INICIO GRABACIÓN

SANJOSÉ
FUNDACIÓN DE EDUCACIÓN SUPERIOR

INDICE

ESTILOS (MODELOS) ARQUITECTÓNICOS

CONCEPTOS

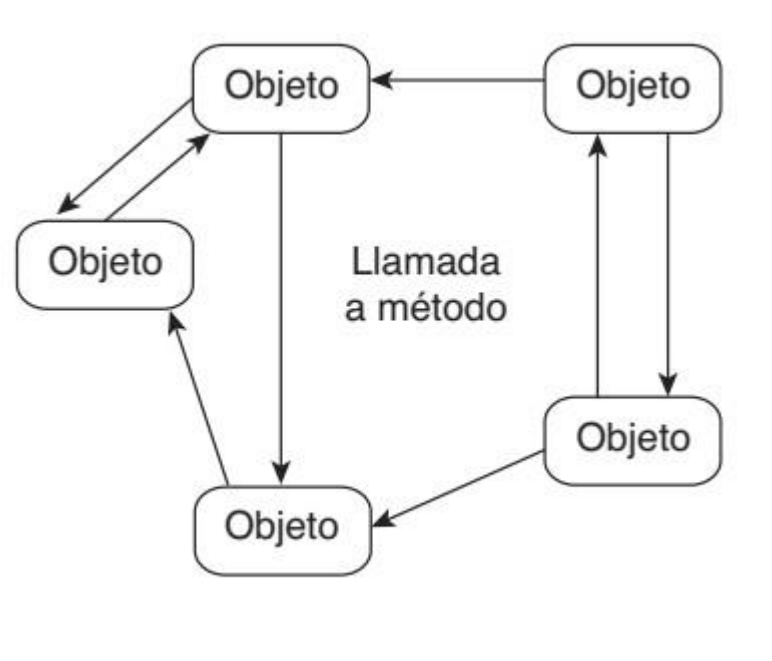
- Consideremos primero la organización lógica de los sistemas distribuidos en componentes de software, también conocida como arquitectura de software, consta de componentes y conectores.

  Por medio de componentes y conectores podemos lograr varias configuraciones, las cuales se han clasificado en estilos arquitectónicos. Varios estilos ya están identificados y los más importantes para sistemas distribuidos son:

  Arquitecturas en capas.
  - Arquitecturas basadas en objetos.
  - Arquitecturas centradas en datos.
  - Arquitecturas basadas en eventos.

Tanenbaum, A. S. (2008). Sistemas Distribuidos / Principios y Paradigmas / 2da ed

Arquitecturas basadas en objetos



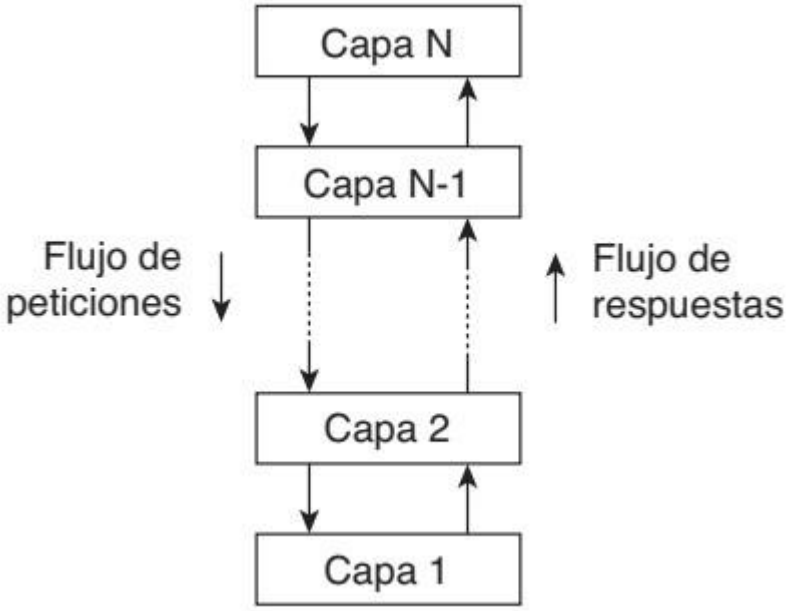
En esencia, cada objeto corresponde a lo que hemos definido como componente, y estos componentes se conectan a través de un mecanismo de llamadas a procedimientos (remotos). Sin ser sorprendente, esta arquitectura de software coincide con la arquitectura de sistemas cliente-servidor.

La arquitectura basada en capas y la basada en objetos aún son los estilos más importantes utilizados para implementar grandes sistemas de software

Arquitecturas en capas

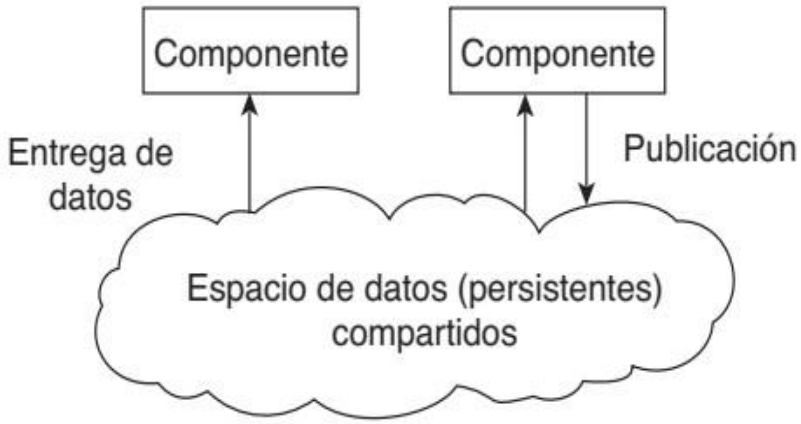
los componentes se estructuran (organizan) a modo de capas, donde al componente de la capa N se le permite llamar a componentes de la capa subyacente N-1, pero no del resto de capas, Este estilo se ha adoptado ampliamente en la comunidad de redes.

Una observación clave es que el control generalmente fluye de capa a capa: las peticiones se mueven hacia abajo en la jerarquía mientras que los resultados se mueven hacia arriba.



Tanenbaum, A. S. (2008). Sistemas Distribuidos / Principios y Paradigmas / 2da ed

Arquitecturas centradas en datos



Las arquitecturas centradas en datos evolucionaron alrededor de la idea de que los procesos se comunican a través de un repositorio común (activo o pasivo). Se puede argumentar que, para sistemas distribuidos, estas arquitecturas son tan importantes como las basadas en capas y objetos.

Un punto a favor que han desarrollado las aplicaciones en red es que se basan en un sistema de archivos distribuidos compartidos donde casi todas las comunicaciones se realizan a través de archivos, De manera similar, los sistemas distribuidos basados en la web se centran bastante en datos.

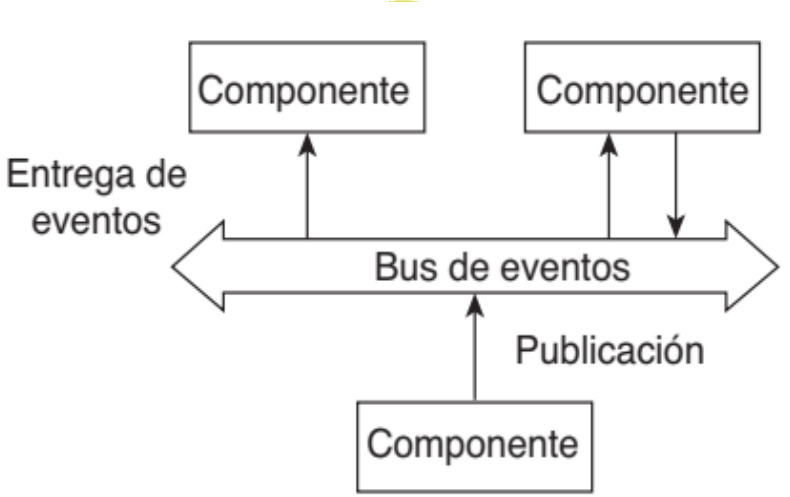
Tanenbaum, A. S. (2008). Sistemas Distribuidos / Principios y Paradigmas / 2da ed

Arquitecturas basadas en eventos.

los procesos se comunican básicamente a través de la propagación de eventos, los que opcionalmente transportan datos.

Para sistemas distribuidos, la propagación de eventos se ha asociado con lo que se conoce como sistemas de publicación-suscripción. La idea básica es que los procesos publican eventos después de los cuales el middleware asegura que sólo aquellos procesos suscritos a tales eventos los recibirán.

La principal ventaja de los sistemas basados en eventos es que los procesos están libremente acoplados. En principio, no necesitan referirse uno a otro explícitamente.



Tanenbaum, A. S. (2008). Sistemas Distribuidos / Principios y Paradigmas / 2da ed

ARQUITECTURA DE SISTEMAS

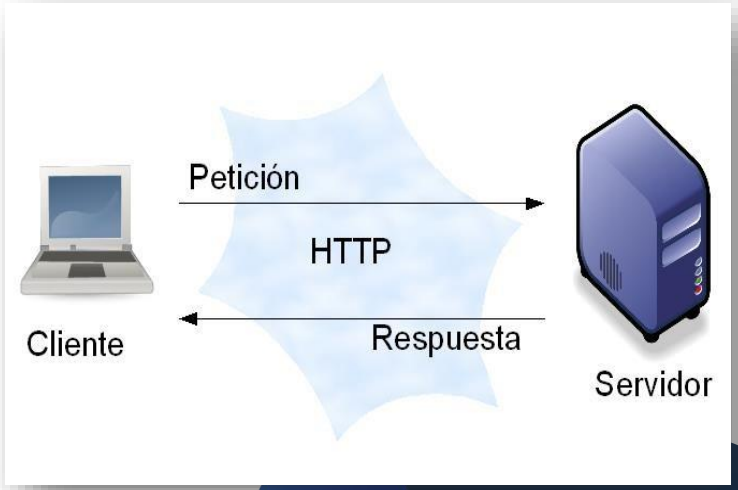
veamos cuántos sistemas distribuidos están realmente organizados considerando el lugar en donde se colocan los componentes de software. Decidir sobre los componentes de software, sobre su interacción y ubicación, da pie a una instancia de arquitectura de software, también conocida como arquitectura de sistemas.

Arquitecturas centralizadas

pensar en términos de clientes que requieren servicios de los servidores nos ayuda a comprender y manejar la complejidad de los sistemas distribuidos.

En el modelo básico cliente-servidor, los procesos de un sistema distribuido se dividen en dos grupo.

1. **Servidor:** es un proceso que implementa un servicio específico, por ejemplo, un servicio de sistema de archivos o un servicio de base de datos.
2. **Cliente:** es un proceso que solicita un servicio a un servidor, enviándole una petición y esperando posteriormente la respuesta. Esta interacción cliente-servidor, también conocida como comportamiento solicitud-respuesta.



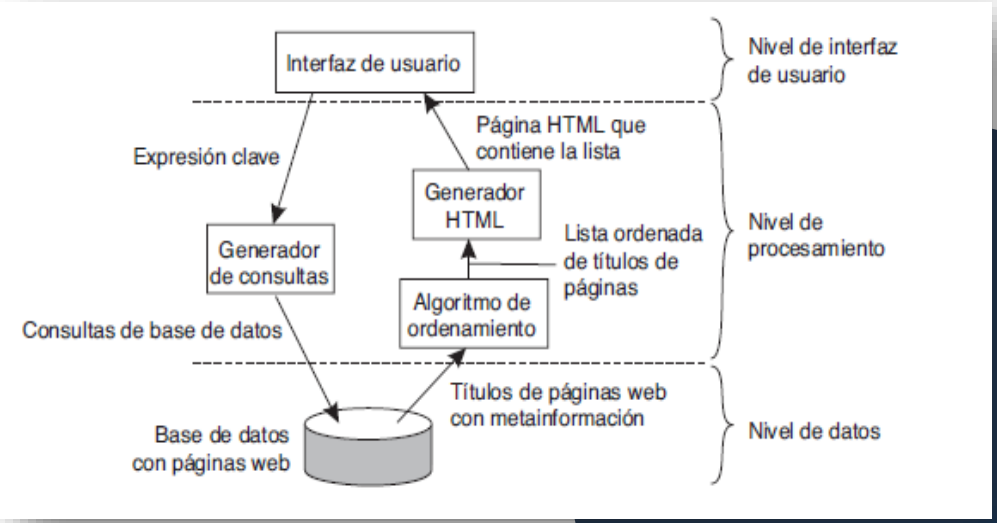
Tanenbaum, A. S. (2008). Sistemas Distribuidos / Principios y Paradigmas / 2da ed

Aplicación de capas

A través del tiempo, el modelo cliente-servidor ha estado sujeto a muchos debates y controversias. Una de las principales cuestiones fue cómo establecer una diferencia clara entre un cliente y un servidor. No sorprende que con frecuencia ésta no exista. Por ejemplo, un servidor para una base de datos distribuida puede actuar continuamente como un cliente, ya que reenvía solicitudes a diferentes servidores de archivos responsables de implementar las tablas de la base de datos. En tal caso, el propio servidor de la base de datos no hace más que procesar consultas.

Sin embargo, si consideramos que muchas aplicaciones cliente-servidor están enfocadas en dar a los usuarios acceso a las bases de datos, mucha gente ha defendido una diferencia entre los siguientes tres niveles, siguiendo básicamente el estilo arquitectónico en capas que se explico.

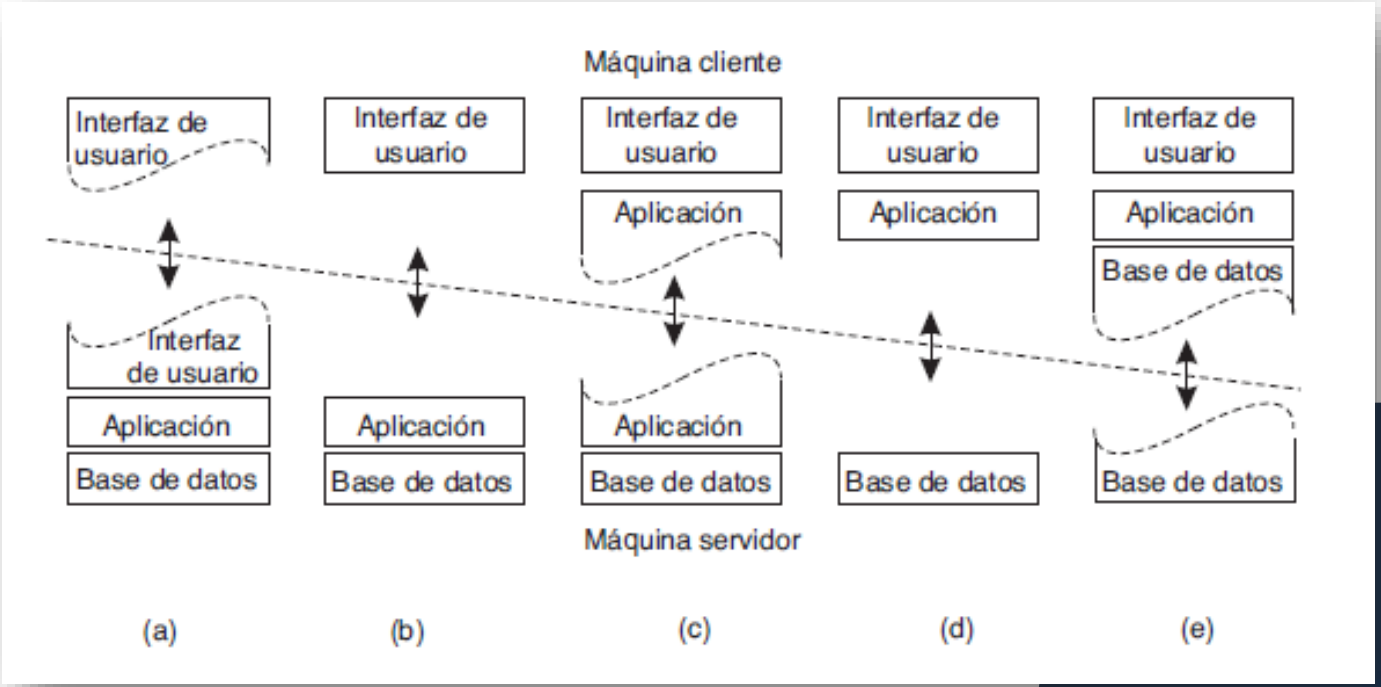
1. El nivel de interfaz de usuario.
2. El nivel de procesamiento.
3. El nivel de datos.



Tanenbaum, A. S. (2008). Sistemas Distribuidos / Principios y Paradigmas / 2da ed

Arquitecturas multiniveles

Un método efectivo para organizar clientes y servidores es distribuir los programas en capas de aplicación (arquitecturas multiniveles), Como primer paso, diferenciamos sólo dos tipos de máquinas: máquinas cliente y máquinas servidor, lo cual nos lleva a lo que se conoce como arquitectura de dos capas.

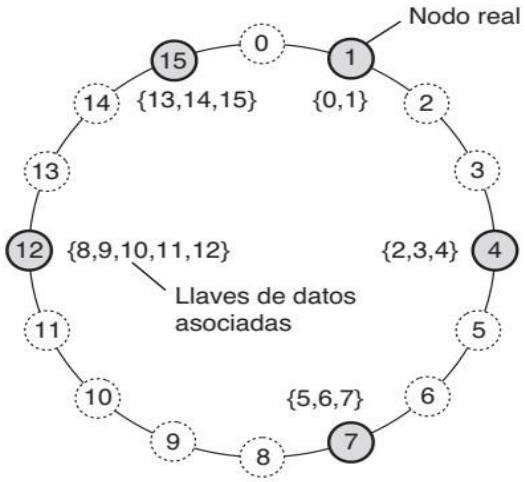


Tanenbaum, A. S. (2008). Sistemas Distribuidos / Principios y Paradigmas / 2da ed

Arquitecturas descentralizadas

Las arquitecturas cliente-servidor multiniveles son una consecuencia directa de dividir aplicaciones para obtener una interfaz de usuario, componentes de procesamiento, y un nivel de datos.

- Arquitecturas estructuradas de punto a punto
- Arquitecturas de punto a punto no estructuradas



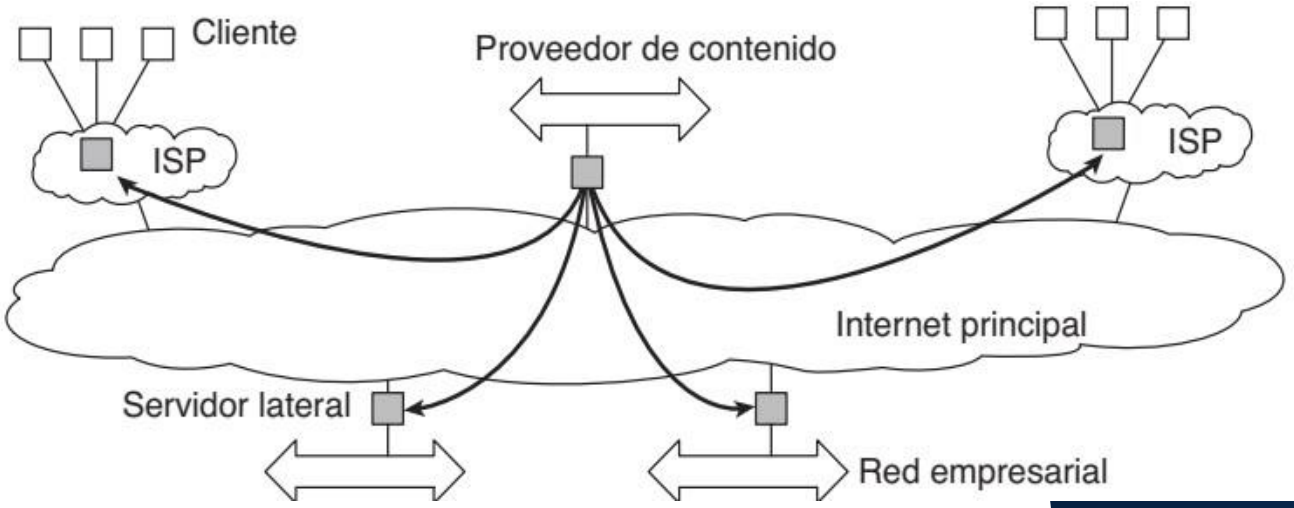
https://www.youtube.com/watch?v=mkSgTE_bEuU

Tanenbaum, A. S. (2008). Sistemas Distribuidos / Principios y Paradigmas / 2da ed

Arquitecturas hibridas

- **Sistemas de servidores al borde**

Los usuarios finales, o clientes en general, se conectan a internet mediante un servidor lateral. El objetivo principal del servidor lateral es proporcionar contenido, probablemente después de realizar un filtrado y descodificar funciones. Más interesante resulta el hecho de que una colección de servidores laterales pueda utilizarse para optimizar la distribución de contenido y aplicaciones. El modelo básico es que, para una organización específica, un servidor lateral actúa como servidor de origen a partir del cual se origina todo el contenido. Ese servidor puede utilizar otros servidores laterales para replicar páginas web y similares

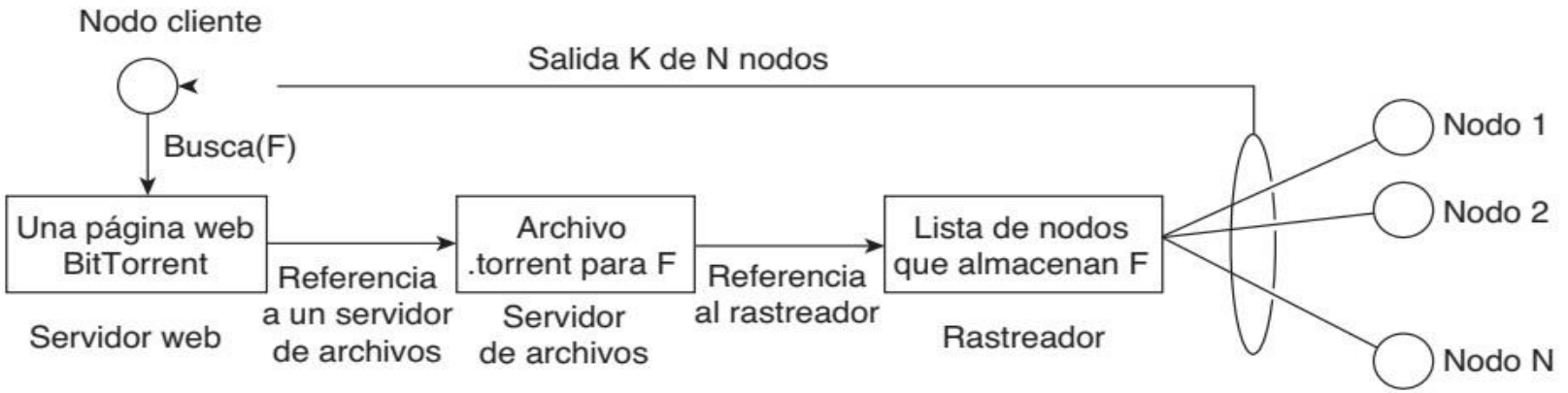


Tanenbaum, A. S. (2008). Sistemas Distribuidos / Principios y Paradigmas / 2da ed

Arquitecturas hibridas

- **Sistemas distribuidos en colaboración**

BitTorrent es un sistema de descarga de archivos de punto a punto. Su funcionamiento principal aparece en la figura 2-14. La idea básica es que cuando un usuario final busca un archivo, BitTorrent descarga partes del archivo de otros usuarios hasta que las partes descargadas pueden ensamblarse y entregar el archivo completo. Un objetivo de diseño importante era garantizar la colaboración. En la mayoría de los sistemas de intercambio de archivos, una fracción importante de participantes simplemente descarga archivos



Tanenbaum, A. S. (2008). Sistemas Distribuidos / Principios y Paradigmas / 2da ed

ARQUITECTURAS VERSUS MIDDLEWARE

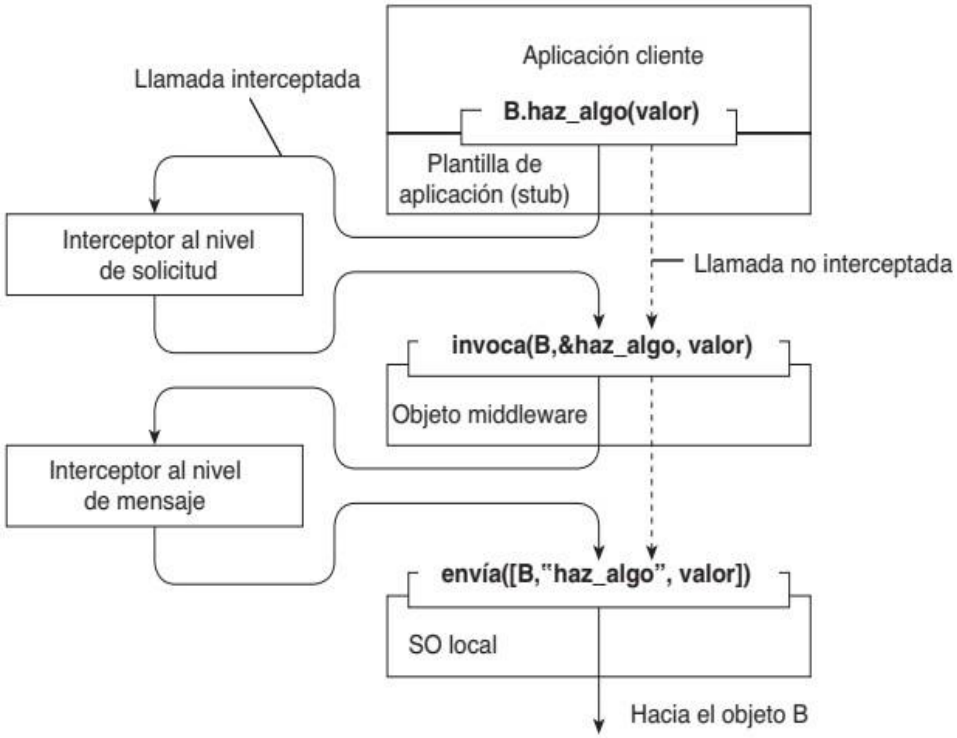
Hacer que el middleware se moldee de acuerdo con un estilo arquitectónico específico tiene el beneficio de que las aplicaciones de diseño pueden volverse más sencillas. Sin embargo, una desventaja evidente es que entonces el middleware ya no puede ser óptimo para lo que un desarrollador de aplicaciones tenía en mente. Además, aunque el middleware tiene como objetivo proporcionar transparencia de distribución, en general, se percibe que soluciones específicas deben ser adaptables a los requerimientos de las aplicaciones. Una solución para este problema es desarrollar diversas versiones de un sistema middleware, en donde cada versión se confeccione para una clase específica de aplicaciones.

https://www.youtube.com/watch?v=UKvfu8xuhvY

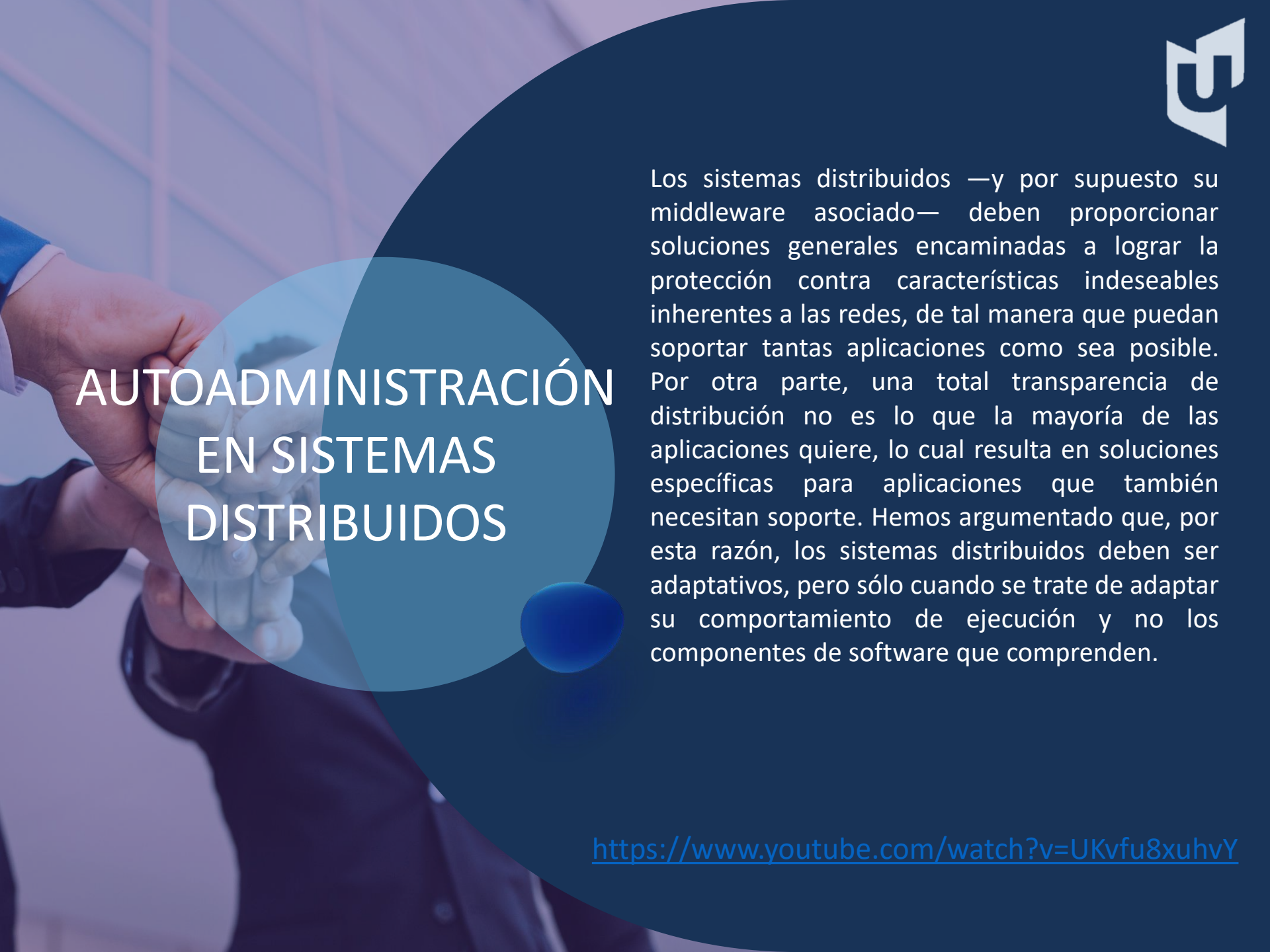
Interceptores

De manera conceptual, un interceptor no es otra cosa que una construcción de software que romperá el flujo usual de control y permitirá que otro código (aplicación específica) se ejecute.

La idea básica es sencilla: un objeto A puede llamar a un método que pertenece a un objeto B, mientras que el objeto B reside en una máquina diferente de A.



Tanenbaum, A. S. (2008). Sistemas Distribuidos / Principios y Paradigmas / 2da ed



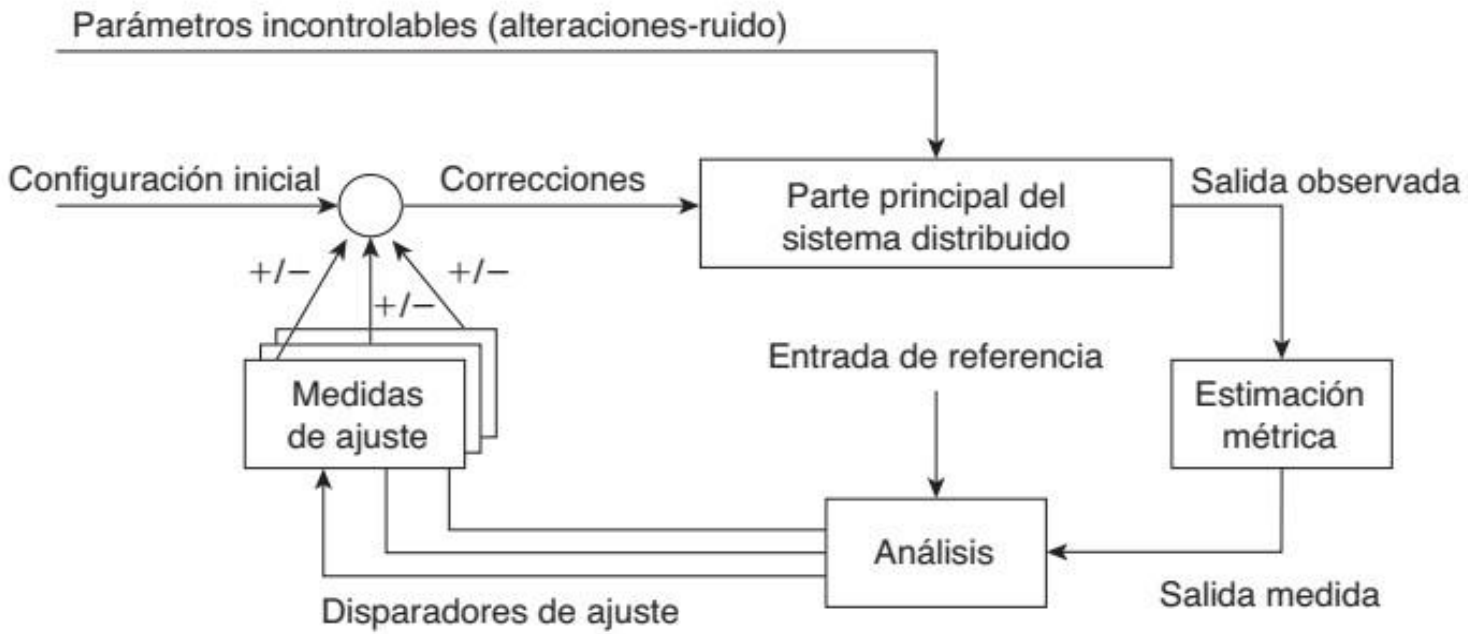
AUTOADMINISTRACIÓN EN SISTEMAS DISTRIBUIDOS

Los sistemas distribuidos —y por supuesto su middleware asociado— deben proporcionar soluciones generales encaminadas a lograr la protección contra características indeseables inherentes a las redes, de tal manera que puedan soportar tantas aplicaciones como sea posible. Por otra parte, una total transparencia de distribución no es lo que la mayoría de las aplicaciones quiere, lo cual resulta en soluciones específicas para aplicaciones que también necesitan soporte. Hemos argumentado que, por esta razón, los sistemas distribuidos deben ser adaptativos, pero sólo cuando se trate de adaptar su comportamiento de ejecución y no los componentes de software que comprenden.

https://www.youtube.com/watch?v=UKvfu8xuhvY

El modelo de control de retroalimentación

Existen muchas opiniones diferentes acerca de los sistemas de autoadministración, pero lo que más tienen en común, es la suposición de que las adaptaciones se llevan a cabo mediante uno o más ciclos de control de retroalimentación. La parte central de un sistema de control de retroalimentación se forma con los componentes que necesitan administrarse. Se supone que estos componentes se manejan mediante parámetros de entrada controlables, pero su comportamiento puede verse influenciado por todo tipo de entrada no controlable



Tanenbaum, A. S. (2008). Sistemas Distribuidos / Principios y Paradigmas / 2da ed

FUNDACIÓN DE EDUCACIÓN SUPERIOR
SANJOSÉ
INSTITUCIÓN TECNOLÓGICA

FIN DE GRABACIÓN