CC5303 – Sistemas Distribuidos

2.- Comunicaciones

Parte 1

Sebastián Blasco V.

Contenidos

1. Fundamentos
   1. Modelo OSI
2. Tipos de Comunicación
3. Tecnologías de comunicación
   1. RPC
   2. Comunicación con Mensajes

Fundamentos

- La comunicación entre procesos se encuentra en el núcleo de todos los sistemas distribuidos.
- La comunicación siempre se basa en el paso de mensajes de bajo nivel.
- Toda la comunicación se basa en el envío y la recepción (de bajo nivel) de mensajes

Fundamentos

Cuando el proceso A debe comunicarse con el proceso B, primero elabora un mensaje en su propio espacio de dirección. Después ejecuta una llamada de sistema y ocasiona que el sistema operativo envíe el mensaje sobre la red hacia B

Existe un acuerdo para la comunicación

Fundamentos

¿Porque es importante dicho acuerdo?

- ¿Cuántos volts deben utilizarse para representar un bit 0, y cuántos para representar un bit 1?
- ¿Cómo sabe el destinatario cuál es el último bit del mensaje?
- ¿Cómo puede detectar si un mensaje se ha dañado o perdido, y qué debe hacer si lo descubre?
- ¿Qué tan largos son los números, cadenas, y otros elementos de datos, y cómo se representan?

El acuerdo estipula reglas estándar que regulen **formato, contenido, y significado** de los mensajes enviados y recibidos.

Fundamentos: Protocolos

Los protocolos son las reglas a que deben apegarse los procesos de comunicación. Nos dan pauta de los acuerdos ya establecidos.

Pueden ser orientados:

- A la conexión
  - El remitente y el destinatario primero establecen explícitamente una conexión, y posiblemente negocian el protocolo que utilizarán.
    - Teléfono
- A la no conexión
  - No se necesita una configuración por adelantado. El remitente sólo transmite el primer mensaje cuando está listo.
    - Una Carta

Fundamentos: Modelo OSI

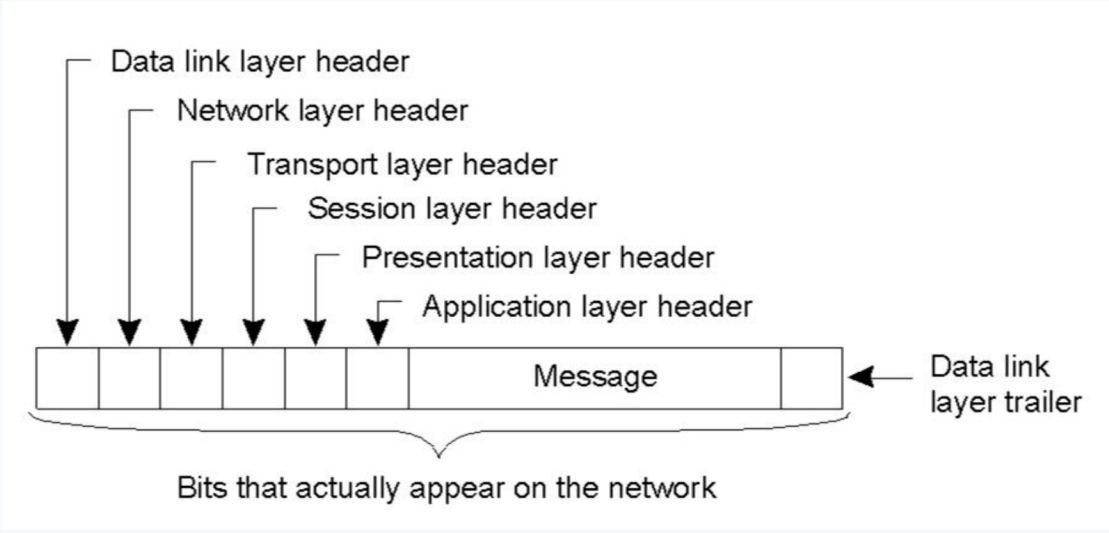
Un modelo de referencia para los protocolos de la red de arquitectura en capas, creado en el año 1980.

- La separación de la comunicación en capas **reduce la complejidad** de la misma limitando las responsabilidades

Capa de Aplicación	Programas de aplicación que usan la red.
Capa de Presentación	Estándar en la forma en que se presentan los datos a las aplicaciones.
Capa de Sesión	Gestión de las conexiones entre aplicaciones cooperativas.
Capa de Transporte	Servicios de detección y corrección de errores.
Capa de Red	Gestión de conexiones para capas superiores.
Capa de Enlace de Datos	Envío de datos al enlace físico.
Capa Física	Define características físicas del medio de transmisión de la red.

Fundamentos: Modelo OSI

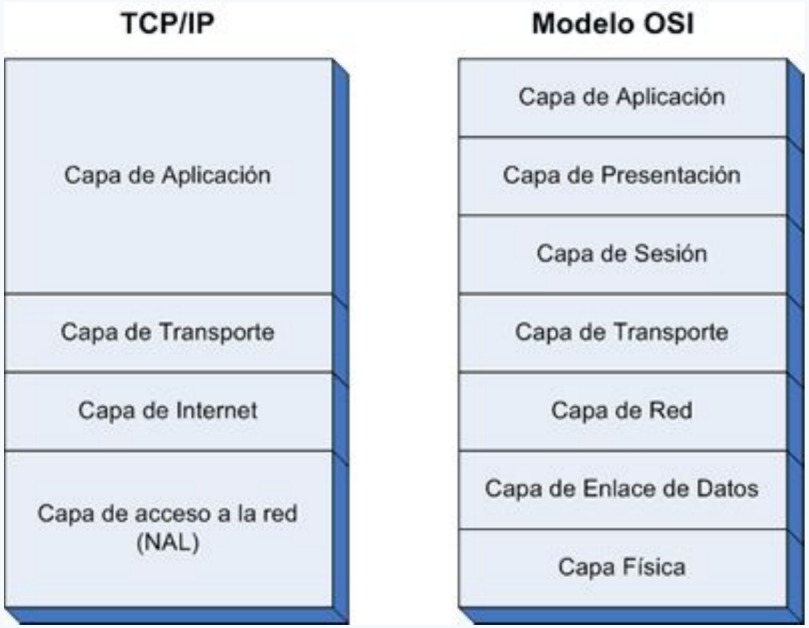
- Cada capa proporciona una interfaz hacia la siguiente capa superior. La interfaz consiste en un conjunto de operaciones que definen el servicio que la capa podrá ofrecer a sus usuarios.



- La información del encabezado de la capa n se utiliza para el protocolo de la capa n

Fundamentos: Modelo OSI

Modelo OSI en la práctica

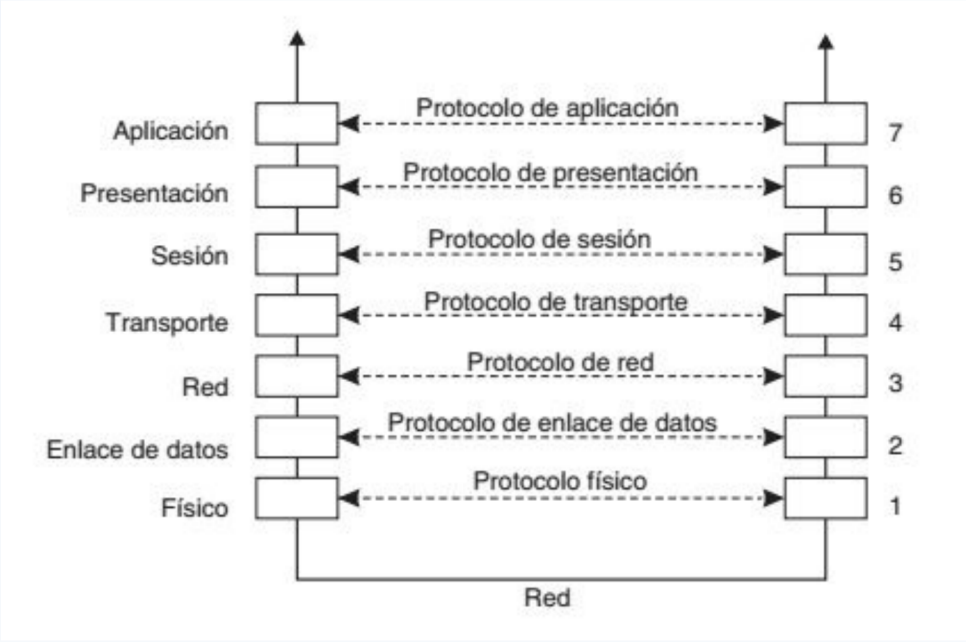


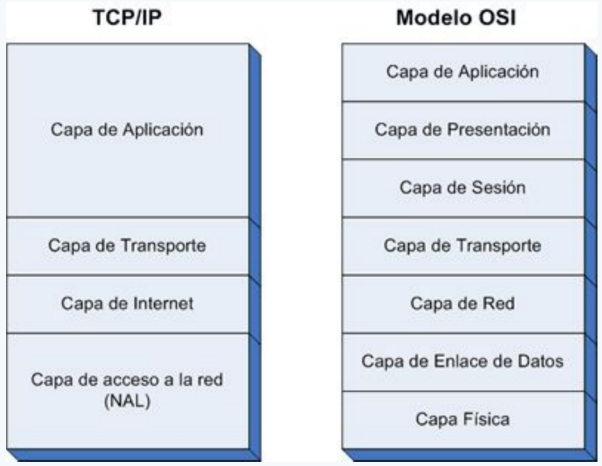
Fundamentos: Modelo OSI

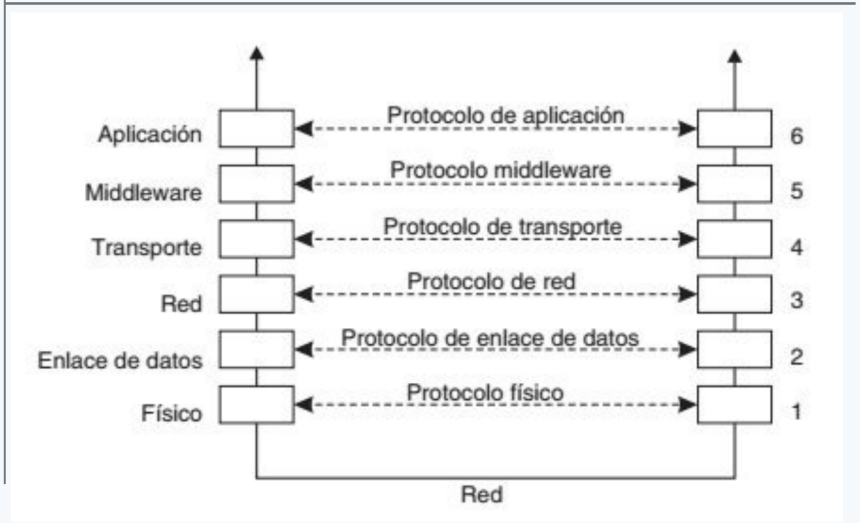
En el caso de SSDD, el middleware es una aplicación que lógicamente reside (la mayor parte del tiempo) en la capa de aplicación.

- Los protocolos de comunicación middleware soportan servicios de comunicación de alto nivel.
  - llamadas a procedimientos remotos
  - servicios de cola de mensajes
  - soporte para comunicación de medios continuos a través de flujos
  - multitransmisión

Fundamentos: Modelo OSI, TCP/IP, Middleware







Tipos de Comunicación

Existen distintas alternativas de comunicación que puede proveer el sistema, según el comportamiento que se desee en la dinámica de colaboración cliente/servidor ofrecido por el sistema mismo (middleware en SSDD)

- Síncrona o asíncrona
- Persistente o Transiente
- Directa o indirecta
- Simétrica o asimétrica
- Envío por copia del mensaje o por referencia
- Mensajes de tamaño fijo o variable
- Etc.

Tipos de Comunicación

- Según modo de envío del mensaje:
  - Comunicación Directa
    - Las primitivas *send* y *receive* explicitan el nombre del proceso con el que se comunican. Es decir se debe especificar cuál va a ser el proceso fuente y cual va a ser el proceso Destino.
  - Comunicación Indirecta
    - Aquella donde la comunicación está basada en una herramienta o instrumento ya que el emisor y el receptor están a distancia.

Tipos de Comunicación

- Según direccionalidad de la comunicación:
  - Comunicación Simétrica (o bidireccional)
    - Todos los procesos pueden enviar o recibir. También llamada bidireccional para el caso de dos procesos. Es una comunicación equilibrada donde tanto emisor como receptor reciben la misma información.
  - Comunicación Asimétrica (o unidireccional)
    - Un proceso puede enviar, los demás procesos sólo reciben. También llamada unidireccional. Suele usarse para hospedar algunos servicios en Internet.

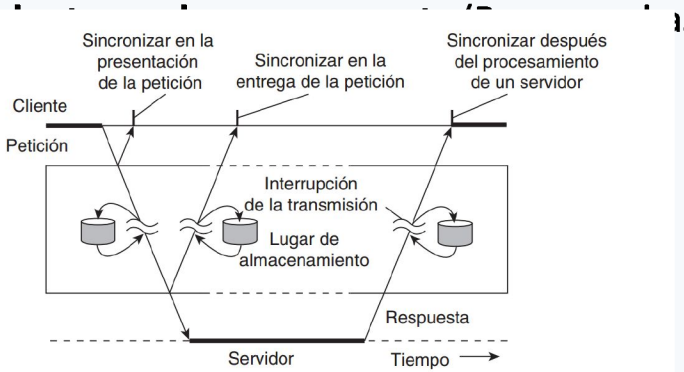
Tipos de Comunicación

- Según persistencia del mensaje:
  - Comunicación Persistente
    - El mensaje es guardado en el sistema de comunicación hasta que el receptor lo procesa, si no está activo lo espera
  - Comunicación Transiente
    - El mensaje es guardado en el sistema mientras el receptor lo procesa, si no está activo lo descarta

En general, todos los servicios de comunicación al nivel de transporte sólo ofrecen comunicación transitoria.

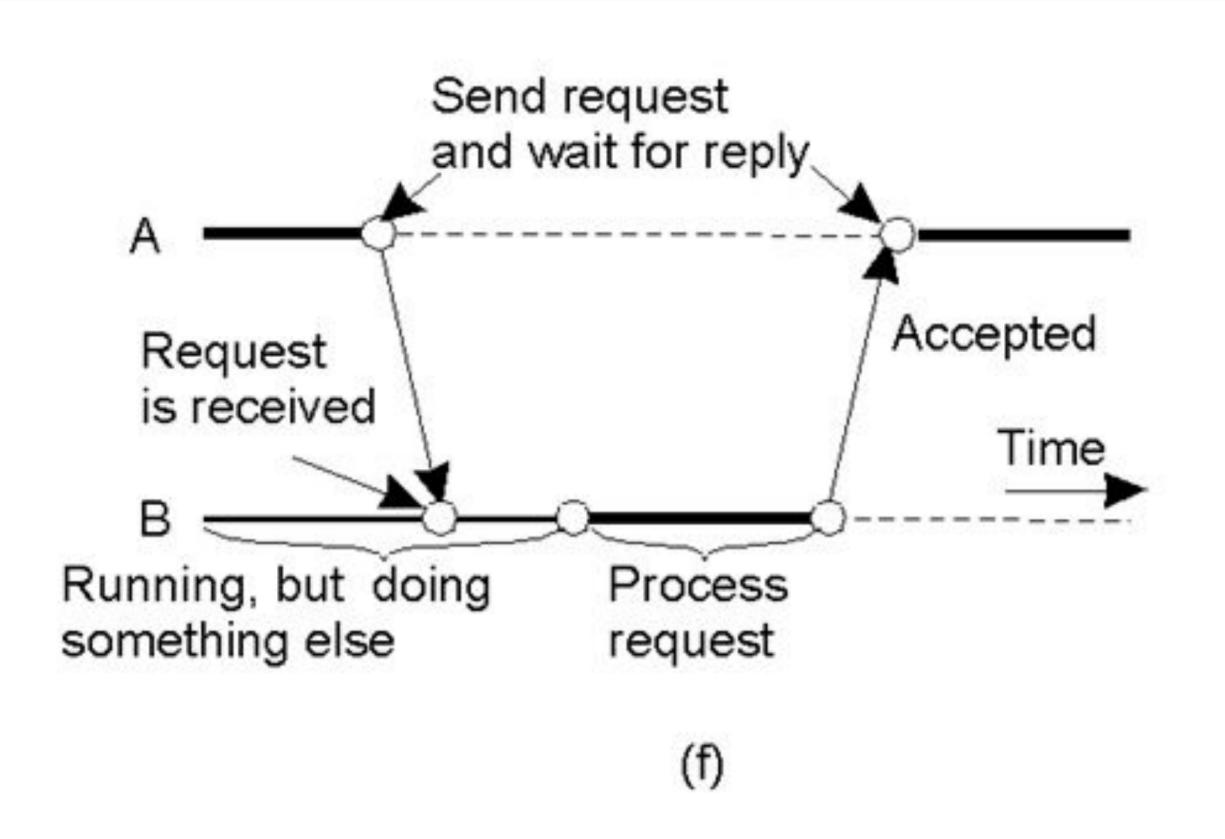
Tipos de Comunicación

- Según bloqueo del emisor:
  - Comunicación Asíncrona (no bloqueante)
    - El remitente continúa inmediatamente después de que ha pasado su mensaje para transmisión.
  - Comunicación Síncrona (bloqueante)
    - El remitente es bloqueado hasta que se sabe que su petición es aceptada.
      - Hasta notificacion de salida del mensjae del sistema local (Receipt-based)
      - Hasta la entrega del mensaje al sistema remoto (Delivery-based)
      - **Has**[illegible]**ased)**



Tipos de Comunicación

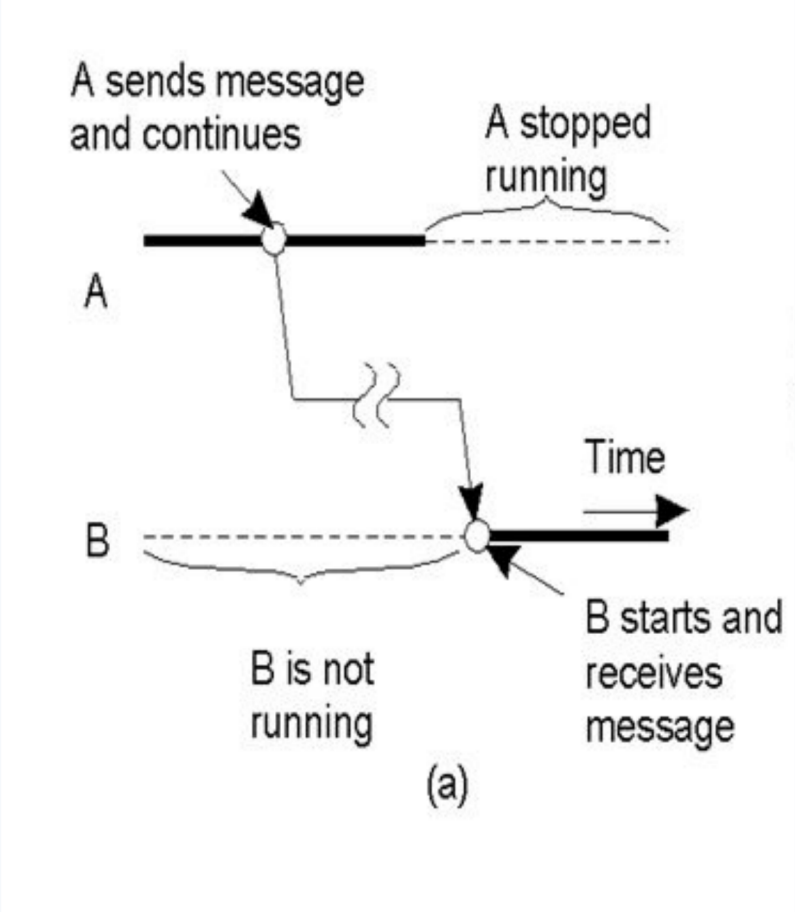
- Síncrono (response-based) + Transiente



(f)

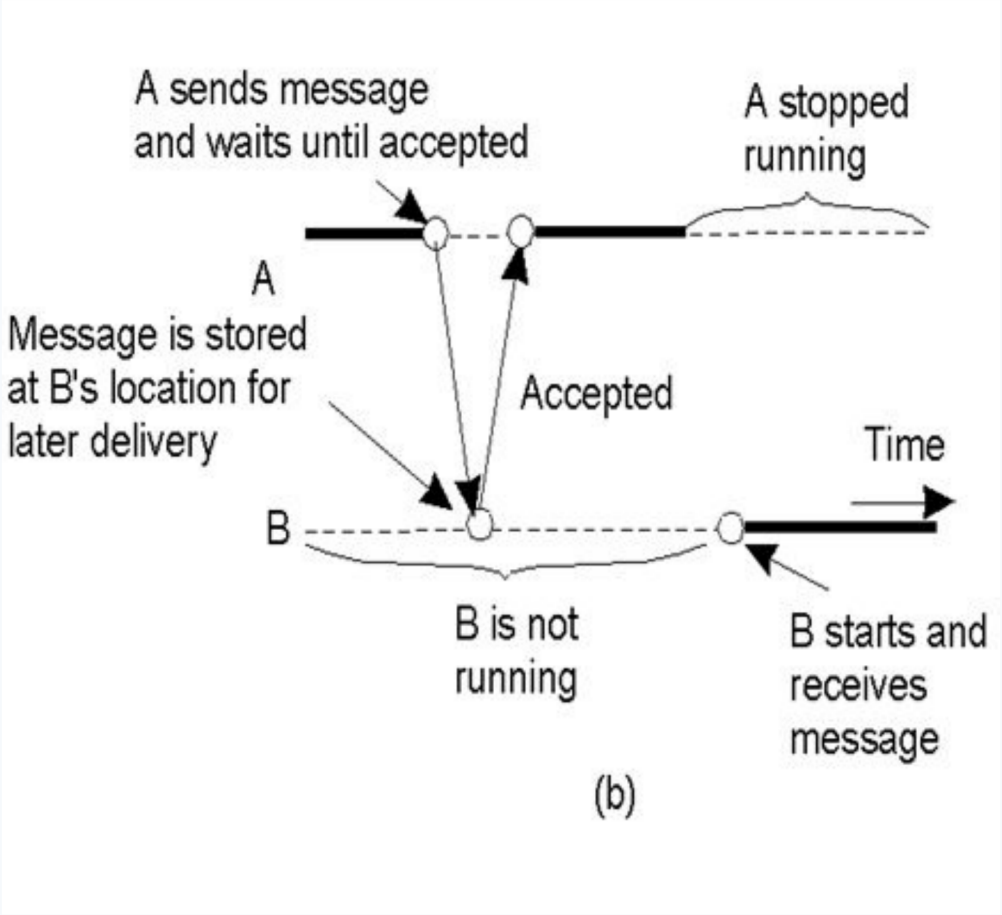
Tipos de Comunicación

- Asíncrono + Persistente



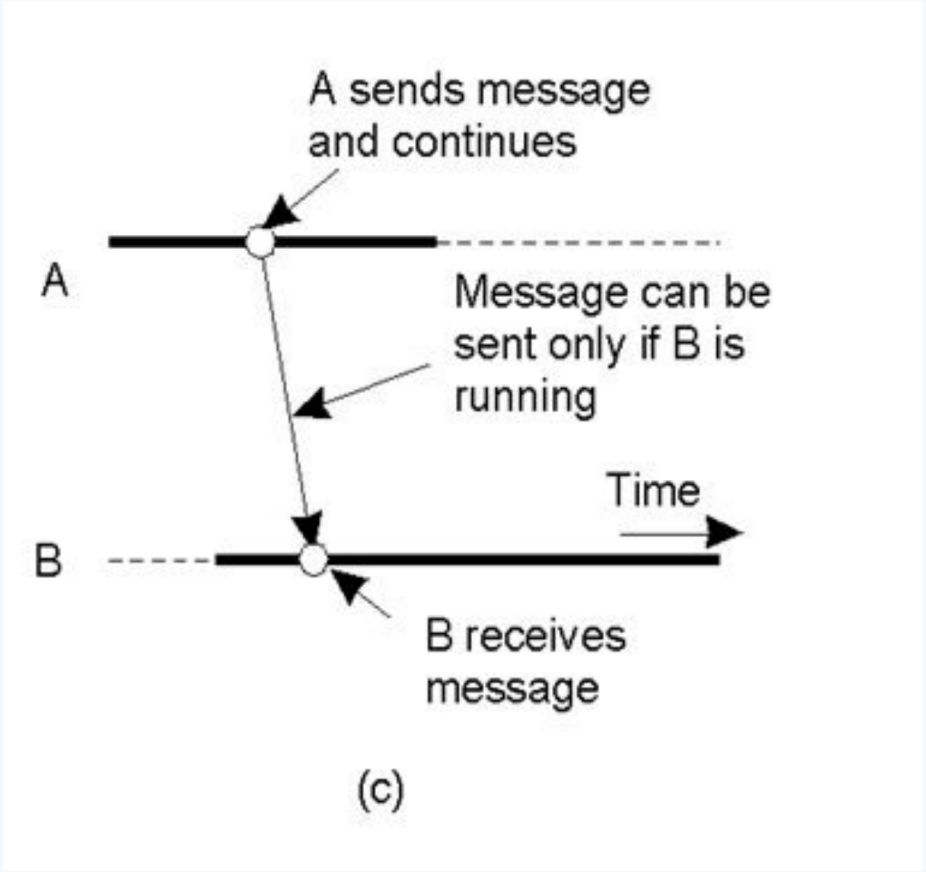
Tipos de Comunicación

- Síncrono (receipt-based) + Persistente



Tipos de Comunicación

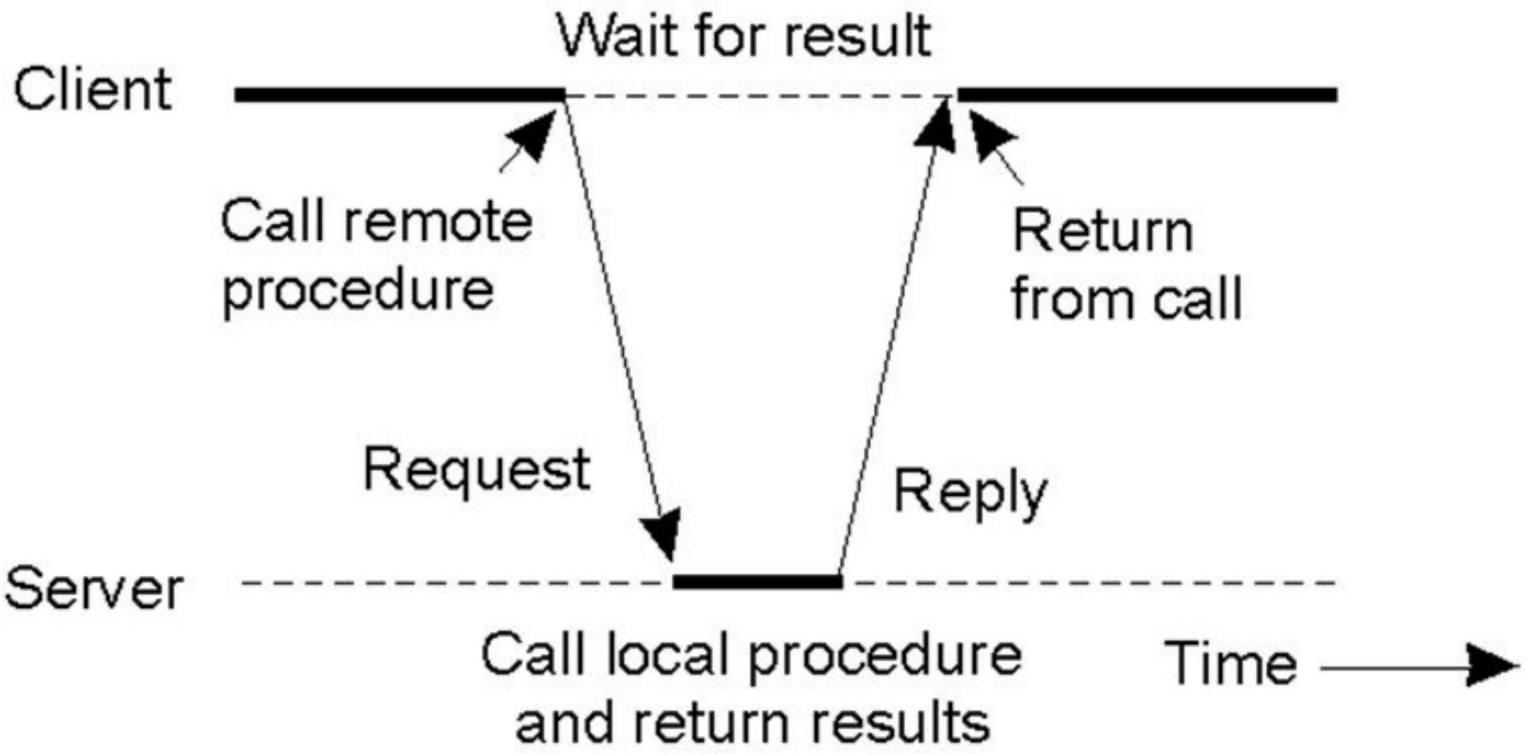
- Asíncrono + Transiente



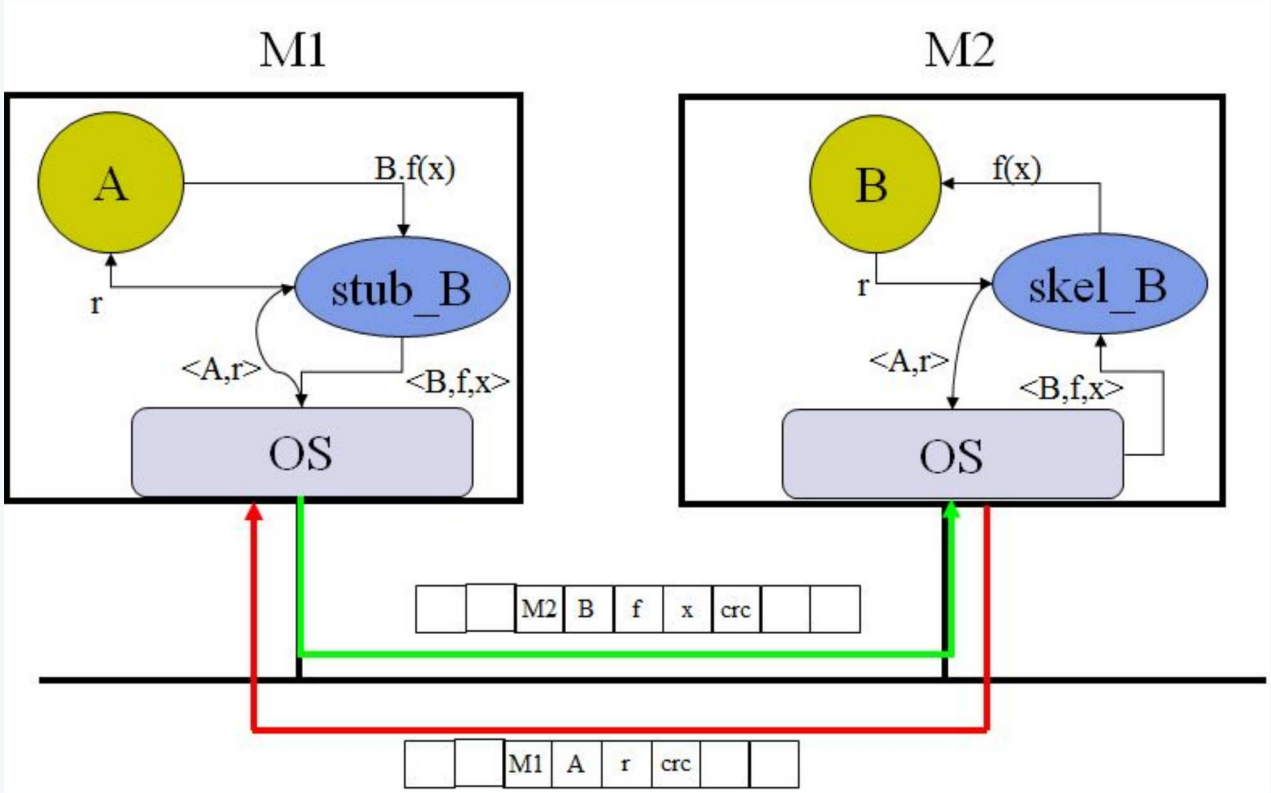
RPC

- Es necesario comunicar procesos. Las llamadas send/receive no ocultan la comunicación
  - Se sugiere permitir que los programas llamaran a procedimientos ubicados en otras máquinas.
  - Cuando un proceso de la máquina A llama a un procedimiento de la máquina B, **el proceso que llama desde A se suspende, y la ejecución del procedimiento llamado ocurre en B.**
  - La información puede transportarse en los parámetros desde quien llama hasta el que es llamado, y puede regresar en el procedimiento resultante.
  - Ningún mensaje de paso es visible para el programador.
  - Este método se conoce como *llamada a procedimiento remoto*

RPC


Wait for result
Client
Call remote
procedure
Return
from call
Request
Reply
Server
Call local procedure
and return results
Time

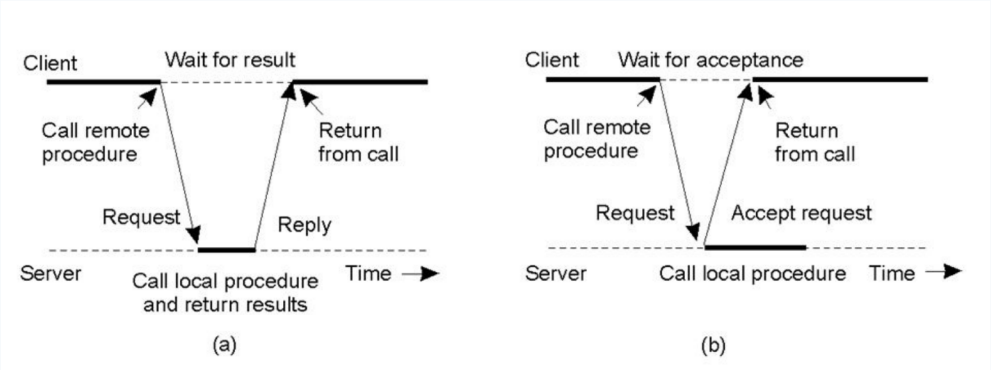
RPC



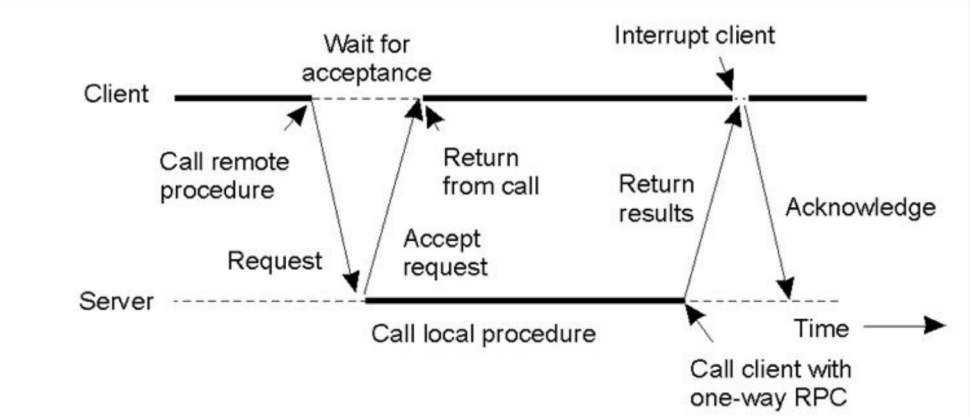
https://docs.oracle.com/javase/7/docs/platform/rmi/spec/rmi-arch2.html

RPC

RPC también implementa mecanismos para llamadas Asíncronas por medio de sincronísmo *Delivery-based*



a) RPC Tradicional
b) RPC Asíncrono



Interacción RPC Asíncrono

RPC

- Si se envía un objeto como parámetro se realiza una copia.
- Si se envía el stub de un objeto como parámetro se está realizando un duplicado de la referencia.

Más info:
http://java.sun.com/docs/books/tutorial/rmi/index.html

Comunicación con Mensajes

RPC suena muy bueno… quizá demasiado (?)

- Necesita un middleware común
- Necesita que el receptor esté en ejecución
- Sincronismo inherente

Ningún mecanismo es perfecto… pero por eso existen otras opciones: **Comunicación con Mensajes**

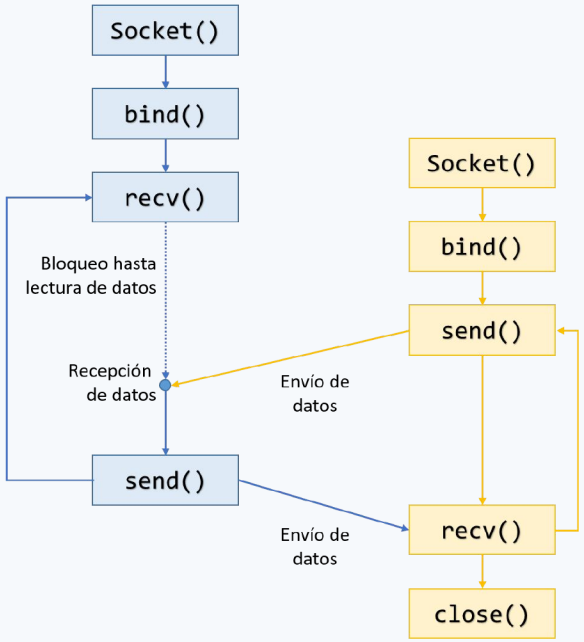
Comunicación con Mensajes

Diseñados e implementados directamente sobre el sencillo modelo orientado a mensajes ofrecido por la capa de transporte *(message-oriented model)*

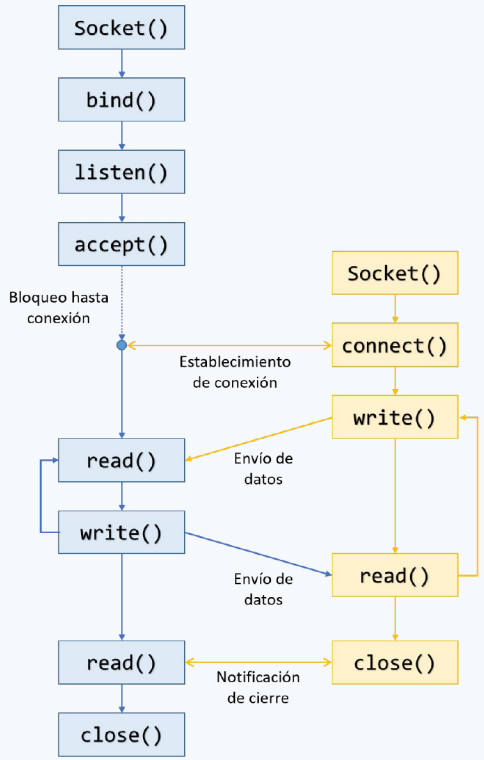
- Transiente
  - Berkeley Sockets
  - MPI
- Persistente:
  - Message-Queuing Systems
  - Message Brokers

Comunicación *Transiente* con Mensajes

Berkeley Sockets



Dinámica Orientada a la no conexión



Dinámica Orientada a la conexión

Comunicación *Transiente* con Mensajes

MPI (Message Passing Interface)

- Con el avance en tecnología, aparecieron computadoras de alto rendimiento. Se necesitó entonces, diseñar un mecanismo para comunicarse eficientemente.
  - ¿Porque no usar los sockets?
    - Escasa (casi nula) abstracción, pues sólo brindan soporte de las primitivas simples para enviar y recibir información.
    - No se consideraron adecuados para los **protocolos propietarios** desarrollados para redes de interconexión de alta velocidad, tales como las utilizadas en clústeres de servidores de alto rendimiento, protocolos requerían una interfaz que pudiese manejar características más avanzadas, como **diferentes formas de utilizar el buffer y de sincronización**, por ejemplo.

Comunicación *Transiente* con Mensajes

MPI (Message Passing Interface)

- Estándar que define la sintaxis y la semántica de las funciones contenidas en una biblioteca de paso de mensajes diseñada para ser usada en programas que exploten la existencia de múltiples procesadores.

- Los procesos invocan diferentes funciones MPI que permiten
  - Iniciar, gestionar y finalizar procesos MPI
  - comunicar datos entre dos procesos
  - Realizar operaciones de comunicación entre grupos de procesos
  - Crear tipos arbitrarios de datos

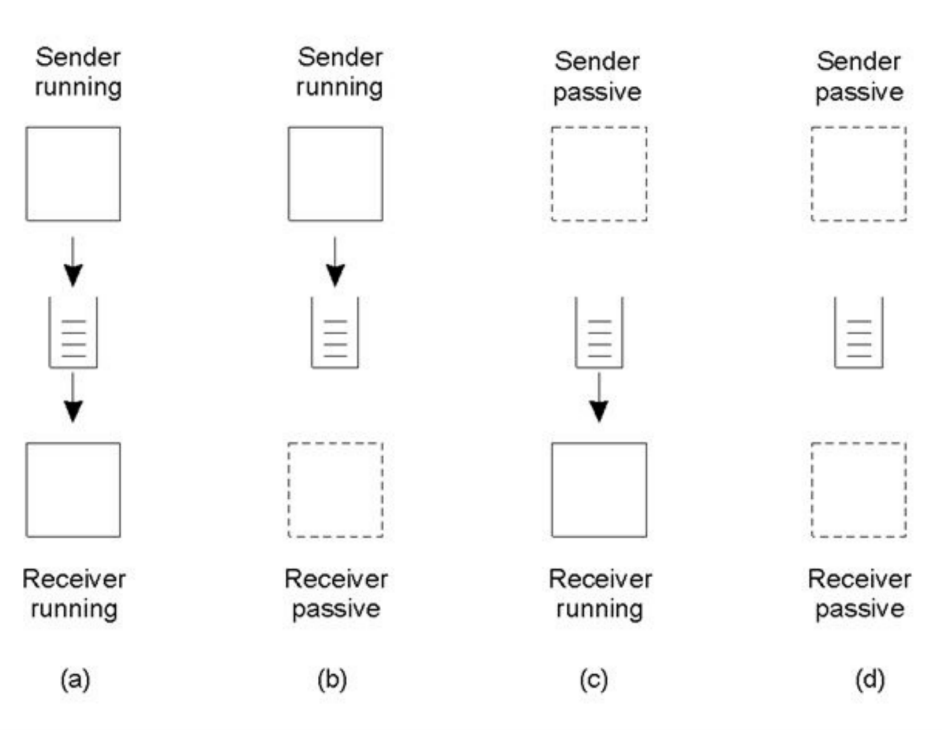
Comunicación *Persistente* con Mensajes

Sistemas de colas de mensajes (Message-Queuing Systems)

- Proporcionan un amplio soporte para comunicación asíncrona persistente.
- La esencia de estos sistemas es que ofrecen capacidad de **almacenamiento de término medio para mensajes**, sin la necesidad de que el remitente o el destinatario estén activos durante la transmisión del mensaje.
- Basado en Message-Queuing Model
- Por lo general, al remitente sólo se le garantiza que su mensaje se insertará en algún momento en la cola del destinatario. **No se dan garantías sobre cuándo, o de si el mensaje en realidad será leído**, lo cual es completamente definido por el comportamiento del destinatario.

Comunicación *Persistente* con Mensajes

Sistemas de colas de mensajes (Message-Queuing Systems)

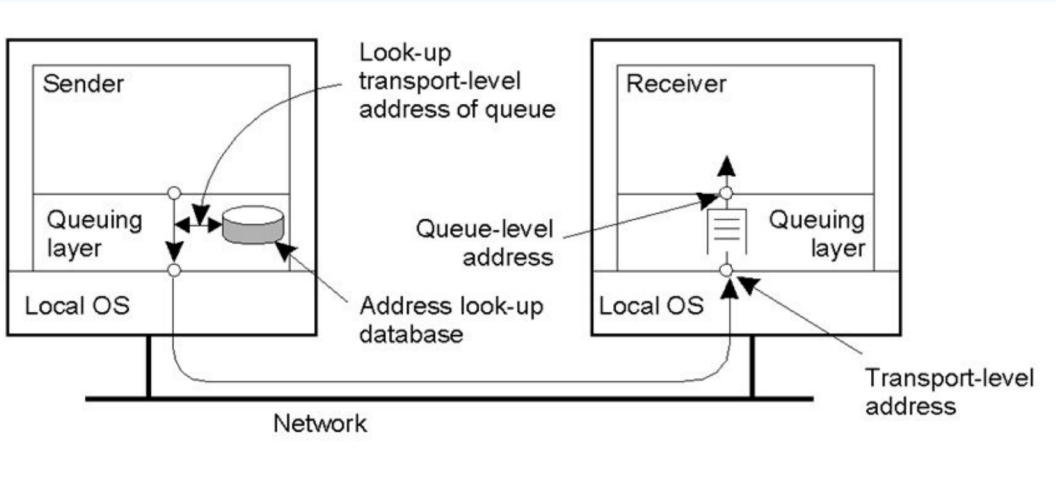


Cuatro combinaciones de comunicaciones muy poco acopladas mediante el uso de colas

Comunicación *Persistente* con Mensajes

Arquitectura de un Sistemas de colas de mensajes

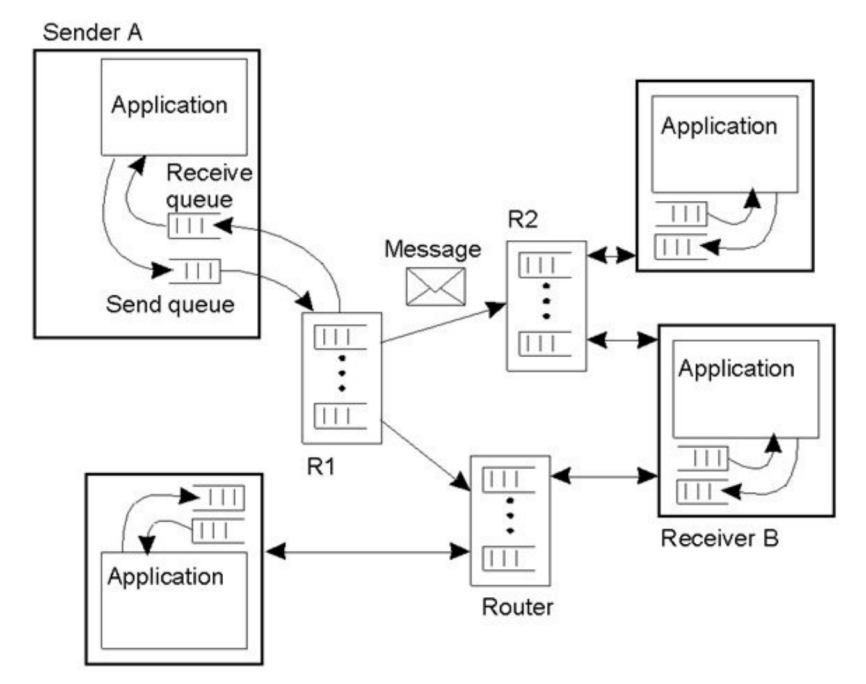
- Para que un sistema de colas de mensajes transfiera mensajes, es necesario que mantenga un mapa de las colas para localización en la red. En la práctica, esto significa que debe mantener una base de datos (probablemente distribuida) de los nombres de las colas para localizarlas en la red



Comunicación *Persistente* con Mensajes

Arquitectura de un Sistemas de colas de mensajes

- En algunos casos se emplean administradores especiales de colas que actúan como enrutadores o **retransmisores**, desacoplando la responsabilidad de la persistencia a una capa inferior.



Comunicación *Persistente* con Mensajes

Message Brokers (Agentes de Mensajes)

- En SSDD que integran aplicaciones de distintas características, es posible que los mensajes de entrada y salida de cada aplicación difieran en forma, lo cual significa un problema para el SSDD completo.
- Un agente de mensaje actúa como una puerta de enlace al nivel de aplicaciones en un sistema de colas de mensajes. Su propósito principal es **convertir los mensajes entrantes de tal manera que la aplicación de destino pueda comprenderlos**.
- Importante para la integración de aplicaciones distintas.

Comunicación *Persistente* con Mensajes

Message Brokers (Agentes de Mensajes)