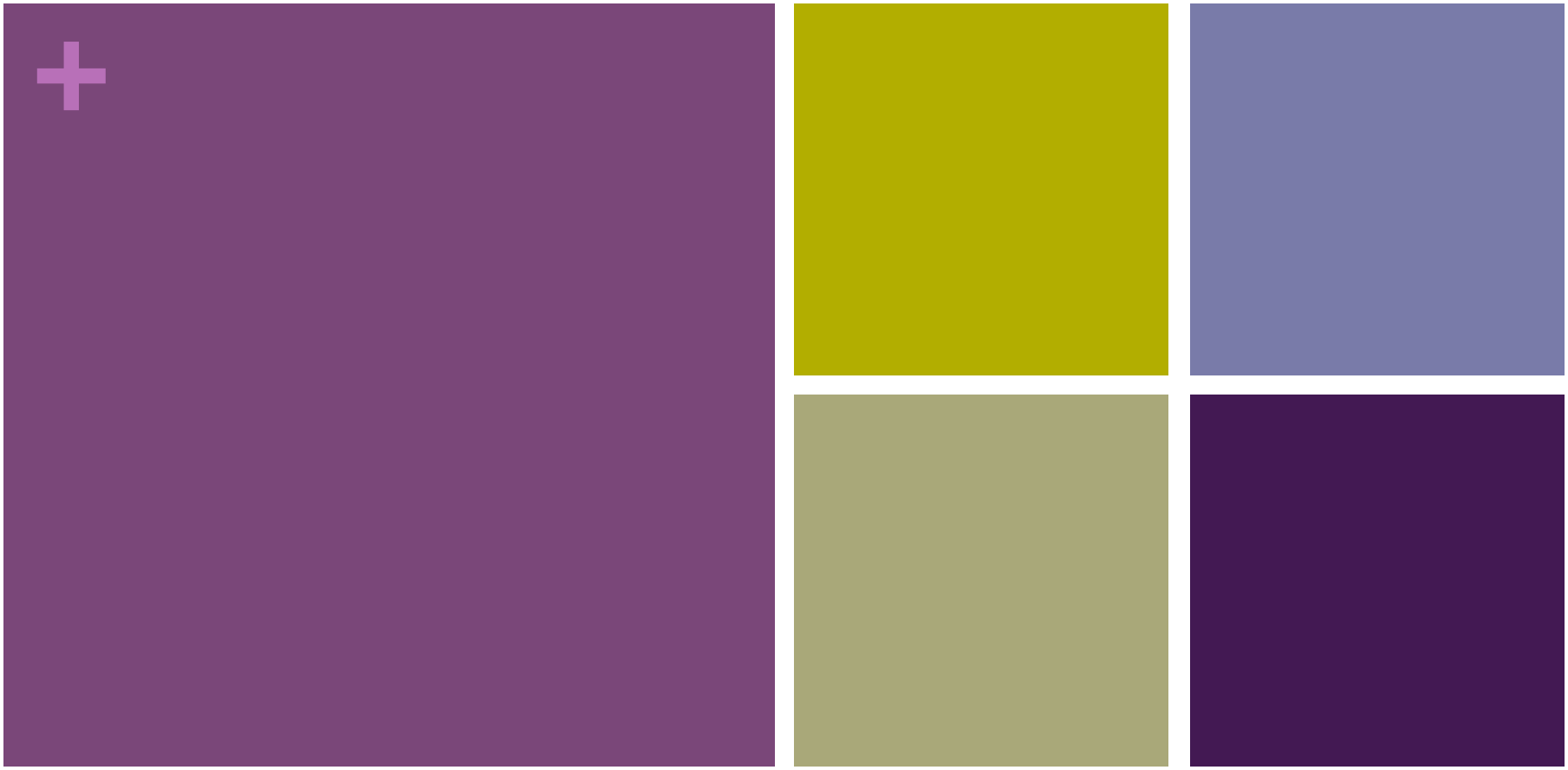


Sistemas Distribuidos
Modelos

Rodrigo Santamaría

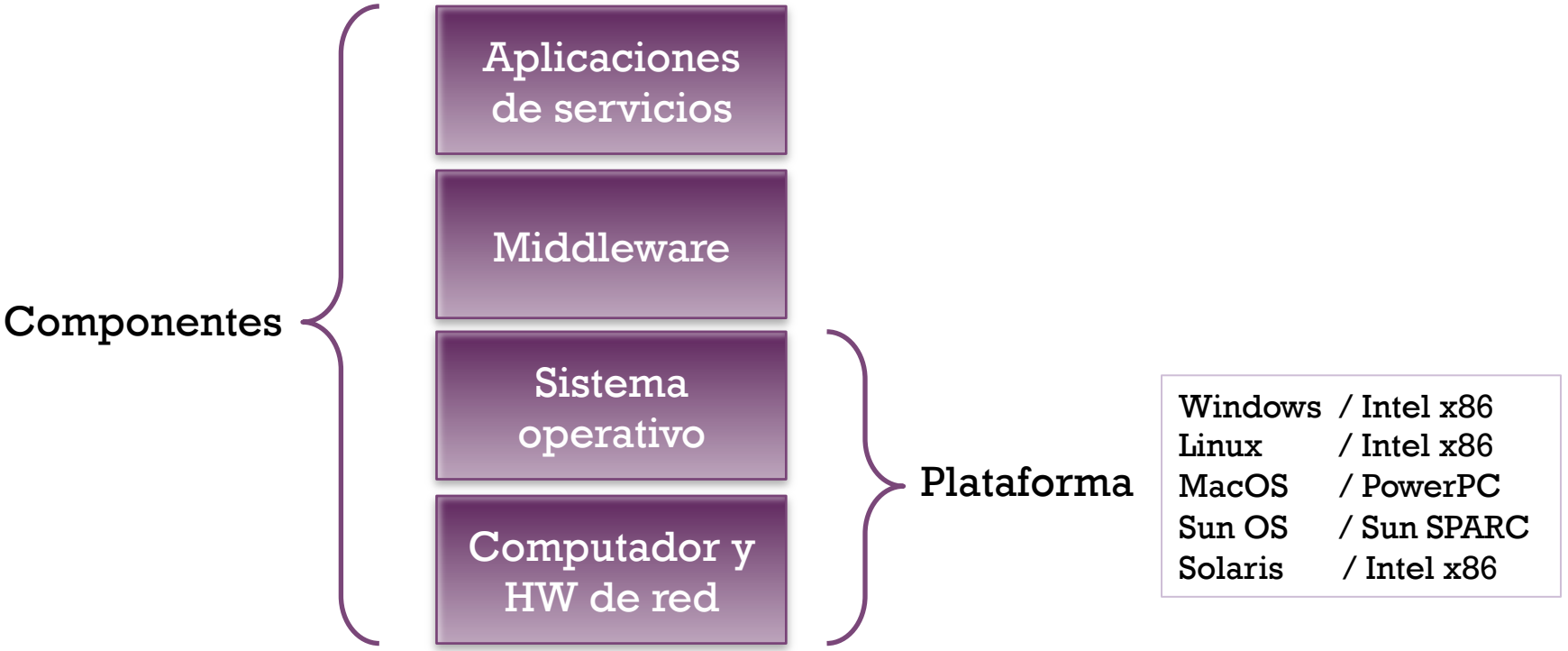
Modelos

- Componentes
  - Plataforma
  - Middleware
- Modelos físicos
- Modelos arquitectónicos
- Modelos fundamentales

Modelos de sistema distribuido

- Todos los tipos de SD tienen características básicas comunes
- **Modelo:** descripción abstracta simplificada pero consistente de cada aspecto relevante del diseño de un SD
- Veremos tres tipos de modelos
  - **Físicos:** representación abstracta de los elementos del sistema
  - **Arquitectónicos**: relación entre los componentes del sistema
  - **Fundamentales**: clasificación según las principales características de un sistema (interacción, fallos, seguridad, etc.)

Componentes de un SD



Componentes de un SD

Middleware

- Capa de SW cuyo propósito es enmascarar la diversidad subyacente y proporcionar un modelo de programación conveniente
- Soporta abstracciones como
  - Procedimientos de invocación remota
  - Comunicación entre grupos de procesos
  - Notificación de eventos
  - Replicación de datos compartidos
  - Transmisión de datos en tiempo real

Componentes de un SD

Middleware

- Algunos ejemplos:
  - **Sun RPC**: Sun Remote Procedure Call
  - **CORBA**: Common Object Request Broker Architecture
  - **Java RMI**: Java Remote Method Invocation

  - **SOAP**: Simple Object Access Protocol
  - **REST**: REpresentational State Transfer

  - Otros
    - ISIS
    - DCOM
    - RM-OPD

Modelos

- Componentes
- Modelos físicos
- Modelos arquitectónicos
- Modelos fundamentales

Modelos físicos

- Representación de los elementos hardware en un SD
  - Abstraen los detalles de las tecnologías de red o de computadores subyacentes
- Sirven para diferenciar las distintas generaciones de SD
  - A partir de un modelo base:
    - Definición de SD: componentes informáticos localizados en una red que se comunican por paso de mensajes
  - Considerando los distintos desafíos
    - Escalabilidad
    - Heterogeneidad
    - Transparencia
    - Calidad (tolerancia a fallos, seguridad, concurrencia)

Modelos físicos

1G - Primeros sistemas distribuidos

- Surgen en respuesta a las primeras redes locales (Ethernet)
  - 10 a 100 nodos
  - Conectividad a Internet limitada
  - Pocos servicios (conexión a impresora, servidores de archivos, email, transferencia de archivos)
  - Nodos homogéneos (no hay mucha necesidad de estándares)

Modelos físicos

2G – Sistemas distribuidos en Internet

- Aparecen a raíz del crecimiento de Internet
  - P. ej. en 1996 Google lanza su primer motor de búsqueda
- El SD empieza a concebirse en un entorno de redes interconectadas, es decir, una red de redes (Internet)
  - Aumenta el número de nodos (escala)
  - Aumenta la heterogeneidad
  - Énfasis en la definición de estándares abiertos (CORBA)

Modelos físicos

3G- Sistemas distribuidos contemporáneos

- En 2G, los nodos de un sistema se consideraban
  - Estáticos: permanecen en un lugar determinado
  - Discretos: no están incluidos en otros dispositivos
  - Autónomos: independientes de otros dispositivos
- Este concepto de nodo se rompe
  - Computación móvil: portátiles, smart phones
  - Computación ubicua: los computadores se encuentran en objetos cotidianos o en el ambiente (chip en la lavadora o el coche)
  - Computación en la nube y arquitecturas de cluster: de un nodo que hace una tarea a grupos de nodos que juntos proveen de un servicio

Modelos físicos

Generaciones

Generación	1ª (Inicios)	2ª (Internet)	3ª (Contemporánea)
Época	1970-1985	1985-2005	2005-
Escalabilidad	Baja	Alta	Muy alta
Heterogeneidad	Limitada (configuraciones homogéneas)	Significativa (plataformas, lenguajes, middleware)	Nuevas dimensiones, (arquitecturas, dispositivos)
Extensibilidad	No es una prioridad	Significativa (estándares)	Desafío (los estándares no cubren sistemas complejos)
Calidad	Inicios	Significativa (algunos servicios)	Desafío (los servicios no cubren sistemas complejos)

Modelos

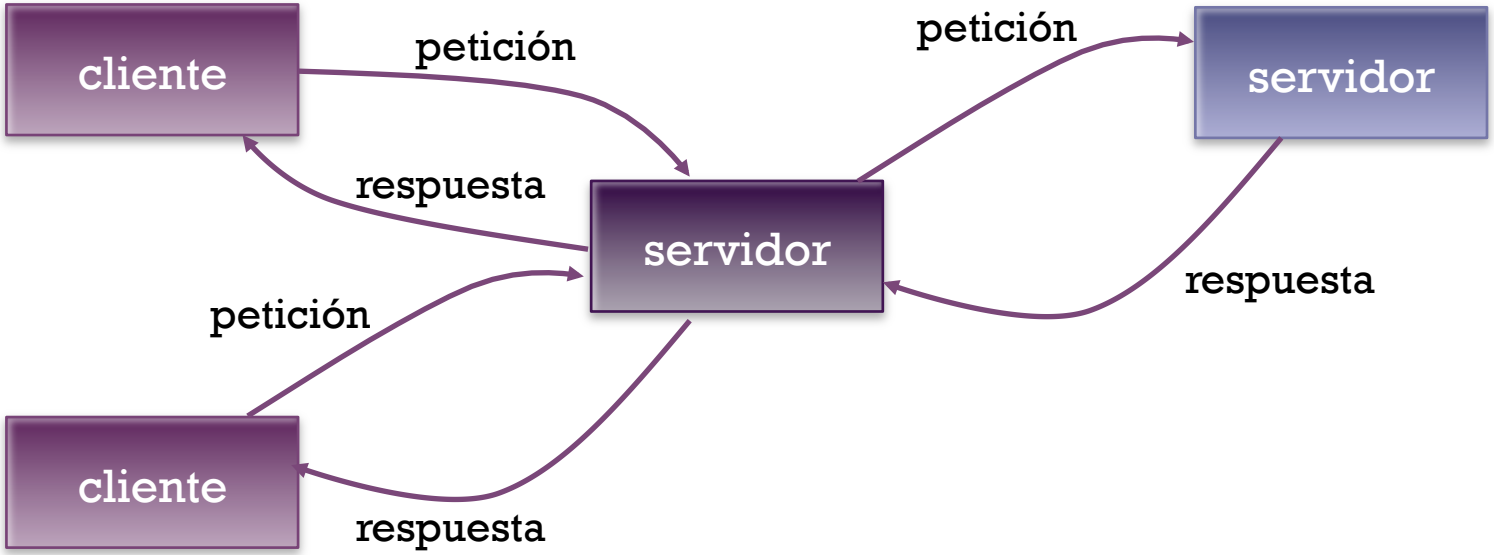
- Componentes
- Modelos arquitectónicos
  - Cliente - servidor
  - Múltiples servidores y proxies
  - Peer to peer
- Modelos fundamentales

Modelos arquitectónicos

Tipos

- Los tipos de modelos arquitectónicos se diferencian en
  - El reparto de responsabilidades entre componentes del sistema
  - La ubicación de los componentes del sistema
- Tipos de modelos
  - Cliente-servidor
    - Servicios proporcionados por múltiples servidores
    - Servidores proxy y cachés
    - Otros derivados
  - Sistemas de igual a igual (peer to peer)

Modelo cliente-servidor



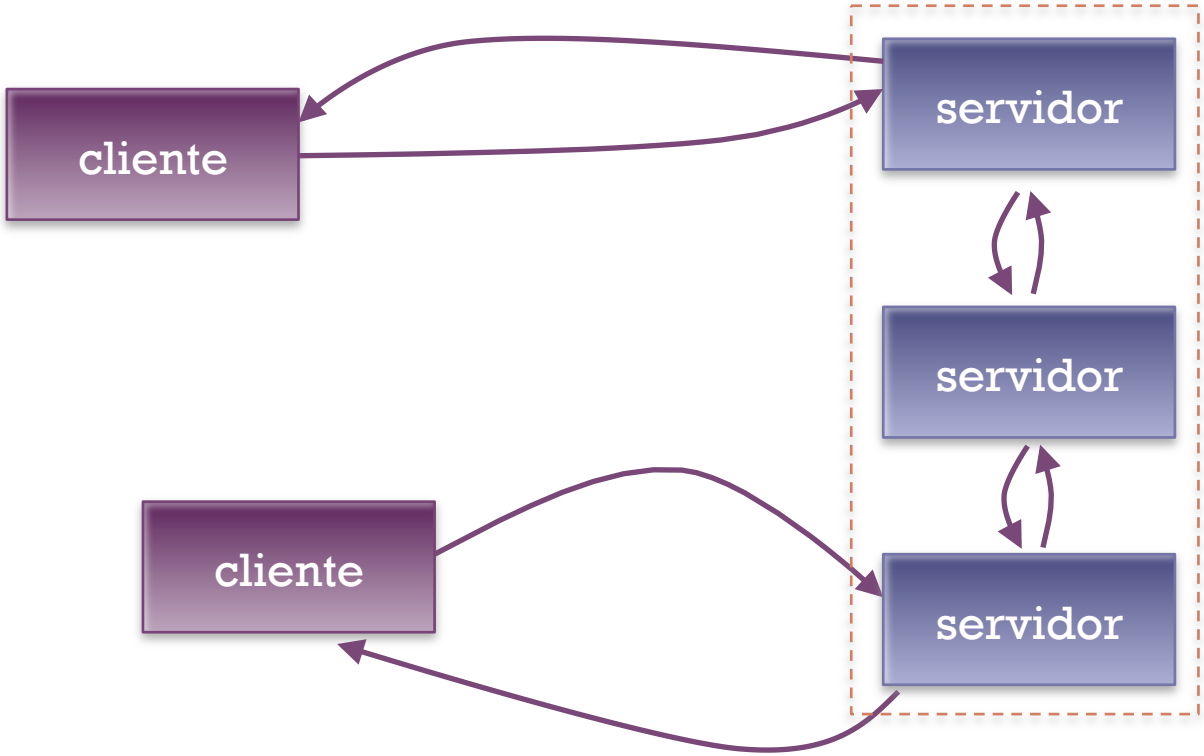
Modelo cliente-servidor

- La arquitectura más común e importante
- Un servidor puede también ser cliente de otros
  - Un servidor web es cliente del servicio DNS que traduce nombres de dominio a direcciones IP
  - Un buscador web es servidor de páginas pero para recopilar la información que presenta es también cliente de otros servidores web, a través de web crawlers
    - Web crawler (araña web): programa que inspecciona páginas web, de forma periódica y automatizada, para obtener información (generalmente realizar copias para su posterior uso en motores de búsqueda)

Modelo cliente-servidor

Servicios proporcionados por múltiples servidores

- Replicación para aumentar prestaciones y disponibilidad
- Muchos servicios web redirigen a varios servidores replicados



Modelo cliente-servidor

Múltiples servidores: Google

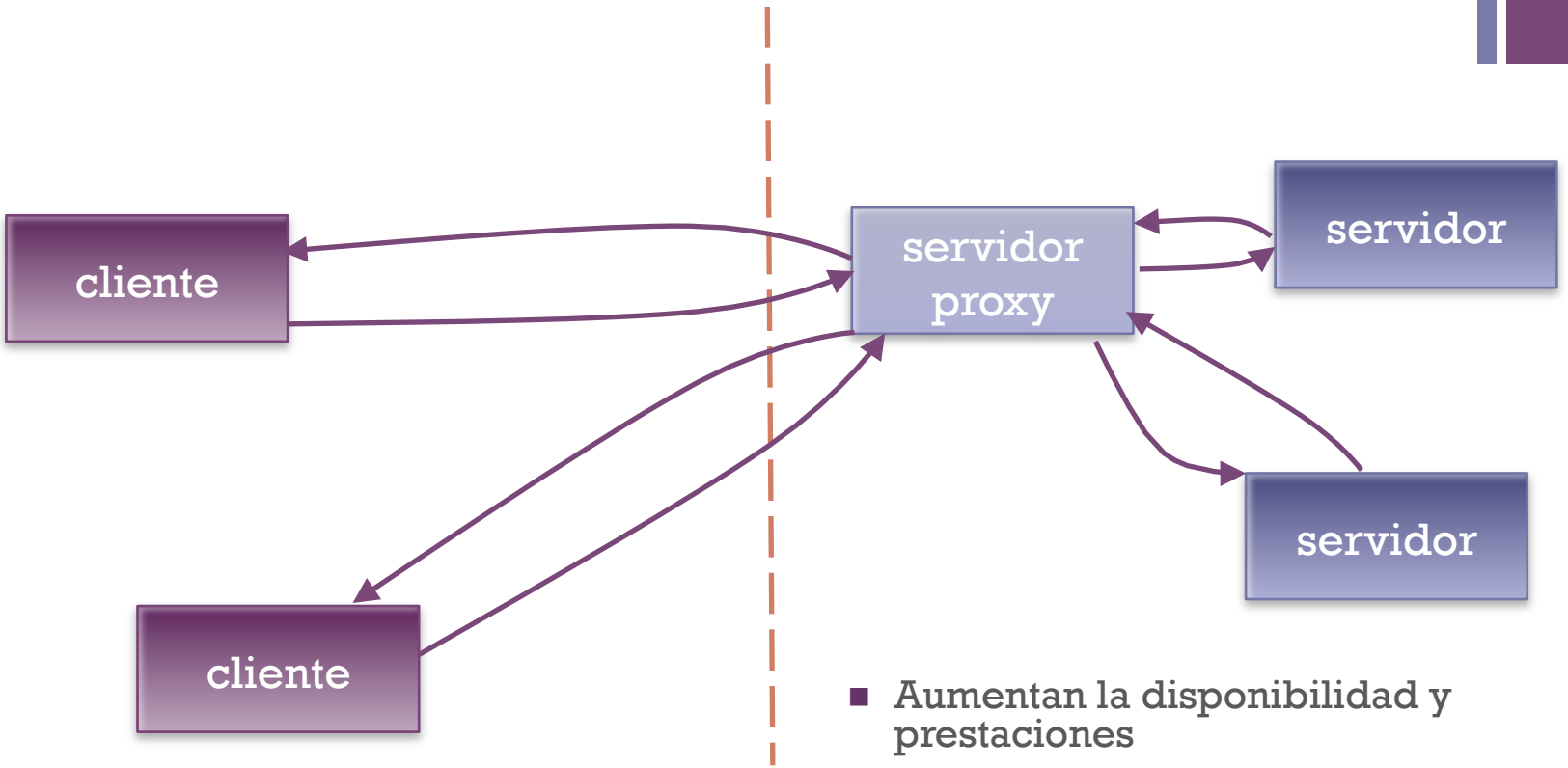
- Google Data Centers (2008)
  - 12 dedicados, 24 compartidos

http://royal.pingdom.com/2008/04/11/map-of-all-google-data-center-locations/

http://www.google.com/about/datacenters/locations/index.html

Modelo cliente-servidor

Servidores proxy



- Aumentan la disponibilidad y prestaciones
  - Usan cachés con los datos más recientemente solicitados por los clientes
- También pueden utilizarse por motivos de seguridad

Modelo cliente-servidor

Derivados

- Código móvil (applets)
  - El código de un programa en el servidor es transferido al cliente y es ejecutado localmente
- Agente móvil
  - Un programa *en ejecución* (código y datos) se traslada/copia de un computador a otro en la red realizando una determinada tarea:
    - Instalación y mantenimiento de software
    - Comparación de precios de productos
    - Cálculo intensivo mediante varios ordenadores
  - Programa gusano: concepto similar al de agente móvil, pero generalmente con connotaciones negativas
    - Virus, spam, phising, etc.

Modelo cliente-servidor

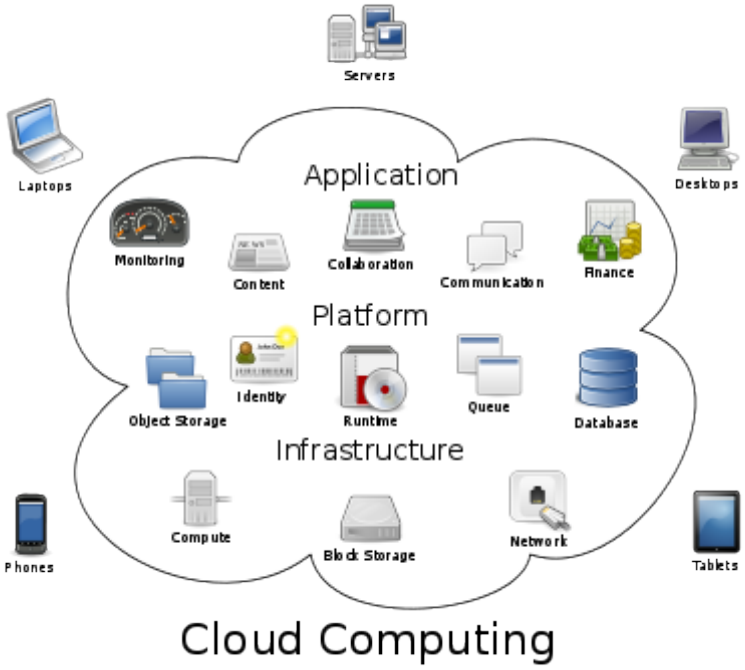
Derivados

- Computadores en red
  - La mayoría de los programas ejecutados por el cliente se descargan de la red, incluido el sistema operativo
  - Las aplicaciones se lanzan localmente pero los archivos están almacenados en un servidor de archivos
- Clientes ligeros
  - Como un computador en red, pero las aplicaciones se lanzan en modo remoto (p. ej. sistema de ventanas X11 de UNIX)
  - Muy útil para cálculos pesados, mediante el uso de potentes servidores (o grupos de servidores –clusters-) de cálculo
    - Inconveniente: actividades altamente interactivas y de cálculo intensivo: interfaces gráficas

Modelo cliente-servidor

Cloud computing

- Evolución del cliente ligero
  - El cliente no pierde su autonomía (sistema operativo, aplicaciones locales) pero puede acceder a servicios y archivos remotos
    - Usualmente a través de un navegador
  - Ejemplos:
    - Amazon Web services: computación en granjas de servidores
      - Amazon Mechanical Turk (p. ej. http://www.tenthousandcents.com/)
    - Web Desktops: almacenamiento y gestión de archivos, sincronización
      - Google Docs, DropBox, iCloud, etc.



Modelo cliente-servidor

Resumen de variantes

Modelo	Ámbito	Ejecución	Almacenamiento
Cliente-servidor	Un servicio	Servidor	Servidor
Código móvil	Un programa	Cliente (previa descarga)	Clientes
Agente móvil	Un programa	Cliente (descarga/viaje inteligente)	Clientes
Computador en red	Todo/casi todo el software	Cliente	Servidor
Cliente ligero	Todo/casi todo el software	Servidor	Servidor
Cloud Computing	Algunos programas	Cliente o Servidor	Servidor

¿Podrías poner ejemplos de sistemas que respondan a cada uno de estos modelos?

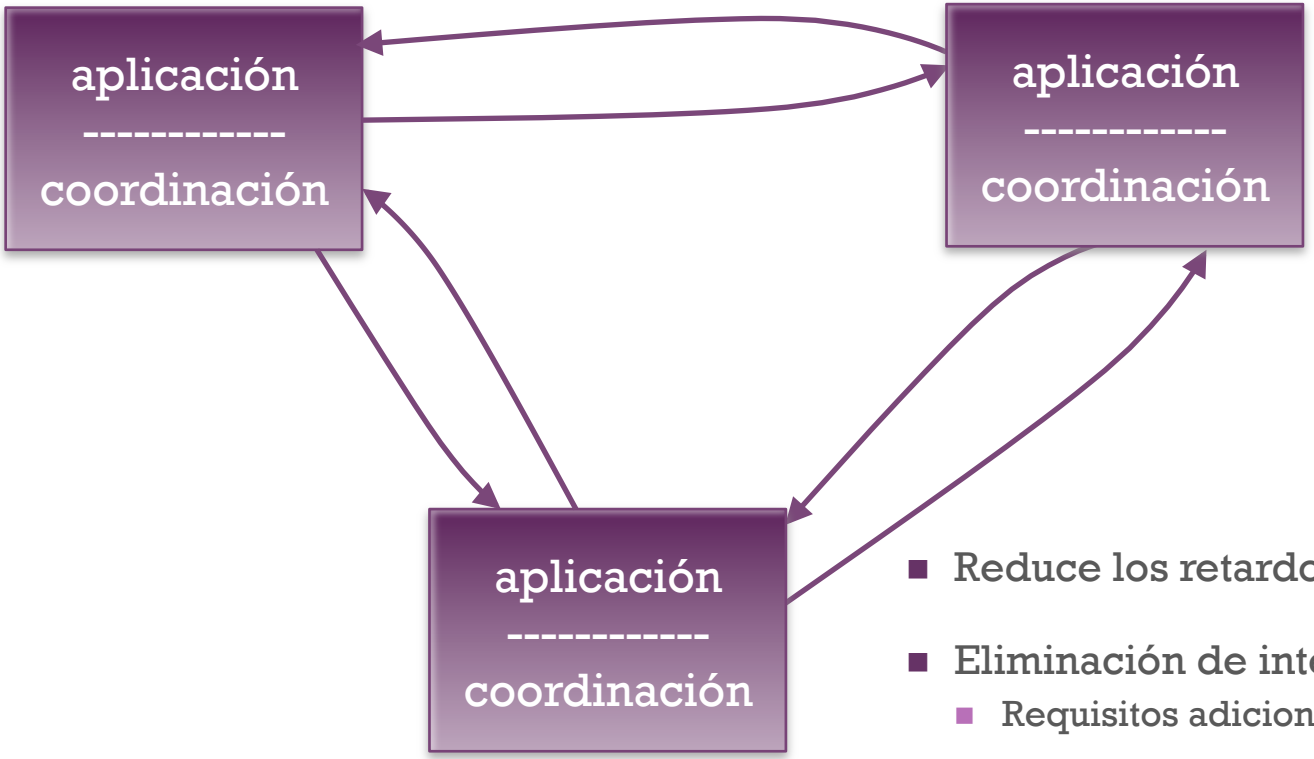
Modelo cliente-servidor

Dispositivos móviles y enlace espontáneo

- Avances tecnológicos
  - Dispositivos más pequeños y transportables
  - Redes inalámbricas de fácil acceso
- El modelo cliente-servidor se adapta a estas dos nuevas características
  - El cliente puede ser un smart phone, una tableta, etc.
  - El enlace se hace a través de un protocolo inalámbrico
- Los clientes pueden ser cada vez más ligeros y confiar en servicios aportados por la nube, con conexión ubicua
  - La conexión puede seguir un modelo peer-to-peer (redes ad hoc)

¿Ves alguna desventaja en este modelo de 'aligeramiento' de los clientes?

Sistemas peer-to-peer



- Reduce los retardos en la comunicación
- Eliminación de intermediarios
  - Requisitos adicionales en los clientes
- Ejemplos:
  - Voz: Skype, VoIP
  - Datos: Napster, BitTorrent
  - Comunicaciones: *ad hoc* networks

Modelos

- Componentes
- Modelos arquitectónicos
- Modelos fundamentales
  - Modelos de interacción
  - Modelos de fallo
  - Modelos de seguridad

Modelos fundamentales

- Todas las arquitecturas comparten algunas propiedades fundamentales:
  - Procesos que se comunican por paso de mensajes a través de una red de computadores
- En particular, trataremos tres aspectos
  - **Interacción**: el modelo debe definir y clasificar la comunicación entre elementos del sistema
  - **Fallos**: el modelo debe definir y clasificar los fallos que pueden darse en el sistema.
  - **Seguridad**: el modelo debe definir y clasificar los tipos de ataque que pueden afectar al sistema.

Modelo de interacción

- Respecto a la interacción, los sistemas distribuidos deben tener en cuenta que
  - Hay limitaciones debidas a la comunicación
  - Es imposible predecir el retraso con el que llega un mensaje
  - Es imposible tener una noción global de tiempo
  - La ejecución es no determinista y difícil de depurar
- Algoritmo distribuido
  - Definición de los pasos que hay que llevar a cabo por cada uno de los procesos del sistema*, incluyendo los mensajes de transmisión entre ellos*

Modelo de interacción

Prestaciones del canal de comunicación

- Latencia
  - Retardo entre el envío de un mensaje y su recepción
- Ancho de banda
  - Información que puede transmitirse en un intervalo de tiempo
- Fluctuación (jitter)
  - Variación del tiempo invertido en repartir una serie de mensajes

Modelo de interacción

Relojes y eventos de tiempo

- Cada computador tiene su propio reloj interno (reloj local)
  - Puede usarse en procesos locales para marcas de tiempo
- Tasa de deriva de reloj (clock drift rate)
  - Evolución de la diferencia entre un reloj local y un reloj de referencia "perfecto"
  - Receptores GPS
  - Network Time Protocol (NTP)
  - Mecanismos de ordenación de eventos
- Dos tipos de modelo de interacción
  - Síncrono y asíncrono

Modelo de interacción

Modelos síncronos

- Conocimiento de características temporales:
  - 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 límite conocido
  - Cada proceso tiene un reloj local cuya tasa de deriva sobre el tiempo de referencia tiene un límite conocido
- A nivel teórico, podemos establecer unos límites para tener una idea aproximada de cómo se comportará el sistema
- A nivel práctico, es imposible garantizar esos límites siempre
  - Aunque a veces se pueden utilizar, por ejemplo como *timeouts*

Modelo de interacción

Modelos asíncronos

- No hay limitaciones en cuanto a
  - Velocidad de procesamiento
  - Retardos en la transmisión de mensajes
  - Tasas de deriva de los relojes
- Los sistemas distribuidos reales suelen ser asíncronos
  - Por ejemplo, Internet
- Una solución válida para un sistema asíncrono lo es también para uno síncrono

Modelo de interacción

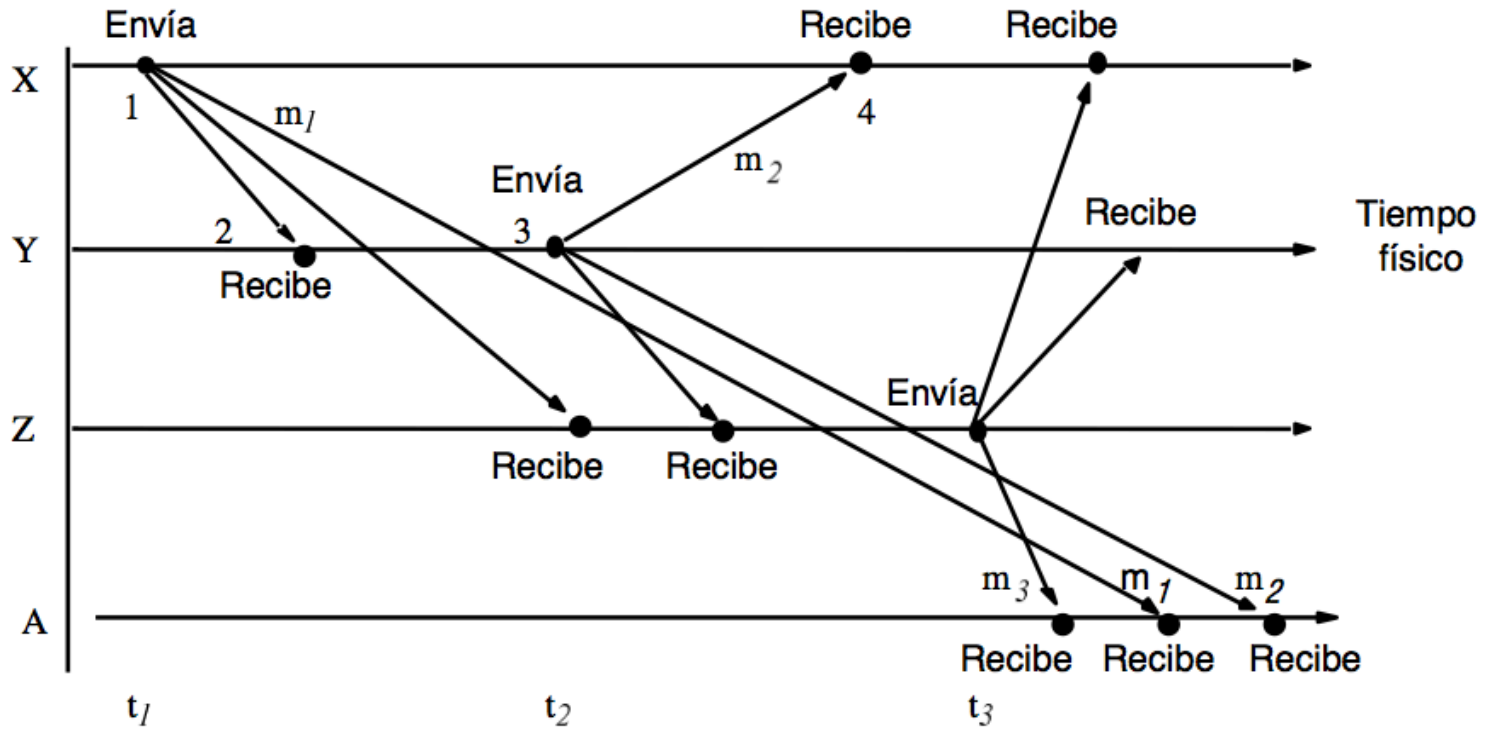
Ordenación de eventos

- Podemos describir un sistema en términos de eventos, solucionando así la falta de precisión de los relojes
- Imaginemos un grupo de usuarios de correo (X, Y, Z, A)
  - X manda un mensaje m_1 con el asunto *Reunión*
  - Y y Z responden con mensajes m_2 y m_3, respectivamente y en ese orden, con el asunto *Re: Reunión*
  - Debido a la independencia en los retardos de cada envío, el usuario A podría ver lo siguiente:

Mensaje	De	Asunto
m_3	Z	Re: Reunión
m_1	X	Reunión
m_2	Y	Re: Reunión

Modelo de interacción

Ordenación de eventos



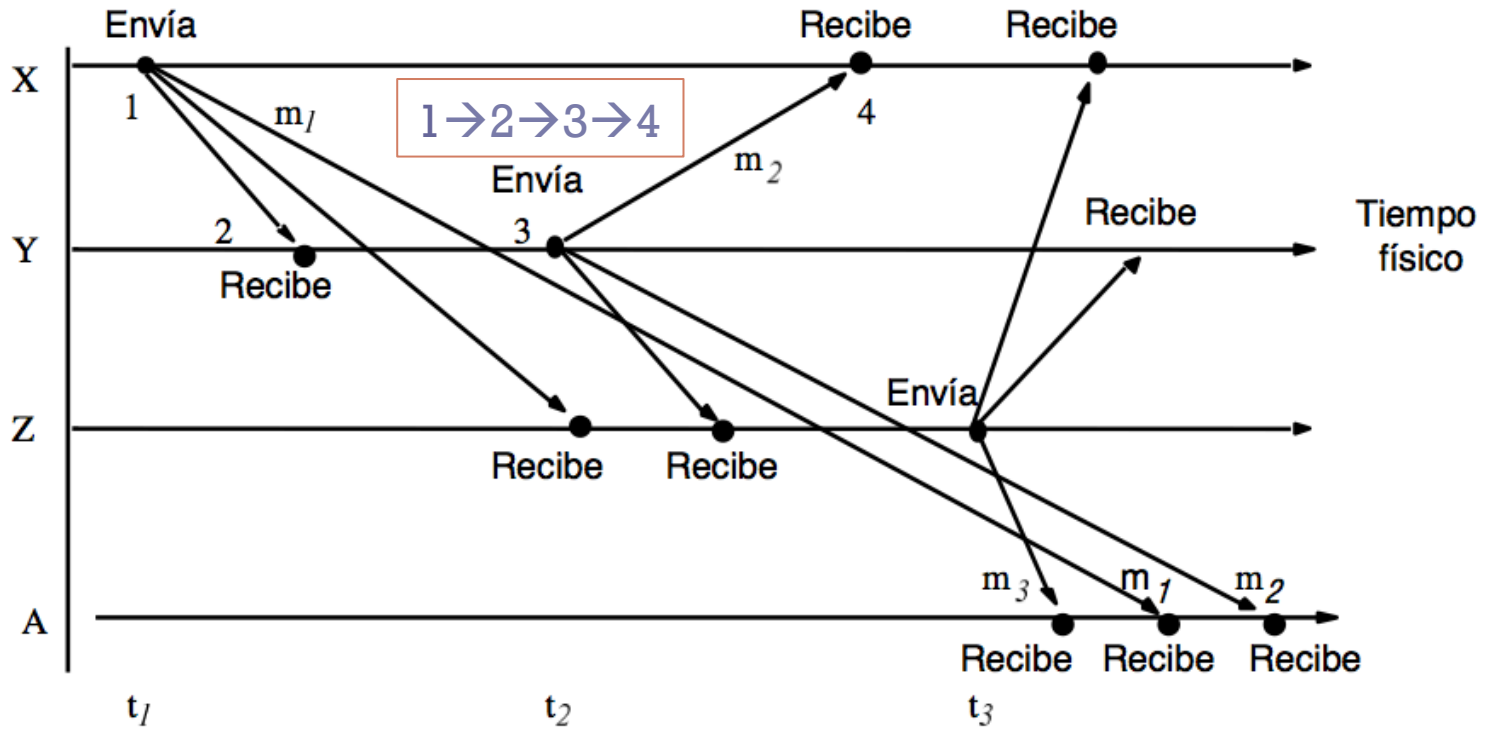
Modelo de interacción

Ordenación de eventos

- Si los relojes de X, Y y Z estuvieran sincronizados, podríamos incluir el tiempo local t_1, t_2, t_3 en los mensajes m_1, m_2, m_3
  - Estaríamos seguros de que $t_1 < t_2 < t_3$
  - Podríamos ordenar los mensajes en concordancia
- Pero los relojes no suelen estar sincronizados
  - Lamport [1978] propuso un modelo de tiempo lógico
  - Infiere el orden de los mensajes sin recurrir al tiempo físico
  - Se basa en las siguientes afirmaciones
    - Un mensaje siempre se recibe después de enviarlo
      - *X* manda m_1 antes de que *Y* reciba m_1
    - La réplica no se envía hasta que no se ha recibido el original
      - *Y* recibe m_1 antes de que envíe m_2

Modelo de interacción

Ordenación de eventos



Modelo de fallo

- Estudio de las causas posibles de fallo
  - Para poder comprender sus consecuencias
- Tipo de fallo según la entidad
  - Fallos de proceso
  - Fallos de comunicación
- Tipo de fallo según el problema
  - Fallos por omisión
    - No se consigue realizar una acción que se debería poder hacer
  - Fallos arbitrarios (bizantinos)
    - Errores de cualquier tipo, fuera del esquema de mensajes
  - Fallos de temporización
    - Superación de tiempos límite en un sistema síncrono

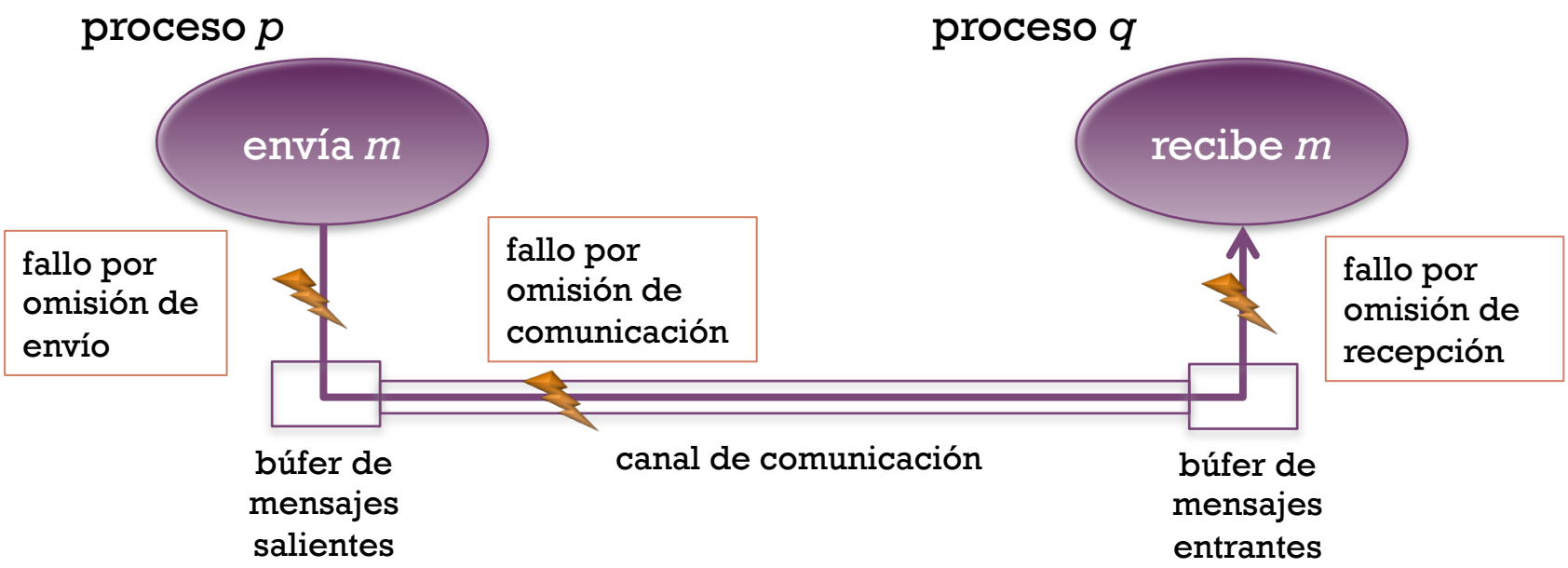
Modelo de fallo

Fallo por omisión en procesos

- Fallo del procesamiento (*crash*)
- Fallo-parada (fail-stop)
  - Fallo de procesamiento que puede ser detectado con certeza por el resto de procesos
- Detección del fallo por timeouts (síncrono)
  - Si el proceso no responde, consideramos que ha habido un fallo
  - En sistemas asíncronos, nunca podemos estar seguros

Modelo de fallo

Fallo por omisión en comunicaciones



Modelo de fallo

Fallos arbitrarios o bizantinos

- En proceso:
  - Se omiten pasos necesarios o deseables del procesamiento
  - Se realizan pasos innecesarios o indeseables en el procesamiento
  - Se omite arbitrariamente la respuesta a mensajes
- En canales de comunicación
  - Corrupción de mensajes
  - Reparto de mensajes inexistentes
  - Duplicación del reparto de mensajes auténticos
- Origen: problema de los generales bizantinos
  - Lamport, Shostak and Pease (1982). *"The Bizantine Generals' Problem"*. ACM Transaction on Programming Languages and Systems 4 (3): 382-401
    - http://vis.usal.es/rodrigo/documentos/papers/Lamport82.pdf

Modelo de fallo

Fallos bizantinos

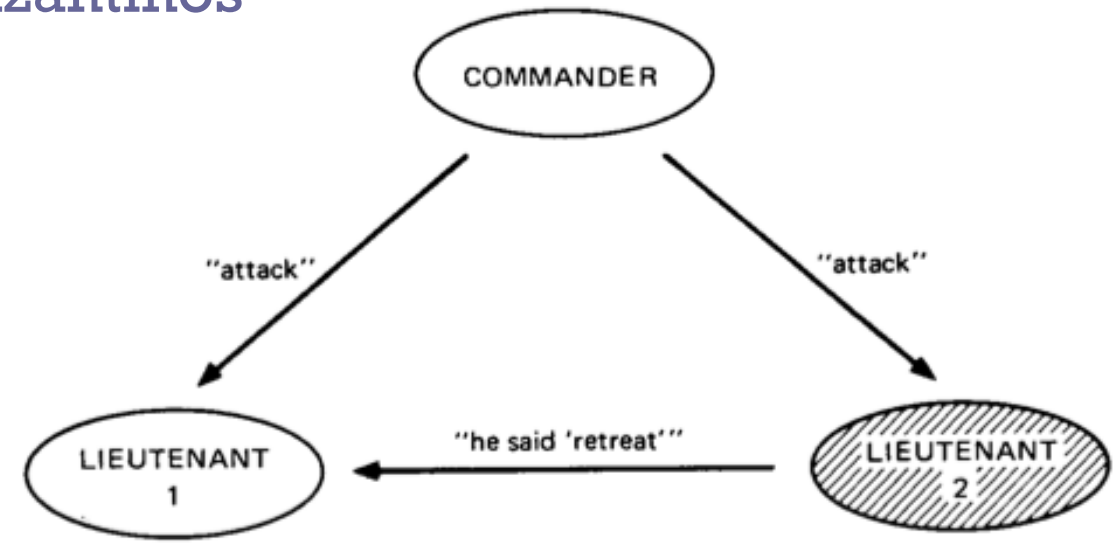


Fig. 1. Lieutenant 2 a traitor.

Reliable computer systems must handle malfunctioning components that give conflicting information to different parts of the system. This situation can be expressed abstractly in terms of a group of generals of the Byzantine army camped with their troops around an enemy city. Communicating only by messenger, the generals must agree upon a common battle plan. However, one or more of them may be traitors who will try to confuse the others. The problem is to find an algorithm to ensure that the loyal generals will reach agreement. It is shown that, using only oral messages, this problem is solvable if and only if more than two-thirds of the generals are loyal; so a single traitor can confound two loyal generals. With unforgeable written messages, the problem is solvable for any number of generals and possible traitors. Applications of the solutions to reliable computer systems are then discussed.

Modelo de fallo

Fallo por omisión y arbitrarios (resumen)

Tipo de fallo	Afecta a	Descripción
Fallo-parada (*fail-stop*)	Proceso	El proceso para y permanece parado Otros procesos pueden detectar la parada
Ruptura (*crash*)	Proceso	El proceso para y permanece parado Otros procesos pueden *no* detectar la parada
Omisión	Canal	Un mensaje insertado en un búfer de mensajes salientes nunca llega al búfer de mensajes entrantes del destino
Omisión de envío	Proceso	Un proceso completa el envío pero no se coloca el mensaje en el búfer de mensajes salientes
Omisión de recepción	Proceso	El mensaje se coloca en el búfer de recepción pero el proceso no lo recibe
Arbitrario (bizantino)	Proceso o canal	El proceso/canal presenta un comportamiento arbitrario: omisiones, paradas, envíos o pasos incorrectos sin patrón claro

Modelo de fallo

Fallos de temporización

- Sistemas síncronos

Tipo de fallo	Afecta a	Descripción
Reloj	Proceso	El reloj local del proceso excede el límite de su tasa de deriva respecto al tiempo de referencia
Prestaciones	Proceso	El proceso excede el límite sobre el intervalo entre dos pasos
Prestaciones	Canal	La transmisión de un mensaje toma más tiempo que el límite permitido

- Sistemas asíncronos
  - No existen fallos de temporización, ya que no se ha dado ninguna garantía al respecto

Modelo de fallo

Enmascaramiento de fallos

- Construcción de servicios fiables a partir de componentes que presenten fallos
  - Por ocultación del error
  - Por conversión a fallos más aceptables
- Por ejemplo:
  - Checksum → de fallo arbitrario a fallo por omisión
    - +retransmisión → de fallo por omisión a ocultación

Modelo de fallo

Comunicación fiable entre dos procesos

- Debe cumplir con dos criterios:
  - Validez
    - Cualquier mensaje en el búfer de mensajes salientes llegará, eventualmente, al búfer de mensajes entrantes
      - Es decir, no hay fallos por omisión en el canal
  - Integridad
    - El mensaje recibido es idéntico al enviado, y no se repiten mensajes
      - Protocolo que adjunta números de secuencia a los mensajes
      - Canales de comunicación seguros
    - No hay fallos bizantinos en el canal

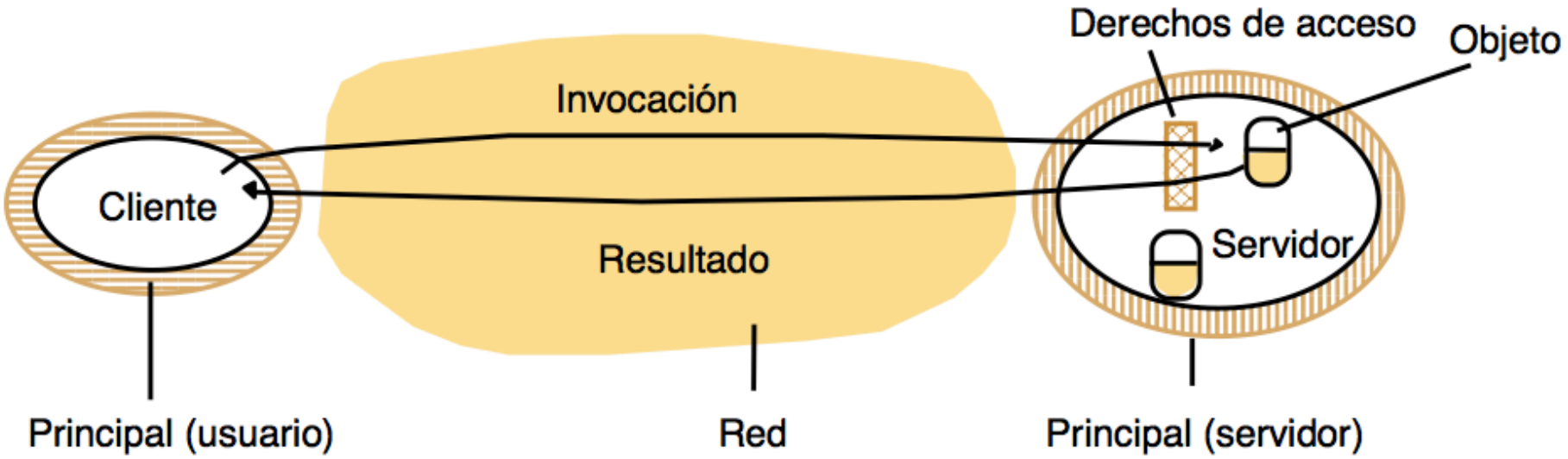
Modelo de seguridad

- La seguridad en un sistema distribuido se basa en la seguridad de los procesos y canales utilizados
  - Entendida como seguridad de objetos
    - Almacenados e invocados por los procesos
    - Transmitidos a través de los canales
- Se logra mediante un sistema de derechos de acceso y distintos tipos de autoridad

Modelo de seguridad

Principal y derechos de acceso

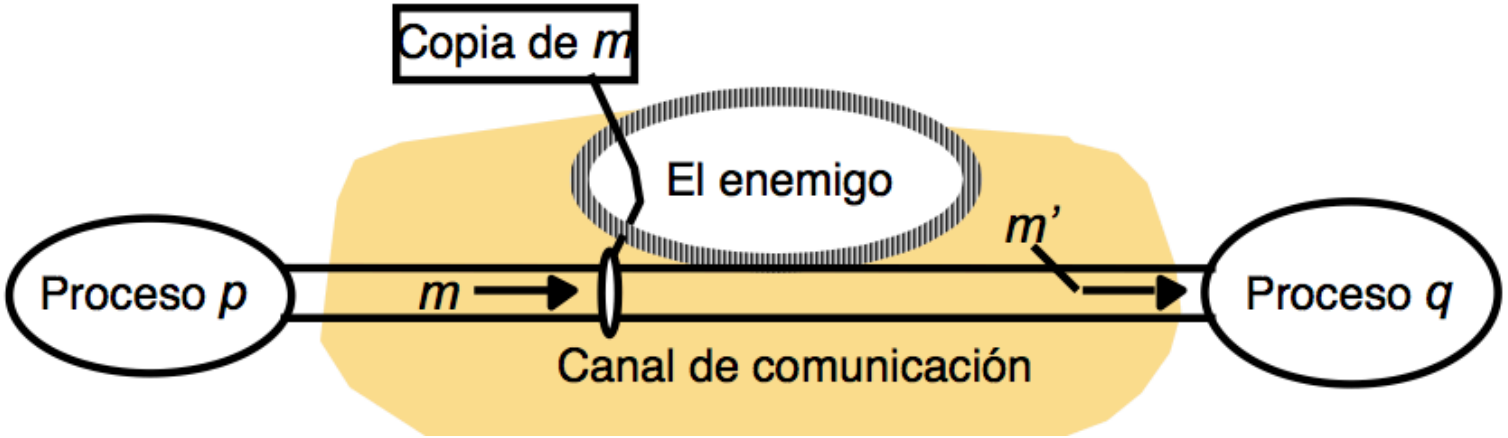
- **Principal**: autoridad con la que se ordena cada invocación de objetos o sus resultados
  - Se contrasta con los derechos de acceso de dicho objeto



Modelo de seguridad

Modelo de enemigo

- **Entidad**
  - Cualquier máquina conectada (de forma autorizada o no) a la red
- **Enemigo**: entidad capaz de
  - Enviar cualquier mensaje a cualquier proceso
  - Leer o copiar cualquier mensaje compartido entre dos procesos
  - Leer mensajes o emitir mensajes falsos de petición de servicios



Modelo de seguridad

Amenazas

- Amenazas a servidores
  - Ciertos servicios no comprueban la identidad del cliente
    - Si la comprueban, no suele ser difícil suplantarla (*spoofing*)
  - En vez de una petición de servicio auténtica se busca, p. ej., obtener información no autorizada o bloquear el servicio (DoS)
- Amenazas a clientes
  - Reciben un resultado falso de la invocación al servicio
    - Generalmente, acompañado de suplantación de identidad
- Amenazas a canales de comunicación
  - Inyección, copia o alteración de mensajes que viajan por el canal
  - Por ejemplo: obtener un mensaje de transferencia de dinero, cambiar la cuenta y reenviarlo después

Modelo de seguridad

Técnicas de seguridad

- **Autenticación**: comprobación de la identidad del proceso
- **Criptografía**: uso de claves públicas y privadas
- **Canales seguros**: canal de comunicación sobre el que dos procesos han establecido una capa de seguridad basada en criptografía + autenticación:
  - Se garantiza la identidad fiable de servidores y clientes
  - Se garantiza la integridad y privacidad de los mensajes enviados
  - Los mensajes incluyen una marca de tiempo para prevenir su repetición o reordenación maliciosa

Resumen

- Un **modelo** da una idea aproximada de lo que nos vamos a encontrar en un sistema distribuido
- Los modelos **físicos** identifican qué tipo de procesos y comunicación tenemos, principalmente en cuanto a **heterogeneidad** y **escala**. Sirven para clasificar los SD en generaciones
- Los modelos **arquitectónicos** determinan la relación entre procesos, desde **cliente/servidor** hasta **pares iguales**
- Los modelos **fundamentales** atienden a cómo es el sistema desde el punto de vista de la sincronización, la seguridad y los fallos
- Los SDs pueden ser **síncronos** o **asíncronos**, dependiendo de si tenemos información sobre los límites temporales de la comunicación o no
- Los SDs pueden tener tres tipos de **fallos**: por **omisión** (el proceso o canal no funciona, anunciándolo o no), **arbitrarios** (el proceso/canal funciona mal) o **temporales** (se exceden los límites de tiempo en sistemas síncronos)
- Los SDs tienen que atender a distintos problemas de **seguridad**, resueltos generalmente con métodos de **autenticación** y **criptografía**, aunque no entraremos en detalle

Referencias

- G. Colouris, J. Dollimore, T. Kindberg and G. Blair. *Distributed Systems: Concepts and Design (5th Ed)*. Addison-Wesley, 2011
  - Capítulo 2: System Models
- Lamport, Shostak and Pease (1982). *"The Bizantine Generals' Problem"*. ACM Transaction on Programming Languages and Systems 4 (3): 382-401
  - http://vis.usal.es/rodrigo/documentos/papers/Lamport82.pdf

Todos los modelos están equivocados, pero algunos son útiles

George E. P. Box (estadístico, famoso por sus trabajos en modelado, predicción e inferencia bayesiana)