Modelos de sistema - 2

Sistemas Distribuidos – ITInformática

César Llamas, febrero 2003

Algunos esquemas de esta presentación están tomados de:

Instructor's Guide for Coulouris, Dollimore and Kindberg
Distributed Systems: Concepts and Design Edn. 3

Índice

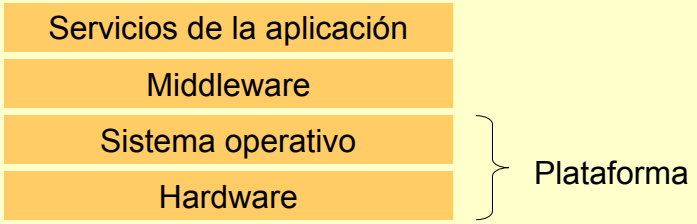
Introducción

- El modelo arquitectónico describe:
  - Interacciones entre componentes
  - Enlace con la plataforma de red: "simplifica y abstrae las funciones y roles de los componentes individuales"
- El enfoque CDK es un poco confuso
  - Estructura de niveles de software
  - Variantes del modelo cliente-servidor
  - Código móvil

Íntroducción

- Modelos de requisitos no funcionales
  - Modelo de interacción
  - Modelo de fallo
  - Modelo de seguridad

Modelos arquitectónicos *capas de soft*



- Componentes importantes:
  - Plataforma
  - Middleware

Modelos arquitectónicos *capas de soft*

- Plataforma
  - Contiene los servicios propios de cada computadora concreta
  - Depende del hardware y del S.O.

Modelos arquitectónicos *Middleware*

- Permite enmascarar la heterogeneidad
- Puede dar un modelo y una interfaz de programación utilizable
- Puede soportar abstracciones como:
  - Llamadas a procedimientos remotos (RPC)
  - Comunicación en grupo
  - Eventos, replicación, servicios multimedia, etc...

Modelos arquitectónicos *middleware*

- ¿Qué forma tiene el middleware? :
  - Bibliotecas adicionales:
    - Procedimientos remotos (RPC)
    - Objetos remotos (RMI, CORBA)
  - Herramientas de programación
    - Lenguajes de definición de interfaces (IDL) + Compiladores para ellos
  - Servicios básicos de ayuda
    - servicios de nombres, para buscar objetos
    - de notificación de eventos,
    - De control de transacciones, etc.

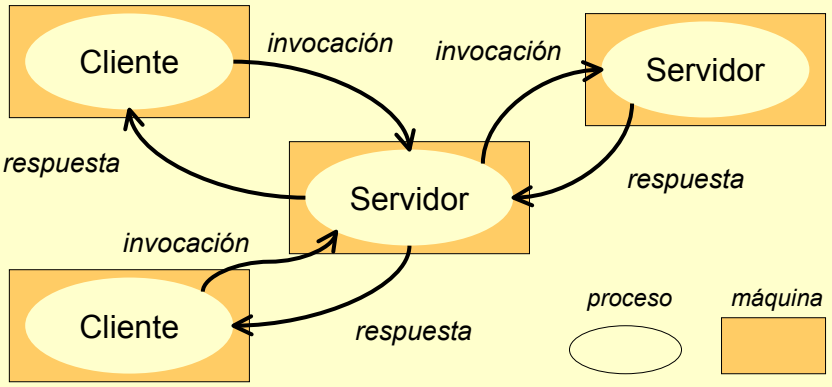
Modelos arquitectónicos *middleware*

- ¿Qué limitaciones impone?
  - Se incrementa la complejidad arquitectónica:
    - Hay más niveles
  - Hay que aprender más herramientas
  - Se pierde el control de bajo nivel sobre los modos de fallo
  - Se depende de terceras partes,
  - ...

Arquitecturas de sistema *Índice*

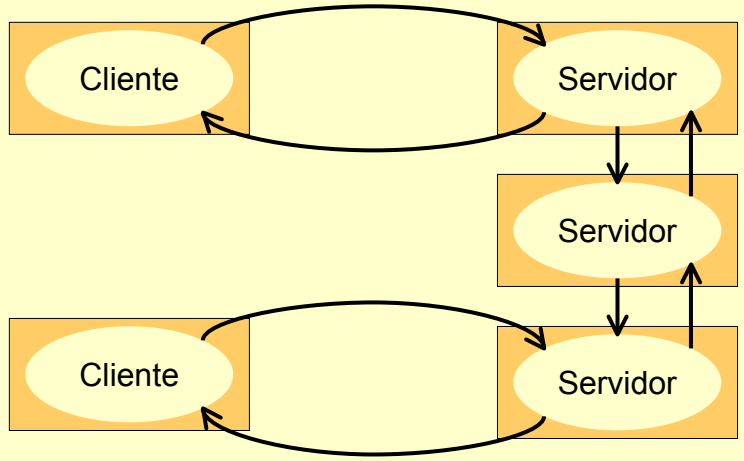
- Modelo cliente-servidor
- Múltiples servidores
- Procesos de igual a igual

Arquitecturas cliente servidor *cliente-servidor*



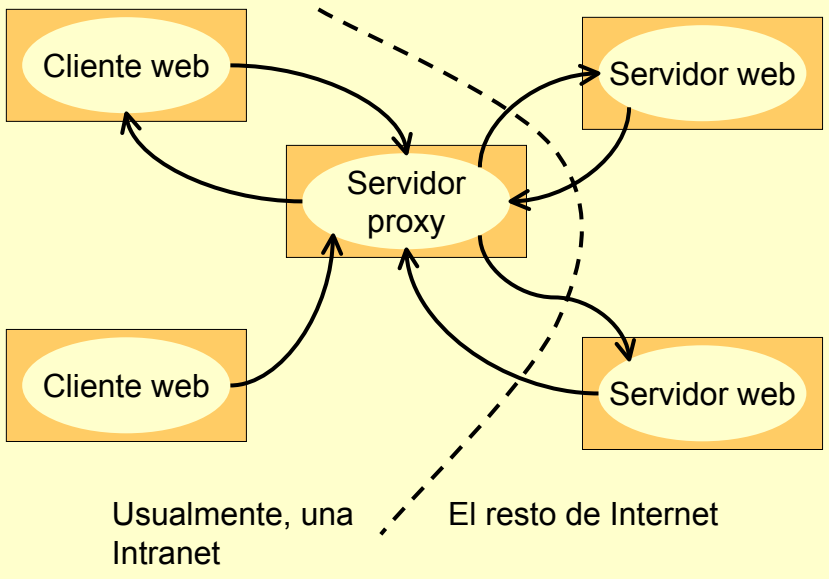
- Muy habitual (DNS, Web, ftp, telnet, ...)
- Un servidor puede ser cliente de otro servicio (servidor web -> crawler)

Arquitecturas cliente servidor *servidores múltiples*

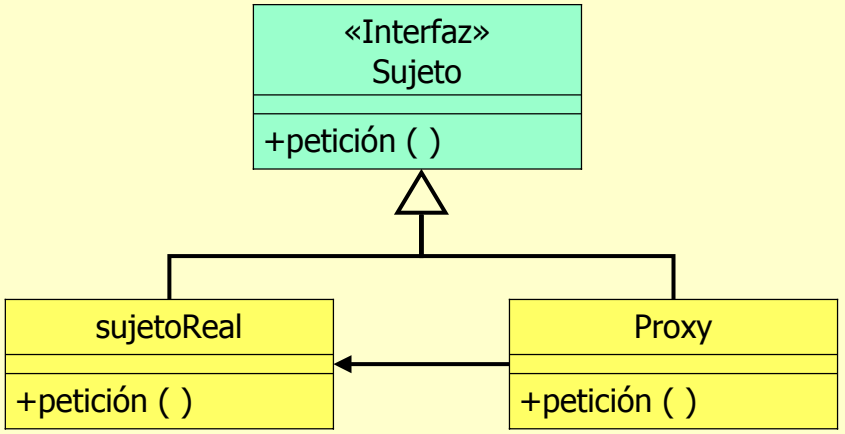


- Muy usada en DNS, Web y NIS

- Caché: almacena los recursos más probablemente usados.
- Un caché puede responder a un esquema de proxy
  - Los servidores proxy para el web aumentan la *disponibilidad*.



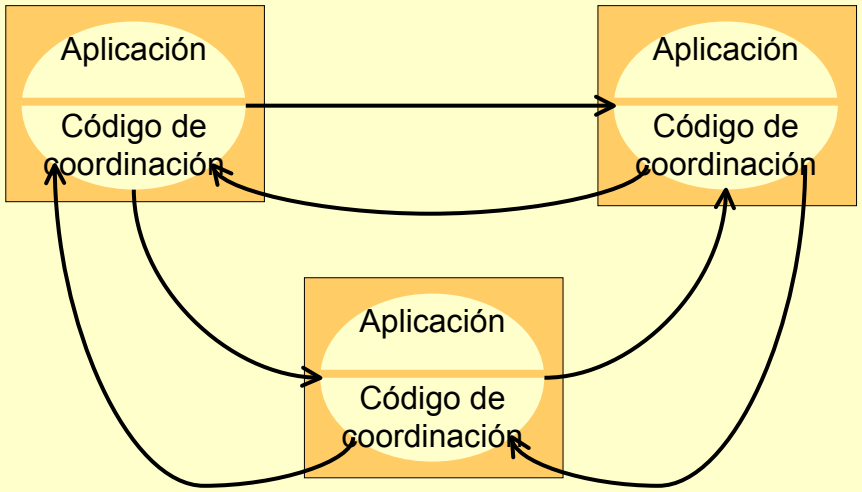
Arquitecturas cliente servidor *proxy y caché*



Arquitecturas de sistema *procesos pares*

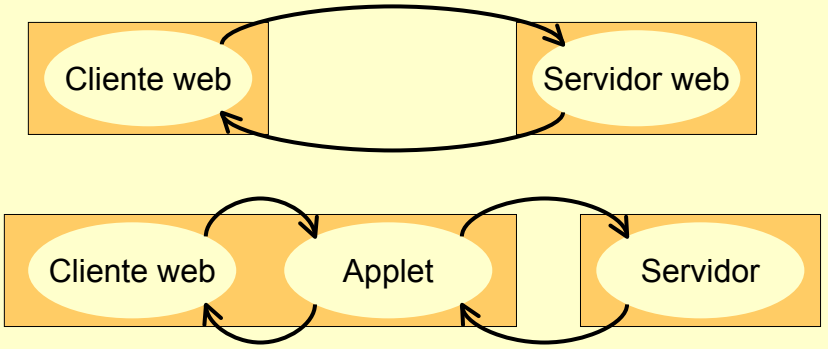
- Cuando los roles entre procesos son de igual a igual (peer-to-peer)
  - Desempeñan tareas semejantes
  - Útil al descomponer aplicaciones en tareas coordinadas
- Ejemplo:
  - Cooperación y coordinación
  - Algoritmos descentralizados
  - (coordinación de agendas, trabajo colaborativo, ...)

Arquitecturas de sistema *procesos pares*



Variaciones en el modelo cliente-servidor *Ejemplos*

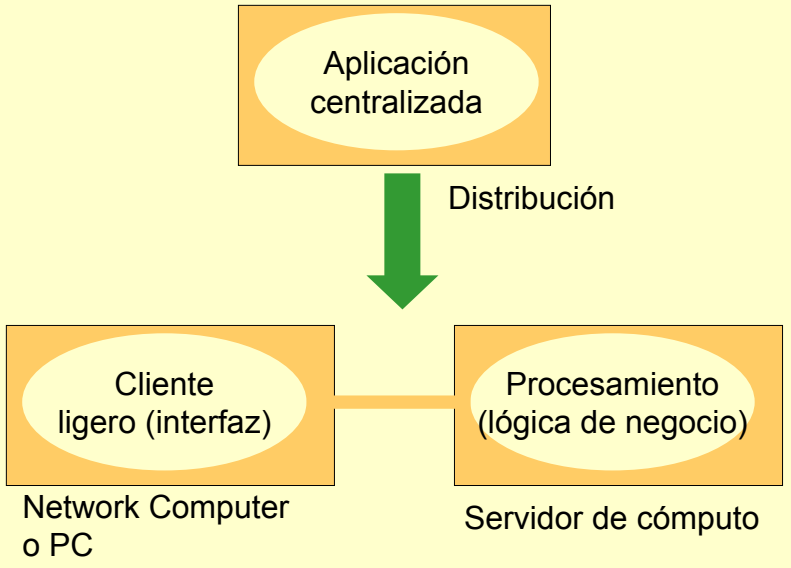
- Ejemplo: applets para clientes Web
  - Para implementar interfaces complejos
  - Para simular un modelo "push" sobre Web



Variaciones en el modelo cliente-servidor *Ejemplos*

- Algunas posibilidades:
  - Según la ubicación del código del proceso cliente:
    - ✓Código estático
    - Código con movilidad (recolocación del proceso)
  - Según la proporción de tareas que recae sobre el cliente y el servidor
    - ✓Clientes al estilo habitual
    - Clientes ligeros de aplicaciones complejas
    - Computadoras de red

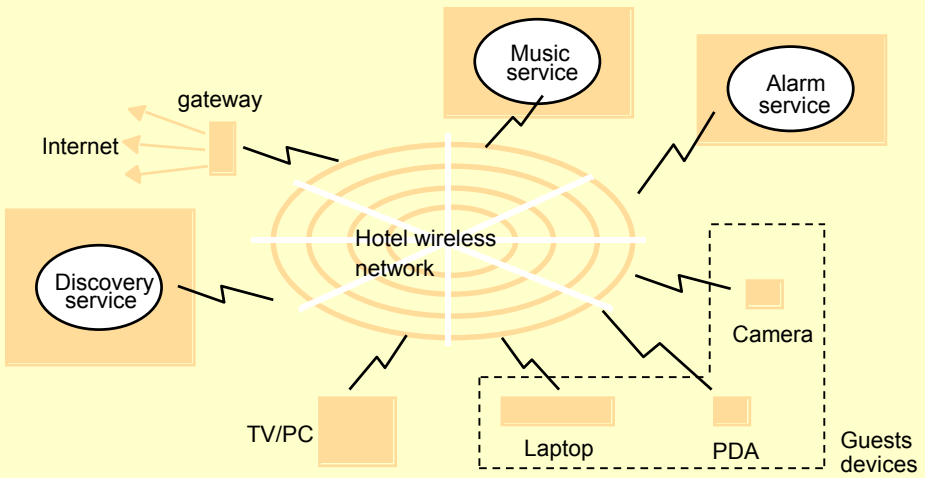
Variaciones en el modelo cliente-servidor *cliente ligero*



Variaciones en el modelo cliente-servidor *red espontánea*

- Más posibilidades
  - Según la conexión:
    - ✓Enlace estático
    - Enlace dinámico (Dispositivos móviles y enlace espontáneo)
- Conexión espontánea
  - Metropolitana (GPRS, UTMS)
  - Media (x0 o x00 m) (Wavelan, Wireless – 802.11b)
  - Corta (x o x0 m) (BlueTooth, infrarojos, HomeRF)

Variaciones en el modelo cliente-servidor *red espontánea*



Variaciones en el modelo cliente-servidor *red espontánea*

- Ventajas del enlace espontáneo
  - Facilidad de conexión a la red local
  - Facilidad de integración con los servicios locales
- Cuestiones a resolver
  - Problemas de conectividad (zonas sombra, cambio de célula, ...)
  - Seguridad
  - Servicios de detección y/o descubrimiento
    - Admisión
    - Búsqueda

Variaciones en el modelo cliente-servidor

- Más posibilidades
  - Según el diseño de la interfaz entre procesos:
    - ✓Interfaz estático
    - Interfaz orientado a llamada a procedimiento remoto
- Interfaz con llamada a procedimiento remoto:
  - ✓Estático (el habitual en RPC)
  - Orientado a objetos (Dinámico)

Interfaces y objetos

- Interfaz de un proceso
  "Conjunto de peticiones a que responde"
- Estilos
  - Mediante interfaces de módulos "módulos"
    Soportado en Modula2, RPC, ...
  - Mediante la interfaz de los objetos en OOP
    Soportado de modo natural para SD en:
    - Java RMI
    - CORBA

Interfaces y objetos para S.D.

"Los procesos contienen objetos cuyos métodos podemos invocar de modo remoto"

- Es deseable que las referencias a los objetos remotos se usen de modo "transparente" (como las locales)
- No podemos hablar exclusivamente de procesos cliente y procesos servidor, sino de "objetos cliente" y "objetos servidor"

Requisitos de diseño para arquitecturas distribuidas

Requisitos no funcionales (Análisis del sistema)

medibles

- Requisitos de rendimiento o prestaciones
- Requisitos de calidad de servicio

- (cuestión relacionada: caché y replicación)

no medibles (fiabilidad)

- Tolerancia
- Seguridad

Requisitos de diseño ... *sobre prestaciones*

- Capacidad de respuesta (responsiveness)
  - Está limitada por:
    - La velocidad de las redes
    - La velocidad de los nodos, y en este caso
    - La cantidad de capas (middleware).
- Productividad (throughput)
  - Tareas completadas /tiempo

Ej: 1000 reservas de billete por segundo.

- Balance de cargas (disminuyen la competición)
  - Entre computadoras alternativas
  - Entre los procesos implicados

Ej: la carga media de cada servidor debe estar balanceada y no superar el 50 %

Requisitos de diseño ... *sobre calidad de servicio*

Calidad de Servicio (en general):

Capacidad de respuesta, Productividad, Balance de carga

\+ fiabilidad, seguridad, adaptabilidad

- Indicador de la aceptabilidad de un servicio
  - Está relacionada con fiabilidad, seguridad y adaptabilidad.
  - Actualmente se recomienda usar QoS para hacer referencia a la capacidad de **garantizar** tasas de transferencia y tiempos límite (en telecomunicaciones)
  - ... Importante para datos críticos en el tiempo (caudales de sonido, de vídeo, etc...)

Requisitos de diseño ... *sobre prestaciones*

- Las prestaciones son un obstáculo principal para el despliegue de soluciones distribuidas
  - Al usar réplicas y
  - caché, se disminuye comunicación y tiempos.
- Ejemplo: protocolo de caché de web (para navegadores y clientes web)
  - Recurso + tiempo de expiración + tiempo local del servidor web

Requisitos de diseño ... *sobre fiabilidad*

- Corrección
- Tolerancia a fallos
  - Se consigue introduciendo redundancia en recursos software y en componentes físicos;
    - Computadoras, procesos, enlaces de red, ...
  - también con protocolos con reconocimiento, y
  - con mecanismos de recuperación.
- Seguridad
  - Protección activa y pasiva contra ataques, y
  - Mecanismos de seguridad criptográfica.

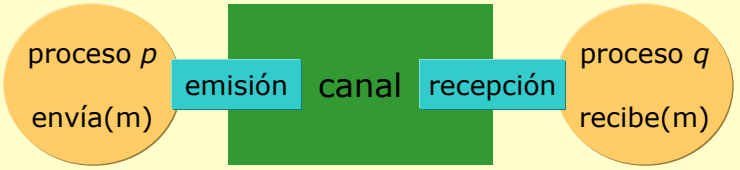
Modelos fundamentales

Modelo: contiene los elementos esenciales para comprender y razonar sobre el sistema

- Manifiesta las premisas del sistema
- Generaliza sobre lo que es posible o no.

- Principales modelos:
  - De interacción
  - De fallo
  - De seguridad

Modelo de Interacción



- La forma en que se produce el "paso de mensajes" entre los procesos restringe los modos de interacción
  - Retrasos, precisión, y tiempo

Modelo de interacción

- Problemas presentados en las prestaciones del canal
  - *Latencia*: Retardo entre la emisión y recepción: acceso + transmisión + propagac. niveles soft.
  - Velocidad limitada (Datos/t)
  - Fluctuación o cortes (jitter)
- Problemas presentados en la noción de tiempo (timestamps de cada evento):
  - Derivas (diferencia horaria) diferentes
  - Tasas de derivas (ritmo de deriva) diferentes

Modelo de interacción *tiempo y sincronicidad*

- En virtud del modelo de comunicación aparecen dos familias de sistemas:
  - Sistemas distribuidos síncronos
  - Sistemas distribuidos asíncronos

	Sistema distribuido síncrono	*Sistema distribuido asíncrono*
Respuesta de los procesos	Timeouts superior e inferior en cada etapa de proceso	Velocidad de proceso muy variable
Tiempo de comunicación	Timeouts de recepción conocidos	Retardos imprevisibles en la transmisión
Error en los relojes	Tasas de deriva locales conocidas	Tasas de de deriva imprevisibles

Modelo de interacción *tiempo y sincronicidad*

- Consecuencia básica de la asincronicidad

  "en un sistema asíncrono los eventos pueden observarse desordenados con respecto a su generación"

Ejemplo:

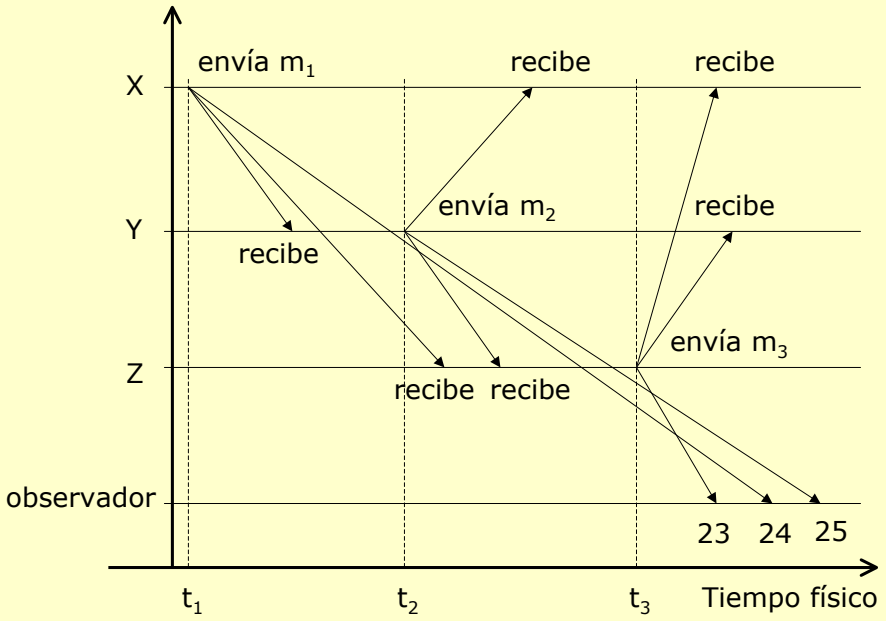
1. El usuario X envía un mensaje con el tema *Reunión*.
2. Los usuarios Y y Z responden con un mensaje con el tema *Re: Reunión*.

Modelo de interacción *tiempo y sincronicidad*

	Bandeja de entrada	
Elemento	Emisor	Asunto
23	Z	*Re: Reunión*
24	X	*Reunión*
25	Y	*Re: Reunión*

- Si rompe la relación de causalidad
- Si los relojes de X, Y y Z pudieran sincronizarse, podríamos observar la secuencia ordenada.

Modelo de interacción *tiempo y sincronicidad*



Modelo de interacción *tiempo y sincronicidad*

- Posibilidad en sistemas asíncronos:

  Relojes lógicos: Proporcionan enteros consecutivos que permiten ordenar eventos marcados por un *timestamp*, con la relación de orden *ocurrió_antes*(S_1,S_2).

  - X envía m_1 antes de que Y reciba m_1

    $T(envía(X,m_1)) < T(recibe(Y,m_1))$

    ocurrió_antes(envía(X,m_1), recibe(Y,m_1))

  Y recibe m_1 antes de enviar m_2

  $T(recibe(Y,m_1)) < T(envía(Y,m_2))$

  ocurrió_antes(recibe(Y,m_1), envía(Y,m_2))

  Luego, por transitividad:

  $T(envía(Y,m_2)) < T(recibe(X,m_2))$

  ocurrió_antes(envía(X,m_1), envía(Y,m_2))

  X envía m_1 antes de que Y envíe m_2

Modelo de fallo *tipos*

- Fallo por omisión (del proceso, o del canal)
  - Los componentes no cumplen sus funciones
- de temporización
- El fallo arbitrario requiere una categoría especial (o binzantinos)

<u>Temas de robustez y fiabilidad frente a fallos</u>

- Enmascaramiento de fallos
- Fiablidad y comunicación uno a uno

Modelo de fallo *tipos*

Rotura o caída (crash)	
El proceso se detiene indefinidamente, y el error puede no ser detectable.	
Opciones: 1. Un protocolo temporizado puede detectar fallos en la operación. 2. Un mecanismo de encuesta (*polling*) puede detectar fallos.	

- Es mejor que *ciertos* procesos caigan antes que fallar y funcionar mal.
  - El diseño es más simple si se supone que los procesos erróneos caen y se detecta la caída.

Modelo de fallo *tipos*

Fallo-detención (fail-stop)	
Caída detectable	
Opciones: Se recoge una notificación de caída	

Posiblemente admite reintentos

Omisión de emisión **R**	
Se completa "envía()", pero no llega a emitirse el mensaje	
Opciones: "envía()" devuelve un mensaje de error, que hay que catalogar	

Modelo de fallo *tipos*

Omisión de recibimiento	**R**	
Se completa la recepción (escucha), pero no se "recibe()" el mensaje		
Opciones: 1. "recibe()" emite un error que hay que catalogar 2. Si "recibe()" está temporizado no se emite error y hay que interrogar a la interfaz de red.		

Modelo de fallo *tipos*

Omisión del canal	**R**	
El mensaje no llega a ser escuchado		
Opciones: 1. Si "recibe()" está temporizado no se emite error y hay que interrogar a la interfaz de red.		

Modelo de fallo *tipos*

Fallo arbitrario (bizantino) **R**	
El sistema presenta un comportamiento arbitrario: omisiones, tiempos arbitrarios, paradas, fallos.	
Opciones: 1. Intentar catalogar el fallo de un modo más preciso mediante sondeos. 2. Incluir comprobaciones para descartar comportamientos puntualmente erróneos (ej: checksums) 3. El sistema puede tener que parar completamente para no ocasionar daños.	

Modelo de fallo *tipos*

Fallo en el reloj **R**	
La tasa de deriva del reloj es excesiva, y afecta al proceso local.	
Opciones: 1. Sincronizar frecuentemente. 2. Considerar inválido el reloj local.	

Modelo de fallo *tipos*

Fallo de prestaciones en el proceso	
El proceso se demora en exceso	
Opciones: 1. Introducir un protocolo de sondeo con procesos de apoyo, para descartar fallos en el canal. 2. Invalidar el proceso mediante *timeouts*.	

Fallo de prestaciones en el canal **R**	
La transmisión se demora en exceso.	
Opciones: 1. Introducir un protocolo de sondeo con procesos de apoyo, para descartar fallos en el proceso.	

Modelo de fallo *enmascaramiento de fallo*

- Aún es posible construir servicios **fiables** con componentes que presenten fallos.
  - Ocultándolos bajo una capa software que se recupere de ciertos fallos
    - Ejemplo: TCP/IP sobre red no fiable. (**Reintentos**) + (**Reordenamiento**)
  - Convirtiéndolo en un fallo más aceptable
    - Ejemplo: un fallo en un checksum de un dato recibido (fallo arbitrario) se puede convertir en un fallo por omisión del canal. (**Comprobaciones auxiliares**)

Modelo de fallo *enmascaramiento de fallo*

- Mediante redundancia y replicación
  - Ej: sistema de varios servidores con réplicas
  - Ej: sistema RAID para archivos.
  - (soluciones ad-hoc) ...
- El enmascaramiento y ocultación del fallo suele requerir el diseño de un servicio de más alto nivel:
  - Ej: la pila de niveles de transporte TCP e ISO/OSI no solo elevan el nivel de abstracción sino que enmascaran y ocultan muchos fallos.

Modelo de seguridad

- Las técnicas de seguridad permiten la comprobación de fallos y la minimización de su posible aparición:

  Comunicación fiable

  - Validez de la comunicación:
    cualquier mensaje enviado() será escuchado.
  - Integridad de la comunicación:
    Cualquier mensaje recibido() es correcto y respeta la secuencialidad.
- Amenazas:
  - Duplicación de mensajes, desorden, corrupción del mensaje, revelación, (y sigue...)

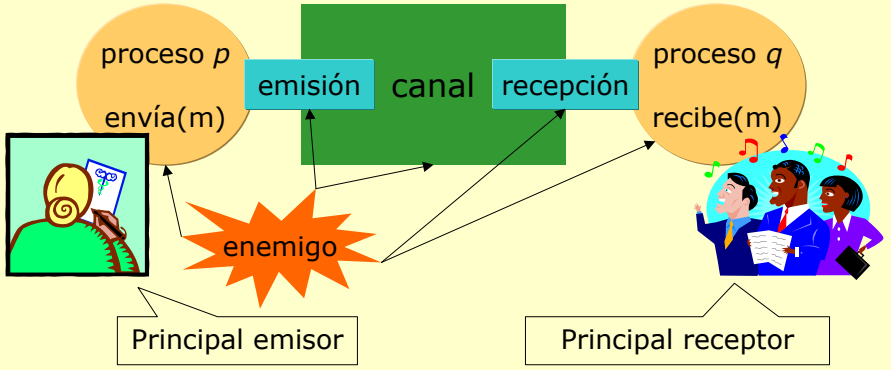
Modelo de seguridad *amenazas*

- A los procesos:
  - Acceso indebido a los recursos
  - Ataque a la integridad del proceso
  - Suplantación de los principales interlocutores
  - Falsificación de servicios
  - Falsificación de peticiones

Modelo de seguridad *amenazas*

- A los canales:
  - Acceso indebido al canal
  - Captura de mensajes
  - Reenvío de mensajes
  - Eliminación de mensajes
  - Modificación de mensajes y de código móvil
- A la disponibilidad del servicio:
  - Ataque a la integridad de los servicios
  - Ataque de denegación de servicio

Modelo de seguridad *amenazas*



- Seguridad de...
  - Las interacciones en procesos y canales
  - Las acciones de acceso a objetos (derechos)

Modelo de seguridad *técnicas*

- Criptografía (de clave secreta y de clave pública)
  - Encriptación para preservar la privacidad
  - Firmas para preservar la autenticidad
  - Autenticación para preservar la identidad
  - Contrato digital para preservar la legalidad
- Ejemplos de servicios seguros
  - Correo electrónico seguro
  - Pagos seguros por Internet

Modelo de seguridad *técnicas*

- Técnicas sobre derechos de acceso
  - Control de acceso a los recursos
  - Control de acceso a los servicios
- Técnicas de filtrado y seguridad de tráfico en redes.
  - Mecanismos de confianza por dominios
  - Protección pasiva
    - Cortafuegos
  - Tunelización
  - ...