Sistemas distribuidos

Conceptos básicos

UNIVERSIDAD NEBRIJA

Índice

- Definición y puntos clave
- Parámetros de diseño
- Modelos de sistema
- Modelos fundamentales

Definición y puntos clave

- Sistema distribuido: una colección de COMPUTADORES INDEPENDIENTES LIGERAMENTE ACOPLADOS que APARECEN ANTE LOS USUARIOS DEL SISTEMA COMO UNA ÚNICA COMPUTADORA
  - Máquinas autónomas
  - Software unificado desde el punto de vista del usuario
- Componentes hardware y software unidos mediante RED que se comunican y coordinan sólo MEDIANTE PASO DE MENSAJES

Seamos más precisos con la definición de Sistemas Distribuidos

- No comparten:
  - Espacio de memoria común
  - Ni un mismo reloj de ejecución

Otras deficiones de Sistemas Distribuidos

“Un sistema distribuido es aquel en el que no puedes trabajar con tu máquina por el fallo de otra máquina que ni siquiera sabías que existía” -Leslie Lamport

“Un sistema distribuido es aquel en el que los computadores localizados en una red comunican y coordinan sus acciones mediante paso de mensajes” -George Coulouris

¿Qué sistemas conocemos?

- | Tercer curso 60 ECTS

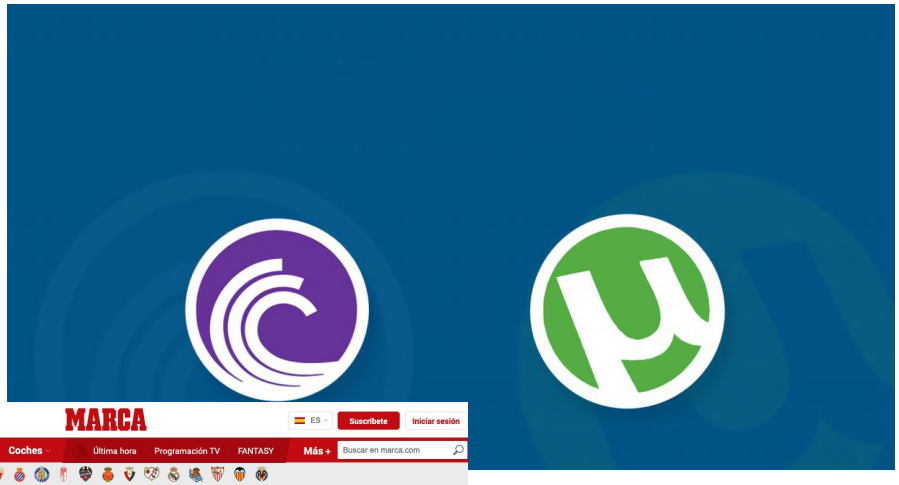
Primer Semestre 30 ECTS

- 6 ECTS | Sistemas Operativos
- 6 ECTS | Arquitectura de Computadores
- 6 ECTS | Sistemas Empotrados de Tiempo Real
- 6 ECTS | Gestión de Proyectos Tecnológicos
- 6 ECTS | Arquitectura y Programación de Sistemas en Internet

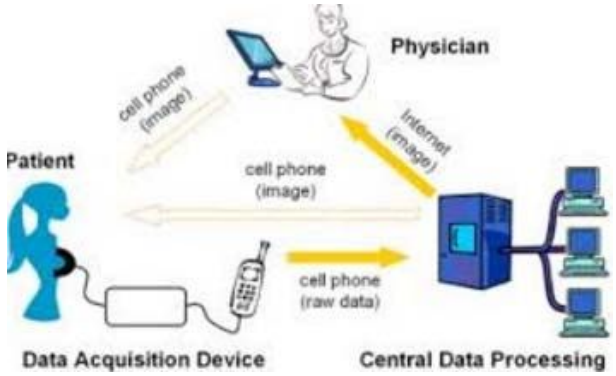
Segundo Semestre 30 ECTS

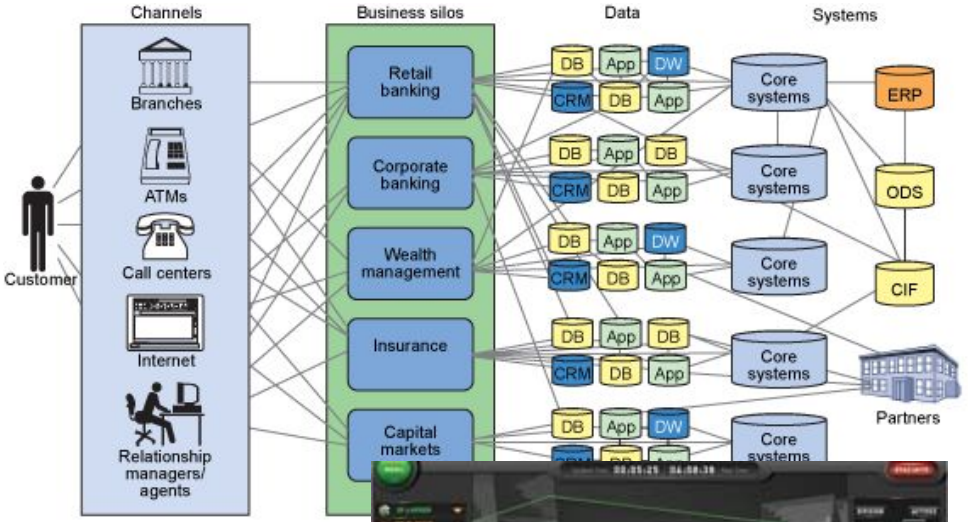
- 6 ECTS | Programación de Interfaces Web
- 6 ECTS | Programación de Sistemas Distribuidos
- 6 ECTS | Programación de Sistemas y Dispositivos
- 6 ECTS | Diseño Automático de Sistemas
- 6 ECTS | Inteligencia Artificial

Definición y puntos clave



Definición y puntos clave





Definición y puntos clave

- Sistema distribuido:
  - Nivel de abstracción:
    - Espacios de objetos y aplicaciones colaborativas
    - Red
    - Llamada a procedimientos remotos e invocación de métodos remotos (Remote Procedure Call-RPC)
    - Intercambio/paso de mensajes (servicio)

Definición y puntos clave

- Motivaciones:
  - Los sistemas distribuidos suelen tener una MEJOR PROPORCIÓN PRECIO/RENDIMIENTO que un sistema centralizado
  - Los sistemas distribuidos pueden alcanzar RENDIMIENTOS INEXISTENTES en máquina CENTRALIZADAS
    - *Ejemplo*: 10.000 microprocesadores vs la CPU más cara del mercado
    - *Ejemplo*: Cadena de supermercados con aplicaciones individuales por tienda y aplicaciones conjuntas de inventario
    - *Ejemplo*: Sistemas donde el control es crítico como centrales nucleares o aviones
  - Permite COMPARTIR RECURSOS de forma rápida y transparente
    - *Ejemplo*: Repartir la CARGA COMPUTACIONAL

Definición y puntos clave

- Los sistemas distribuidos tienen como principal objetivo COMPARTIR RECURSOS. Entendemos por recursos componentes hardware (discos duros, impresoras, etc.) y componentes software (ficheros, bases de datos, etc.)
    - Componentes conectados en red se coordinan ÚNICAMENTE mediante el PASO DE MENSAJES
    - CONCURRENCIA en los componentes DIVERSOS
    - CARENCIA de un reloj global
    - Posibilidad de FALLOS en cada uno de los componentes

Definición y puntos clave

- Concurrencia: Es necesaria la coordinación de los programas existentes para, por ejemplo, la correcta compartición de recursos. Esta coordinación se realiza mediante el intercambio de mensajes

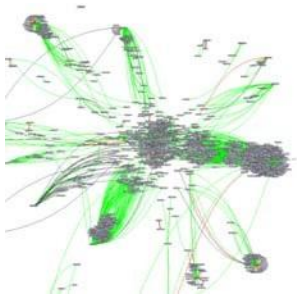
- Carencia de reloj global: Para el correcto intercambio de mensajes se requiere conocer el **momento exacto** en el que se emiten. La precisión para sincronizar relojes de diferentes componentes no es suficiente en la mayoría de los casos

- Fallos en cada uno de los componentes: Deben planificarse las acciones a aplicar cada vez que se produzca un fallo. En función de su origen los fallos tendrán una u otra consecuencia. Por ejemplo, no tendrá el mismo efecto un fallo en la red que un fallo en un nodo de cálculo

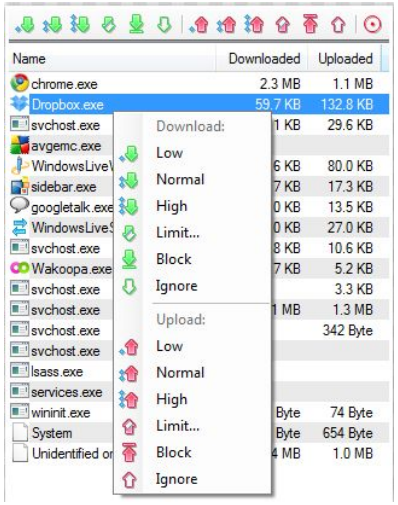
Definición y puntos clave

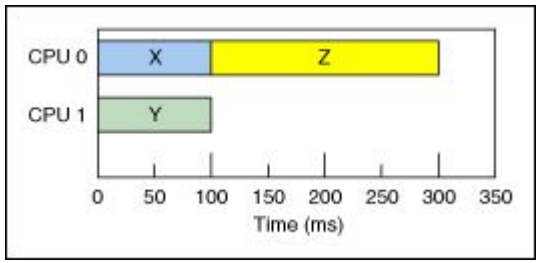
- Otras problemáticas presentes:
  - Aumento de la complejidad del software a diseñar
  - Totalmente ligado a las topologías de red
    - *Ejemplo*: Saturación del tráfico de red (cuellos de botella)
  - Los datos se comparten con otros sistemas por lo que la seguridad se puede ver comprometida

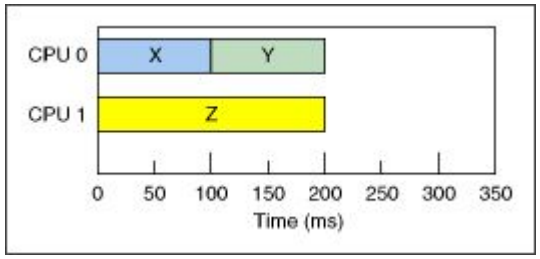


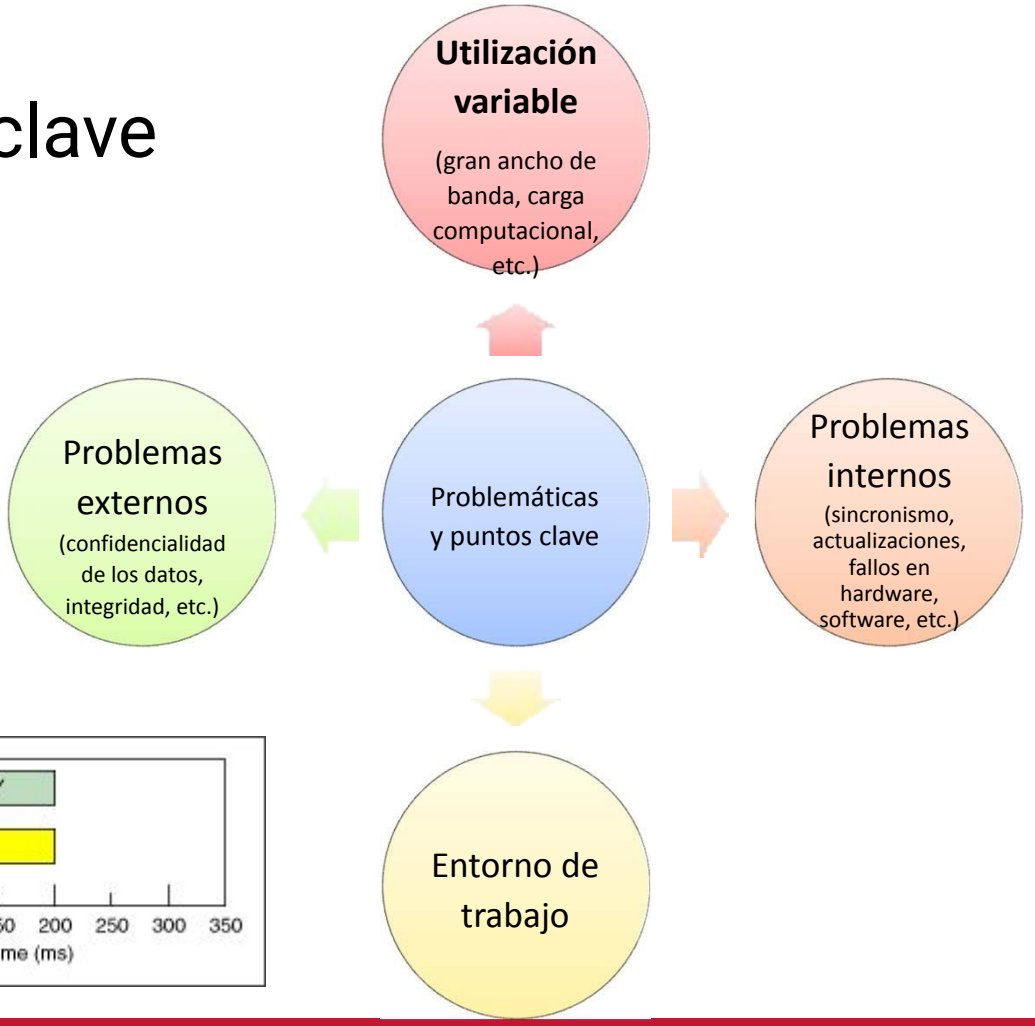




Definición y puntos clave

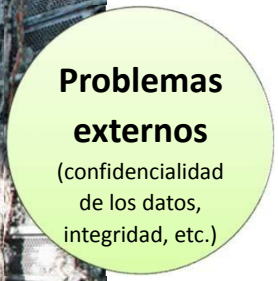




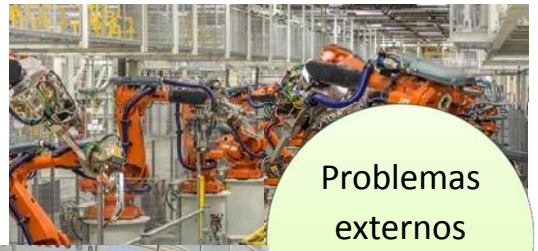




Definición y puntos clave



Definición y puntos clave



Problemas externos (confidencialidad de los datos, integridad, etc.)

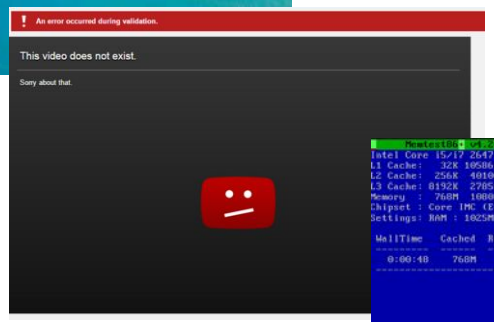
Utilización variable (gran ancho de banda, carga computacional, etc.)

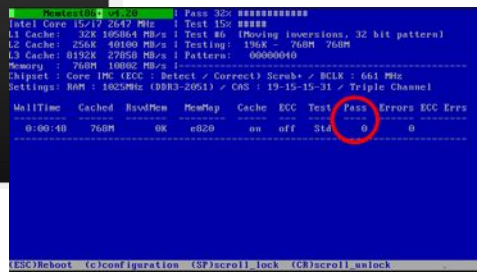
Problemáticas y puntos clave

Problemas internos (sincronismo, actualizaciones, fallos en hardware, software, etc.)

Entorno de trabajo

Definición y puntos clave





Utilización variable

(gran ancho de banda, carga computacional, etc.)

Problemas externos

(confidencialidad de los datos, integridad, etc.)

Problemáticas y puntos clave

Problemas internos

(sincronismo, actualizaciones, fallos en hardware, software, etc.)

Entorno de trabajo

UNIVERSIDAD NEBRIJA

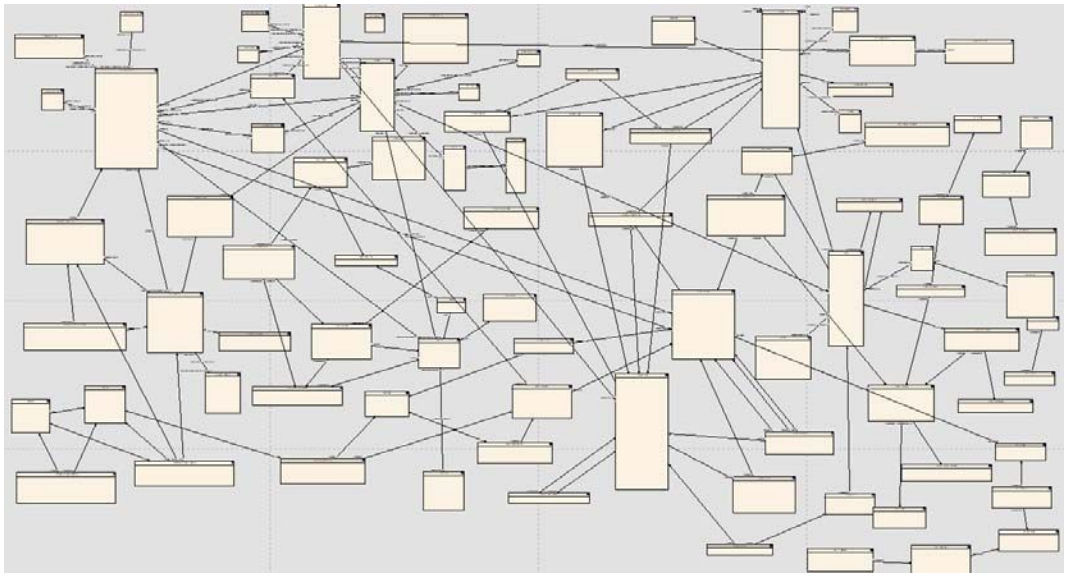
Actividad

Indica si para cada una de las actividades descritas se produce o no una computación distribuída:

- Utilizar un cliente de mensajería instatánea
- Usar un procesador de textos como Open Office
- Acceder a una web
- Jugar a un videojuego en una máquina multiprocesador sin conexión a internet

Parámetros de diseño

- Heterogeneidad
- Extensibilidad
- Seguridad
- Escalabilidad
- Gestión de los fallos
- Concurrencia
- Transparencia



Parámetros de diseño

- Heterogeneidad: Variedad y diferencias en los componentes del sistema
  - Hardware de los nodos (Intel, AMD, Xilinx, etc.)
  - Sistemas operativos de los nodos (Windows, Linux, etc.)
  - Lenguajes de programación para las aplicaciones (Java, C++, C#, etc.)
  - Redes (Ethernet, X25, etc.)
  - Representación de los datos
  - Las diferencias y variedades en los sistemas no desaparecen, pero es necesario buscar una cierta estandarización (PROTOCOLOS)

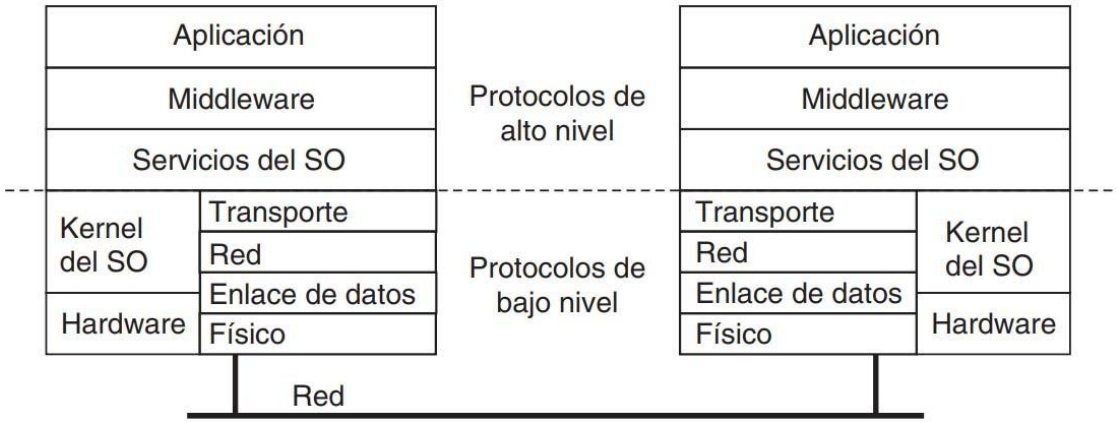
Parámetros de diseño

- Heterogeneidad: Variedad y diferencias en los componentes del sistema
  - Middleware: enmascaramiento de/abstracción de la heterogeneidad
    - Proporciona un modelo COMPUTACIONAL UNIFORME
    - Permite la invocación de objetos OCULTANDO que se intercambio de mensajes en red
      - *Ejemplos*: CORBA, Java RMI, etc.

  - Código móvil: código que puede ser ejecutado en cualquier máquina ejemplo, el basado en máquina virtual

Parámetros de diseño

- Heterogeneidad: Variedad y diferencias en los componentes del sistema
  - Middleware: enmascaramiento de/abstracción de la heterogeneidad



Actividad

Dibujar mediante un diagrama de eventos y uno de secuencia cúal es el intercambio de mensajes entre un cliente web (un navegador) y un servidor.

Parámetros de diseño

- Extensibilidad: Posibilidad de re-implementación (nuevas funcionalidades)
  - Posibilidad de añadir nuevos servicios que:
    - Compartan recursos
    - Se extiendan a usuarios heterogéneos (programas cliente)
  - Open distributed systems: Independientes de proveedores concretos
  - Especificaciones y documentación software disponibles para los desarrolladores: INTERFACES PÚBLICAS
  - Permite la invocación de objetos ocultando que se realiza un intercambio de mensajes en red

Parámetros de diseño

- Seguridad:
  - Confidencialidad: Protección contra el descubrimiento de individuos no autorizados
  - Integridad: Protección contra la alteración o corrupción
  - Autentificación: Nadie te puede suplantar
  - Disponibilidad: Protección contra el difícil acceso a los recursos
  - *Problemática:* En los sistemas distribuidos la información *recorre nodos en los que no controlamos* qué uso se hace de la información, cómo se trata esta información y qué retardos van a introducir los sistemas

Parámetros de diseño

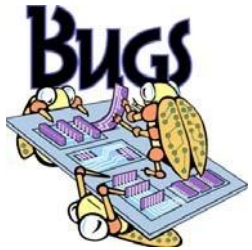
- Escalabilidad: Aumento del número de recursos DEL MISMO TIPO y del número de Usuarios. **IMPORTANTE: NO es EXTENSIBILIDAD**

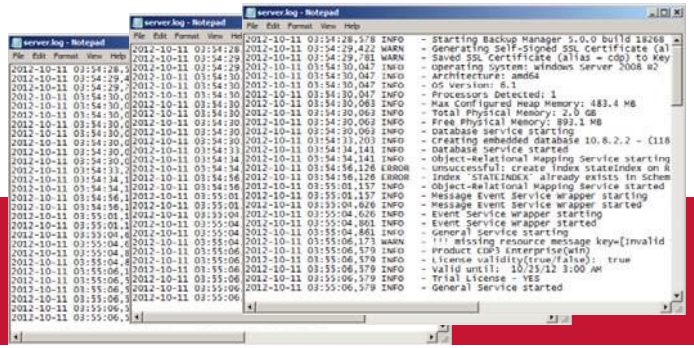
  - Mantener una relación lineal entre recursos y demandas
    Ejemplo: *n* clientes para un servidor, *2n* clientes para dos servidores
  - Mantener constantes las prestaciones del servicio
    Ejemplo: Si el QoS de un sistema es del 99% con *n* clientes, debe mantenerse igual para *kn* clientes
  - Mantener la estabilidad numérica de los sistemas
    Ejemplo: Correcto dimensionamiento de los datos representados con el fin de evitar errores por truncamientos o pérdidas de precisión
  - Gestionar los cuellos de botella
    Ejemplo: Insertar memorias caché para el acceso rápido a la información

Parámetros de diseño

- Gestión de fallos: Los sistemas distribuidos suelen tener fallos parciales (algunos componentes funcionan correctamente mientras otros dejan de funcionar) lo cual DIFICULTA LA GESTIÓN DE FALLOS

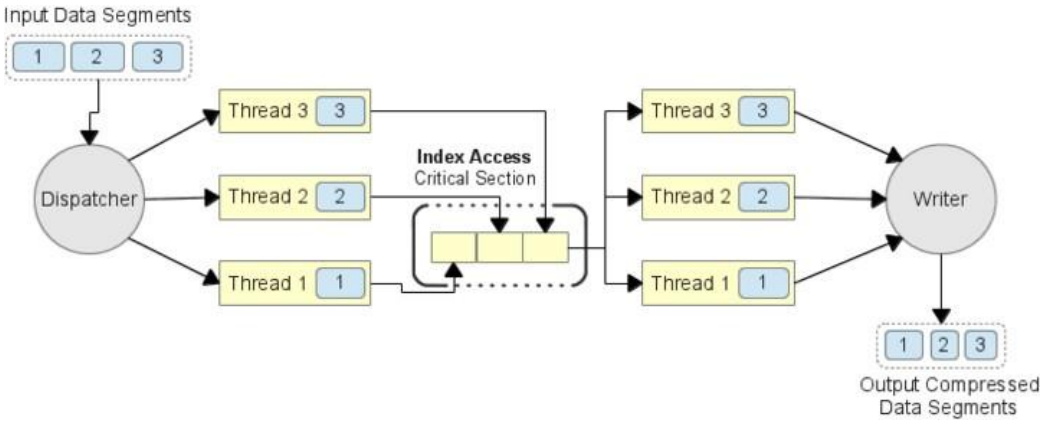
  - Detección/corrección de fallos: CRC's, algoritmos FEC, etc.
  - Enmascaramiento de fallos: retransmisión de la información, duplicidad de archivos
  - Recuperación de estados previos: bases de datos roll back
  - Redundancia: replicación de elementos dentro del sistema (repositorios, rutado múltiple en la red, duplicidad de las bases de datos, etc.)





Parámetros de diseño

- Concurrencia: Acceso simultáneo a un mismo recurso por clientes diferentes
  - Múltiples hilos que se ejecutan en paralelo y que pueden entrar en conflicto al intentar utilizar un recurso compartido. Una posible solución es el uso de semáforos



Parámetros de diseño

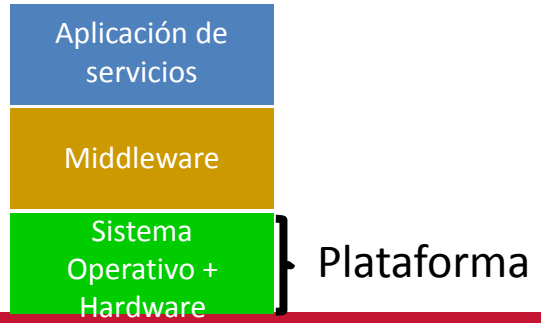
- Transparencia: Percepción del sistema como un elemento ÚNICO no como una colección de elementos interdependientes
  - Tipos de transparencia:
    - Acceso (Heterogeneidad): se utilizan los recursos como si fueran LOCALES
    - Ubicación (Seguridad): se desconoce la localización real del recurso
    - Concurrencia: se comparten recursos SIN que cada usuario sea consciente de ello
    - Replicación: se utilizan múltiples instancias del mismo recurso sin que los usuarios finales sean conscientes de ello
    - Fallos: se puede UTILIZAR el sistema aunque se haya producido algún tipo de ERROR
    - Escalado: se puede modificar la estructura del sistema SIN afectar a los clientes

Parámetros de diseño

- Transparencia: Percepción del sistema como un elemento ÚNICO no como una colección de elementos interdependientes
  - Tipos de transparencia:
    - Prestaciones: se puede modificar la configuración del sistema para MANTENER el rendimiento
    - Movilidad: se puede alterar la disposición de los recursos sin que el sistema deje de funcionar

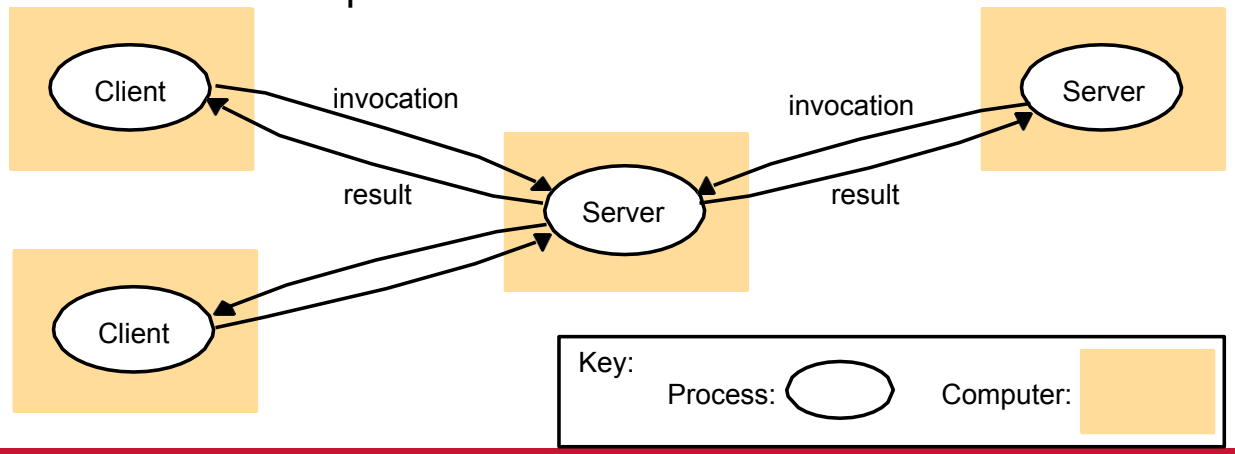
Modelos de sistema

- Definiciones para sistemas distribuidos:
  - Plataforma: nivel hardware (procesado e infraestructura de red)+ Sist. Operativo
  - Middleware: procesos, objetos y servicios que interactúan entre sí para facilitar la compartición de recursos (debe respetar los parámetros de diseño anteriores)
    - Invocación remota
    - Notificación de eventos (excepciones)
    - Transmisión de datos
  - Aplicación de servicios finales



Modelos de sistema

- Arquitecturas:
  - Modelo cliente-servidor: Los procesos clientes interaccionan con los procesos servidores que gestionan el uso de los diferentes recursos
    - Los servidores pueden ser a su vez clientes de otros servidores



Modelos de sistema

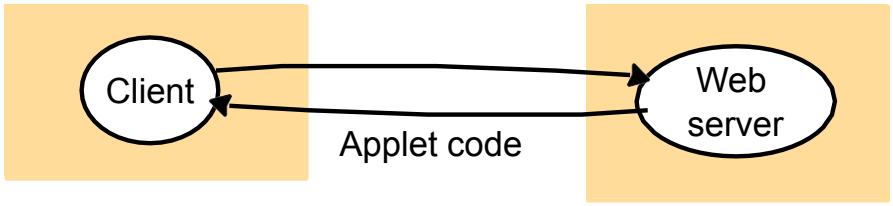
- Arquitecturas:
  - Modelo cliente-servidor (Variaciones)
    - Código móvil: se descarga el código a ejecutar de un servidor, pero la aplicación se ejecuta en el cliente
    - Agentes móviles: programa en ejecución que se traslada de un nodo a otro de la red recolectando información o solicitando recursos locales de los lugares que visita
    - Thin clients (clientes ligeros/delgados): capa de aplicación que incluye una interfaz de usuario, mientras que el procesado se realiza en un computador remoto. Sufre de fuertes latencias cuando el intercambio de información es muy elevado (CAD)

      Ejemplo: www.citrix.com

Modelos de sistema

- Arquitecturas:
  - Modelo cliente-servidor (Variaciones)

a) client request results in the downloading of applet code



b) client interacts with the applet

Modelos de sistema

- Arquitecturas:
  - Modelo cliente-servidor

Client side attacks using evil JAVA applets

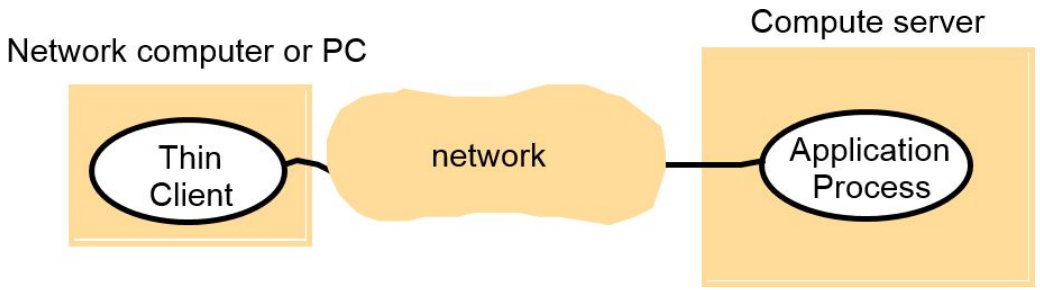
Código móvil

Agente móvil

ATLAS@Home

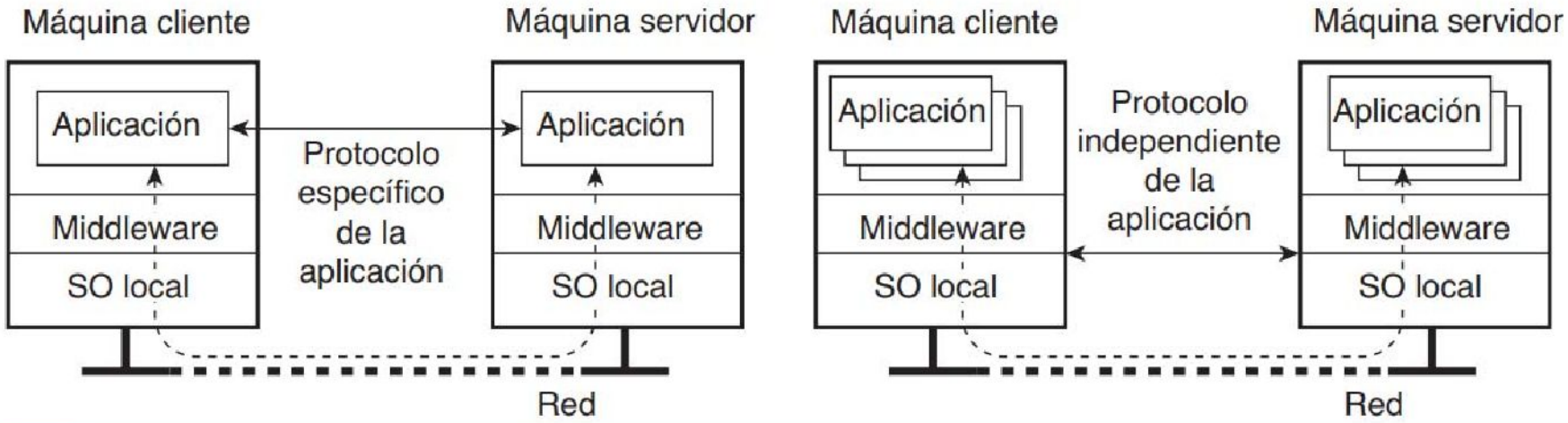
Modelos de sistema

- Arquitecturas:
  - Modelo cliente-servidor (Variaciones)



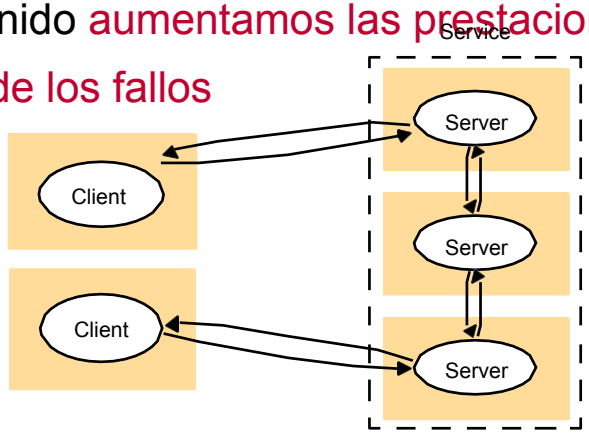
Modelos de sistema

- Arquitecturas:
  - Modelo cliente-servidor (Variaciones)



Modelos de sistema

- Arquitecturas:
  - Modelo multiservidor: Los servidores pueden dividir el conjunto de objetos en los que está basado el servicio y mantener distribuirlos entre diferentes nodos o incluir réplicas del mismo recurso en diferentes servidores
    - Replicando el mismo contenido aumentamos las prestaciones, la disponibilidad y la gestión de los fallos



Modelos de sistema

- Arquitecturas:
  - Modelo peer-to-peer: donde los procesos asumen el mismo rol, no hay distinciones, y en general supone una programación más compleja. Cada proceso debe ser capaz de hacer peticiones, de recibir su repuesta, de recibir peticiones de otros procesos, etc. Esa homogeneidad en lso procesos permite crear grandes redes de computadores donde todos ellos se comunican con todos.

Modelos de sistema

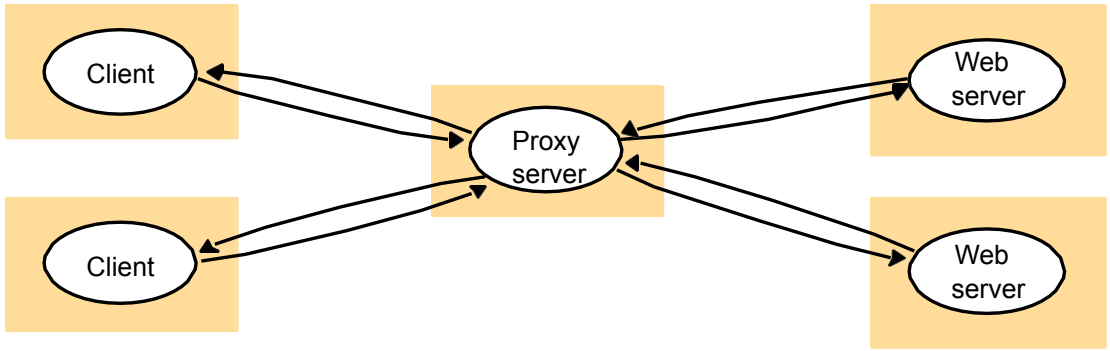
- Arquitecturas:
  - Proxy y cachés:
    - La arquitectura proxy, consiste en situar una capa de servicio entre los servidores y los clientes
    - Dicha capa ofrece principalmente un servicio de redireccionamiento hacia los servidores, que son los que ofrecen un servicio de procesamiento

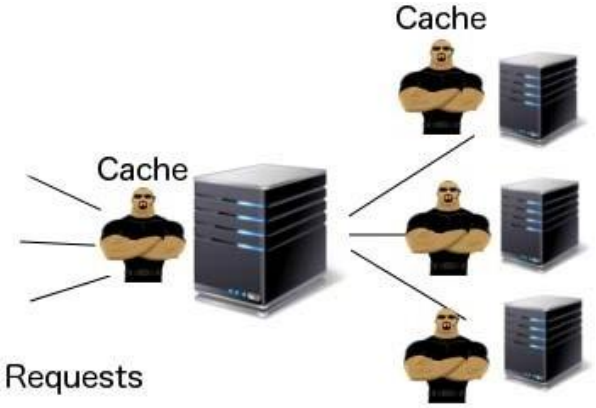
    - La caché almacena los recursos utilizados recientemente con el fin de reducir los tiempos de acceso entre los clientes y los servidores
    - Las caché pueden situarse en cada uno de los clientes o en el servidor

    - Los servidores proxy y las caché pretenden incrementar la disponibilidad, la distribución de los servicios y la seguridad

Modelos de sistema

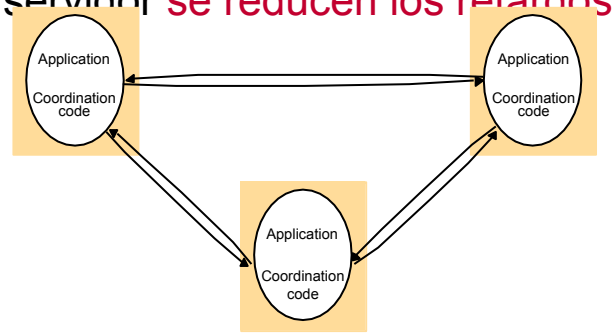
- Arquitecturas:
  - Proxy y cachés:





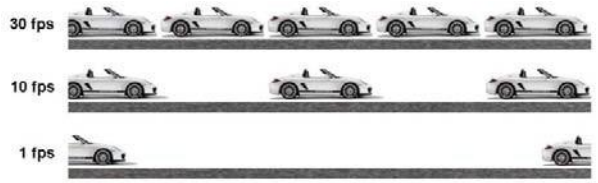
Modelos de sistema

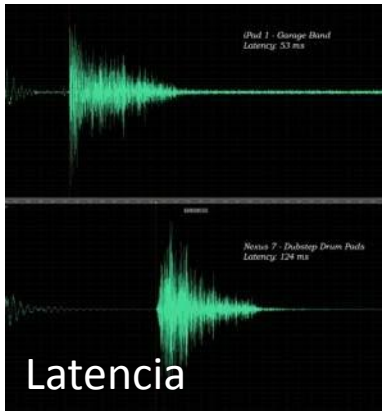
- Arquitecturas:
  - Peer to peer: Todos los componentes del sistema desempeñan tareas similares.
    - Cooperan para poder completar una tarea entre todos SIN distinción entre clientes y servidores
    - Se requiere mantener la consistencia en recursos y sincronización
    - Al no existir un servidor se reducen los retardos de comunicación



Modelos de sistema

- Requisitos de diseño:
  - Prestaciones (Interacción):
    - Responsiveness: Rapidez y consistencia con las interfaces. Latencia
    - Throughput: Tasa de transmisión de la información
    - Balanceo de la carga computacional
  - QoS:
    - Fiabilidad (gestión de fallos)
    - Seguridad
    - Disponibilidad de recursos





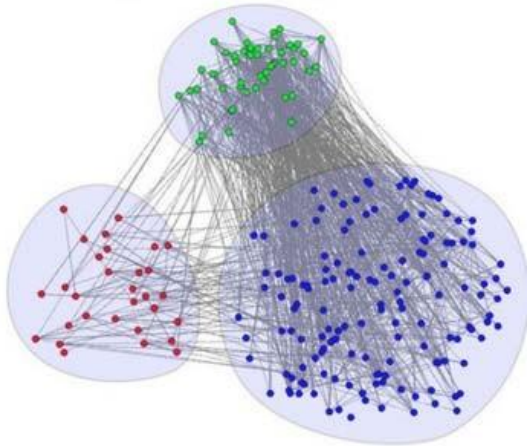
Actividad

¿Qué tipo de sistemas distribuído sería el más indicado en cada caso: cliente-servidor, peer-to-peer o sistema de mensajes?

a) Notificador de canastas de un equipo de baloncesto
b) Servidor de nombres de DNS
c) Spotify (comparte música)
d) Un canal de mensajería instantánea coordinado por un administrador
e) Sistema de apuestas online

Modelos fundamentales

- Interacción:
  - Paso de mensajes
  - Flujo de información
  - Coordinación de actividades: noción del tiempo
- Fallo:
  - Clasificación
  - Análisis
  - Corrección
- Seguridad:
  - Clasificación
  - Análisis
  - Prevención/solución



Modelos fundamentales

- Interacción:
  - Prestaciones del canal:
    - **Latencia**: retardo entre el envío de un mensaje y su recepción
      Latencia = Retardo de **propagación** + Retardo de **red**+ Retardo del **OS**
      - **Ancho de banda** (throughput): cantidad total de datos en un intervalo de tiempo dado
      - **Jitter**: fluctuación del tiempo de llegada de los datos
  - Relojes de computadores y eventos de temporización:
    - Tasa de **deriva de los relojes**: diferencia entre un reloj y su referencia
    - Definición de eventos para intentar sincronizar las operaciones
  - Variantes de modelos de interacción:
    - Sistemas distribuidos **síncronos**
    - Sistemas distribuidos **asíncronos**

Modelos fundamentales

- Interacción:
  - Sistemas síncronos
    - El **retardo** del canal (retardo de propagación) **está modelado**
    - Se **conoce la deriva del reloj** (oscilador) local
    - El tiempo de ejecución tiene unos **límites inferiores y superiores acotados**

  - Se define un TIMEOUT, superado el cual se supone que se ha producido un fallo en el proceso (ya que no se ha ajustado a los márgenes modelados)

  - Sistemas asíncronos
    - Retardo del **canal arbitrario**
    - Deriva de **reloj arbitraria**
    - Velocidad de **procesado arbitraria**

Modelos fundamentales

- Interacción:
  - Ordenación de eventos: Necesitamos saber qué evento ocurre antes,
  - después o simultáneamente. NO ES NECESARIO RELOJES PRECISOS

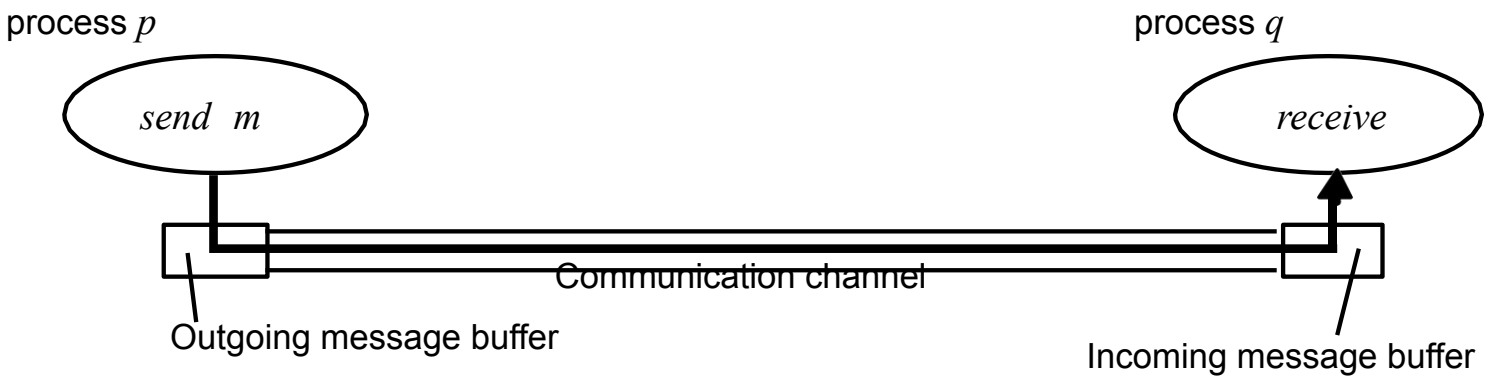
  - Se han propuesto alternativas basadas en un tiempo lógico del sistema: los mensajes que se intercambian llevan una marca temporal

Modelos fundamentales

- Fallo:
  - Fallos por omisión de procesos: Ruptura accidentada de un proceso (*crash*)
    - Es deseable que los procesos funcionen correctamente o se detengan
    - Se pueden utilizar *timeouts* para reiniciar los procesos:
      - Sistema síncrono, si no responde dentro del tiempo existe un fallo
    - Sistema asíncrono, si no responde puede que falle o puede que simplemente se retrase
  - Fallos por omisión de comunicaciones: Se pierde el mensaje debido a saturación de buffers de entrada o salida, o fallo del canal físico
    - Fallos por omisión de envío: Se dan cuando se pierden mensajes entre el proceso emisor y el buffer de mensajes de salida
    - Fallos por omisión de recepción: Se dan cuando se pierden mensajes entre el buffer de mensajes de entrada y el proceso receptor
    - Fallos por omisión de canal: Se pierden mensajes entre los dos buffers

Modelos fundamentales

- Fallo:
  - Fallos por omisión de comunicaciones



Modelos fundamentales

- Fallo:
  - Fallos arbitrarios: mensajes corrompidos, mensajes duplicados, etc.
    - Son muy poco habituales ya que existen múltiples sistemas para detectarlos
  - Fallos de temporización (sólo sistemas síncronos): se excede el tiempo en cualquiera de los parámetros definidos (retardo de canal, procesado, red)

  - Enmascaramiento de fallos: consisten en la ocultación de un fallo o la conversión de un fallo en un error de otro tipo más aceptable

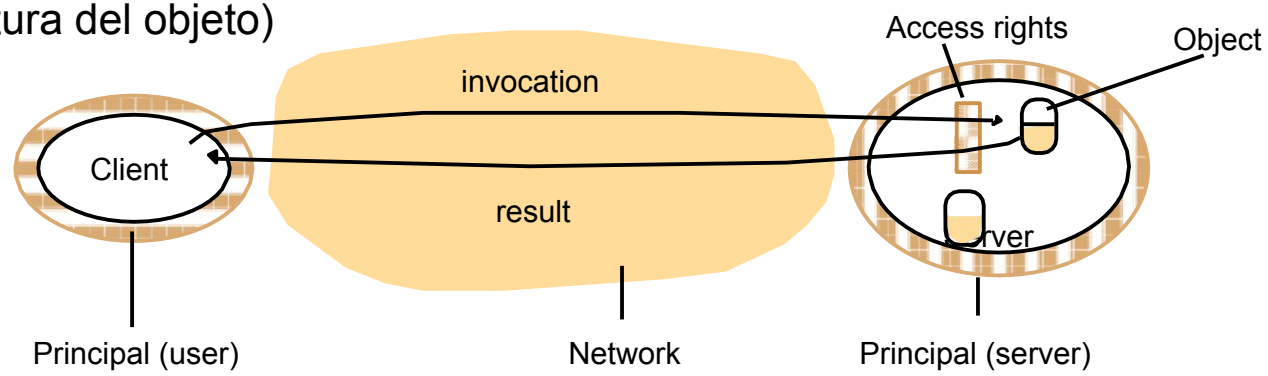
Modelos fundamentales

- Seguridad: asegurar modelos y canales protegiendo los objetos que se intercambian contra el acceso no autorizado
  - Protección de objetos: los usuarios lanzan programas cliente que envían invocaciones al servidor para realizar operaciones sobre los objetos. El servidor realiza la operación y envía el resultado al cliente. Para asegurar su protección COMPRUEBA LOS DERECHOS DE ACCESO (lectura o escritura del objeto)

    - En los sistemas distribuidos, este sistema de derechos de acceso puede estar en una localización diferente a la del objeto o recurso, ya que dicho objeto o recurso puede estar almacenado de forma distribuida, o puede que tenga la propiedad de movilidad lo que dificultaría el tema de los derechos de acceso

Modelos fundamentales

- Seguridad: asegurar modelos y canales protegiendo los objetos que se intercambian contra el acceso no autorizado
  - Protección de objetos: los usuarios lanzan programas cliente que envían invocaciones al servidor para realizar operaciones sobre los objetos. El servidor realiza la operación y envía el resultado al cliente. Para asegurar su protección COMPRUEBA LOS DERECHOS DE ACCESO (lectura o escritura del objeto)

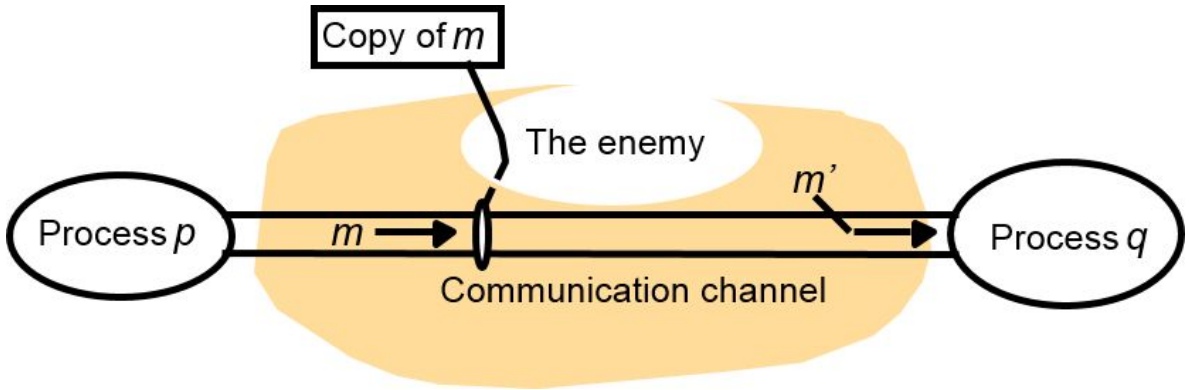


Modelos fundamentales

- Seguridad:
  - Protección de procesos: los mensajes intercambiados entre procesos están expuestos a posibles ataques en la red
    - Enemigo: puede enviar, leer o copiar cualquier mensaje a cualquier proceso
    - Amenazas a procesos: cuando un proceso preparado para admitir peticiones recibe un mensaje de cualquier otro proceso y no es capaz de determinar la identidad
      - Como los servidores reciben peticiones de múltiples clientes es más complicado determinar la identidad de una invocación en particular
      - Por parte de los clientes, la no identificación clara de los servidores origina problemas de suplantación o de trabajo con

Modelos fundamentales

- Seguridad:
  - Protección de procesos: los mensajes intercambiados entre procesos están expuestos a posibles ataques en la red



Modelos fundamentales

- Seguridad:
  - Autenticación: se envía un fragmento encriptado con la identidad del solicitante, identidad del recurso y marca temporal
  - Canales seguros: conocen la identidad de los participantes y sus derechos; asegura la privacidad y la integridad de los datos; sellos de carácter temporal (físicos o lógicos) para evitar el reenvío o la reordenación de los mensajes