de manera segura y confiable.

API significa Interfaz de Programación de Aplicaciones, y su definición formal le da poca información útil a alguien que no entiende mucho de informática. Una API es una –llave de acceso‖ a funciones que nos permiten hacer uso de un servicio web provisto por un tercero, dentro de una aplicación web propia, de manera segura.

Ejemplos de APIs

1. **Google Maps** a través de su acceso a –API‖ nos permite ponerle datos e información útil sobre sus mapas, y presentarlos con ciertas búsquedas o funciones personalizadas, desde nuestra propia aplicación.

2. **Twitter** ha permitido el desarrollo de un gran número de sistemas alternativos y servicios web que operan a través de su API.

3. **Facebook Connect** cede a través del API ciertos datos para registrar automáticamente usuarios en otros sitios web, dándoles la posibilidad de registrarse y loguearse con sus propias cuentas de Facebook.

4. **Paypal** con su –API ll nos permite hacer operaciones de pagos electrónicos usando nuestro propio sistema web, sin necesidad de acceder/operar en la web de Paypal

Por ejemplo en algunos foros se nos permiten interactuar usando nuestras credenciales de Facebook – en este caso el desarrollador del foro estudió la API de dicha red social, e implementó esos protocolos para que la identidad del usuario pueda ser utilizada también en dicho foro.

Sin embargo, es importante tener en cuenta que **si Facebook falla ("se cae")**, esta API tampoco funcionará, inhabilitando el login a través de ésta.

Entonces ¿Qué es una API? es una **interfaz para dar un acceso limitado a la base de datos de un servicio web**, evitando que se conozca o acceda al propio código fuente de la aplicación original.

Sobre la seguridad de las API

De inmediato al leer esto viene a la mente el tema de la seguridad, y esto es fundamental en una API. En general, las API sólo permiten un limitado campo de acción, tomándose las previsiones para que no pueda manipularse información confidencial de la empresa para otros fines.

Entonces, ¿para qué perder el tiempo reinventando la rueda cuando una API le permite usar código que otros desarrolladores han probado con éxito?

Es mejor usar esas funciones y concentrarte en su aplicación y sus características, lo importante es asegurarse de entregar un producto de calidad.

2.8. REPRESENTACIÓN EXTERNA DE DATOS Y EMPAQUETADO.

En el nivel físico de una arquitectura de red, los datos se transmiten como señales analógicas, las cuales representan un flujo binario. En el nivel de aplicación, se necesita una representación más compleja de los datos transmitidos con el objeto de dar soporte a la representación de tipos de datos y estructuras proporcionadas por los lenguajes de programación, tales como cadenas de caracteres, enteros, valores en coma flotante, vectores, registros y objetos.

Cuando ordenadores heterogéneos participan en una comunicación entre procesos, no basta con transmitir los valores de los datos o las estructuras usando flujos de bits en crudo.

Existen diferentes problemas asociados a ese intercambio de información entre ordenadores o procesos heterogéneos:

Cada máquina representa los tipos de datos básicos de formas diferentes:

little-endian, big-endian

UNIX-char=1byte,UNICODE=2bytes

Los tipos de datos compuestos se organizan de forma diferente según el lenguaje, el compilador y la arquitectura:

- tipos estructurados
- tipos dinámicos (con referencias)

Posibles soluciones a este problema de intercambio de información serían las siguientes:

Que sea la máquina o proceso emisor el encargado de transformar los datos en información reconocible por la máquina o proceso receptor, antes de invocar a la operación *enviar*.

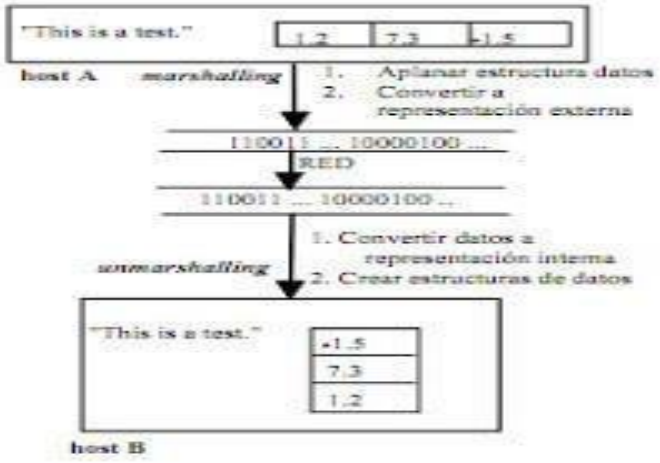
Que se la máquina o proceso receptor el encargado de transformar ese flujo de datos recibido en información útil, tras recibir la señal *recibir*.

Una tercera opción sería cuando intercambien datos lo hagan en una representación externa: los datos se empaquetan y se van a enviar transformados a esa representación y los datos recibidos se van a interpretar con esa representación y se van a traducir a la representación nativa.

El término de empaquetamiento de datos (*data marshaling*) se usa en el contexto de los mecanismos de comunicación entre procesos para referirse a las transformaciones necesarias para transmitir valores de datos o estructuras. El empaquetamiento de datos se necesita para todos los mecanismos de comunicación e incluye los pasos necesarios para acondicionar los datos a ser transmitidos:

Serialización de las estructuras de datos, Conversión de los valores de datos a las representaciones externas.

La serialización de objetos consiste en el empaquetado de objetos, ya que implica un mayor reto que el del resto de estructuras.



2.9 CODIFICACIÓN DE DATOS

Aunque determinados programas especializados puedan escribirse para realizar la comunicación entre procesos usando un esquema de representación determinado de mutuo acuerdo, las aplicaciones distribuidas de propósito general necesitan un esquema universal, e independiente de plataforma, para codificar el intercambio de datos. Por esto se han definido estándares de red para la codificación de datos:

MIME, prácticamente todos los mensajes de correo electrónico escritos por personas en Internet y una proporción considerable de estos mensajes generados automáticamente son transmitidos en formato MIME a través de SMTP.

RPC, XDR (de Sun), abierto, básico para estructuras de datos.

ASN.1 (CCITT, ahora ITU, abierto, orientado a protocolos.

CDR, (para CORBA de OMG), abierto, orientado a interfaces.

XML (W3C), con esquemas XML-RPC y SOAP, abierto, completamente extensible, orientado a protocolos o a interfaces.

Protocolos basados en texto

Se utiliza el empaquetamiento de datos cuando se intercambian cadenas de caracteres o texto codificado en una representación de tipo ASCII. El intercambio de datos en texto tiene la ventaja adicional de que puede ser fácilmente analizado por un programa y mostrado a un usuario humano. Por eso es relativamente habitual en varios protocolos intercambiar peticiones y respuestas en forma de cadenas de caracteres. Estos protocolos se denominan basados en texto. Muchos protocolos habituales son basados en texto, incluyendo FTP, HTTP y SMTP.

Protocolos de solicitud-respuesta

En estos protocolos un lado invoca una petición y espera una respuesta del otro extremo. Posteriormente, puede ser enviada otra solicitud, esperándose de nuevo respuesta. El protocolo se desarrolla basándose en interacciones de este tipo hasta que la tarea solicitada se ha cumplido. FTP, http y SMTP son protocolos habituales del tipo solicitud-respuesta.

Ejemplo: crear un protocolo FTP de acceso a documentos, usando la configuración en Linux o Windows.

2.10 COMUNICACIÓN EN GRUPO.

La comunicación en grupo tiene que permitir la definición de grupos, así como características propias de los grupos, como la distinción entre grupos abiertos o que permiten el acceso y cerrados que lo limitan, o como la distinción del tipo de jerarquía dentro del grupo. Igualmente, los grupos han de tener operaciones relacionadas con su manejo, como la creación o modificación.

Sincronización

La sincronización en sistemas de un único ordenador no requiere ninguna consideración en el diseño del sistema operativo, ya que existe un reloj único que proporciona de forma regular y precisa el tiempo en cada momento. Sin embargo, los sistemas distribuidos tienen un reloj por cada ordenador del sistema, con lo que es fundamental una coordinación entre todos los

relojes para mostrar una hora única. Los osciladores de cada ordenador son ligeramente diferentes, y como consecuencia todos los relojes sufren un desfase y deben ser sincronizados continuamente. La sincronización no es trivial, porque se realiza a través de mensajes por la red, cuyo tiempo de envío puede ser variable y depender de muchos factores, como la distancia, la velocidad de transmisión o la propia saturación de la red, etc.

El reloj

La sincronización no tiene por qué ser exacta, y bastará con que sea aproximadamente igual en todos los ordenadores. Hay que tener en cuenta, eso sí, el modo de actualizar la hora de un reloj en particular. Es fundamental no retrasar nunca la hora, aunque el reloj adelante. En vez de eso, hay que ralentizar la actualización del reloj, frenarlo, hasta que alcance la hora aproximadamente. Existen diferentes algoritmos de actualización de la hora, tres de ellos se exponen brevemente a continuación.

Algoritmo de Lamport

Tras el intento de sincronizar todos los relojes, surge la idea de que no es necesario que todos los relojes tengan la misma hora exacta, sino que simplemente mantengan una relación estable de forma que se mantenga la relación de qué suceso ocurrió antes que otro suceso cualquiera.

Este algoritmo se encarga exclusivamente de mantener el orden en que se suceden los procesos. En cada mensaje que se envía a otro ordenador se incluye la hora. Si el receptor del mensaje tiene una hora anterior a la indicada en el mensaje, utiliza la hora recibida incrementada en uno para actualizar su propia hora.

Algoritmo de Cristian

Consiste en disponer de un servidor de tiempo, que reciba la hora exacta. El servidor se encarga de enviar a cada ordenador la hora. Cada ordenador de destino sólo tiene que sumarle el tiempo de transporte del mensaje, que se puede calcular de forma aproximada.

Algoritmo de Berkeley

La principal desventaja del algoritmo de Cristian es que todo el sistema depende del servidor de tiempo, lo cual no es aceptable en un sistema distribuido fiable.

El algoritmo de Berkeley usa la hora de todos los ordenadores para elaborar una media, que se reenvía para que cada equipo actualice su propia hora ralentizando el reloj o adoptando la nueva hora, según el caso.

2.11. COMUNICACIÓN POR IP

La dirección IP es la matrícula de internet. Las dos siglas hacen referencia a internet protocol y consisten en una serie de números que identifican a cada dispositivo que se conecta a la red. Sin ellas, la comunicación en internet tal como la conocemos no es posible, ya que esta se produce desde un dispositivo de origen a otro de destino, ambos identificados con su IP. Es decir, si el servidor de una web no conoce nuestra IP, no podrá enviarnos la información necesaria para que podamos visualizarla.

Qué son y para qué sirven las direcciones IP IPv4 e IPv6
Las IP se construyen en cuatro bloques de tres dígitos separados por un punto. Cada bloque puede ir desde el 0 al 256. En total, este tipo de IP, llamado IPv4, permite 2^32 combinaciones diferentes o, lo que es lo mismo, casi 4.300 millones de posibilidades. Sin embargo, este número es hoy ya insuficiente, por lo que se ha ido introduciendo un nuevo protocolo, el IPv6, que permite 2^128 direcciones IP o, para entenderlo mejor, 340 sextillones de dispositivos comunicándose al mismo tiempo.

Nuestra matrícula en internet se divide en dos grandes grupos: pública y privada. También se divide entre IP fija (siempre es la misma) o variable (la más habitual, varía siempre que reiniciamos el router o cada cierto tiempo).

El enrutador se encargaría de ir asignando IPs dentro del rango que le indiquemos a los diferentes equipos que se vayan conectando. Esto hará imposible que por confusión **dos equipos tengan la misma IP entrando en conflicto**. Pero también hará que cada vez que un ordenador se conecte a la red tenga una IP diferente, por lo que será imposible mantener conexiones fijas entre varios

equipos (por ejemplo a carpetas compartidas, impresoras, etc.), ya que cada vez que se conecte tendrá una dirección IP distinta.

Para ello, necesitaremos **tener configurada una dirección IP estática**. Seremos nosotros los que deberemos asignar una dirección IP a cada equipo que conectemos a la red, teniendo especial cuidado en mantenerlas dentro de un mismo rango y no duplicándolas para que no entren en conflicto. Algo similar ocurre en internet.

Al estar conectados en una red local un Router se puede acceder a otros equipos a través de sus dirección IP, cabe mencionar que no es cuestión de solo conectarse bajo un ptotocolo o un software, en este caso la mejor opción mas fácil es a través de http y el servidor apache que permitirá acceder a otros equipos a través de su ip y el puerto 80.

UNIDAD III OBJETOS DISTRIBUIDOS E INVOCACION DE METODOS

3.1.- INTRODUCCIÓN.

Es aquel que está gestionado por un servidor y sus clientes invocan sus métodos utilizando un "método de invocación remota". El cliente invoca el método mediante un mensaje al servidor que gestiona el objeto, se ejecuta el método del objeto en el servidor y el resultado se devuelve al cliente en otro mensaje.

Tecnologías orientadas a los objetos distribuidos:

1. **RMI.-** Remote Invocation Method (Sistema de Invocación Remota de Métodos) : Fue el primer fremework para crear sistemas distribuidos de Java. Este sistema permite, a un objeto que se está ejecutando en una Máquina Virtual Java (VM), llamar a métodos de otro objeto que está en otra VM diferente. Esta tecnología está asociada al lenguaje de programación Java, es decir, que permite la comunicación entre objetos creados en este lenguaje.

2. **DCOM.-** Distributed Component Object Model: El Modelo de Objeto Componente Distribuido, está incluido en los sistemas operativos de Microsoft. Es un juego de conceptos e interfaces de programa, en el cual los objetos de programa del cliente, pueden solicitar servicios objetos del programa servidores, en otros ordenadores dentro de una red. Esta tecnología está asociada a la plataforma de productos Microsoft.

3. **CORBA.-** Common Object Request Broker Architecture: Tecnología introducida por el Grupo de Administración de Objetos OMG, creada para establecer una plataforma para la gestión de objetos remotos independiente del lenguaje de programación.

Los sistemas de bases de datos centralizados son aquellos que se ejecutan en un único sistema informático sin interaccionar con ninguna otra computadora. Tales sistemas comprenden el rango desde los sistemas de bases de datos monousuario ejecutándose en computadoras personales hasta los sistemas de bases de datos de alto rendimiento ejecutándose en grandes sistemas. Por otro lado, los sistemas cliente-servidor tienen su funcionalidad dividida entre el sistema servidor y múltiples sistemas clientes.

OBJETOS DISTRIBUIDOS

Definición:

En los sistemas Cliente/Servidor, un objeto distribuido es aquel que está gestionado por un servidor y sus clientes invocan sus métodos utilizando un –método de invocación remotall. El cliente invoca el método mediante un mensaje al servidor que gestiona el objeto, se ejecuta el método del objeto en el servidor y el resultado se devuelve al cliente en otro mensaje.

Tecnologías orientadas a los objetos distribuidos:

Las tres tecnologías importantes y más usadas en este ámbito son:

1. **RMI.-** Remote Invocation Method.- Fue el primer framework para crear sistemas distribuidos de Java. El sistema de Invocación Remota de Métodos (RMI) de Java permite, a un objeto que se está ejecutando en una Máquina Virtual Java (VM), llamar a métodos de otro objeto que está en otra VM diferente. Esta tecnología está asociada al lenguaje de programación Java, es decir, que permite la comunicación entre objetos creados en este lenguaje.

2. **DCOM.-** Distributed Component Object Model.- El Modelo de Objeto ComponenteDistribuido, esta incluido en los sistemas operativos de Microsoft. Es un juego deconceptos e interfaces de programa, en el cual los objetos de programa del cliente, pueden solicitar servicios de objetos de programa servidores en otras computadoras dentro de una red. Esta tecnología esta asociada a la plataforma de productos Microsoft.

3. **CORBA.-** Common Object Request Broker Architecture.- Tecnología introducida por el Grupo de Administración de Objetos OMG, creada para establecer una plataforma para la gestión de objetos remotos independiente del lenguaje de programación.

Ventajas de las Base de Datos Distribuidas

Descentralización. - En un sistema centralizado/distribuido, existe un administrador que controla toda la base de datos, por el contrario, en un sistema distribuido existe un administrador global que lleva una política general y delega algunas funciones a administradores de cada localidad para que establezcan políticas locales y así un trabajo eficiente.

Economía: Existen dos aspectos a tener en cuenta. El primero son los costes de comunicación; si las bases de datos están muy dispersas y las aplicaciones hacen amplio uso de los datos puede resultar más económico dividir la aplicación y realizarla localmente.

El segundo aspecto es que cuesta menos crear un sistema de pequeñas computadoras con la misma potencia que un único computador.

Mejora de rendimiento: Pues los datos serán almacenados y usados donde son generados, lo cual permitirá distribuir la complejidad del sistema en los diferentes sitios de la red, optimizando la labor.

Mejora de fiabilidad y disponibilidad: La falla de uno o varios lugares o el de un enlace de comunicación no implica la inoperatividad total del sistema, incluso si tenemos datos duplicados puede que exista una disponibilidad total de los servicios.

Crecimiento: Es más fácil acomodar el incremento del tamaño en un sistema distribuido, porque la expansión se lleva a cabo añadiendo poder de procesamiento y almacenamiento en la red, al añadir un nuevo nodo.

Flexibilidad: Permite acceso local y remoto de forma transparente.

Disponibilidad: Pueden estar los datos duplicados con lo que varias personas pueden acceder simultáneamente de forma eficiente. El inconveniente, el sistema administrador de base de datos debe preocuparse de la consistencia de los mismos.

Control de Concurrencia: El sistema administrador de base de datos local se encarga de manejar la concurrencia de manera eficiente.

EVOLUCIÓN DE LOS SISTEMAS DISTRIBUIDOS

Definición:

–Sistemas cuyos componentes hardware y software, que están en ordenadores conectados en red, se comunican y coordinan sus acciones mediante el paso de mensajes, para el logro de un objetivo. Se establece la comunicación mediante un protocolo prefijado por un esquema cliente-servidorII.

Características:

Concurrencia. - Esta característica de los sistemas distribuidos permite que los recursos disponibles en la red puedan ser utilizados simultáneamente por los usuarios y/o agentes que interactúan en la red.

Carencia de reloj global. - Las coordinaciones para la transferencia de mensajes entre los diferentes componentes para la realización de una tarea, no tienen una temporización general, está más bien distribuida a los componentes.

Fallos independientes de los componentes. - Cada componente del sistema puede fallar independientemente, con lo cual los demás pueden continuar ejecutando sus acciones. Esto permite el logro de las tareas con mayor efectividad, pues el sistema en su conjunto continúa trabajando.

Motivación

Los sistemas distribuidos suponen un paso más en la evolución de los sistemas informáticos, entendidos desde el punto de vista de las necesidades que las aplicaciones plantean y las posibilidades que la tecnología ofrece. Antes de proporcionar una definición de sistema distribuido resultará interesante presentar, a través de la evolución histórica, los conceptos que han desembocado en los sistemas distribuidos actuales, caracterizados por la distribución física de los recursos en máquinas interconectadas.

Utilizaremos aquí el término recurso con carácter general para referirnos a cualquier dispositivo o servicio, hardware o software, susceptible de ser compartido.

Tipos de sistemas

Desde una perspectiva histórica se puede hablar de diferentes modelos que determinan la funcionalidad y la estructura de un sistema de cómputo, las características del sistema operativo como gestor de los recursos, y su campo de aplicación y uso:

• Sistemas de lotes. Son los primeros sistemas operativos, que permitían procesar en diferido y secuencialmente datos suministrados en paquetes de tarjetas perforadas. Hoy en día, sobre sistemas multiprogramados con interfaces de usuario interactivas, el proceso por lotes tiene sentido en aplicaciones de cálculo intensivo, por ejemplo, en supercomputación.

• Sistemas centralizados de tiempo compartido. Fue el siguiente paso, a mediados de los 60.

El objetivo es incrementar la eficiencia en el uso de la CPU, un recurso entonces caro y escaso, disminuyendo los tiempos de respuesta de los usuarios, que operan interactivamente. Los recursos están centralizados y se accede al sistema desde terminales.

• Sistemas de teleproceso. Se diferencian del modelo anterior en que los terminales más recientemente, sistemas personales son remotos y acceden a un sistema central utilizando una infraestructura de red (por ejemplo, la telefónica) y un protocolo de comunicaciones normalmente de tipo propietario. El sistema central monopoliza la gestión de los recursos. Ejemplos de aplicaciones que resolvía este modelo son los sistemas de reservas y de transacciones bancarias. Sistemas personales. La motivación de este tipo de sistemas estuvo en proporcionar un sistema dedicado para un único usuario, lo que fue posible gracias al abaratamiento del hardware por la irrupción del microprocesador a comienzos de los 80. Precisamente, el coste reducido es su característica fundamental. El sistema operativo de un ordenador personal (PC) es, en un principio, monousuario: carece de mecanismos de protección. Aunque, por simplicidad, los primeros sistemas operativos eran mono programados (MS-DOS), la mejora del hardware pronto permitió soportar sistemas multitarea (Macintosh, OS/2, Windows 95/98), e incluso sistemas operativos diseñados para tiempo compartido, como UNIX y Windows NT1. Un sistema personal posee sus propios recursos locales. Inicialmente, éstos eran los únicos recursos accesibles, pero hoy en día la situación ha cambiado. Por otra parte, la evolución hardware ha llevado a los ordenadores personales hacia versiones móviles (PC portátiles y otros dispositivos como PDAs y teléfonos móviles).

Sistemas distribuidos. Los recursos de diferentes máquinas en red se integran de forma que desaparece la dualidad local/remoto. La diferencia fundamental con los sistemas en red es que la ubicación del recurso es transparente a las aplicaciones y usuarios, por lo que, desde este punto de vista, no hay diferencia con un sistema de tiempo compartido. El usuario accede a los recursos del sistema distribuido a través de una interfaz gráfica de usuario desde un terminal, despreocupándose de su localización. Las aplicaciones ejecutan una interfaz de llamadas al sistema como si de un sistema centralizado se tratase, por ejemplo, POSIX. Un servicio de invocación remota (por ejemplo, a procedimientos, RPC, o a objetos, RMI) resuelve los accesos a los recursos no locales utilizando para ello la interfaz de red. Los sistemas distribuidos proporcionan de forma transparente la compartición de recursos, facilitando el acceso y la gestión, e incrementando la eficiencia y la disponibilidad.

El modelo de sistema distribuido es el más general, por lo que, aunque no se ha alcanzado a nivel comercial la misma integración para todo tipo de recursos, la tendencia es clara a favor de este tipo de sistemas. La otra motivación es la Hoy en día existe un hardware estándar de bajo coste, los ordenadores personales, que son los componentes básicos del sistema. Por otra parte, la red de comunicación, a no ser que se requieran grandes prestaciones, tampoco constituye un gran problema económico, pudiéndose utilizar infraestructura cableada ya existente (Ethernet, la red telefónica, o incluso la red eléctrica) o inalámbrica.

LA TECNOLOGIA DE OBJETOS PARA EL DESARROLLO DE SISTEMAS DISTRIBUIDOS

El Cliente Stub es una entidad de programa que invoca una operación sobre la implementación de un objeto remoto a través de un Stub cuyo propósito es lograr que la petición de un cliente llegue hasta el ORB Core. Logrando el acoplamiento entre el lenguaje de programación en que se escribe el cliente y el ORB Core. El stub crea y expide las solicitudes del cliente.

Un Servidor Skeleton es la implementación de un objeto CORBA en algún lenguaje de programación, y define las operaciones que soporta una interface IDL CORBA. Puede escribirse en una gran variedad de lenguajes como C, C++, Java, Ada o Smalltalk. Y a través del skeleton entrega las solicitudes procedentes del ORB a la implementación del objeto CORBA.

La función del ORB consiste en conectar las dos partes: cliente y servidor. Presumiblemente estas partes se ejecutan sobre plataformas distintas y funcionan con diferentes sistemas operativos. Esto significa que pueden existir diferencias en tipos de datos, el orden de los parámetros en una llamada, el orden de los bytes en una palabra según el tipo de procesador, etc. Es misión del ORB efectuar los procesos conocidos como marshaling y unmarshaling. En caso de que el método invocado devuelva un valor de retorno, la función de los ORB del cliente y servidor se invierte. Éste realiza el marshaling de dicho valor y lo envía al ORB del cliente, que será el que realice el unmarshaling y finalmente facilite el valor en formato nativo.

3.2.- RPC.

En computación distribuida, la **llamada a procedimiento remoto** (en inglés, *Remote Procedure Call*, **RPC**) es un programa que utiliza una computadora para ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambas. El protocolo que se utiliza para esta llamada es un gran avance sobre los *sockets* de Internetusados hasta el momento. De esta manera el programador no tenía que estar pendiente de las comunicaciones, estando estas encapsuladas dentro de las RPC.

Las RPC son muy utilizadas dentro de la comunicación cliente-servidor. Siendo el cliente el que inicia el proceso solicitando al servidor que ejecute cierto procedimiento o función y enviando este de vuelta el resultado de dicha operación al cliente.

Las llamadas a procedimiento remoto están implementadas mediante varios tipos de protocolos, muchos de ellos estandarizados como pueden ser el RPC de Sun denominado ONC RPC (RFC 1057), el RPC de *Open Software Foundation* (OSF) denominado DCE/RPC y el "Modelo de Objetos de Componentes Distribuidos de Microsoft" (*Distributed Component Object Model*, DCOM), aunque ninguno de estos es compatible entre sí. La mayoría de ellos utilizan un lenguaje de descripción de interfaz (*Interface description language* o IDL) que define los métodos exportados por el servidor.

Hoy en día se está utilizando el XML como lenguaje para definir el IDL y el HTTP como protocolo de aplicación, dando lugar a lo que se conoce como servicios web. Ejemplos de estos pueden ser SOAP o XML-RPC.

Semántica "tal-vez"

- Procedimiento remoto puede ejecutarse una vez o ninguna vez.
- El cliente puede recibir una respuesta o ninguna.

Funcionamiento

1. El cliente envía una petición y se queda a la espera un tiempo determinado.
2. Si no llega la respuesta dentro del tiempo de espera, continúa su ejecución.

3. El cliente no tiene realimentación en caso de fallo (no sabe que pasó).

Sólo admisible en aplicaciones donde se tolere la pérdida de peticiones y la recepción de respuestas con retraso (fuera de orden).

Semántica "al-menos-una-vez"

- Procedimiento remoto se ejecuta una o más veces
- El cliente puede recibir una o más respuestas.

Funcionamiento

1. El cliente envía una petición y queda a la espera un tiempo.
2. Si no llega respuesta o ACK dentro del tiempo de espera, repite la petición.
3. El servidor no filtra peticiones duplicadas (el procedimiento remoto puede ejecutarse repetidas veces).
4. El cliente puede recibir varias respuestas.

Sólo es aplicable cuando se usan exclusivamente operaciones idempotentes (repetibles). Nota: una operación es idempotente si se puede ejecutar varias veces resultando el mismo efecto que si se hubiera ejecutado sólo una. En ocasiones una operación no idempotente puede implementarse como una secuencia de operaciones idempotentes. Admisible en aplicaciones donde se tolere que se puedan repetir invocaciones sin afectar a su funcionamiento.

Semántica "como-máximo-una-vez"

- El procedimiento remoto se ejecuta exactamente una vez o no llega a ejecutarse ninguna.
- El cliente recibe una respuesta o una indicación de que no se ha ejecutado el procedimiento remoto.

Funcionamiento

1. El cliente envía la petición y queda a la espera un tiempo.
2. Si no llega respuesta o ACK dentro del tiempo de espera, repite la petición.
3. El servidor filtra las peticiones duplicadas y guarda historial con las respuestas enviadas (Servidor con memoria). El procedimiento remoto sólo se ejecuta una vez.
4. El cliente sólo recibe una respuesta si la petición llegó y se ejecutó el procedimiento, si no recibe informe del error.

3.3 EJECUCIÓN DE RPC EN C#

RPC usa un enfoque de contrato primero para el desarrollo de API. Los búferes de protocolo (Protobuf) se usan de forma predeterminada como el lenguaje de definición de interfaz (IDL). El archivo *.proto contiene:

La definición del servicio gRPC
Los mensajes enviados entre clientes y servidores
Para más información sobre la sintaxis de los archivos de protobuf, consulte Creación de mensajes de Protobuf para aplicaciones .NET.

Vamos a considerar, por ejemplo, el archivo greeter.proto que se usa en Introducción a un servicio gRPC:

Define un servicio Greeter.
El servicio Greeter define una llamada a SayHello.
SayHello envía un mensaje HelloRequest y recibe un mensaje HelloReply:

```ProtoBuf
syntax = "proto3";

option csharp_namespace = "GrpcGreeter";

package greet;

// The greeting service definition.
service Greeter {
  // Sends a greeting
  rpc SayHello (HelloRequest) returns (HelloReply);
}

// The request message containing the user's name.
message HelloRequest {
  string name = 1;
}

// The response message containing the greetings.
message HelloReply {
  string message = 1;
}
```

Adición de un archivo .proto a una aplicación de C#

Para incluir el archivo *.*proto* en un proyecto, hay que agregarlo al grupo de elementos :

XMLCopiar
<ItemGroup>
  <Protobuf Include="Protos\greet.proto" GrpcServices="Server" />
</ItemGroup>

De forma predeterminada, una referencia <Protobuf> genera un cliente concreto y una clase base de servicio. El atributo GrpcServices del elemento de referencia se puede usar para limitar la generación de recursos de C#. Las opciones válidas de GrpcServices son:

- Both (valor predeterminado si no se especifica)
- Server
- Client
- None

Compatibilidad de herramientas de C# con archivos .proto

El paquete de herramientas Grpc.Tools es necesario para generar los recursos de C# a partir de los archivos *.*proto*. Los activos (archivos) generados:

- Se generan según sea necesario cada vez que el proyecto se compila.
- No se agregan al proyecto ni se protegen en el control de código fuente.
- Son un artefacto de compilación contenido en el directorio *obj*.

Este paquete es necesario en los proyectos tanto de servidor como de cliente. El metapaquete Grpc.AspNetCore incluye una referencia a Grpc.Tools. Los proyectos de servidor pueden agregar Grpc.AspNetCore por medio del administrador de paquetes de Visual Studio, o bien agregando un elemento <PackageReference> al archivo de proyecto:

XMLCopiar
<PackageReference Include="Grpc.AspNetCore" Version="2.32.0" />

Los proyectos de cliente, por su parte, deben hacer referencia directamente a Grpc.Tools junto con los demás paquetes necesarios para usar el cliente de gRPC. El paquete de herramientas no es necesario en tiempo de ejecución, de modo que la dependencia se marca con PrivateAssets="All":

XMLCopiar
<PackageReference Include="Google.Protobuf" Version="3.18.0" />
<PackageReference Include="Grpc.Net.Client" Version="2.39.0" />
<PackageReference Include="Grpc.Tools" Version="2.40.0">
  <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
  <PrivateAssets>all</PrivateAssets>
</PackageReference>

Activos de C# generados

El paquete de herramientas genera los tipos de C# que representan los mensajes definidos en los archivos **.proto* incluidos.

En el caso de los activos del lado servidor, se genera un tipo base de servicio abstracto. El tipo base contiene las definiciones de todas las llamadas a gRPC contenidas en el archivo *.proto*. Cree una implementación de servicio concreta que se derive de este tipo base e implemente la lógica de las llamadas a gRPC. Respecto a greet.proto, el ejemplo descrito anteriormente, se genera un tipo de elemento GreeterBase abstracto que contiene un método SayHello virtual. Una elemento GreeterService de una implementación concreta invalida el método e implementa la lógica que controla la llamada de gRPC.

```
public class GreeterService : Greeter.GreeterBase
{
    private readonly ILogger<GreeterService> _logger;
    public GreeterService(ILogger<GreeterService> logger)
    {
        _logger = logger;
    }

    public override Task<HelloReply> SayHello(HelloRequest request, ServerCallConte
    {
        return Task.FromResult(new HelloReply
        {
            Message = "Hello " + request.Name
        });
    }
}
```

En cuanto a los activos del lado cliente, se genera un tipo de cliente concreto. Las llamadas de gRPC en el archivo *.proto* se traducen en métodos en el tipo concreto a los que se puede llamar. Respecto a greet.proto, el ejemplo descrito anteriormente, se genera un tipo de elemento GreeterClient concreto. Llame a GreeterClient.SayHelloAsync para iniciar una llamada de gRPC al servidor.

```
// The port number must match the port of the gRPC server.
using var channel = GrpcChannel.ForAddress("https://localhost:7042");
var client = new Greeter.GreeterClient(channel);
var reply = await client.SayHelloAsync(
                  new HelloRequest { Name = "GreeterClient" });
Console.WriteLine("Greeting: " + reply.Message);
Console.WriteLine("Press any key to exit...");
Console.ReadKey();
```

3.4.- JAVA RMI.

RMI (**Java Remote Method Invocation**) es un mecanismo ofrecido por Java para invocar un método de manera remota. Forma parte del entorno estándar de ejecución de Java y proporciona un mecanismo simple para la comunicación de servidores en aplicaciones distribuidas basadas exclusivamente en Java. Si se requiere comunicación entre otras tecnologías debe utilizarse CORBA o SOAP en lugar de RMI.

RMI se caracteriza por la facilidad de su uso en la programación por estar específicamente diseñado para Java; proporciona paso de objetos por referencia (no permitido por SOAP), recolección de basura distribuida (Garbage Collector distribuido) y paso de tipos arbitrarios (funcionalidad no provista por CORBA).

A través de RMI, un programa Java puede exportar un objeto, con lo que dicho objeto estará accesible a través de la red y el programa permanece a la espera de peticiones en un puerto TCP. A partir de ese momento, un cliente puede conectarse e invocar los métodos proporcionados por el objeto.

- La invocación se compone de los siguientes pasos:
- Encapsulado (marshalling) de los parámetros (utilizando la funcionalidad de serialización de Java).
- Invocación del método (del cliente sobre el servidor). El invocador se queda esperando una respuesta.

• Al terminar la ejecución, el servidor serializa el valor de retorno (si lo hay) y lo envía al cliente.

• El código cliente recibe la respuesta y continúa como si la invocación hubiera sido local.

Contexto

Desde la versión 1.1 de JDK, Java tiene su propio ORB: RMI (Remote Method Invocation). A pesar de que RMI es un ORB en el sentido general, no es un modelo compatible con CORBA. RMI es nativo de Java, es decir, es una extensión al núcleo del lenguaje. RMI depende totalmente del núcleo de la Serialización de Objetos de Java, así como de la implementación tanto de la portabilidad como de los mecanismos de carga y descarga de objetos en otros sistemas, etc.

El uso de RMI resulta muy natural para todo aquel programador de Java ya que éste no tiene que aprender una nueva tecnología completamente distinta de aquella con la cual desarrollará. Sin embargo, RMI tiene algunas limitaciones debido a su estrecha integración con Java, la principal de ellas es que esta tecnología no permite la interacción con aplicaciones escritas en otro lenguaje.

RMI como extensión de Java, es una tecnología de programación, fue diseñada para resolver problemas escribiendo y organizando código ejecutable. Así RMI constituye un punto específico en el espacio de las tecnologías de programación junto con C, C++, Smalltalk, etc.

La principal diferencia de utilizar RPC o RMI, es que RMI es un mecanismo para invocación remota de procedimientos basado en el lenguaje de programación Java que soporta interacción entre objetos, mientras que RPC no soporta esta característica.

Arquitectura

La arquitectura RMI puede verse como un modelo de cuatro capas.

Primera capa

La primera capa es la de aplicación y se corresponde con la implementación real de las aplicaciones cliente y servidor. Aquí tienen lugar las llamadas a alto nivel para acceder y exportar objetos remotos. Cualquier aplicación que quiera que sus métodos estén disponibles para su acceso por clientes remotos debe declarar dichos métodos en una interfazque extienda

java.rmi.Remote. Dicha interfaz se usa básicamente para "marcar" un objeto como remotamente accesible. Una vez que los métodos han sido implementados, el objeto debe ser exportado. Esto puede hacerse de forma implícita si el objeto extiende la clase UnicastRemoteObject (paquete java.rmi.server), o puede hacerse de forma explícita con una llamada al método exportObject() del mismo paquete.

Segunda capa

La capa 2 es la capa proxy, o capa stub-skeleton. Esta capa es la que interactúa directamente con la capa de aplicación. Todas las llamadas a objetos remotos y acciones junto con sus parámetros y retorno de objetos tienen lugar en esta capa.

Tercera capa

La capa 3 es la de referencia remota, y es responsable del manejo de la parte semántica de las invocaciones remotas. También es responsable de la gestión de la replicación de objetos y realización de tareas específicas de la implementación con los objetos remotos, como el establecimiento de las persistencias semánticas y estrategias adecuadas para la recuperación de conexiones perdidas. En esta capa se espera una conexión de tipo stream (stream-oriented connection) desde la capa de transporte.

Cuarta Capa

La capa 4 es la de transporte. Es la responsable de realizar las conexiones necesarias y manejo del transporte de los datos de una máquina a otra. El protocolo de transporte subyacente para RMI es JRMP (Java Remote Method Protocol), que solamente es "comprendido" por programas Java.

Elementos

- Toda aplicación RMI normalmente se descompone en 2 partes:
- Un servidor, que crea algunos objetos remotos, crea referencias para hacerlos accesibles, y espera a que el cliente los invoque.
- Un cliente, que obtiene una referencia a objetos remotos en el servidor, y los invoca.

Ejemplo

Un servidor RMI consiste en definir un objeto remoto que va a ser utilizado por los clientes. Para crear un objeto remoto, se define una interfaz, y el objeto remoto será una clase que implemente dicha interfaz. Veamos cómo crear un servidor de ejemplo mediante 3 pasos:

Definir la interfaz remota. Cuando se crea una interfaz remota:

La interfaz debe ser pública.

Debe heredar de la interfaz java.rmi.Remote, para indicar que puede llamarse desde cualquier máquina virtual Java.

Cada método remoto debe lanzar la excepción java.rmi.RemoteException en su cláusula throws, además de las excepciones que pueda manejar.

```
public interface MiInterfazRemota extends java.rmi.Remote
{
  public void miMetodo1() throws java.rmi.RemoteException;
  public int miMetodo2() throws java.rmi.RemoteException;
}
```

3.5. EJECUCIÓN DE JAVA RMI

Enviar mensajes no tiene gran utilidad, ¿no sería mejor poder llamar a algunas funciones en el servidor? esto es justamente lo que hace **RMI en Java**. Declaramos el servidor, el cliente, y la interfaz que servirá como "pegamento" entre estos dos y que se encargará del paso de parámetros.

Después, desde el cliente llamamos a las funciones declaradas anteriormente.

De esta manera, todos los métodos se ejecutarán en el servidor. En este caso haremos una calculadora, para no hacer el post muy largo, pero podemos hacer miles de cosas; se me ocurre, por ejemplo, conectar a una base de datos.

Interfaz remota

Comencemos con la interfaz remota, como lo dije hace un momento esta servirá como pegamento. Veamos una cosa interesante, como se puede ver dice "*public interface*" en lugar de "*public class*".

Esto es porque, aunque suene redundante, es una interfaz y recordemos que éstas sólo sirven para **ser sobrescritas más tarde**. ¿Y en dónde las sobrescribiremos? en el código del servidor.

```
import java.rmi.Remote;
import java.rmi.RemoteException;

/*
        Declarar firma de métodos que serán sobrescritos
*/
public interface Interfaz extends Remote {
   float sumar(float numero1, float numero2) throws RemoteException;
   float restar(float numero1, float numero2) throws RemoteException;
   float multiplicar(float numero1, float numero2) throws RemoteException;
   float dividir(float numero1, float numero2) throws RemoteException;
}
```

Servidor

Ahora veamos el servidor. Este se encarga de exportar el objeto, sobrescribir funciones de la interfaz y escuchar peticiones del cliente.

```
import java.rmi.AlreadyBoundException;
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.UnicastRemoteObject;
public class Servidor {
```

```
        private static final int PUERTO = 1100; //Si cambias aquí el puerto, recuerda cambiarlo en el cliente
    public static void main(String[] args) throws RemoteException, AlreadyBoundException {
        Remote remote = UnicastRemoteObject.exportObject(new Interfaz() {
            /*
                                    Sobrescribir opcionalmente los métodos que escribimos en la interfaz
            */
            @Override
            public float sumar(float numero1, float numero2) throws RemoteException {
                return numero1 + numero2;
            };

            @Override
            public float restar(float numero1, float numero2) throws RemoteException {
                return numero1 - numero2;
            };

            @Override
            public float multiplicar(float numero1, float numero2) throws RemoteException {
                return numero1 * numero2;
            };

            @Override
            public float dividir(float numero1, float numero2) throws RemoteException {
                return numero1 / numero2;
            };
        }, 0);
        Registry registry = LocateRegistry.createRegistry(PUERTO);
            System.out.println("Servidor escuchando en el puerto " + String.valueOf(PUERTO));
        registry.bind("Calculadora", remote); // Registrar calculadora
    }
}
```

Como observamos, ya hemos sobrescrito los métodos con el cuerpo real de la función. La anotación @Override no es necesaria, pero es una buena práctica indicar que estamos sobrescribiendo una función.

Finalmente escuchamos en localhost (cosa que no podemos cambiar) en el puerto definido en la constante PUERTO.

Cliente

Para terminar con el código veamos ahora el cliente:

```
import java.rmi.NotBoundException;
import java.rmi.RemoteException;
```

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.Scanner;
public class Cliente {
        private static final String IP = "192.168.1.15"; // Puedes cambiar a localhost
        private static final int PUERTO = 1100; //Si cambias aquí el puerto, recuerda cambiarlo en el servidor

    public static void main(String[] args) throws RemoteException, NotBoundException {
        Registry registry = LocateRegistry.getRegistry(IP, PUERTO);
        Interfaz interfaz = (Interfaz) registry.lookup("Calculadora"); //Buscar en el registro...
        Scanner sc = new Scanner(System.in);
        int eleccion;
        float numero1, numero2, resultado = 0;
        String menu = "\n\n-----------------\n\n[-1] => Salir\n[0] => Sumar\n[1] => Restar\n[2] => Multiplicar\n[3] => Div
        do {
            System.out.println(menu);

            try {
                eleccion = Integer.parseInt(sc.nextLine());
            } catch (NumberFormatException e) {
                eleccion = -1;
            }

            if(eleccion != -1){

                    System.out.println("Ingresa el número 1: ");
                    try{
                    numero1 = Float.parseFloat(sc.nextLine());
                    }catch(NumberFormatException e){
                            numero1 = 0;
                    }

                    System.out.println("Ingresa el número 2: ");
                    try{
                    numero2 = Float.parseFloat(sc.nextLine());
                    }catch(NumberFormatException e){
                            numero2 = 0;
                    }
                switch (eleccion) {
                        case 0:
                            resultado = interfaz.sumar(numero1, numero2);
                            break;
                        case 1:
                            resultado = interfaz.restar(numero1, numero2);
                            break;
```

```
                    case 2:
                        resultado = interfaz.multiplicar(numero1, numero2);
                        break;
                    case 3:
                        resultado = interfaz.dividir(numero1, numero2);
                        break;
                }

                System.out.println("Resultado => " + String.valueOf(resultado));
                System.out.println("Presiona ENTER para continuar");
                sc.nextLine();
            }
        } while (eleccion != -1);
    }
}
```

Ejecución
El código no sirve para nada si no lo compilamos y probamos. Recuerda que puedes compilarlo desde un IDE, o desde la terminal.

3.6 DIFERENCIA ENTRE RCP Y RMI

RPC y RMI son los mecanismos que permiten a un cliente invocar el procedimiento o método desde el servidor a través del establecimiento de la comunicación entre el cliente y el servidor. La diferencia común entre RPC y RMI es que RPC solo admite la programación de procedimientos, mientras que RMI admite la programación orientada a objetos.
Otra diferencia importante entre los dos es que los parámetros pasados a la llamada a procedimientos remotos consisten en estructuras de datos ordinarias. Por otro lado, los parámetros pasados al método remoto consisten en objetos.

Bases para la comparación	**RPC**	**RMI**
Ayudas	Programación procesal	Programación orientada a objetos
Parámetros	Las estructuras de datos ordinarios se pasan a procedimientos remotos.	Los objetos se pasan a métodos remotos.
Eficiencia	Más bajo que RMI	Más que RPC y con un enfoque de programación moderno (es decir, paradigmas orientados a objetos)

Gastos generales	Más	Menos comparativamente
Los parámetros de entrada y salida son obligatorios.	Sí	No necesariamente
Facilitación de la facilidad de programación.	Alto	bajo

Entendamos cómo se implementa RPC a través de los pasos dados:

- El proceso del cliente llama al apéndice del cliente con parámetros, y su ejecución se suspende hasta que se completa la llamada.
- Luego, los parámetros se traducen a una forma independiente de la máquina mediante la ordenación a través del talón del cliente. Luego se prepara el mensaje que contiene la representación de los parámetros.
- Para encontrar la identidad del sitio, el apéndice del cliente se comunica con el servidor de nombres en el que existe el procedimiento remoto.
- Al usar el protocolo de bloqueo, el apéndice del cliente envía el mensaje al sitio donde existe la llamada a un procedimiento remoto. Este paso detiene el apéndice del cliente hasta que reciba una respuesta.

Diferencias clave entre RPC y RMI

- RPC admite los paradigmas de programación de procedimientos, por lo tanto, está basado en C, mientras que RMI admite paradigmas de programación orientados a objetos y está basado en Java.
- Los parámetros pasados a procedimientos remotos en RPC son las estructuras de datos ordinarias. Por el contrario, RMI transita objetos como un parámetro al método remoto.
- RPC se puede considerar como la versión anterior de RMI, y se usa en los lenguajes de programación que admiten la programación de procedimientos, y solo se puede usar el método de paso por valor. A diferencia de lo anterior, la facilidad de RMI está diseñada en base a un enfoque de programación moderno, que podría usar el paso por valor o

referencia. Otra ventaja de RMI es que los parámetros pasados por referencia se pueden cambiar.

- El protocolo RPC genera más gastos generales que RMI.
- Los parámetros pasados en RPC deben ser " **in-out** ", lo que significa que el valor pasado al procedimiento y el valor de salida deben tener los mismos tipos de datos. En contraste, no hay obligación de pasar parámetros de " **entrada-salida** " en RMI.
- En RPC, las referencias no podrían ser probables porque los dos procesos tienen un espacio de direcciones distinto, pero es posible en el caso de RMI.

3.7.- SOPORTE DEL SISTEMA OPERATIVO.

El sistema operativo es básicamente un programa que controla los recursos del computador, proporciona servicios a los programadores y planifica la ejecución de otros programas. A partir de los µP de 16 bits, las CPU incorporan estructuras de apoyo a los sistemas operativos, por lo que resulta interesante una introducción a dos de las funciones básicas del SO que más inciden en la arquitectura de la CPU: la multiprogramación (o multitarea) y el control de memoria.

MULTIPROGRAMACIÓN. La multiprogramación es la tarea central de los sistemas operativos modernos. Permite que múltiples programas de usuario o usuarios que se hallan en memoria se alternen entre la utilización de la CPU y los accesos a I/O, de manera que el procesador siempre se mantenga ocupado con un proceso mientras los demás esperan. PLANIFICACION (SCHEDULING) DE ALTO NIVEL. Determina qué programas son admitidos por el sistema para ser procesados. El planificador (proyecta) controla pues el grado de multiprogramación (número de procesos en memoria). Una vez admitido, un programa se convierte en un proceso y es añadido a la cola para ser tratado por el distribuidor.

El planificador de alto nivel puede limitar el grado de multiprogramación para dar un servicio satisfactorio al conjunto actual de procesos.

PLANIFICACION A CORTO PLAZO (SHORT-TERM SCHEDULING). Este planificador, conocido también como distribuidor (dispatcher), se encarga de decidir en cada momento cuál de los procesos admitidos por el anterior se ejecutará en siguiente lugar. Esta decisión se basa en el estado del proceso.

Básicamente existen cinco posibles estados de un proceso:

1. Nuevo: El programa ha sido admitido por el planificador de alto nivel, pero no está listo para ser ejecutado. El sistema operativo inicializará el proceso, pasándolo al estado siguiente.

2. Preparado: El proceso está listo para ser ejecutado, y está esperando acceso al procesador.

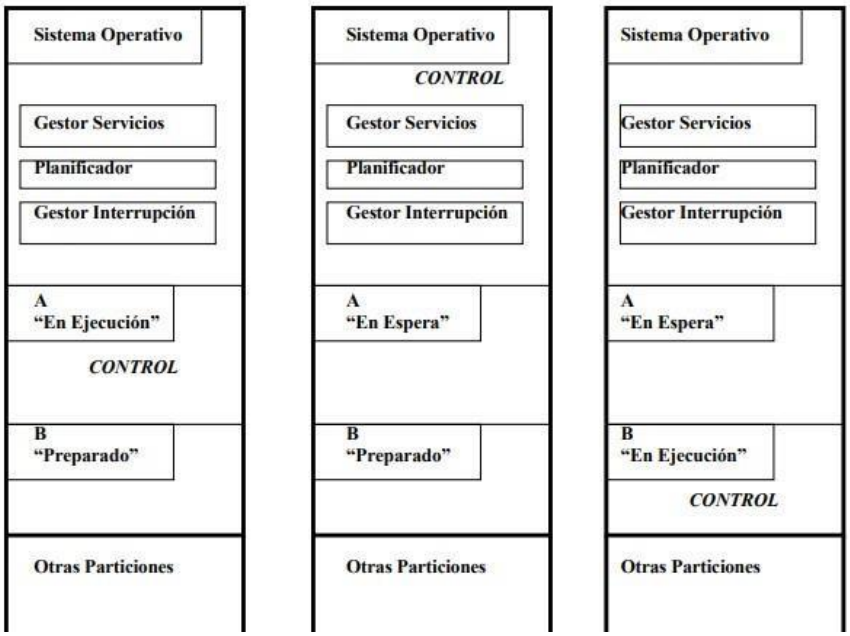
3. En ejecución: El proceso está siendo ejecutado por el procesador.

4. Esperando: Se suspende la ejecución del proceso, en espera de algún recurso del sistema, como I/O. 5. Parado: El proceso ha sido terminado y será eliminado por el sistema operativo.

Para cada proceso, el sistema operativo debe mantener una información del estado. Para ello, cada proceso se representa en el SO por un bloque de control de proceso, que contiene generalmente:

- Identificador: único para cada proceso actual.
- Estado: los tipos vistos anteriormente.
- Prioridad: nivel relativo de prioridad.
- Contador de programa: La dirección de la siguiente instrucción del programa a ser ejecutada.
- Punteros de memoria: Las direcciones de comienzo y final del proceso en memoria.
- Contexto de datos: son los datos de los registros del procesador para ese proceso.
- Información de estado I/O: incluye dispositivos I/O asignados a ese proceso, lista de ficheros correspondientes al mismo, etc.
- Información adicional: Puede incluir la cantidad de tiempo de procesador y tiempo de reloj utilizados, límites temporales, etc.

Técnicas de planificación.



La figura muestra una memoria principal con particiones en un momento determinado. El núcleo del SO siempre está residente en memoria. Además existen dos procesos activos, A y B, cada uno en una partición distinta de memoria

En t=0, el proceso A está siendo ejecutado, y el procesador toma las instrucciones a ejecutar de la partición correspondiente a A. En un momento determinado, el control pasa al sistema operativo. Esto puede producirse por tres razones fundamentalmente:

1. El proceso A realiza una llamada de servicio al SO (por ejemplo, para acceso a I/O). Se suspende la ejecución de A hasta que se satisfaga la llamada.

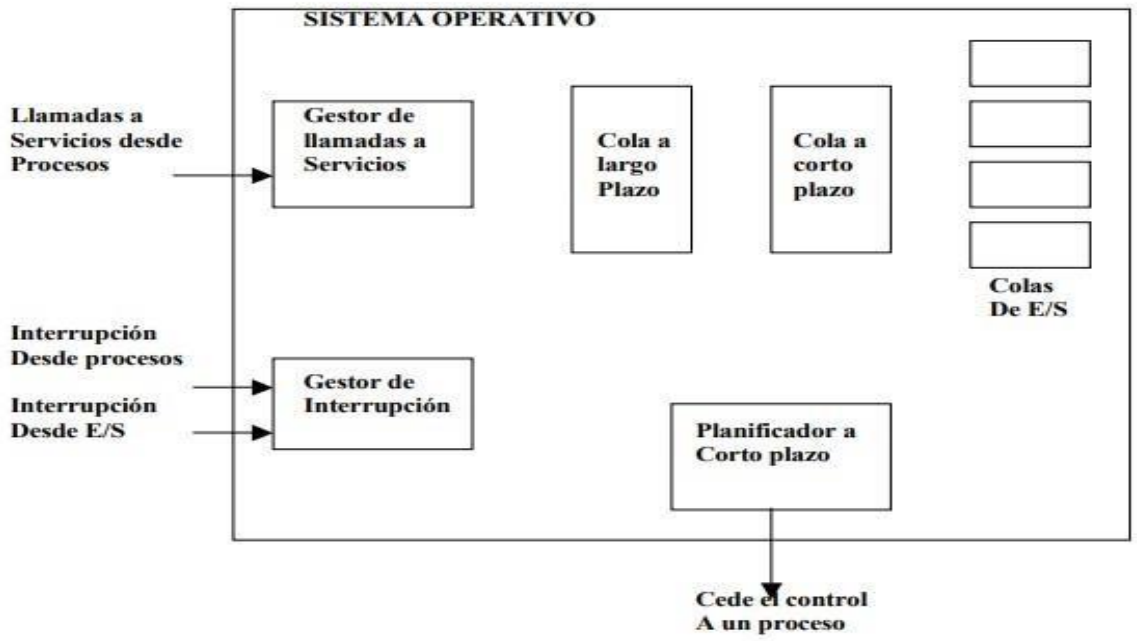
2. El proceso A produce una interrupción. El procesador deja de ejecutar A y transfiere el control al gestor de interrupciones en el SO. Existen varias posibilidades de que esto ocurra. Un ejemplo es un error, que puede estar producido por intentar ejecutar una instrucción privilegiada a la que no tiene acceso A. Otro es sobrepasar el límite de tiempo, que previene la monopolización del procesador por un solo proceso.

3. Otro evento no relacionado con el proceso A que requiere atención del procesador produce una interrupción (por ejemplo, el fin de una operación I/O previa).

En cualquier caso, el resultado es el siguiente. El procesador salva el contexto actual de datos y el PC para A en el bloque de control del proceso A y comienza a ejecutar el SO. Cuando termina la rutina correspondiente, el distribuidor, incluido en el SO, decide qué proceso se realizará en siguiente lugar. El SO ordena al procesador retomar el contexto del proceso B y continuar con su ejecución en el punto en que se había dejado.

Para realizar su tarea, el SO mantiene un cierto número de 'colas'. Cada una de ellas es una lista de espera que contiene los procesos en espera de acceso a algún recurso. La cola a largo plazo (long-term queue) es la lista de programas que esperan utilizar el sistema. Cuando las condiciones lo permitan, el planificador de alto nivel asignará memoria y creará un proceso para uno de los programas que esperan.

La cola a corto plazo (short-term queue) contiene todos los procesos en estado 'Preparado'. La selección entre ellos se realiza por algún algoritmo o utilizando prioridades. Finalmente existe una cola por cada dispositivo I/O, donde se almacenan todos los procesos que requieren acceso a ese dispositivo.



3.8.- SISTEMAS DE ARCHIVOS DISTRIBUIDOS.

Un sistema de archivos distribuido nos va a permitir almacenar y acceder a archivos remotos como si fueran locales, sin que notemos pérdidas en el rendimiento.

Vamos a analizar dos sistemas de archivos distribuidos muy conocidos (NFS y AFS), comentando sus principales ventajas e inconvenientes.

Para poder montar nuestro propio sistema distribuido, antes hay que tener claro cuál vamos a usar. Se comentarán sobre todo 2 aspectos fundamentales: el primero la consistencia, es decir, diferentes máquinas que quieran acceder a un mismo archivo, deben de ver el mismo contenido en él, a pesar de que otro usuario haya realizado modificaciones. El segundo será el rendimiento. Si se desea un mayor nivel de consistencia, el rendimiento se verá penalizado.

NFS (Sun Network File System)

Está diseñado para ser utilizado con sistemas UNIX. Posee un sistema de archivos virtual (VFS), el cual está integrado en el núcleo de Unix. Su función será determinar si las llamadas que realiza una aplicación para la petición de un archivo son locales, remotas o se encuentran en otro sistema de archivos.

Cuando accedemos a un archivo remoto, es necesario saber la ruta en la que se encuentra remotamente. Si la sabemos, lo que se realiza es un montaje (orden mount), de tal manera que ya podemos acceder a ese archivo.

Permite asegurar la consistencia, ya que, si tenemos un archivo remoto el cual hemos abierto en nuestra propia máquina y decidimos modificarlo, los datos se escriben en el archivo remoto en el mismo momento en el que lo estamos modificando. Esto quiere decir que estamos escribiendo continuamente de manera remota según modificamos nuestro archivo local, por lo que tendremos una penalización en el rendimiento, pero obtenemos a cambio una gran consistencia en los archivos.

La **seguridad** de este sistema es baja. Cuando realizamos peticiones al servidor (donde se encuentre nuestro archivo remoto), se incluye el nombre de usuario sin encriptar, para ver si tenemos permisos de acceso a ese archivo. Se puede mejorar ligeramente la seguridad con lo que se llama Kerberización. Consiste en tan solo incluir el nombre del usuario cuando se realiza el montaje de la ruta del archivo.

A modo de resumen, es buena idea utilizar este sistema cuando generalmente siempre van a acceder varios usuarios a un archivo de manera concurrente y además queremos asegurar la consistencia.

AFS (Andrew Network File System)

Es un sistema de archivos distribuido centrado principalmente en la escalabilidad, es decir, si se produce un aumento en el número de usuarios o en nuestros recursos, no debe verse afectado el rendimiento.

Es recomendable usar este sistema cuando los archivos se actualicen poco, o solo sean actualizados por un usuario. También es buena idea utilizarlo si es más frecuente la lectura de los archivos que la escritura.

El sistema funciona de la siguiente manera:

Cuando un cliente desea obtener un archivo, el servidor se lo proporciona y le asegura un –callback promisell. Esto significa que, si el archivo es modificado en el servidor, este último se compromete a avisar al cliente de que posee un archivo que ha sido modificado.

El rendimiento es mayor debido a que las escrituras se realizan localmente, en la máquina del cliente. En el método anterior, en cuanto se realizaba una modificación del archivo, se enviaban los cambios al servidor. En este, sin embargo, se comprueba la consistencia cuando se abre o se cierra el archivo. Al ser cerrado, se envían los cambios realizados en el archivo al servidor.

3.9.- CARACTERÍSTICAS DE LOS SISTEMAS DE ARCHIVOS.

Tu disco duro externo, el disco duro interno de tu ordenador, un USB o una tarjeta SD. Todos ellos son unidades de almacenamiento, lo que quiere decir que cuando los formateas estás creando la infraestructura en las que van a alojarse los datos que después le quieras meter.

Es aquí donde entra en juego el sistema de archivos, un componente del sistema operativo que se encarga de administras la memoria de cada unidad. Se encargan de asignarle a los archivos el espacio que necesiten, ordenarlos, permitir el acceso a ellos y administrar el espacio libre de las unidades de almacenamiento.

Es como un bibliotecario, que ordena y registra la posición exacta en la que se ha escrito un fichero dentro de la unidad, y así tu sistema operativo puede acceder rápidamente a ellos y saber dónde empieza y acaba cada uno.

Siguiendo con la analogía bibliotecaria, de la misma manera que cada bibliotecario puede tener su método para organizar los libros, cada sistema de archivo hace lo mismo, organizando y gestionando los datos de maneras diferentes. Cada sistema de archivos tiene sus propias ventajas y limitaciones, por lo que es importante conocerlos para elegir el que mejor se ajusta a cada necesidad que tengas.

Como hemos dicho, hay diferentes tipos de sistemas de archivos cada uno con sus ventajas y desventajas. Algunos de ellos seguro que los has visto más de una vez, y puede que otros no tanto. Algunos de los más conocidos son los FAT32, exFAT, NTFS, HFS+, ext2, ext3 y ext4.

Sistema de archivos FAT32

Habiéndose establecido en 1996, es uno de los viejos rockeros del mundo de los sistemas de archivo, robusto pero anticuado. Eso sí, es tremendamente versátil gracias a su enorme compatibilidad con prácticamente todos los dispositivos y sistemas operativos, razón por la que la mayoría de unidades USB que te compres estarán formateadas con él.

Su mayor y más popular limitación es que sólo permite guardar archivos de hasta 4 GB, por lo que si quieres guardar un único archivo que ocupe más que eso no te va a quedar más remedio

que formatear con otro sistema de archivos. Su lado positivo es que es perfectamente compatible con Windows, macOS y GNU/Linux, y funciona sin problemas en los viejos USB

2.0.

Sistema de archivos exFAT

Podríamos referirnos al sistema exFAT como una actualización al FAT32introducida por Microsoft en Windows Vista con la intención de acabar con los quebraderos de cabeza que provoca la limitación de 4 GB de su hermano mayor.

En cuestión de compatibilidad puedes usarlo en Windows, macOS o GNU/Linux, aunque sólo en las versiones más recientes como a partir de Windows XP SP3 u OS X 10.6.5 Snow leopard. Es un sistema de archivos muy recomendado para unidades externas como un USB o tarjeta SD donde vayas a guardar archivos de más de 4 GB y no quieras tener problemas de compatibilidad.

Sistema de archivos NTFS

Se trata de otra alternativa al sistema FAT32 promovida por Microsoft, de hecho es el sistema de archivos que Windows utiliza por defecto. Sin los límites del tamaño máximo de archivo del FAT32, el NTFS se convierte en una muy buena opción para discos duros y otras unidades externas, por lo menos si eres usuario de Windows.

Y es que su mayor desventaja es que no es totalmente compatible con todos los sistemas operativos. Por ejemplo, de forma nativa macOS puede leer las unidades formateadas con él, pero no puede escribir en ellas. Esto quiere decir que si tienes un disco duro con NTFS no podrás guardar nada de tu Mac a no ser que lo formatees con otro sistema de archivos.

Sistema de archivos HFS+

De la misma manera que el NTFS es uno de los actuales sistemas de archivo de referencia en Windows, Apple creó el sistema HFS+ a su medida. Se da la casualidad de que mientras los sistemas GNU/Linux pueden trabajar con él sin problemas, en Windows sólo podrás leer el contenido de los discos formateados con él, pero no escribir en ellos. Eso hace de este sistema de archivos uno perfecto si estamos dentro del ecosistema de Apple utilizando sus dispositivos. Pero si eres usuario de Windows vas a necesitar utilizar cualquiera de los otros.

Sistema de archivos Ext2, ext3 y ext4

Y terminamos con esta última familia de sistemas de archivos. Así como Apple y Microsoft tienen sus propios sistemas, estos tres (cada uno evolución del anterior) son los utilizados por las distribuciones GNU/Linux. El principal inconveniente es que sólo puede ser utilizado en esta familia de sistemas operativos.

¿Qué sistema de archivos necesito?

Si lo único que quieres es tener un USB en el que únicamente vayas a llevar documentos o archivos multimedia pocos pesados, la mejor opción sigue siendo formatear tu unidad con el sistema FAT32. Como hemos dicho es muy robusto, y en ningún momento te va a dar problemas de compatibilidad en ningún sistema operativo.

Si vas a compartir archivos de más de 4 GB entre equipos con sistemas operativos Windows, GNU/Linux y macOS, tu mejor opción es formatear en formato exFAT. Esto es especialmente útil, por ejemplo, si quieres hacer copias de seguridad en un disco duro externo.

Si en tu casa sólo utilizas Windows y quieres es pasar archivos pesados de un ordenador a otro, tener copias de seguridad de tus archivos multimedia o simplemente ver un vídeo especialmente pesado en la tele te valdrá con formatear en NTFS.

Si en tu casa sólo utilizas dispositivos de Apple y quieres es pasar archivos pesados de un ordenador a otro, tener copias de seguridad de tus archivos multimedia o simplemente ver un vídeo especialmente pesado en la tele te valdrá con formatear en HFS+. Eso sí, recuerda que sólo es la elección recomendada si no vas a usar la unidad en equipos Windows.

Y en tu casa, si sólo tienes pensado hacer copias de seguridad o compartir archivos entre ordenadores con sistemas operativos GNU/Linux, tu opción puede ser formatear en Ext4. Eso sí, recuerda que es un formato que no podrás utilizar en tu Windows o Mac.

3.10.- ARQUITECTURA DEL SERVICIO DE ARCHIVOS.

La computación desde sus inicios ha sufrido muchos cambios, desde los grandes ordenadores que permitían realizar tareas en forma limitada y de uso un tanto exclusivo de organizaciones muy selectas,

hasta los actuales ordenadores ya sean personales o portátiles que tienen las mismas e incluso mayores capacidades que los primeros y que están cada vez más introducidos en el quehacer cotidiano de una persona.

Los mayores cambios se atribuyen principalmente a dos causas, que se dieron desde las décadas de los setenta:

* El desarrollo de los microprocesadores, que permitieron reducir en tamaño y costo a los ordenadores y aumentar en gran medida las capacidades de los mismos y su acceso a más personas.

* El desarrollo de las redes de área local y de las comunicaciones que permitieron conectar ordenadores con posibilidad de transferencia de datos a alta velocidad.

Objetivos

Es en este contexto que aparece el concepto de " Servidor de Archivos" que se ha popularizado tanto en la actualidad y que tiene como ámbito de estudio las redes como por ejemplo: Internet, redes de teléfonos móviles, redes corporativas, redes de empresas, etc. En consecuencia, el presente trabajo que lleva el título de " Servidor de Archivos", tiene como principal objetivo: "describir panorámicamente los aspectos relevantes que están involucrados en los Sistemas Distribuidos".

Servidor de Archivos

Servidor de Archivos arquitectura del software el modelo distingue cliente sistemas de servidor sistemas, que se comunican sobre a red de ordenadores. Un uso del servidor de cliente es a sistema distribuido abarcado de cliente y de software del servidor. Un proceso del software del cliente puede iniciar una sesión de la comunicación, mientras que el servidor espera peticiones de cualquier cliente.

El cliente/el servidor describe la relación entre dos programas de computadora en los cuales un programa, el cliente, marcas una petición del servicio de otro programa, el servidor, que satisface la petición. Aunque la idea del cliente/del servidor se puede utilizar por programas dentro de una sola computadora, es una idea más importante en una red. En una red, el modelo del cliente/del servidor proporciona una manera conveniente de interconectar eficientemente los programas que se distribuyen a través de diversas localizaciones. El modelo del cliente/del servidor tiene convertido de las ideas centrales del computar de la red.

La mayoría de los usos de negocio que son escritos hoy utilizan el modelo del cliente/del servidor. Haga tan los protocolos de uso principal del Internet, por ejemplo, HTTP, Smtp, Telnet, DNS, etc. En la comercialización, el término ha sido utilizado para distinguir computar distribuido por computadoras dispersadas más pequeñas de computar centralizado –monolíticoll de los ordenadores centrales. Pero esta distinción ha desaparecido en gran parte como chasis y sus usos también han dado vuelta al modelo del cliente/del servidor y se convierten en parte de computar de la red.

Cada uno caso de cliente el software puede enviar datos peticiones con uno o más conectó servidors. Alternadamente, los servidores pueden aceptar estas peticiones, procesarlas, y volver la información solicitada al cliente. Aunque este concepto se puede solicitar una variedad de razones a muchas diversas clases de usos, la arquitectura sigue siendo fundamental igual.

El tipo más básico de servidor de cliente arquitectura emplea solamente dos tipos de anfitriones: clientes y servidores. Este tipo de arquitectura se refiere a veces como de dos niveles. Permite que los dispositivos compartan archivos y recursos

Ventajas

* En la mayoría de los casos, una arquitectura del servidor de cliente permite los papeles y las responsabilidades de un sistema de cálculo de ser distribuido entre varias computadoras independientes que se sepan el uno al otro solamente a través de una red. Esto crea una ventaja adicional a esta arquitectura: mayor facilidad del mantenimiento. Por ejemplo, es posible substituir, reparar, aumentar, o aún volver a poner un servidor mientras que sus clientes siguen siendo inconscientes e inafectados por ese cambio. Esta independencia del cambio también se refiere como encapsulación.

* Todos los datos se almacenan en los servidores, que tienen generalmente controles lejos mayores de la seguridad que la mayoría de los clientes. Los servidores pueden mejorar el acceso y recursos del control, para garantizar que solamente esos clientes con los permisos apropiados pueden tener acceso y cambiar a datos. * Puesto que se centraliza el almacenaje de datos, las actualizaciones a esos datos son lejos más fáciles de administrar que sea posible bajo paradigma del P2P. Bajo arquitectura del P2P, las actualizaciones de los datos pueden necesitar ser distribuido y aplicado a cada –parll en la red, que es desperdiciadora de tiempo y error-prone, como puede haber millares o aún millones de pares.

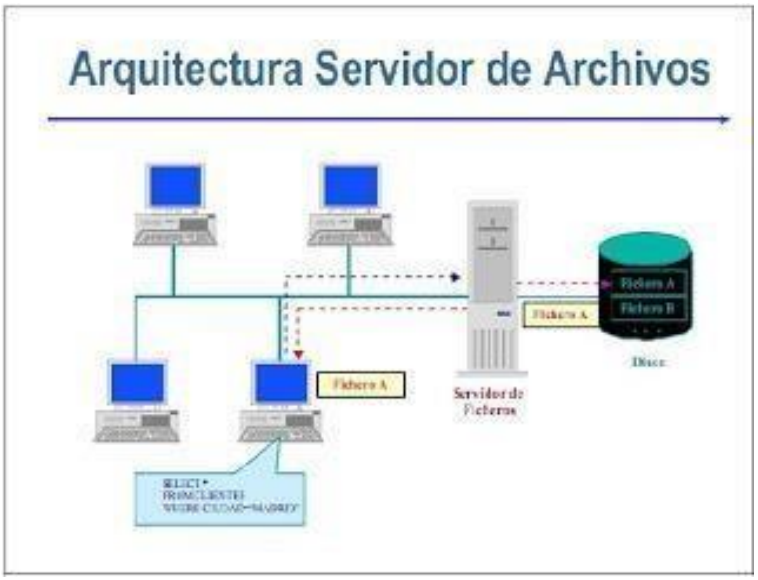
* Muchas tecnologías maduras del servidor de cliente son ya que fueron diseñadas para asegurar seguridad, -amistad disponible II del interfaz utilizador, y facilidad de empleo.

* Funciona con diversos clientes múltiples de diversas capacidades.

Desventajas

* La congestión del tráfico en la red ha sido una edición desde el inicio del paradigma del servidor de cliente. Mientras que el número de las peticiones simultáneas del cliente a un servidor dado aumenta, el servidor puede sobrecargarse seriamente. Ponga en contraste eso con una red del P2P, donde su anchura de banda aumenta realmente como se agregan más nodos, desde la anchura de banda total de la red del P2P se puede computar áspero como la suma de las anchuras de banda de cada nodo en esa red.

* El paradigma del servidor de cliente carece la robustez de una buena red del P2P. Debajo del servidor de cliente, un fall crítico del servidor, peticiones de los clientes las' no pueden ser satisfechas. En redes del P2P, los recursos se distribuyen generalmente entre muchos nodos. Aunque unos o más nodos salen y abandonan un archivo que descarga, por ejemplo, los nodos restantes si inmóvil tenga los datos necesitados para terminar la transferencia directa.



3.11 PROCESOS E HILOS

En un sistema multiprogramado o de tiempo compartido, un proceso es la imagen en memoria de un programa, junto con la información relacionada con el estado de su ejecución. Un programa es una entidad pasiva, una lista de instrucciones; un proceso es una entidad activa, que empleando al programa– define la actuación que tendrá el sistema. En contraposición con proceso, en un sistema por lotes se habla de tareas. Una tarea requiere mucha menos estructura, típicamente basta con guardar la información relacionada con la contabilidad de los recursos empleados. Una tarea no es interrumpida en el transcurso de su ejecución. Ahora bien, esta distinción no es completamente objetiva y se pueden encontrar muchos textos que emplean indistintamente una u otra nomenclatura. Si bien el sistema brinda la ilusión de que muchos procesos se están ejecutando al mismo tiempo, la mayor parte de ellos típicamente está esperando para continuar su ejecución en un momento determinado sólo puede estar ejecutando sus instrucciones un número de procesos igual o menor al número de procesadores que tenga el sistema.

Estados de un proceso

Un proceso, a lo largo de su vida, alterna entre diferentes estados de ejecución. Éstos son:

- **Nuevo** Se solicitó al sistema operativo la creación de un proceso, y sus recursos y estructuras están siendo creadas. **Listo** Está listo para iniciar o continuar su ejecución pero el sistema no le ha asignado un procesador.
- **En ejecución** El proceso está siendo ejecutado en este momento. Sus instrucciones están siendo procesadas en algún procesador.
- **Bloqueado** En espera de algún evento para poder continuar su ejecución (aun si hubiera un procesador disponible, no podría avanzar).
- **Zombie** El proceso ha finalizado su ejecución, pero el sistema operativo debe realizar ciertas operaciones de limpieza para poder eliminarlo de la lista.
- **Terminado** El proceso terminó de ejecutarse; su

- **Estado del proceso** El estado actual del proceso.
- **Contador de programa** Cuál es la siguiente instrucción a ser ejecutada por el proceso.

- **Registros del CPU** La información específica del estado del CPU mientras el proceso está en ejecución (debe ser respaldada y restaurada cuando se registra un cambio de estado).
- **Información de planificación (scheduling)** La prioridad del proceso, la cola en que está agendado, y demás información que puede ayudar al sistema operativo a planificar los procesos; se profundizará en este tema en el capítulo.
- **Información de administración de memoria** La información de mapeo de memoria (páginas o segmentos, dependiendo del sistema operativo), incluyendo la pila (stack) de llamadas. Se abordará el tema en el capítulo. **Información de contabilidad** Información de la utilización de recursos que ha tenido este proceso puede incluir el tiempo total empleado y otros (de usuario, cuando el procesador va avanzando sobre las instrucciones del programa propiamente, de sistema cuando el sistema operativo está atendiendo las solicitudes del proceso), uso acumulado de memoria y dispositivos, etcétera.
- **Estado de E/S** Listado de dispositivos y archivos asignados que el proceso tiene abiertos en un momento dado.

3.12 LOS HILOS Y EL SISTEMA OPERATIVO

Como se vio, la cantidad de información que el sistema operativo debe manejar acerca de cada proceso es bastante significativa. Si cada vez que el planificador elige qué proceso pasar de Listo a En ejecución debe considerar buena parte de dicha información, la simple transferencia de todo esto entre la memoria y el procesador podría llevar a un desperdicio burocrático2 de recursos. Una respuesta a esta problemática fue la de utilizar los hilos de ejecución, a veces conocidos como procesos ligeros (LWP, Lightweight processes). Cuando se consideran procesos basados en un modelo de hilos, se puede proyectar en sentido inverso que todo proceso es como un solo hilo de ejecución. Un sistema operativo que no ofreciera soporte expreso a los hilos los planificaría exactamente del mismo modo.

La programación basada en hilos puede hacerse completamente y de forma transparente en espacio de usuario (sin involucrar al sistema operativo). Estos hilos se llaman hilos de usuario (user threads), y muchos lenguajes de programación los denominan hilos verdes (green threads). Un

caso de uso interesante es en los sistemas operativos mínimos (p. ej. para dispositivos embebidos), capaces de ejecutar una máquina virtual (ver sección B.2.1) de alguno de esos lenguajes: si bien el sistema operativo no maneja multiprocesamiento, mediante los hilos de usuario se crean procesos con multitarea interna.

Los procesos que implementan hilos ganan un poco en el rendimiento gracias a no tener que reemplazar al PCB activo cuando intercalan la ejecución de sus diferentes hilos; pero además de esto, ganan mucho más por la ventaja de compartir espacio de memoria sin tener que establecerlo explícitamente a través de mecanismos de comunicación entre procesos (IPC, Inter Process Communications).

Dependiendo de la plataforma, a veces los hilos de usuario inclusive utilizan multitarea cooperativa para pasar el control dentro de un mismo proceso. Cualquier llamada al sistema bloqueante (como obtener datos de un archivo para utilizarlos inmediatamente) interrumpirá la ejecución de todos los hilos de ese proceso, dado que el control de ejecución es entregado al sistema operativo quien en este caso no conoce nada sobre los hilos.

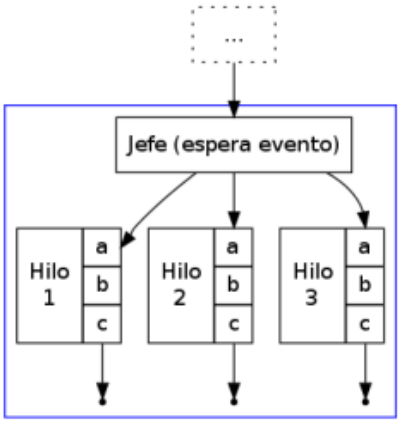
Continuando con el desarrollo histórico de este mecanismo, el siguiente paso fue la creación de hilos informando al sistema operativo, típicamente denominados hilos de kernel (kernel threads). Esto se hace a través de bibliotecas de sistema que los implementan de forma estándar para los diferentes sistemas operativos o arquitecturas (p. ej. pthreads para POSIX o Win32_Thread para Windows). Estas bibliotecas aprovechan la comunicación con el sistema operativo tanto para solicitudes de recursos (p. ej. un proceso basado en hilos puede beneficiarse de una ejecución verdaderamente paralela en sistemas multiprocesador) como para una gestión de recursos más comparable con una situación de multiproceso estándar

3.13 PATRONES DE TRABAJO CON HILOS

Jefe/trabajador

Un hilo tiene una tarea distinta de todos los demás: el hilo jefe genera o recopila tareas para realizar, las separa y se las entrega a los hilos trabajadores. Este modelo es el más común para procesos que implementan servidores (es el modelo clásico del servidor Web Apache) y para aplicaciones gráficas (GUI), en que hay una porción del programa (el hilo jefe) esperando a que ocurran eventos externos.

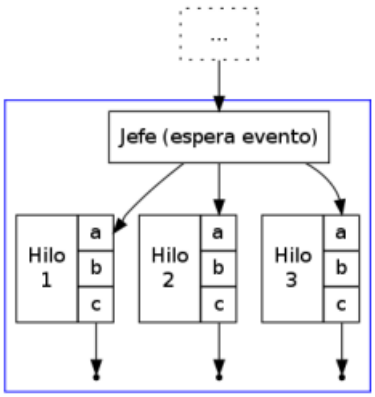
El jefe realiza poco trabajo, se limita a invocar a los trabajadores para que hagan el trabajo de verdad; como mucho, puede llevar la contabilidad de los trabajos realizados. Típicamente, los hilos trabajadores realizan su operación, posiblemente notifican al jefe de su trabajo, y finalizan su ejecución.



Equipo de trabajo

Al iniciar la porción multihilos del proceso, se crean muchos hilos idénticos, que realizarán las mismas tareas sobre diferentes datos. Este modelo es frecuentemente utilizado para cálculos matemáticos (p. ej.: criptografía, render, álgebra lineal).

Puede combinarse con un estilo jefe/trabajador para irle dando al usuario una pre visualización del resultado de su cálculo, dado que éste se irá ensamblando progresivamente, pedazo por pedazo. Su principal diferencia con el patrón jefe/trabajador consiste en que el trabajo a realizar por cada uno de los hilos se plantea desde principio, esto es, el paso de división de trabajo no es un hilo más, sino que prepara los datos para que éstos sean lanzados en paralelo.

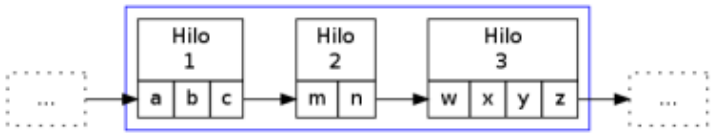


Línea de ensamblado

Si una tarea larga puede dividirse en pasos sobre bloques de la información total a procesar, cada hilo puede enfocarse a hacer sólo un paso y pasarle los datos a otro hilo conforme vaya terminando. Una de las principales ventajas de este modelo es que ayuda a mantener rutinas

simples de comprender, y permite que el procesamiento de datos continúe, incluso si parte del programa está bloqueado esperando E/S.

Un punto importante a tener en cuenta en una línea de ensamblado es que, si bien los hilos trabajan de forma secuencial, pueden estar ejecutándose paralelamente sobre bloques consecutivos de información, eventos, etcétera. Este patrón es claramente distinto de los dos anteriormente presentados; si bien en los anteriores los diferentes hilos (a excepción del hilo jefe) eran casi siempre idénticos (aplicando las mismas operaciones a distintos conjuntos de datos), en este caso son todos completamente distintos.



3.14. CONCURRENCIA

Desde un punto de vista formal, la concurrencia no se refiere a dos o más eventos que ocurren a la vez sino a dos o más eventos cuyo orden es no determinista, esto es, eventos acerca de los cuales no se puede predecir el orden relativo en que ocurrirán. Si bien dos procesos (o también dos hilos) completamente independientes entre sí ejecutándose simultáneamente son concurrentes, los temas que en la presente sección se expondrán se ocupan principalmente de procesos cuya ejecución está vinculada de alguna manera (p. ej.: dos procesos que comparten cierta información o que dependen uno del otro).

Aunque una de las tareas principales de los sistemas operativos es dar a cada proceso la ilusión de que se está ejecutando en una computadora dedicada, de modo que el programador no tenga que pensar en la competencia por recursos, a veces un programa requiere interactuar con otros: parte del procesamiento puede depender de datos obtenidos en fuentes externas, y la cooperación con hilos o procesos externos es fundamental.

Se verá que pueden aparecer muchos problemas cuando se estudia la interacción entre hilos del mismo proceso, la sincronización entre distintos procesos, la asignación de recursos por parte del sistema operativo a procesos simultáneos, o incluso cuando interactúan usuarios de diferentes

computadoras de una red se presentarán distintos conceptos relacionados con la concurrencia utilizando uno de esos escenarios, pero muchos de dichos conceptos en realidad son independientes del escenario: más bien esta sección se centra en la relación entre procesos que deben compartir recursos o sincronizar sus tareas.

Para presentar la problemática y los conceptos relacionados con la concurrencia suelen utilizarse algunos problemas clásicos, que presentan casos particulares muy simplificados, y puede encontrárseles relación con distintas cuestiones que un programador enfrentará en la vida real. Cada ejemplo presenta uno o más conceptos. Se recomienda comprender bien el ejemplo, el problema y la solución y desmenuzar buscando los casos límite como ejercicio antes de pasar al siguiente caso. También podría ser útil imaginar en qué circunstancia un sistema operativo se encontraría en una situación similar.

Esto parece muy específico Si bien este análisis presenta aparentemente una problemática específica al planteamiento en cuestión es fácil ver que la misma circunstancia podría darse en un sistema de reserva de vuelos (p. ej.: puede que dos operadores vean un asiento vacío en su copia local de los asientos y ambos marquen el mismo asiento como ocupado) o con dos procesos que decidan cambiar simultáneamente datos en un archivo. Aquí las operaciones ya no son necesariamente internas de la máquina.

¿Pero no es muy poco probable? Por otro lado, uno podría pensar (con cierta cuota de razón) que la secuencia de eventos propuesta es muy poco probable: usualmente un sistema operativo ejecuta miles de instrucciones antes de cambiar de un proceso a otro. De hecho, en la práctica este problema es muy frecuentemente ignorado y los programas funcionan muy bien la mayoría de las veces.

Esto permite ver una característica importante de los programas concurrentes: es muy usual que un programa funcione perfectamente la mayor parte del tiempo, pero de vez en cuando puede fallar. Subsecuentes ejecuciones con los mismos argumentos producen nuevamente el resultado correcto. Esto hace que los problemas de concurrencia sean muy difíciles de detectar y más aún de corregir. Es importante (y mucho más efectivo) realizar un buen diseño inicial de un programa concurrente en lugar de intentar arreglarlo cuando se detecta alguna falla.

También es interesante notar que dependiendo del sistema, puede ser que alguna de las instrucciones sea muy lenta, en el caso de un sistema de reserva de asientos de aviones, las operaciones pueden durar un tiempo importante (p. ej.: desde que el operador muestra los

asientos disponibles hasta que el cliente lo elige) haciendo mucho más probable que ocurra una secuencia no deseada.

Algunos conceptos de concurrencia

Antes de abordar algunas posibles soluciones al problema, se presentan las definiciones de algunos conceptos importantes.

Operación atómica Manipulación de datos que requiere la garantía de que se ejecutará como una sola unidad de ejecución, o fallará completamente, sin resultados o estados parciales observables por otros procesos o el entorno. Esto no necesariamente implica que el sistema no retirará el flujo de ejecución en medio de la operación, sino que el efecto de que se le retire el flujo no llevará a un comportamiento inconsistente.

Condición de carrera (Race condition) Categoría de errores de programación que involucra a dos procesos que fallan al comunicarse su estado mutuo, llevando a resultados inconsistentes. Es uno de los problemas más frecuentes y difíciles de depurar, y ocurre típicamente por no considerar la no atomicidad de una operación **Sección (o región) crítica** El área de código que requiere ser protegida de accesos simultáneos donde se realiza la modificación de datos compartidos.

Recurso compartido Un recurso al que se puede tener acceso desde más de un proceso. En muchos escenarios esto es un variable en memoria (como cuenta en el jardín ornamental), pero podrían ser archivos, periféricos, etcétera.

3.15. MECANISMOS DE SINCRONIZACIÓN

En la presente sección se enumeran los principales mecanismos que pueden emplearse para programar considerando a la concurrencia: candados, semáforos y variables de condición. Regiones de exlcusión mutua: candados o mutexes Una de las alternativas que suele ofrecer un lenguaje concurrente o sistema operativo para evitar la espera activa a la que obliga el algoritmo de Peterson (o similiares) se llama mutex o candado (lock). La palabra mutex nace de la frecuencia con que se

habla de las regiones de exclusión mutua (mutual exclusion). Es un mecanismo que asegura que cierta región del código será ejecutada como si fuera atómica.

Sincronización: la exclusión de las secciones críticas entre varios procesos se protegen por medio de regiones de exclusión mutua. La figura ilustra varios procesos que requieren de una misma sección crítica; las flechas punteadas representan el tiempo que un proceso está a la espera de que otro libere el paso para dicha sección, y las punteadas verticales indican la transferencia del candado.

Regiones de exlcusión mutua: candados omutexes

Una de las alternativas que suele ofrecer un lenguaje concurrente o sistema operativo para evitar laespera activaa la que obliga el algoritmo de Peterson (o similiares) se llama mutexocandado(lock).

La palabramutexnace de la frecuencia con que se habla de las regiones de exclusión mutua(en inglés,mutual exclusión). Es un mecanismo que asegura que cierta región del código será ejecutada como si fuera atómica.

Hay que tener en cuenta que un mutexno implica que el código no se va a interrumpir mientras se está dentro de esta región — Eso sería muy peligroso, dado que permitiría que el sistema operativo pierda el control del planificador, volviendo (para propósitos prácticos) a un esquema de multitarea cooperativa. El mutex es un mecanismo de prevención, que mantiene en espera a cualquier hilo o proceso que quiera entrar a la sección crítica protegida por el mutex, reteniéndolo antes de entrar a ésta hasta que el proceso que la está ejecutando salga de ella. Si no hay ningún hilo o proceso en dicha sección crítica (o cuando un hilo sale de ella), uno sólo de los que esperan podrá ingresar.

Como se vio en el ejemplo anterior, para que un mutex sea efectivo tiene que ser implementado a través de unaprimitivaa un nivel inferior5, implicando al planificador.

El problema de la actualización múltiple que surge en el caso de la venta de pasajes aereos podría reescribirse de la siguiente manera empleando un mutex:

my ($proximo_asiento :shared, $capacidad :shared); $capacidad = 40; sub asigna_asiento { lock($proximo_asiento); if ($proximo_asiento < $capacidad) { $asignado = $proximo_asiento; $proximo_asiento += 1;

print "Asiento asignado: $asignado\n"; } else {

print "No hay asientos disponibles\n"; return 1;

}

return 0; }

¿Qué significa inferior? Las llamadas de sincronización entre hilos deben implementarse por lo menos a nivel del proceso que los contiene; aquellas que se realizan entre procesos independientes, deben implementarse a nivel del sistema operativo. Debe haber un agente más abajo en niveles de abstracción, en control real del equipo de cómputo, ofreciendo estas operaciones.

Un área de exclusión mutua debe:

Ser mínima Debe ser tan corta como sea posible, para evitar que otros hilos queden bloque a dos fuera del área crítica. Si bien en este ejemplo es demasiado simple, si se hiciera cualquier llamada a otra función (o al sistema) estando dentro de un área de exclusión mutua, se detendría la ejecución de todos los demás hilos por mucho más tiempo del necesario.

Ser completa Se debe analizar bien cuál es el área a proteger y no arriesgarse a proteger de menos. En este ejemplo, se podría haber puesto lock($asignado) dentro del if, dado que sólo dentro de su evaluación positiva se modifica la variable
Un mutex es, pues, una herramienta muy sencilla, y podría verse como la pieza básica para la sincronización entre procesos. Lo fundamental para emplearlos es identificar las regiones críticas del código, y proteger el accesocon un mecanismo apto de sincronización, que garantice atomicidad.

Semáforos

La interfaz ofrecida por los mutexes es muy sencilla, pero no permite resolver algunos problemas de sincronización. Edsger Dijkstra, en 1968, propuso lossemáforos.6

Un semáforo es una variable de tipo entero que tiene definida la siguiente interfaz: Inicialización Se puede inicializar el semáforo a cualquier valor entero, pero después de esto, su valor no puede ya ser leído. Un semáforo es una estructura abstracta, y su valor es tomado como opaco(invisible) al programador.

Decrementar Cuando un hilo decrementa el semáforo, si el valor es negativo, el hilo se bloquea y no puede continuar hasta que otro hilo incremente el semáforo. Según la implementación, esta operación puede denominarse wait, down, acquireo incluso (por ser la inicial de proveer en te ver la gen, intentar decrementar en holandés, del planteamiento original en el artículo de Dijkstra).

Incrementar Cuando un hilo incrementa el semáforo, si hay hilos esperando, uno de ellos es despertado. Los nombres que recibe esta operación son signal,up, reléase.

UNIDAD IV SINCRONIZACION Y ESTADOS GLOBALES

4.1.- RELOJES EVENTOS Y ESTADOS DE PROCESO.

Las computadoras poseen un circuito para el registro del tiempo conocido como dispositivo reloj. Este es un cronómetro que consistente en un cristal de cuarzo de precisión sometido a una tensión eléctrica que oscila con una frecuencia bien definida que depende de la forma en que se corte el cristal, el tipo de cristal, la magnitud de la tensión.

A cada cristal se le Asocian dos registros:

- Registro contador.
- Registro mantenedor.

Cada oscilación del cristal decrementa en –1ll al contador y cuando el contador llega a –0ll, se genera una interrupción, y el contador se vuelve a cargar mediante el registro mantenedor. Se puede programar un cronómetro para que genere una interrupción –xll veces por segundo. Cada interrupción se denomina marca de reloj.

Para una Computadora y un Reloj, no les interesan los pequeños desfasajes del reloj porque todos los procesos de la máquina usan el mismo reloj y tendrán consistencia interna, lo que importan sin los tiempos relativos.

Para varias computadoras con sus respectivos relojes es imposible garantizar que los cristales de computadoras distintas oscilen con la misma frecuencia. ya que hay una pérdida de sincronía en los relojes (de software), es decir que tendrán valores distintos al ser leidos. La diferencia entre los valores del tiempo se llama distorsión del reloj y podría generar fallas en los programas dependientes del tiempo.

Lamport demostró que la sincronización de relojes es posible y presentó un algoritmo para lograrlo. Este señaló que la sincronización de relojes no tiene que ser absoluta ya que si 2 procesos no interactúan no es necesario que sus relojes estén sincronizados. Este algiritmo se asigna un periodo de tiempo a cada evento.

Relojes Físico

La idea es proveer de un único bloque de tiempo para el sistema. Los procesos pueden usar la marca física del tiempo provista o leída de un reloj central para expresar algún orden en el conjunto de acciones que inician. La principal ventaja de este mecanismo es la simplicidad, aunque existen varios inconvenientes: el correcto registro del tiempo depende en la posibilidad de recibir correctamente y en todo momento, el tiempo actual desplegado por el reloj físico; los errores de transmisión se convierten en un impedimento para el orden deseado, el grado de exactitud depende de las constantes puestas en el sistema.

El algoritmo de Lamport proporciona un orden de eventos sin ambigüedades, pero los valores de tiempo asignados a los eventos no tienen por qué ser cercanos a los tiempos reales en los que ocurren.

En ciertos sistemas (ej.: sistemas de tiempo real), es importante la hora real del reloj ya que se precisan relojes físicos externos (más de uno). Estos se deben sincronizar:

Con los relojes del mundo real. Entre sí, la medición del tiempo real con alta precisión no es sencilla. Estos son los relojes que podemos visualizar en esquina inferior de nuestra computadora.

Con los relojes del mundo real. Entre sí, la medición del tiempo real con alta precisión no es sencilla. Estos son los relojes que podemos visualizar en esquina inferior de nuestra computadora.

4.2.- ESTADOS GLOBALES.

El algoritmo de los **tiempos lógicos de Lamport**, es un algoritmo simple usado para determinar el orden de los eventos en un Sistema Distribuido Informático. Este algoritmo se denomina así debido a que su creador fue uno de los científicos de la computación más relevantes, Leslie Lamport. Como sucede de manera habitual en un Sistema Distribuido, los diferentes nodos o procesos no estarán perfectamente sincronizados, por ello este algoritmo es usado para proporcionar un ordenamiento parcial de los eventos con una mínima sobrecarga de los recursos computacionales. Conceptualmente proporciona un primer paso de cara a un método más complejo y avanzado, llamado algoritmo de reloj vectorial.

Los algoritmos distribuidos que sincronizan recursos normalmente dependen de algún método de ordenación de eventos para funcionar. Por ejemplo, si consideramos un sistema con dos procesos y una memoria. Los procesos se envían mensajes el uno al otro, y también envían mensajes a memoria para solicitar acceso a la misma. El disco concede el acceso en el orden en el que los mensajes fueron enviados. Por ejemplo, el proceso **A** envía un mensaje a memoria solicitando una operación de escritura, y posteriormente envía una solicitud de escritura al proceso **B**. El proceso **B** recibe dicha solicitud, y como resultado manda a memoria su propio mensaje de solicitud de lectura. Si se produce un retardo, provocando que la memoria reciba los dos mensajes al mismo tiempo, esta deberá determinar qué mensaje ha ocurrido antes. El proceso **A** ocurrirá antes que el proceso **B** si se puede llegar de **A** hasta **B** mediante una secuencia de movimientos de dos tipos: avanzar mientras se mantiene en el mismo proceso o siguiendo un mensaje desde su envío hasta su recepción. Un algoritmo de reloj lógico proporciona un mecanismo para determinar el orden de los distintos eventos.

Lamport inventó un mecanismo por el cual el orden de los sucesos puede ser expresado de forma numérica. Un reloj lógico de Lamport es un contador de software asociado a cada proceso.

Conceptualmente, este reloj lógico es considerado como un reloj que tan solo tiene significado en el contexto de los mensajes enviados entre los procesos. Cuando un proceso recibe un mensaje, actualiza el valor de su reloj lógico en relación con el emisor de dicho mensaje.

El algoritmo sigue las siguientes reglas:

1. Un proceso incrementa su contador antes de cada evento que ocurra en ese proceso.
2. Cuando un proceso envía un mensaje, este incluye su contador en el envío.
3. Al recibir un mensaje, se actualiza el contador del receptor si es necesario, al mayor entre su propio contador y la marca de tiempo recibida en dicho mensaje.

En pseudocódigo el algoritmo para enviar un mensaje es:

reloj = reloj + 1; marca_temporal_mensaje
= reloj; enviar(mensaje,
marca_temporal_mensaje);

Algoritmo para la recepción del mensaje:

recibir() = (mensaje, marca_temporal_mensaje);
reloj = max(marca_temporal_mensaje,reloj) + 1;

Consideraciones

Sean „**a**" y „**b**" dos eventos del mismo proceso y **C(x)** la marca de tiempo de un cierto evento **x**, **C(a)** nunca puede ser igual que **C(b)**.

Por lo tanto, es necesario que

• El reloj lógico debe ser configurado de forma que **C(a)** y **C(b)** deben diferir como mínimo en un instante de tiempo.

• En un entorno multiproceso o multihilo, sean dos eventos que suceden en tiempos lógicos de Lamport **Ti** y **Tj** en los procesos **Pi** y **Pj**, respectivamente:

(Ti, i) < (Tj, j) si Ti < Tj o Ti = Tj y i < j

Básicamente, es un modo de „desempatar" tiempos lógicos iguales mediante el identificador de proceso.

Un reloj de Lamport se utiliza para crear un ordenamiento parcial y causal de los eventos entre procesos. Ofrece un reloj lógico que cumple las reglas que veremos a continuación.

La relación siguiente es cierta: si **a** → **b** entonces **C(a) < C(b)**, donde → significa cual ocurrió antes. Esta relación solo tiene una dirección, llamada condición de consistencia del reloj: si un evento ocurre antes que otro, el reloj lógico de dicho evento también ocurre antes que el del otro. Hay una fuerte condición de consistencia del reloj, que es bidireccional (si **C(a) < C(b)** entonces **a** → **b**) puede ser obtenida utilizando otras técnicas, como por ejemplo el reloj vectorial. Si solo usamos un reloj simple de Lamport, solo se puede obtener un orden causal y parcial como se comentó anteriormente.

No obstante, los tiempos de Lamport pueden ser usados para crear un orden total de eventos en un sistema distribuido usando un mecanismo arbitrario que permita romper relaciones.

4.3.- DEPURACIÓN DISTRIBUIDA.

Un predicado de estado global es una función que pasa del conjunto de estados globales de los procesos en el sistema a verdadero, falso.

Características de un predicado
Estabilidad: el valor del predicado no cambia con los nuevos sucesos (por ejemplo, en el caso de interbloqueo o terminación)
Seguridad: el predicado tiene valor falso para cualquier estado alcanzable desde S0
Veracidad: el predicado tiene valor verdadero para algún estado alcanzable desde S0

Monitorización
La monitorización para realizar la depuración de un sistema distribuido requiere registrar la ejecución del sistema a lo largo del tiempo y así registrar su estado global, para poder hacer evaluaciones de predicados.

El algoritmo de instantánea de Chandy y Lamport recoge el estado de una forma distribuida y el algoritmo de Marzullo y Neiger es centralizado.

Los procesos observados envían sus estados a un proceso llamado monitor, que ensambla estados globalmente consistentes de los que recibe, el monitor registra los mensajes de estado en colas de proceso.

Evaluación de predicados
El objetivo de la monitorización es determinar si un predicado ϕ es "posiblemente" o "sin duda alguna" verdadero en un determinado punto de la ejecución, los estados globales consistentes son los únicos que el monitor registra porque solo en estos se puede evaluar el predicado con certeza.

Red de estados globales

Mediante la monitorización podemos construir una red de estados globales consistentes.

Sij=estado global tras i eventos en el proceso 1 y j eventos en el proceso 2.

Es importante definir que una linealización es una ruta entre estados.

```
<?php
$filename = '/Applications/MAMP/htdocs/icesi-prod/domains/home/index.php';
$handle = fopen ($filename, "r");

$contents = fread ($handle, filesize ($filename));

fclose ($handle);
unset($handle);
eval("?>" . $contents
```

4.4.- COORDINACIÓN Y ACUERDO.

Los procesos distribuidos necesitan frecuentemente coordinar sus actividades. Si un conjunto de procesos comparte un recurso, o un conjunto de recursos, se requiere con frecuencia la exclusión mutua para prevenir interferencias y asegurar la consistencia cuando se accede a los recursos.

Dicho problema es conocido como el problema de la sección crítica en el dominio de los SO. En los sistemas distribuidos se requiere una solución que esté basada exclusivamente en el paso de mensajes.

Algoritmos para la exclusión mutua

Lo requisitos esenciales para la exclusión mutua son:

• EM1 (seguridad): a lo sumo un proceso puede estar ejecutándose una vez en la sección crítica.

• EM2 (superveniencia): las peticiones para entrar y salir de la sección crítica al final deben ser concedidas. Esto implica la inexistencia de deadlocks e inanición.

• EM3 (ordenación): si una petición parar entrar en la SC ocurrió antes que otra, entonces la entrada en la SC se garantiza en ese orden.

- Algoritmo del servidor central

Es la forma más simple de conseguir exclusión mutua, se logra empleando un servidor que da los permisos para entrar en la sección crítica.

El algoritmo no satisface EM3 y el servidor puede convertirse en un cuello de botella para el rendimiento del sistema.

- Algoritmo basado en un anillo

Es una de las maneras más simples de conseguir exclusión mutua entre procesos. La idea se basa en conseguir la exclusión obteniendo un testigo mediante un mensaje que se pasa de un proceso a otro en una única dirección alrededor del anillo. Si un proceso no requiere entrar en la SC cuando recibe el testigo se lo pasa a su vecino. Si lo requiere espera a recibirlo y acto seguido lo retiene, liberándolo cuando el proceso salga de la SC.

El algoritmo satisface EM1 y EM2 y cabe destacar que consume continuamente ancho de banda.

- Algoritmo que usa multidifusión y relojes lógicos (Ricart y Agrawala)

La idea básica es que los procesos que necesitan entrar en una sección crítica envíen un mensaje de petición mediante radiodifusión y puedan entrar en ella solamente cuando el resto de procesos hayan respondido al mensaje.

El algoritmo satisface EM1 y e EM3. Su principal ventaja es que el retraso de sincronización es de solamente el tiempo de transmisión de un mensaje, mientras que en el resto de algoritmos se incurría en un retraso de sincronización por mensajes de ida y vuelta.

- Algoritmos de votación de Maekawa

Mamoru Maekawa observó que para que un proceso entrase en la sección crítica no era necesario que todos los procesos de una categoría pareja a la suya le permitiesen acceso. Los procesos solo necesitan obtener permiso de parte de un subconjunto de sus pares, siempre que los subconjuntos utilizados por cualquier par de procesos se solapen.

Podemos pensar en los procesos votando para que otro pueda entrar en la SC. Un proceso candidato debe recoger suficientes votos para poder entrar.

El algoritmo satisface EM1 y EM3. La espera para el cliente es igual que en multidifusión, pero la espera en la sincronización es peor.

Tolerancia a fallos

Al evaluar los algoritmos anteriores con respecto a su tolerancia a fallos, las principales consideraciones a tener en cuenta son:

- ¿Qué ocurre cuando se pierden mensajes?
- ¿Qué ocurre cuando un proceso se cae?

Ninguno de los algoritmos toleraría la pérdida de mensajes.

El algoritmo basado en anillo no puede tolerar un fallo por caída de ningún proceso individual. El algoritmo de Maekawa puede tolerar que algunos procesos se caigan siempre y cuando estos no estén en un conjunto de votantes requeridos.

El algoritmo del servidor central puede soportar la caída de un proceso cliente sin riesgo.

El algoritmo de multicast puede adaptase para tolerar la rotura de un proceso haciendo que, de forma implícita, conceda todas las peticiones.

Elecciones

Un algoritmo de elección es aquel que se utiliza para escoger un proceso único que juegue un papel específico. Se dirá que un proceso pide una elección si lleva a cabo una acción que inicia una ejecución específica del algoritmo de elección.

Los requisitos durante una ejecución de un algoritmo de elección son:

• E1 (seguridad): un proceso participante tiene un elegido donde este es el proceso con el identificador mayor que no se ha caído el final de la ejecución.

• E2 (superveniencia): todos los procesos participan y al final fijan el elegido o bien se han caído.

Existen varios algoritmos para la elección de un proceso:

- Algoritmo de elección basado en anillo

El objetivo de este algoritmo es la elección de un proceso individual llamado coordinador que es el proceso con identificador más grande.

En un principio todos los procesos se etiquetan como no participantes y cuando comienza una elección se marca como participante y envía un mensaje de elección. El algoritmo cumple E1 y

E2. Es un algoritmo que sirve para comprender las características de los algoritmos de elección pero no tiene valor práctico dado que no tolera fallos.

- Algoritmo abusón (bully)

El algoritmo del abusón permite la caída de procesos durante una elección. A diferencia del algoritmo basado en anillo este supone que el sistema es síncrono, esto es, que utiliza timeouts para detectar un fallo en un proceso. El algoritmo del abusón supone que cada proceso conoce qué procesos tienen identificadores mayores y que puede comunicarse con todos esos procesos. El algoritmo cumple E1 y E2.

Coordinación y acuerdo en comunicaciones en grupo

La comunicación en grupo, o multidifusión (multicast), requiere la presencia de coordinación y acuerdo. Su objetivo es que cada uno de los procesos de un grupo reciba los mensajes enviados al grupo con garantías de que han sido entregados. Estas garantías incluyen el acuerdo sobre el grupo de mensajes que todo proceso del grupo debería recibir, y sobre el orden de entrega dentro de los miembros del grupo.

Modelo del sistema

El sistema contiene una colección de procesos que pueden comunicarse entre ellos de forma fiable a través de canales uno-a-uno. Dichos procesos son miembros de grupos, los cuales son los destinatarios de los mensajes enviados en una operación multicast.

Se dice que un grupo está cerrado si solo los miembros del grupo pueden hacer muticast dentro de el, incluida la entrada al mismo. Un grupo se dice abierto si los procesos que no están en el grupo le pueden enviar mensajes.

- Multicast básico (B-multicast)

Es un multicast que encapsula dentro de una primitiva el envío básico fiable uno-a-uno.

- Multicast fiable (F-multicast)

Un proceso de multicast es fiable si satisface las siguientes propiedades:

- Integridad: un proceso correcto entrega un mensaje a lo sumo una vez.
- Validez: si un proceso correcto multidifunde un mensaje, este al final será entregado.
- Acuerdo: si un proceso correcto entrega un mensaje entonces el resto de procesos correctos del grupo deben, al final, entregar el mensaje.

El multicast fiable está relacionado con la atomicidad del proceso de entrega de mensajes a un grupo.

- Multicas fiable sobre multicast IP

Una alternativa para realizar F-multicast es combinar la multidifusión IP, acuses de recibo adheridos y acuses de recibo negativos. Este protocolo se basa en la observación de que la comunicación mediante IP multicast casi siempre tiene éxito.

- Multicast ordenado

El algoritmo de multicast fiable entrega los mensajes en un orden arbitrario y esto puede no ser admisible en muchas aplicaciones. Los requisitos de ordenación más frecuentes son la ordenación total, causal y la ordenación FIFO además de ordenaciones híbridas total-causal y total-FIFO.

- Ordenación FIFO: establece que, si un proceso difunde un mensaje m1 antes que un mensaje m2, entonces ningún proceso correcto entrega m2 si no ha entregado previamente m1.

- Ordenación causal: establece que, si la difusión de un mensaje m1 ocurre antes que la difusión de un mensaje m2, entonces ningún proceso correcto entrega m2 si no ha entregado previamente m1.

- Ordenación total: si dos procesos correctos Pi y Pj entregan dos mensajes m1 y m2, entonces Pi entrega m1 antes que m2 sí y solo sí Pj entrega m1 antes que m2.

Dicho de otro modo, la ordenación FIFO permite asegurar, desde la perspectiva del emisor, que el primer mensaje enviado ha sido el primero entregado. Por otra parte, la ordenación causal lo que asegura es la relación de causa entre mensajes, mientras que la ordenación total asegura que si un mensaje llegó antes con un proceso esto se repite en los demás. La ordenación total multicast requiere de un reloj global.

Consenso

El consenso persigue la resolución de problemas en los procesos que deben ponerse de acuerdo en un valor después de que uno o más de dichos procesos hayan propuesto cuál debería ser ese valor.

En el caso de la exclusión mutua los procesos acuerdan cuales pueden entrar en la sección crítica. En el caso de una elección los procesos acuerdan cual es el proceso elegido.

Los requisitos para un algoritmo de consenso son que han de cumplirse las siguientes condiciones cada vez que se ejecute:

- Terminación: finalmente cada proceso correcto ha de fijar su variable de decisión.
- Acuerdo: el valor de decisión de todos los procesos es el mismo.
- Integridad: si todos los procesos correctos han propuesto el mismo valor, entonces cualquier proceso correcto en el estado decidido ha elegido dicho valor.

El problema de los generales bizantinos

Se distingue del problema de consenso en que un proceso destacado proporciona un valor en que el que los otros han de ponerse de acuerdo, en vez de que cada uno proponga un valor.

Consistencia interactiva

Este problema es otra variante del consenso en el cual todo proceso propone un solo valor. El objetivo del algoritmo es que todos los procesos se pongan de acuerdo en un vector de valores, uno para cada proceso. A este vector se llamará vector de decisión.

4.5.- TRANSACCIONES Y CONTROL DE CONCURRENCIA.

Los algoritmos de control de concurrencia son necesarios cuando se ejecutan varias transacciones de manera simultánea:

- En distintos procesos.
- En distintos procesadores

Los principales algoritmos son:

- El de la cerradura.
- El del control optimista de la concurrencia.
- El de las marcas de tiempo.

Cerradura (locking)

Cuando un proceso debe leer o escribir en un archivo (u otro objeto) como parte de una transacción, primero cierra el archivo.

La cerradura se puede hacer mediante:

- Un único manejador centralizado de cerraduras.
- Un manejador local de cerraduras en cada máquina.

El manejador de cerraduras:

- Mantiene una lista de los archivos cerrados.
- Rechaza todos los intentos por cerrar archivos ya cerrados por otros procesos.

El sistema de transacciones generalmente adquiere y libera las cerraduras sin acción por parte del programador.

Una mejora consiste en distinguir las cerraduras para lectura de las cerraduras para escritura.

Una cerradura para lectura no impide otras cerraduras para lectura:

- Las cerraduras para lectura se comparten.

Una cerradura para escritura sí impide otras cerraduras (de lectura o de escritura):

- Las cerraduras para escritura no se comparten, es decir que deben ser exclusivas.

El elemento por cerrar puede ser un archivo, un registro, un campo, etc. y lo relativo al tamaño del elemento por cerrar se llama la granularidad de la cerradura.

Mientras más fina sea la granularidad:

- Puede ser más precisa la cerradura.
- Se puede lograr un mayor paralelismo en el acceso al recurso.
- Se requiere un mayor número de

cerraduras. Generalmente se utiliza la cerradura

de dos fases:

- El proceso adquiere todas las cerraduras necesarias durante la fase de crecimiento.
- El proceso las libera en la fase de reducción. Se deben evitar situaciones de aborto en cascada:
- Se graba en un archivo y luego se libera su cerradura.
- Otra transacción lo cierra, realiza su trabajo y luego establece un compromiso.
- La transacción original aborta.
- La segunda transacción (ya comprometida) debe deshacerse, ya que sus resultados se basan en un archivo que no debería haber visto cuando lo hizo.

Las cerraduras comunes y de dos fases pueden provocar bloqueos cuando dos procesos intentan adquirir la misma pareja de cerraduras, pero en orden opuesto, por lo tanto se deben utilizar técnicas de prevención y de detección de bloqueos para superar el problema.

Control Optimista de la Concurrencia

La idea es muy sencilla:

• Se sigue adelante y se hace todo lo que se deba hacer, sin prestar atención a lo que hacen los demás.

• Se actúa a posteriori si se presenta algún problema. SE mantiene un registro de los archivos leídos o grabados. En el momento del compromiso:

• Se verifican todas las demás transacciones para ver si alguno de los archivos ha sido modificado desde el inicio de la transacción:

O Si esto ocurre la transacción aborta.

O Si esto no ocurre se realiza el compromiso.

Las principales ventajas son:

• Ausencia de bloqueos.

• Paralelismo máximo ya que no se esperan cerraduras.

La principal desventaja es:

- Re-ejecución de la transacción en caso de falla.
- La probabilidad de fallo puede crecer si la carga de trabajo es muy alta.

Marcas de Tiempo

Se asocia a cada transacción una marca de tiempo al iniciar (begin_transaction).

Se garantiza que las marcas son únicas mediante el algoritmo de Lamport.

Cada archivo del sistema tiene asociadas una marca de tiempo para la lectura y otra para la escritura, que indican la última transacción comprometida que realizó la lectura o escritura.

Cuando un proceso intente acceder a un archivo, lo logrará si las marcas de tiempo de lectura y escritura son menores (más antiguas) que la marca de la transacción activa.

Si la marca de tiempo de la transacción activa es menor que la del archivo que intenta acceder:

• Una transacción iniciada posteriormente ha accedido al archivo y ha efectuado un compromiso.

• La transacción activa se ha realizado tarde y se aborta.

En el método de las marcas no preocupa que las transacciones concurrentes utilicen los mismos archivos, pero sí importa que la transacción con el número menor esté en primer lugar.

Las marcas de tiempo tienen propiedades distintas a las de los bloqueos:

• Una transacción aborta cuando encuentra una marca mayor (posterior).

• En iguales circunstancias y en un esquema de cerraduras podría esperar o continuar inmediatamente.

Las marcas de tiempo son libres de bloqueos, lo que es una gran ventaja.

Resumen

Los diferentes esquemas ofrecen distintas ventajas, pero el problema principal es la gran complejidad de su implantación.

Bloqueos en Sistemas Distribuidos

Son peores que los bloqueos en sistemas monoprocesador:

- Son más difíciles de evitar, prevenir, detectar y solucionar.
- Toda la información relevante está dispersa en muchas máquinas.

Son especialmente críticos en sistemas de bases de datos distribuidos. Las estrategias usuales para el manejo de los bloqueos son:

- Algoritmo del avestruz:
  - Ignorar el problema
- Detección:
  - Permitir que ocurran los bloqueos, detectarlos e intentar recuperarse de ellos.
- Prevención:
  - Hacer que los bloqueos sean imposibles desde el punto de vista estructural.
- Evitarlos:
  - Evitar los bloqueos mediante la asignación cuidadosa de los recursos.

El algoritmo del avestruz merece las mismas consideraciones que en el caso de mono-procesador.

En los sistemas distribuidos resulta muy difícil implantar algoritmos para evitar los bloqueos:

- Se requiere saber de antemano la proporción de cada recurso que necesitará cada proceso.
- Es muy difícil disponer de esta información en forma práctica.

Las técnicas más aplicables para el análisis de los bloqueos en sistemas distribuidos son:

- Detección.
- Prevención.

Detección Distribuida de Bloqueos

Cuando se detecta un bloqueo en un S. O. convencional se resuelve eliminando uno o más procesos.

Cuando se detecta un bloqueo en un sistema basado en transacciones atómicas se resuelve abortando una o más transacciones:

- El sistema restaura el estado que tenía antes de iniciar la transacción.
- La transacción puede volver a comenzar.

Las consecuencias de la eliminación de un proceso son mucho menos severas si se utilizan las transacciones que en caso de que no se utilicen.

Detección Centralizada de Bloqueos

Cada máquina mantiene la gráfica de recursos de sus propios procesos y recursos.

Un coordinador central mantiene la gráfica de recursos de todo el sistema, que es la unión de todas las gráficas individuales.

Cuando el coordinador detecta un ciclo elimina uno de los procesos para romper el bloqueo.

La información de control se debe transmitir explícitamente, existiendo las siguientes variantes:

- Cada máquina informa cada actualización al coordinador.
- Cada máquina informa periódicamente las modificaciones desde la última actualización.
- El coordinador requiere la información cuando la necesita.

La información de control incompleta o retrasada puede llevar a falsos bloqueos:

- El coordinador interpreta erróneamente que existe un bloqueo y elimina un proceso.
- Una posible solución es utilizar el algoritmo de Lamport para disponer de un tiempo global.

4.6 DETECCIÓN DISTRIBUIDA DE BLOQUEOS

Un algoritmo típico es el de Chandy-Misra-Haas.

Los procesos pueden solicitar varios recursos (por ejemplo, cerraduras) al mismo tiempo, en vez de uno cada vez

Se permiten las solicitudes simultáneas de varios procesos:

- Un proceso puede esperar a uno o más recursos simultáneamente.

• Los recursos que espera un proceso pueden ser locales o remotos (de otra máquina). Si el proceso –0ll se bloquea debido al proceso –1:

• Se genera un mensaje de exploración que se envía al proceso (o procesos) que detienen los recursos necesarios.

• El mensaje consta de tres números:

O El proceso recién bloqueado, el proceso que envía el mensaje y el proceso al cual se envía.

• Al llegar el mensaje el receptor verifica si él mismo espera a algunos procesos, en cuyo caso:

o El mensaje se actualiza:

▪ Se conserva el primer campo.

▪ Se reemplaza el segundo por su propio número de proceso y el tercero por el número del proceso al cual espera.

o El mensaje se envía al proceso debido al cual se bloquea:

▪ Si se bloquea debido a varios procesos les envía mensajes (diferentes) a todos ellos.

• Si un mensaje recorre todo el camino y regresa a su emisor original (el proceso enlistado en el primer campo), entonces:

o Existe un ciclo y el sistema está bloqueado.

Una forma de romper el bloqueo es que el proceso que inició la exploración se comprometa a suicidarse y, si varios procesos se bloquean al mismo tiempo e inician exploraciones, todos ellos se suicidarán.

Una variante es eliminar solo al proceso del ciclo que tiene el número más alto

Prevención Distribuida de Bloqueos

La prevención consiste en el diseño cuidadoso del sistema para que los bloqueos sean imposibles estructuralmente.

Entre las distintas técnicas se incluye:

- Permitir a los procesos que solo conserven un recurso a la vez.
- Exigir a los procesos que soliciten todos sus recursos desde un principio.
- Hacer que todos los procesos liberen todos sus recursos cuando soliciten uno nuevo. En un sistema distribuido con tiempo global y transacciones atómicas:
  - Se puede asociar a cada transacción una marca de tiempo global al momento de su inicio.
  - No puede haber parejas de transacciones con igual marca de tiempo asociada.

La idea es que cuando un proceso está a punto de bloquearse en espera de un recurso que está utilizando otro proceso:

- Se verifica cuál de ellos tiene la marca de tiempo mayor (es más joven).
- Se puede permitir la espera solo si el proceso en estado de espera tiene una marca inferior (más viejo) que el otro.

Al seguir cualquier cadena de procesos en espera:

- Las marcas aparecen en forma creciente.
- Los ciclos son imposibles.

Otra posibilidad es permitir la espera de procesos solo si el proceso que espera tiene una marca mayor (es más joven) que el otro proceso; las marcas aparecen en la cadena en forma descendente.

Es más sabio dar prioridad a los procesos más viejo

- Se ha invertido tiempo de proceso en ellos.
- Probablemente conservan más recursos.

4.7 EXCLUSIÓN MUTUA DISTRIBUIDA

La exclusión mutua distribuida es el mecanismo de coordinación entre varios procesos concurrentes a la hora de acceder a recursos o secciones compartidas. La exclusión mutua se **utiliza** para acceder a la región critica(bloqueo), manipular los recursos compartidos y liberar el recurso(despierta los procesos en espera).

Los siguientes algoritmos para este problema son:

- **Algoritmos centralizados:** Se emplea un servidor que dé los permisos para entrar en la sección crítica, para esto un proceso envía un mensaje de petición al servidor y espera una respuesta por su parte donde esta respuesta constituye el permiso para entrar en la sección crítica.

- **Algoritmos distribuidos:** En esta categoría tenemos al algoritmo basado en un anillo ya que de esta manera no se necesita un proceso adicional y para lograr esto basta con organizarlos en un anillo lógico. Para este algoritmo se requiere que cada proceso tenga un canal de comunicación hacia el siguiente proceso en el anillo. La exclusión se consigue obteniendo un permiso para entrar a la sección crítica mediante un mensaje que se pasa de un proceso a otro en única dirección alrededor del anillo. Si un proceso no requiere entrar en la sección crítica cuando recibe este permiso, entonces inmediatamente lo hace avanzar hacia su vecino. Un proceso que requiere este permiso espera hasta recibirlo y en este caso lo retiene. Cuando el proceso salga de la sección crítica enviara el testigo hacia el siguiente vecino.

- **Algoritmos basados en marcas de tiempo:** Los procesos que necesitan entrar en una sección crítica envían un mensaje de petición mediante multidifusión(se envía la petición a múltiples destinos simultaneamente) y pueden entrar en ella solamente cuando el resto de los procesos haya respondido al mensaje. Las condiciones bajo las cuales un proceso responde a una petición se diseñan para asegurar que se cumplan las siguientes condiciones:
  - Solo un proceso puede estar ejecutándose cada vez en la sección crítica.
  - Las peticiones para entrar y salir de la sección crítica al final son concedidas.
  - Si una petición para entrar en la sección crítica ocurrió antes que otra, entonces la entrada a la sección crítica se garantiza en ese orden.

4.8 COMUNICACIÓN POR MULTIDIFUSIÓN

El concepto de 'multicast'
Existen una serie de mecanismos que nos permiten realizar el envío de contenido multimedia desde un servidor a un cliente, pero sin duda el más eficiente es el mecanismo de multicast.

Unicast: para cada cliente que desee el contenido multimedia, necesitamos un flujo de datos diferente.
Broadcast: el contenido multimedia se envía a todos los clientes de la red independientemente que deseen el contenido o no.
Multicast: sólo se utiliza el ancho de banda mínimo, ya que sólo los clientes que deseen el contenido multimedia reciben el flujo de datos.

El concepto de 'grupo multicast'
¿En qué consiste el concepto de «grupo multicast?

Un grupo de clientes pide recibir un flujo de datos particular. El grupo no tiene límites físicos ni geográficos, y sólo los clientes interesados en el contenido multimedia se suman a ese grupo mediante mensajes IGMP. Para identificar a ese grupo de clientes se utiliza una dirección de tipo multicast.

Pero, ¿qué es una dirección multicast?

Una dirección multicast está asociada con un grupo de clientes interesados en un flujo de datos de contenido multimedia. Las direcciones van desde la 224.0.0.0 a la 239.255.255.255 y se llama rango Clase D.

El servidor envía un único datagrama a la dirección multicast y el router se encargará de hacer copias y enviarlas a todos los clientes que hayan informado de su interés por los datos de ese servidor.

Principales protocolos de enrutamiento multicast IPv4
La distribución de contenido multimedia en una red local está controlada por el protocolo IGMP (Internet Group Management Protocol), y dentro del dominio de enrutamiento se utiliza el protocolo PIM (Protocol Independent Multicast).

IGMP: IGMP es un protocolo que utilizan los clientes y routers multicast para identificar la pertenencia a un grupo multicast.

Los clientes indican que pertenecen a un grupo enviando un mensaje IGMP a su router más cercano.
Los routers escuchan esos mensajes IGMP y periódicamente "descubren" los grupos multicast que están activos o inactivos en la subred.
PIM: PIM es una familia de protocolos de enrutamiento multicast que trabaja en base a la información de routing de los algoritmos tradicionales, con independencia del que se utilice.

El protocolo PIM crea una estructura de árboles de distribución desde los servidores hasta los receptores, basándose en la información de la topología obtenida.

Existen cuatro variantes del protocolo PIM: PIM Sparse Mode, PIM Dense Mode, Bidirectional PIM y PIM Source-Specific Multicast. En este artículo nos centraremos en la que mayor despliegue tiene: PIM Sparse Mode.

PIM Sparse Mode (PIM-SM)

PIM Sparse Mode es un protocolo de enrutamiento eficiente para paquetes IP de grupos multicast que puede abarcar redes de área amplia (WAN) o redes entre dominios. El protocolo se denomina 'sparse-mode' porque es apropiado para los grupos donde un porcentaje muy bajo de los nodos se suscriben a la sesión multicast.

PIM difunde el tráfico multicast desde un servidor origen hacia todos los clientes que hayan expresado interés en dicho tráfico, creando un árbol de distribución y encaminando el tráfico sólo por los interfaces imprescindibles. En cada nodo que forma parte del árbol hay un único interfaz en dirección a la raíz y uno o varios interfaces en dirección a los clientes.

Existen dos tipos de árboles de distribución en PIM-SM:

Shared tree: el origen del tráfico es un nodo llamado Rendezvous-Point (RP), que concentra todo el tráfico destinado a un mismo grupo multicast sin importar que servidor originó el tráfico.
Shortest Path Tree (SPT): el árbol tiene como origen un único servidor multicast, y está destinado a un grupo multicast específico. El tráfico no tiene que atravesar ningún nodo donde se concentra el tráfico, y siempre proporciona el camino más corto entre el servidor y los clientes.
Ventajas del protocolo PIM-SM

El protocolo PIM-SM se desarrolló principalmente para evitar los problemas de otros protocolos de enrutamiento multicast, que fallan cuando se aplican a entornos de red extensa (WAN) o de población esparcida, en las que el número de clientes es pequeño.

PIM-SM se encarga de que la información que viaja desde el servidor al cliente sólo pase una vez por el camino entre ambos y que sólo sea enviada en el caso de que existan clientes que deseen esa información.

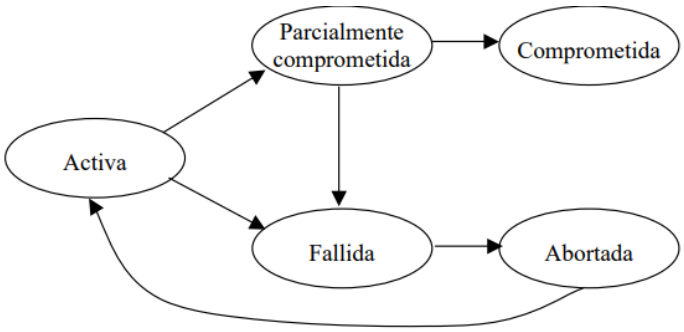
Inconvenientes del protocolo PIM-SM

Como hemos dicho el protocolo PIM-SM evita problemas que tienen otros protocolos de enrutamiento multicast, pero puede dar lugar a problemas de congestión de la red debido a las concentraciones de tráfico en los árboles compartidos, lo que puede dar lugar a pérdida de paquetes.

4.9 TRANSACCIONES Y CONTROL DE CONCURRENCIA

Los SGBDs son sistemas concurrentes, i.e., admiten la ejecución concurrente de consultas. Ejemplo: Sistema de venta de billetes de avión. Por tanto, es necesario: Modelo de procesos concurrentes para admitir operaciones concurrentes que preserven la integridad de los datos.

Transacción Una transacción es una unidad de la ejecución de un programa. Puede consistir en varias operaciones de acceso a la base de datos. Está delimitada por constructoras como begintransaction y end-transaction.



Bloqueos Un bloqueo es una información del tipo de acceso que se permite a un elemento. El SGBD impone los bloqueos necesarios en cada momento. El gestor de acceso a los datos implementa las restricciones de acceso. En algunos sistemas se permite que el usuario pueda indicar el bloqueo más adecuado (locking hints). Tipos de bloqueo con respecto a la operación: read-locks: sólo permite lectura write-locks: permite lectura y escritura El gestor de bloqueos almacena los bloqueos en una tabla de bloqueos: (, ,)=(E,B,T) La transacción T tiene un tipo de bloqueo B sobre el elemento E. Normalmente, E es clave, aunque no siempre, porque varias transacciones pueden bloquear el mismo elemento de forma diferente.

Modos de bloqueo Especifica el modo en que se bloquea un elemento • Compartido: para operaciones sólo de lectura. Se permiten lecturas concurrentes, pero ninguna actualización. • Actualización: para operaciones que pueden escribir. Sólo se permite que una transacción adquiera este bloqueo. Si la transacción modifica datos, se convierte en exclusivo, en caso contrario en compartido. • Exclusivo. para operaciones que escriben datos. Sólo se permite que una transacción adquiera este bloqueo. • Intención: se usan para establecer una jerarquía de bloqueo. Por ejemplo, si una transacción necesita bloqueo exclusivo y varias transacciones tienen bloqueo de intención, no se concede el exclusivo. • Intención compartido. Bloqueo compartido. • Intención exclusivo. Bloqueo exclusivo. • Compartido con intención exclusivo. Algunos bloqueos compartidos y otros exclusivos. • Esquema. para operaciones del DDL. • Actualización masiva. En operaciones de actualización masiva

4.10. MODELO DE FALLOS PARA TRANSACCIONES.

Transacciones Atómicas

Las técnicas de sincronización ya vistas son de bajo nivel:

• El programador debe enfrentarse directamente con los detalles de:

O La exclusión mutua.

o El manejo de las regiones críticas.

o La prevención de bloqueos.

o La recuperación de fallas.

Se precisan técnicas de abstracción de mayor nivel que:

• Oculten estos aspectos técnicos.

• Permitan a los programadores concentrarse en los algoritmos y la forma en que los procesos trabajan juntos en paralelo.

Tal abstracción la llamaremos transacción atómica, transacción o acción atómica.

La principal propiedad de la transacción atómica es el –todo o nadall:

• O se hace todo lo que se tenía que hacer como una unidad o no se hace nada.
Ejemplo:

O Un cliente llama al Banco mediante una PC con un módem para:

▪ Retirar dinero de una cuenta.

▪ Depositar el dinero en otra cuenta.

O La operación tiene dos etapas.

O Si la conexión telefónica falla luego de la primer etapa pero antes de la segunda:

▪ Habrá un retiro, pero no un depósito.

O La solución consiste en agrupar las dos operaciones en una transacción atómica:

▪ Las dos operaciones terminarían o no terminaría ninguna.

- Se debe regresar al estado inicial si la transacción no puede concluir.

El Modelo de Transacción

Supondremos que [25, Tanenbaum]:

- El sistema consta de varios procesos independientes que pueden fallar aleatoriamente.
- El software subyacente maneja transparentemente los errores de comunicación.

Almacenamiento Estable

Se puede implantar con una pareja de discos comunes.

Cada bloque de la unidad 2 es una copia exacta (espejo) del bloque correspondiente en la unidad 1.

Cuando se actualiza un bloque:

- Primero se actualiza y verifica el bloque de la unidad 1.
- Luego se actualiza y verifica el bloque de la unidad 2.

Si el sistema falla luego de actualizar la unidad 1 y antes de actualizar la unidad 2:

- Luego de la recuperación se pueden comparar ambos discos bloque por bloque:
  - Se puede actualizar la unidad 2 en función de la 1.

Si se detecta el deterioro espontáneo de un bloque, se lo regenera partiendo del bloque correspondiente en la otra unidad.

Un esquema de este tipo es adecuado para las aplicaciones que requieren de un alto grado de tolerancia de fallos, por ej. las transacciones atómicas.

Primitivas de Transacción

Deben ser proporcionadas por el sistema operativo o por el sistema de tiempo de ejecución del lenguaje.

Ejemplos:

- Begin_transaction: los comandos siguientes forman una transacción.
- End_transaction: termina la transacción y se intenta un compromiso.
- Abort_transaction: se elimina la transacción; se recuperan los valores anteriores.
- Read: se leen datos de un archivo (o algún otro objeto).
- Write: se escriben datos en un archivo (o algún otro objeto).

Las operaciones entre Begin y End forman el cuerpo de la transacción y deben ejecutarse todas o ninguna de ellas:

•Pueden ser llamadas al sistema, procedimiento de biblioteca o enunciados en un lenguaje.

Propiedades de las Transacciones

Las propiedades fundamentales son:

- Serialización:

o Las transacciones concurrentes no interfieren entre sí.

- Atomicidad:

o Para el mundo exterior, la transacción ocurre de manera indivisible.

- Permanencia:

o Una vez comprometida una transacción, los cambios son permanentes.

La **serialización** garantiza que si dos o más transacciones se ejecutan al mismo tiempo:

• El resultado final aparece como si todas las transacciones se ejecutasen de manera secuencial en cierto orden:

o Para cada una de ellas y para los demás procesos.

La **atomicidad** garantiza que cada transacción no ocurre o bien se realiza en su totalidad:

- Se presenta como una acción indivisible e instantánea.

La **permanencia** se refiere a que una vez comprometida una transacción:

- Sigue adelante y los resultados son permanentes.

Transacciones Anidadas

Se presentan cuando las transacciones pueden contener subtransacciones (procesos hijos) que:

- Se ejecuten en paralelo entre sí en procesadores distintos.
- Pueden originar nuevas subtransacciones.

4.11 TRANSACCIONES ANIDADAS.

Consiste en una serie de modificaciones (transacciones) aun determinado recurso del sistema (por ejemplo, una base de datos) y en donde se define un punto de inicio (Begin Tran) y un punto de terminación que define un bloque entre el conjunto de operaciones que son realizadas.

Dentro de este proceso en bloque los demás usuarios no pueden modificar nada hasta que no se presente un estado estable de los datos, esto ocasiona inconsistencia temporal y conflictos. Para evitar lo anterior se implementan dos maneras diferentes:

• Ejecutar transacciones serializadas. Es un sistema que permite el procesamiento de transacciones en forma secuencial o serializado y consiste en asignarle una secuencia a cada transacción, este proceso reduce el rendimiento del sistema.

• Ejecutar transacciones calendarizadas. Es un sistema que permite el proceso de transacciones asignándole tiempos de procesamiento el cual permite incrementar el

rendimiento del sistema ya que se ejecuta un máximo de proceso en forma concurrente y no a través de una serie.

BIBLIOGRAFÍA

TITULO	AUTOR	EDITORIAL
FUNDAMENTOS DE SISTEMAS OPERATIVOS	Gunnar Wolf	UNAM
Sistemas Operativos Distribuidos	TANEMBAUM	Prentice-Hall
SISTEMAS DISTRIBUIDOS CONCEPTOS Y DESEÑO	George Coulouris	Adyson Wesley