Modelos Arquitectónicos de Sistemas Distribuidos

Virginia Padilla

Universidad Nacional Experimental de Guayana

virginiapadillas@gmail.com

26 de mayo de 2021

Contenido

Modelos Arquitectónicos

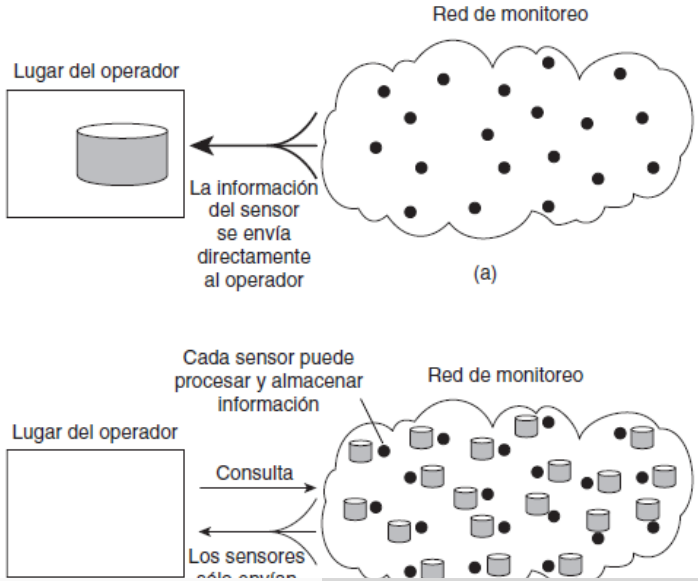
En [Coulouris, 2011]

Para comprender el modelo arquitectónico de un sistema distribuido, el autor, [Coulouris, 2011], considera cuatro preguntas claves :

- ¿Cuáles son las entidades que se comunican en el sistema distribuido?
- ¿Cómo se comunican o, más específicamente, qué paradigma de comunicación usan?
- ¿Qué roles y responsabilidades está presente en una arquitectura?
- ¿Cómo se asignan a la infraestructura física distribuida (cuál es su correspondencia)?

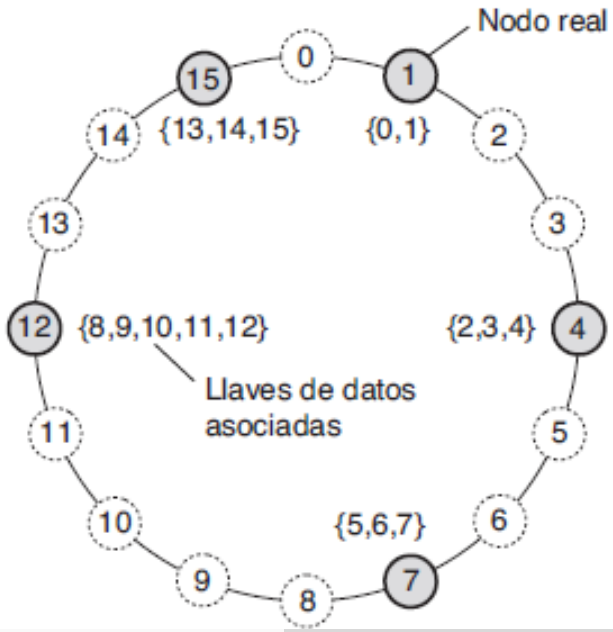
Procesos

Procesos Conducen a visión predominante de un sistema distribuido como procesos acoplados y paradigmas de comunicación entre procesos. En la figura 1 se ilustra la organización de una base de datos de una red de monitoreo y los procesos que se ejecutan en ella, en el lugar del operador (a), o en los sensores (b). [Steen, 2017]



Nodos

Nodos En algunos entornos primitivos, como las redes de sensores, los sistemas operativos pueden que no admitan abstracciones de proceso y, las entidades que se comunican en este tipo de sistemas son **nodos**. Ver figura 2 la organización de los nodos en la arquitectura Chord. Más de Chord en: Chord



Preguntas

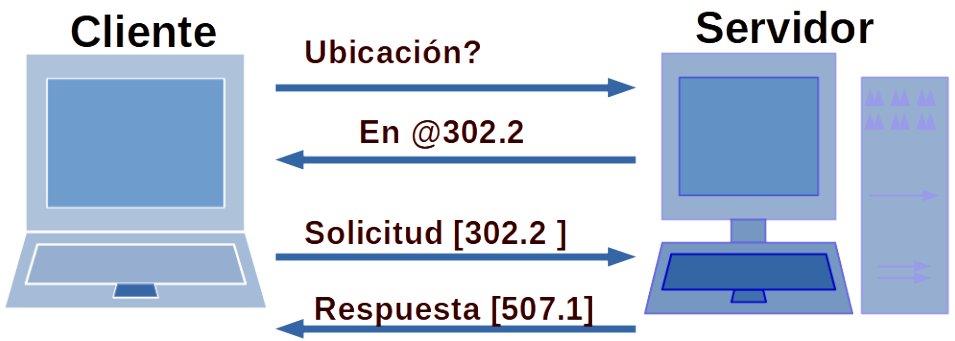
Chord

- Chord es un protocolo y algoritmo para la implementación de tablas hash distribuidas para sistemas P2P. Ver enlce Chord
- ¿Como se hace una búsqueda en Chord?

Hilos

Hilos En la mayoría de los entornos de sistemas distribuidos, los procesos se complementan con **hilos**, que son los puntos finales de la comunicación.

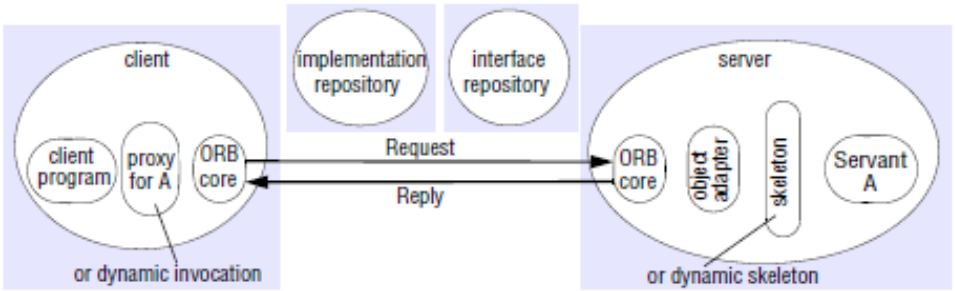
- En la figura 3 se presenta los hilos que se establecen en la comunicación del cliente y el servidor con el protocolo solicitud-respuesta.



Objetos

Objetos En el enfoque de sistemas distribuido basado en objetos, un cálculo consiste en una serie de objetos interactivos que representan unidades para el dominio del problema dado.

- Se accede a los objetos a través de interfaces, con un *interface definition language, IDL* que proporciona un especificación de los métodos definidos en un objeto.
- Por ejemplo, en la figura 4, se muestra la constitución de la arquitectura de Corba, detallando los objetos y sus funciones. Detalles del estándard Corba en: Corba



Componentes

- Se parecen a los objetos porque ofrecen abstracciones orientadas a problemas para construir sistemas distribuidos y también son accedido a través de interfaces.
- La diferencia es que los **componentes** especifican no solo sus interfaces, también las supuestos que hacen en términos de otros componentes/interfaces que deben estar presentes para que un componente cumpla su función.

Componentes

- La figura 5 muestra la arquitectura de un sistema de archivos simple que proporciona una interfaz a otros usuarios y, a su vez, que requiere conexión a un componente de servicio de directorio y un componente de servicio de archivo plano

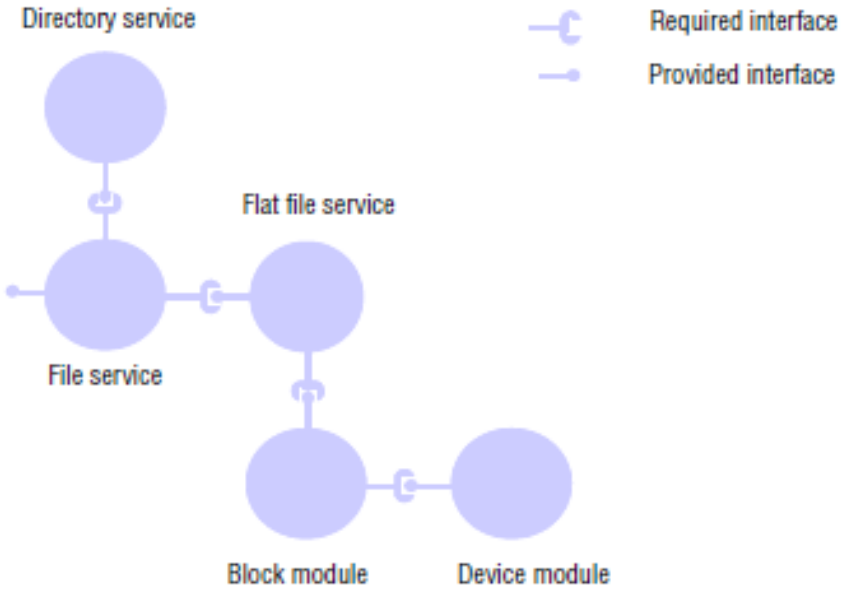


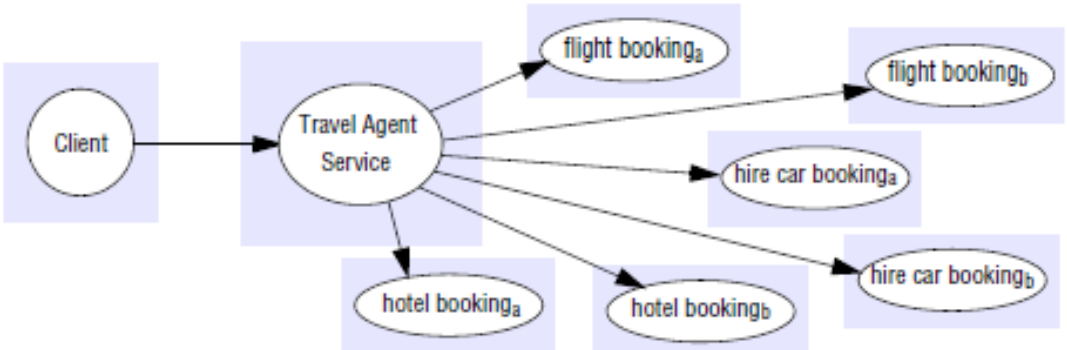
Figura: Arquitectura de software basado en componentes

Servicios Web

- Los servicios web están estrechamente relacionado con objetos y componentes, adoptando un enfoque basado en la encapsulación de comportamiento y acceso a través de interfaces.
- Están integrados en la *World Wide Web*, utilizando estándares web para representar y descubrir servicios.
- Servicios WEB, SOA, es un estándard de la OMG. Puede verlo en SOA

Servicios Web

- El esquema de la figura 6 detalla el servicio Agente de Viaje y su integración con otros servicios.
- Ejemplo: Considere el hecho de que las personas reservan vuelos, hoteles y autos de alquiler para viajes en línea utilizando una variedad de sitios web diferentes.
- Si cada uno de estos sitios web proporcionara una interfaz de servicio web estándar, entonces un "servicio de agente de viajes"podría utilizar sus operaciones para proporcionar al viajero una combinación de estos servicios.



Paradígmas de comunicación que usan

Considere tres tipos de paradigma de comunicación:

- Comunicación entre procesos,
- Invocación remota
- Comunicación indirecta

Comunicación entre Procesos

Comunicación entre procesos Se refiere al soporte de bajo nivel para comunicación entre procesos en sistemas distribuidos

- Pase de mensaje
- Socket
- Multidifusión

Invocación remota

Invocación remota Representa el paradigma de comunicación más común en sistemas distribuidos, cubren una gama de técnicas basadas en un intercambio bidireccional entre entidades comunicantes.

- Protocolo Solicitud-respuesta
- Llamada a procedimientos remotos
- Llamada a métodos remotos

Invocación Remota

Protocolo solicitud-respuesta Es un patrón impuesto en un servicio subyacente de transmisión de mensajes para admitir la informática cliente-servidor. Este paradigma es bastante primitivo y se usa en sistemas embebidos donde el rendimiento es primordial. El enfoque también se utiliza en el protocolo HTTP.

Invocación Remota

Llamadas a procedimiento remoto *(Remote Procediment Call, RPC)* es computación cliente-servidor con servidores que ofrece un conjunto de operaciones a través de un interfaz de servicio y clientes que llaman a estas operaciones directamente como si estuvieran disponibles en la zona.

Invocación de método remoto la invocación de método remoto *(Remote Method Invocation, RMI)* se parece mucho a llamadas a procedimiento remoto pero en un mundo de objetos distribuidos.

Comunicación Indirecta

Comunicación Indirecta Se realiza a través de una tercera entidad, lo que permite un fuerte grado de desacoplamiento entre remitentes y receptores.

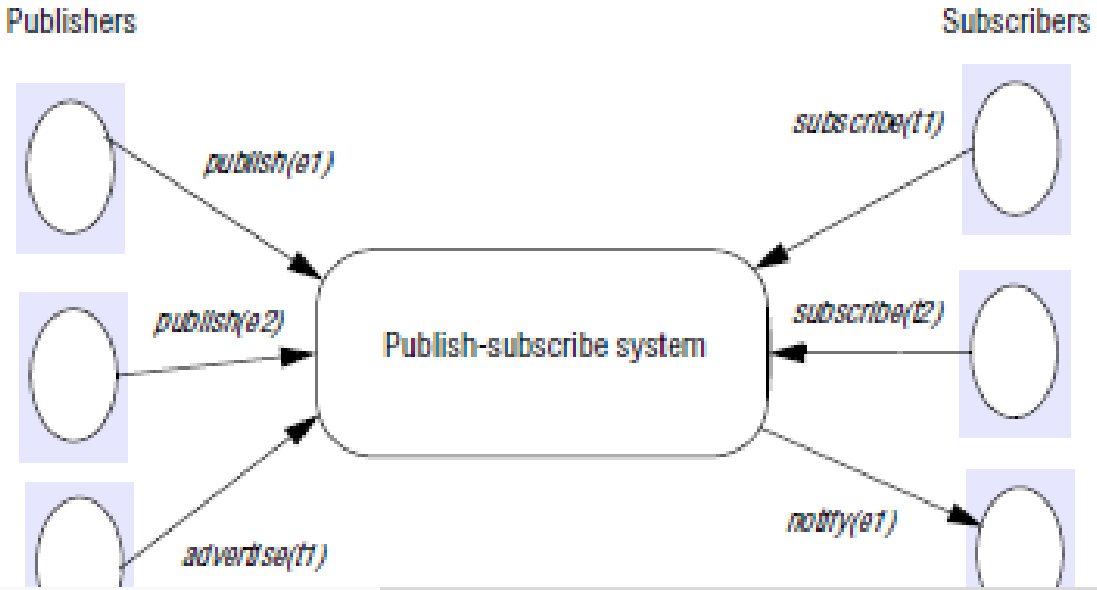
- Los remitentes no necesitan saber a quién están enviando mensajes.
- No es necesario que los emisores y los receptores existan al mismo tiempo.

Comunicación Indirecta

Comunicación grupal La comunicacion grupal se basa en la abstracción de un grupo que está representado en el sistema por un identificador de grupo .

Comunicación Indirecta

Sistemas de publicación-suscripción Los sistemas pub-sub donde el número de productores distribuye elementos de informacion de interes o eventos, a un número de comsumidores (llamado también, sistemas basados en eventos distribuidos). Ver esquema de la arquitectura en la figura 7.



Comunicación Indirecta

Colas de mensajes Servicio punto a punto mediante el cual el productor los procesos pueden enviar mensajes a una cola específica y los procesos del consumidor pueden recibir mensajes de la cola o ser notificado de la llegada de nuevos mensajes en el cola. Ver figura 8.

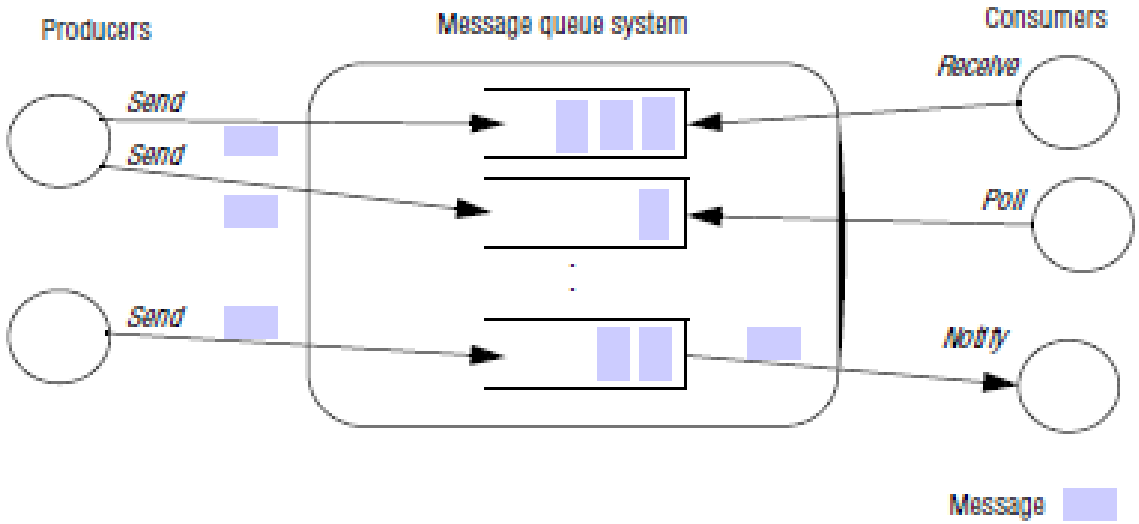


Figura: Arquitectura Colas de Mensajes.

Comunicación Indirecta

Espacios de tupla Los procesos pueden colocar elementos arbitrarios de datos estructurados o tuplas, en un espacio de tuplas persistente; otros procesos pueden leer o eliminar tales tuplas del espacio de tuplas especificando patrones de interés.

Memoria compartida distribuida proporcionan un abstracción para compartir datos entre procesos que no comparten memoria física. A los programadores se les presenta una abstracción de lectura o escritura de estructuras de datos (compartidas) como si estuvieran en sus propios espacios de direcciones locales.

Comunicación Indirecta

Los estilos de arquitectura según los roles y responsabilidades que cumplen los nodos, incluyen:

- Rol que juegan como clientes y servidores en la arquitectura cliente-servidor.
- Mismas funciones que cumplen los nodos en una arquitectura p2p.
- Rol como publicadores o suscriptores de eventos en una arquitectura pub-sub.
- Rol como productores o consumidores de mensajes en una arquitectura basada en colas.

Correspondencia cn Infrestructura Física

En esta clasificación se considera cómo las entidades, objetos o servicios se asignan o forman parte de la infraestructura física distribuida subyacente. Se exploran tres aspectos:

- Elementos que conforman la arquitectura
- Patrones que usa la arquitecura y
- Soluciones basadas en middleware

Correspondencia con múltiples servidores

Correspondencia de servicios a múltiples servidores Los servicios pueden implementarse como varios servidores de procesos en computadoras separadas que interactúan según sea necesario para proporcionar un servicio a procesos del cliente. En la figura 9 se muestra un ejemplo.

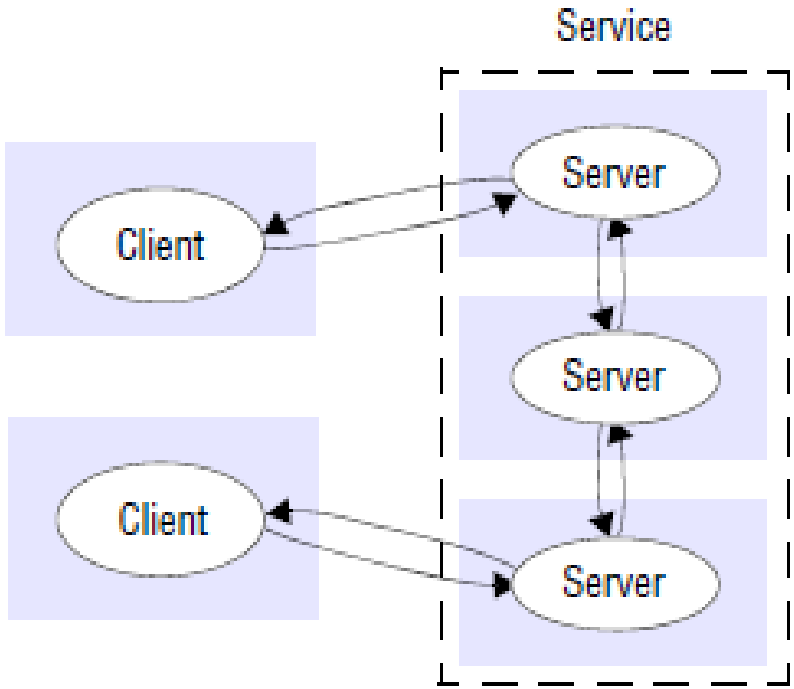


Figura: Arquitectura con múltiples

Cache

Cache. Los navegadores web mantienen un cache con las páginas web visitadas y otros recursos en el sistema de archivos local del cliente, y usa una solicitud HTTP para verificar con el servidor original que las páginas en caché están actualizadas. Un servidor proxy web (figura 10) proporciona un caché compartido de recursos web para las máquinas cliente en un sitio o en varios sitios.

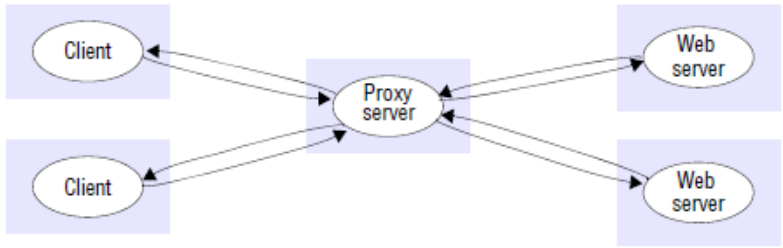


Figura: Arquitectura web proxy.

Código móvil

Código móvil El applet es un ejemplo de código móvil: el usuario que ejecuta un navegador selecciona un enlace a un subprograma cuyo código se almacena en un servidor web; el código se descarga en el navegador y lo ejecuta , como se muestra en la figura 11.

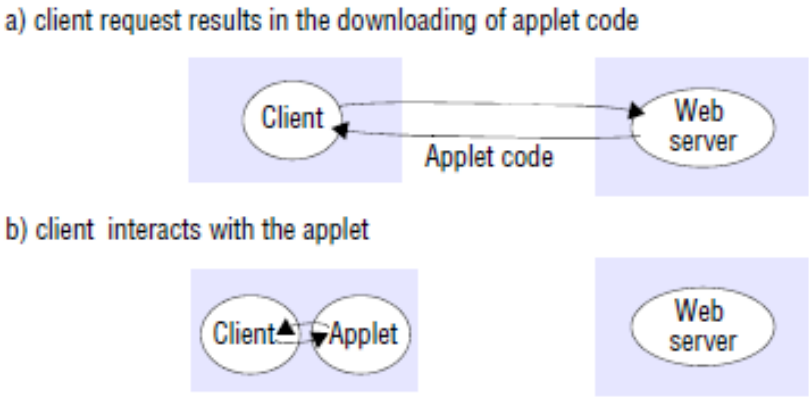


Figura: Arquitectura web applet.

Agentes móviles

Agentes móviles

- Realizar invocaciones a los recursos locales en cada sitio que visita, por ejemplo, acceder a entradas individuales de la base de datos.
- Pueden usarse para instalar y mantener software en las computadoras dentro de una organización o para comparar los precios de productos de varios proveedores visitando el sitio de cada proveedor y realizando una serie de operaciones de base de datos.

Patrones Arquitectónicos

Patrones Arquitectónicos

- Los patrones arquitectónicos, o patrones de arquitectura, dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones.
- Expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones.
- No son soluciones completas en sí mismas, sino que ofrecen conocimientos parciales que, cuando se combinan con otros patrones, llevan al diseñador a una solución para un dominio de problema dado.

Patrones Arquitectónicos

Patrones Arquitectónicos

Se presentan los siguientes patrones:

- Capas
- Arquitectura de capas
- Cliente flacos
- Otras clasificaciones

Patrones Arquitectónicos

Capas

- En un enfoque por capas, un sistema complejo se divide en varias capas, con un capa dada haciendo uso de los servicios ofrecidos por la capa siguiente.
- La capa referida ofrece una abstracción de software, con capas superiores que desconocen detalles de su implementación, o de cualquier otra capa debajo de ellos.
- En términos de sistemas distribuidos, esto equivale a una organización vertical de servicios en capas de servicio. Ejemplo de esta estructura es el middleware.

Patrones Arquitectónicos

Arquitectura de capas

- Es una técnica para organizar la funcionalidad de una capa determinada y colocarla en servidores apropiados y, como consideración secundaria, en los nodos físicos.
- Se asocia con la organización de aplicaciones y servicios como se observa en la figura 12.

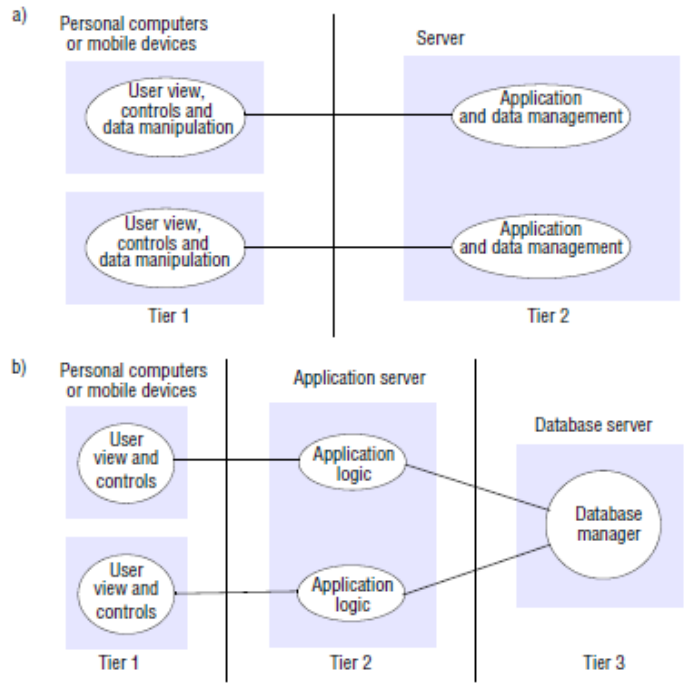


Figura: Arquitectura de capas de a) dos niveles y b) tres niveles.

Patrones Arquitectónicos

Clientes flacos

- Computación distribuida aleja la complejidad del dispositivo del usuario final hacia los servicios en Internet.
- Cliente ligero permite el acceso a servicios en red, proporcionados por soluciones en la nube, con pocas demandas en el dispositivo del cliente. Ver Figura 13.

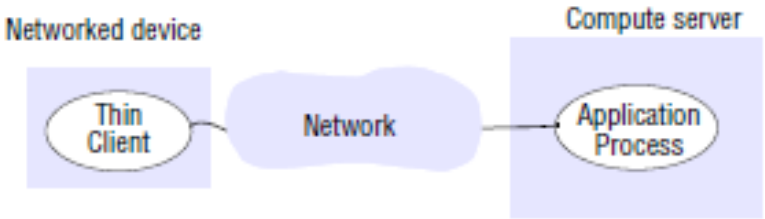
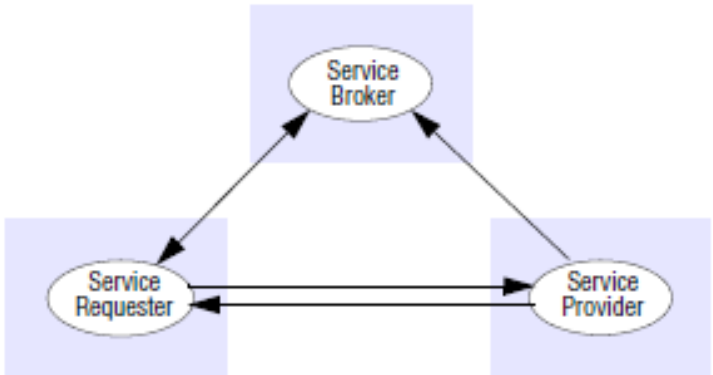


Figura: Arquitectura de cliente ligero.

Patrones Arquitectónicos

Otras clasificaciones

- El **patron proxy**, diseñados para apoyar la transparencia de la ubicación en RPC o RMI.
- El uso de **corredores** o *brokerage* en servicios web es un patrón que admite la interoperabilidad en infraestructuras distribuidas potencialmente complejas. Consta del trío de proveedores de servicios, solicitante de servicios y corredor de servicios, ver la Figura 14.



Patrones Arquitectónicos

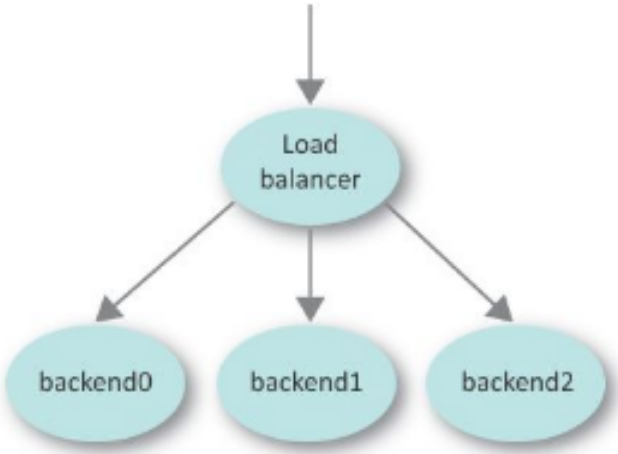
Otras patrones

- En [Limoncelli, 2014], presenta una clasificación basada en patrones.
- Estos patrones forman parte de la estructura de los sistemas distribuidos, los mismos están compuesto por múltiples componentes que aportan la funcionalidad que presentan los sistemas distribuidos, como la disponibilidad y escalamiento de servicios.
- Se detallan tres patrones en particular:
  - Balanceador de carga con múltiple *backend* de replicas
  - Servidor con múltiples *backend*
  - Árbol de servidores

Patrones Arquitectónicos

Balanceador de Carga con múltiples réplicas de backend

- En este patrón, Figura 15, las solicitudes se envían al servidor del equilibrador de carga.
- Para cada solicitud, selecciona un *backend* y se reenvía la solicitud allí.
- La respuesta vuelve al servidor del equilibrador de carga, que a su vez la transmite al solicitante original.



Patrones Arquitectónicos

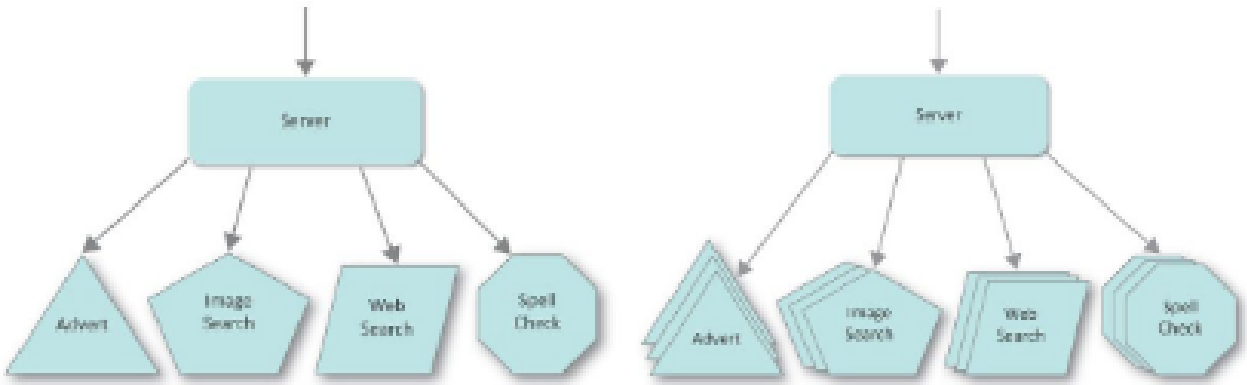
Servidor con varios backends

- El servidor recibe una solicitud, envía consultas a muchos servidores backend y redacta la respuesta final combinando esas respuestas.
- Este enfoque se utiliza normalmente cuando la consulta original se puede descomponer en una serie de consultas independientes que se pueden combinar para formar la respuesta final.

Patrones Arquitectónicos

Servidor con varios backends

- La figura 16a ilustra cómo un motor de búsqueda simple procesa una consulta con la ayuda de múltiples *backends*. La interfaz recibe la solicitud y transmite la consulta a los servidores *backend*.
- La figura 16b ilustra la misma arquitectura con *backends* replicados y equilibrio de carga. Es el mismo principio, pero el sistema puede escalar y sobrevivir mejor a las fallas.



Patrones Arquitectónicos

Árbol de servidores

- Se utiliza para acceder a un gran conjunto de datos o corpus. Cada hoja almacena una fracción de los datos.
- La raíz recibe la consulta y la reenvía a los padres, quienes la reenvían a los servidores hoja.
- Las hoja envía sus hallazgos a los padres, que clasifican y filtran los resultados antes de enviarlos a la raíz. La raíz toma las respuestas, combina los resultados y responde.

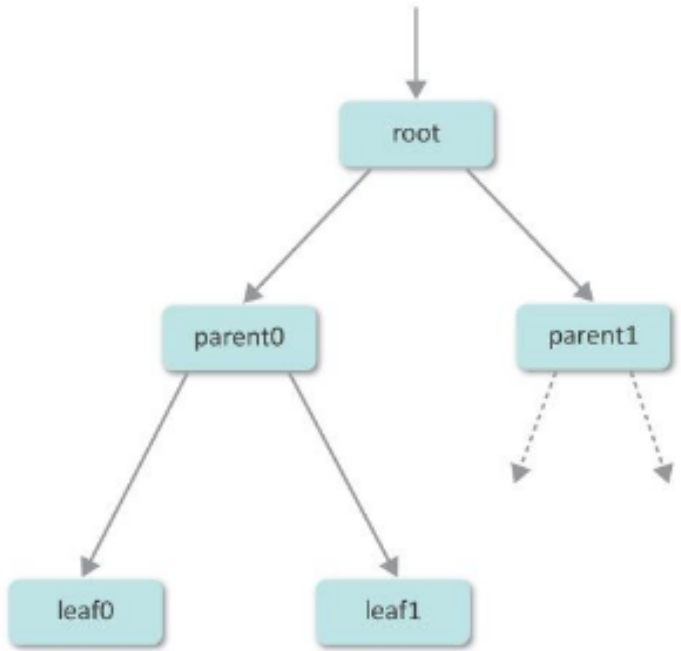


Figura: Árbol de Servidores.

Middleware

Árbol de servidores

- Su tarea es proporcionar un nivel superior abstracción de programación para el desarrollo de sistemas distribuidos y, con las capas, abstraer la heterogeneidad en la infraestructura subyacente para promover interoperabilidad y portabilidad.
- Hay arquitecturas más complejas como las que se presentan en la tabla 2.

Categoría	Subcategoría	Ejemplos
Objetos distribuidos	Plataforma	CORBA
	Plataforma	Java RMI
Componentes distribuidos	Componentes ligeros	Fractal
	Servidores de aplicaciones	SUN EJB
	Servidores de aplicaciones	CORBA Component Model
	Servidores de aplicaciones	JBoss
Publicación-suscripción	-	CORBA
	-	Event Service
	-	JMS

Cuadro: Categorías de middleware.

Categoría	Subcategoría	Ejemplos
Colas de mensajes	-	Websphere MQ
	-	JMS
Servicios web	Servicios web	Apache Axis
Peer-to-peer	Enrutamiento superpuesto	PASTRY
	Enrutamiento superpuesto	Tapestry
	Específica a la aplicación	OceanStore
	Específica a la aplicación	Gnutella

Cuadro: Categorías de middleware.

References


Van Steen M , Tanenbaum, A (2017)
Distributed Systems
Pearson Education, Inc.


Coulouris G, Dollimore J, Kindberg T, Blair G (2011)
Distributed Systems: Concepts and Design
Addison-Wesley Publishing Company.


Veríssimo, P. and Rodrigues, L. (2012)
Distributed Systems for System Architects
Springer.


Limoncelli T, Strata R, Hogan C. (2014)
The Practice of Cloud System Administration: Designing and Operating Large Distributed Systems
Addison Wesley.

Fin