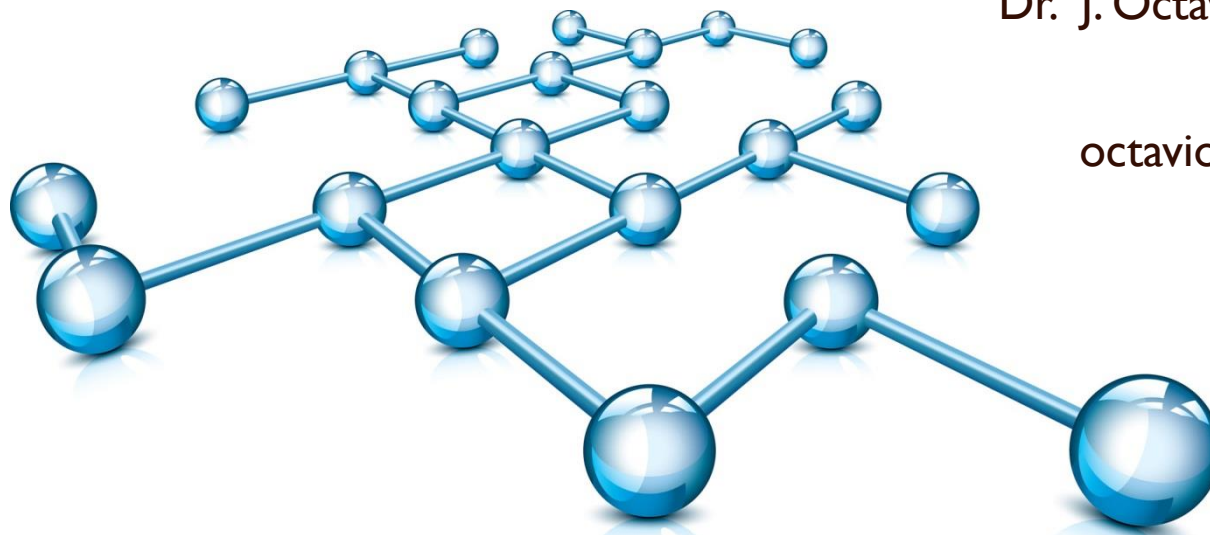




Modelos de Comunicación Indirecta



Profesor:

Dr. J. Octavio Gutiérrez García

octavio.gutierrez@itam.mx

Comunicación indirecta

- Comunicación entre entidades en un SD a través de un intermediario sin acoplamiento directo entre emisores y receptores
 - One-to-one / One-to-many
- Facilita la adaptación a cambios, aunque introduce retrasos



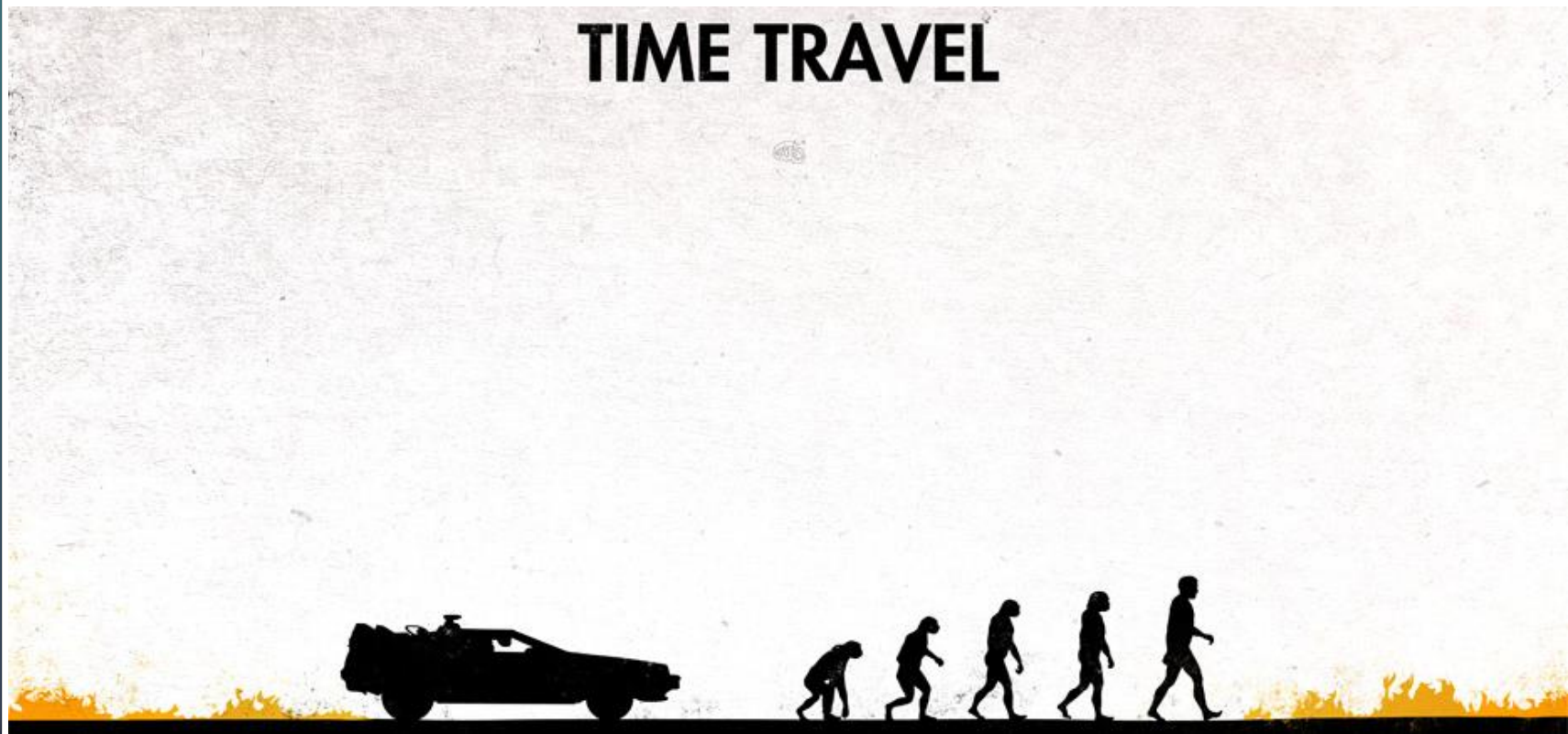
Comunicación indirecta

- **Desacoplamiento espacial**
 - El emisor no necesita conocer al receptor y viceversa.
 - Mayor libertad para tratar cambios: los participantes se pueden reemplazar, actualizar, replicar o migrar



Comunicación indirecta

- **Desacoplamiento temporal**
 - Los participantes pueden tener líneas de existencias independientes.



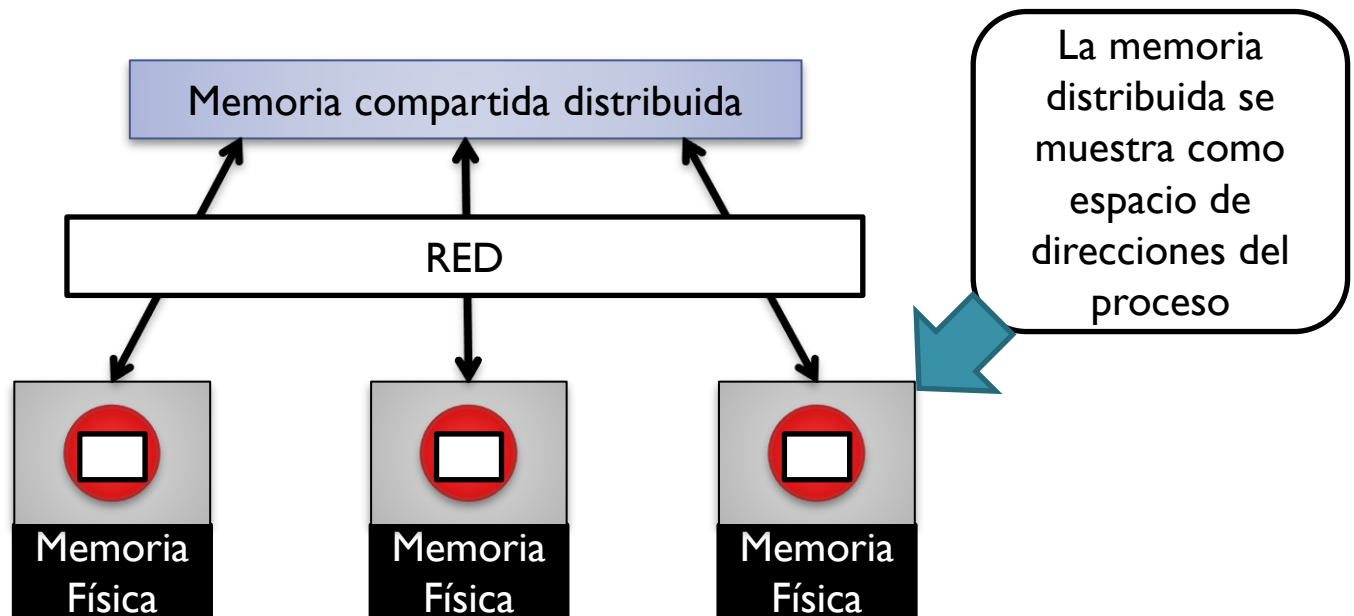
Comunicación indirecta

- SDs donde se anticipan **cambios**:
 - Ambientes altamente **volátiles** (ambiente móviles)
 - Diseminación de eventos donde los **usuarios** pueden **cambiar** y/o son **desconocidos**



Comunicación - Memoria Distribuida Compartida

- Abstracción utilizada para el intercambio de datos entre equipos que no comparten memoria física.
- Los **datos** almacenados se consideran **persistentes**
- **No** hay pasos de mensajes **entre** procesos



Comunicación grupal

- El **mensaje** se manda a un **grupo** y se difunde a todos los miembros del mismo
- El **emisor no conoce** a los receptores
- Los miembros pueden unirse y abandonar a un grupo
- Abstracción respecto a **multicast**



Comunicación grupal

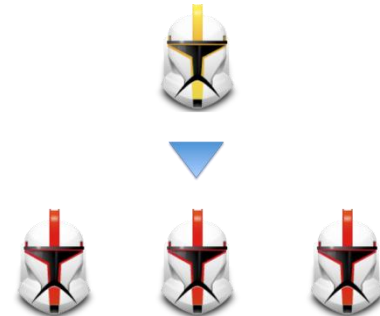
Es una base importante para:

- Diseminación fiable de información a muchos clientes



- Apoyo a aplicaciones colaborativas, para que los usuarios tengan la misma vista del sistema

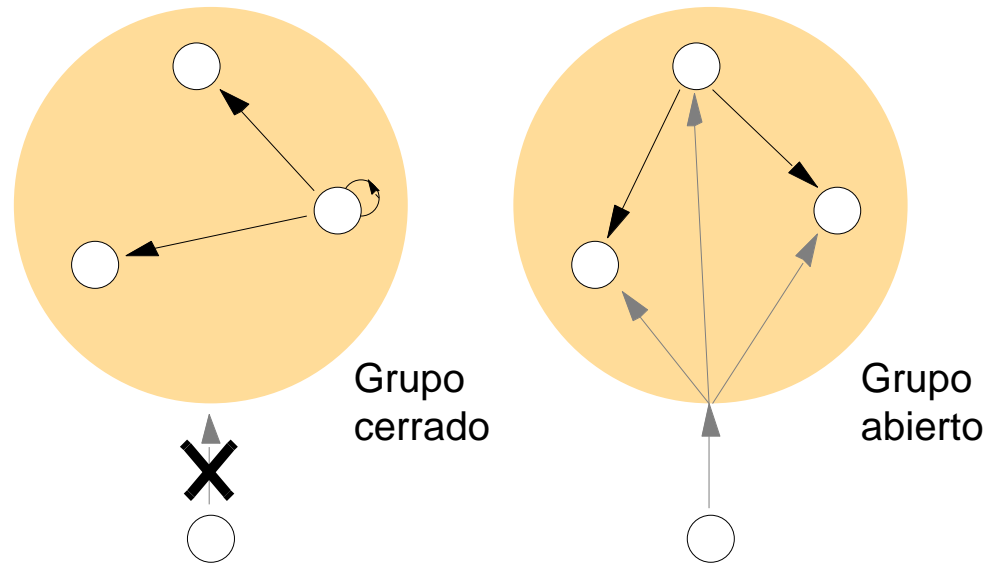
- Apoyo a estrategias de tolerancia a fallos para mantener coherente información replicada.



- Apoyo a sistemas de monitorización

Comunicación grupal

- Grupos **abiertos** vs Grupos **cerrados**



- Grupos con y sin **sobrelapamiento**

Comunicación grupal

Fiabilidad: garantías en la entrega



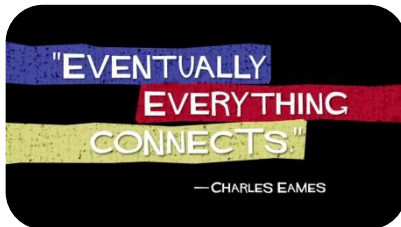
Acuerdo sobre los
mensajes que reciben
los miembros y en su
orden.

Comunicación grupal

Fiabilidad en **multicast** se define mediante tres propiedades:



- **Integridad:** El mensaje que se recibe es el mismo que se envió



- **Validez:** Los mensajes enviados se entregan en algún momento

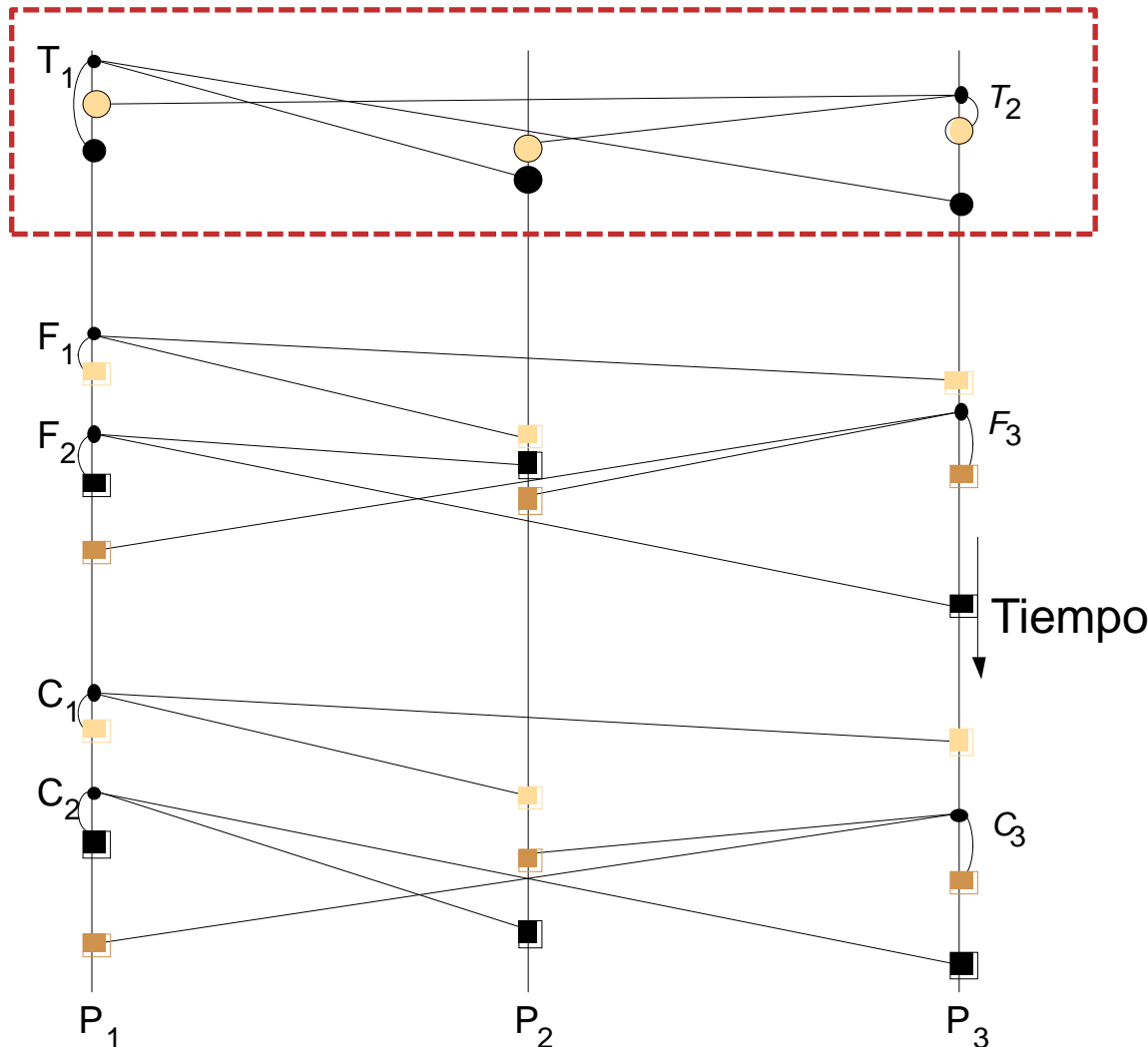


- **Acuerdo:** si el mensaje se entrega a un proceso, se entrega a todos los del grupo.

Comunicación grupal: Ordenamiento de mensajes

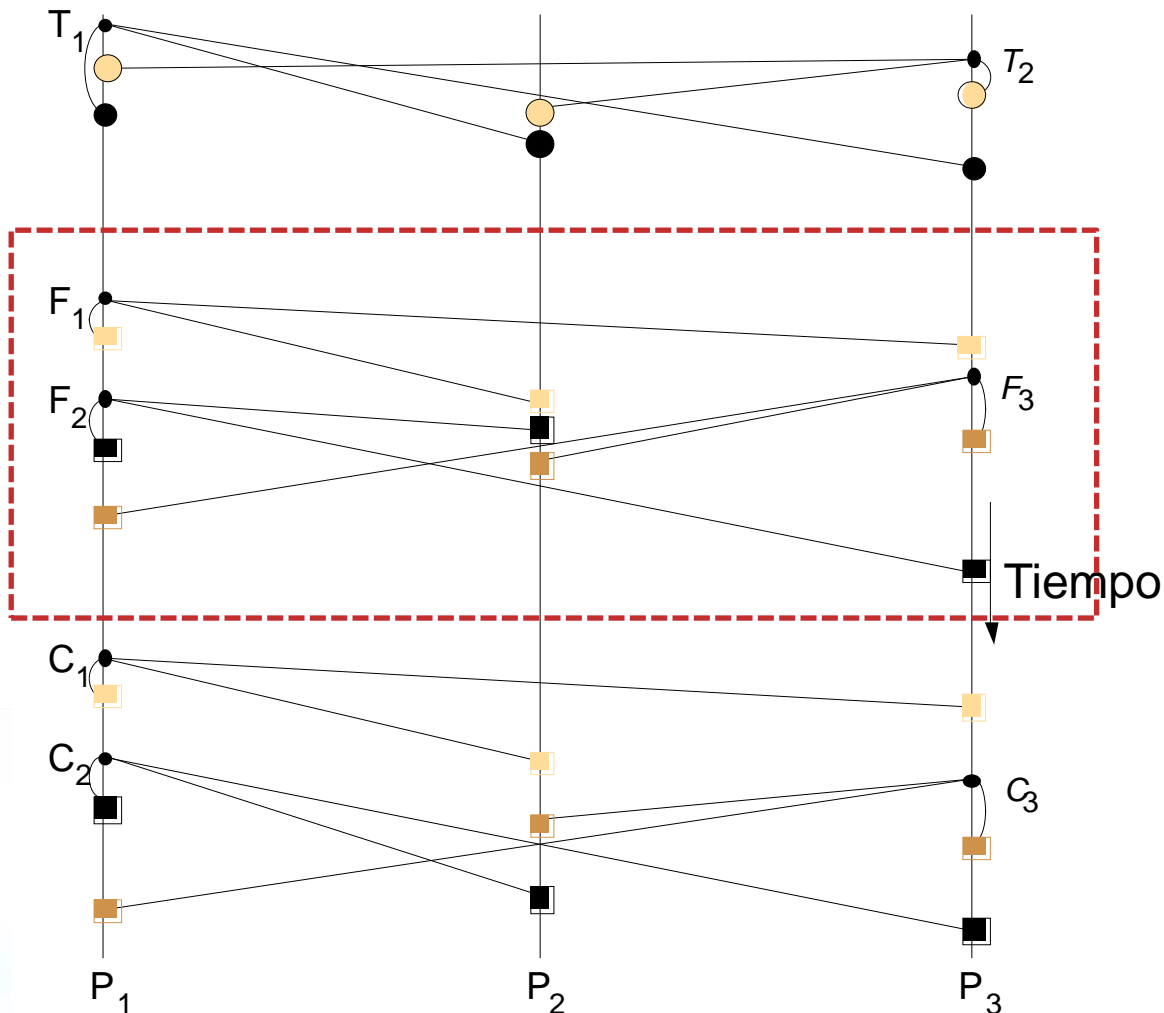
Total. Todos los mensajes son entregados en el mismo orden a todos los nodos

TOTAL



Comunicación grupal: Ordenamiento de mensajes

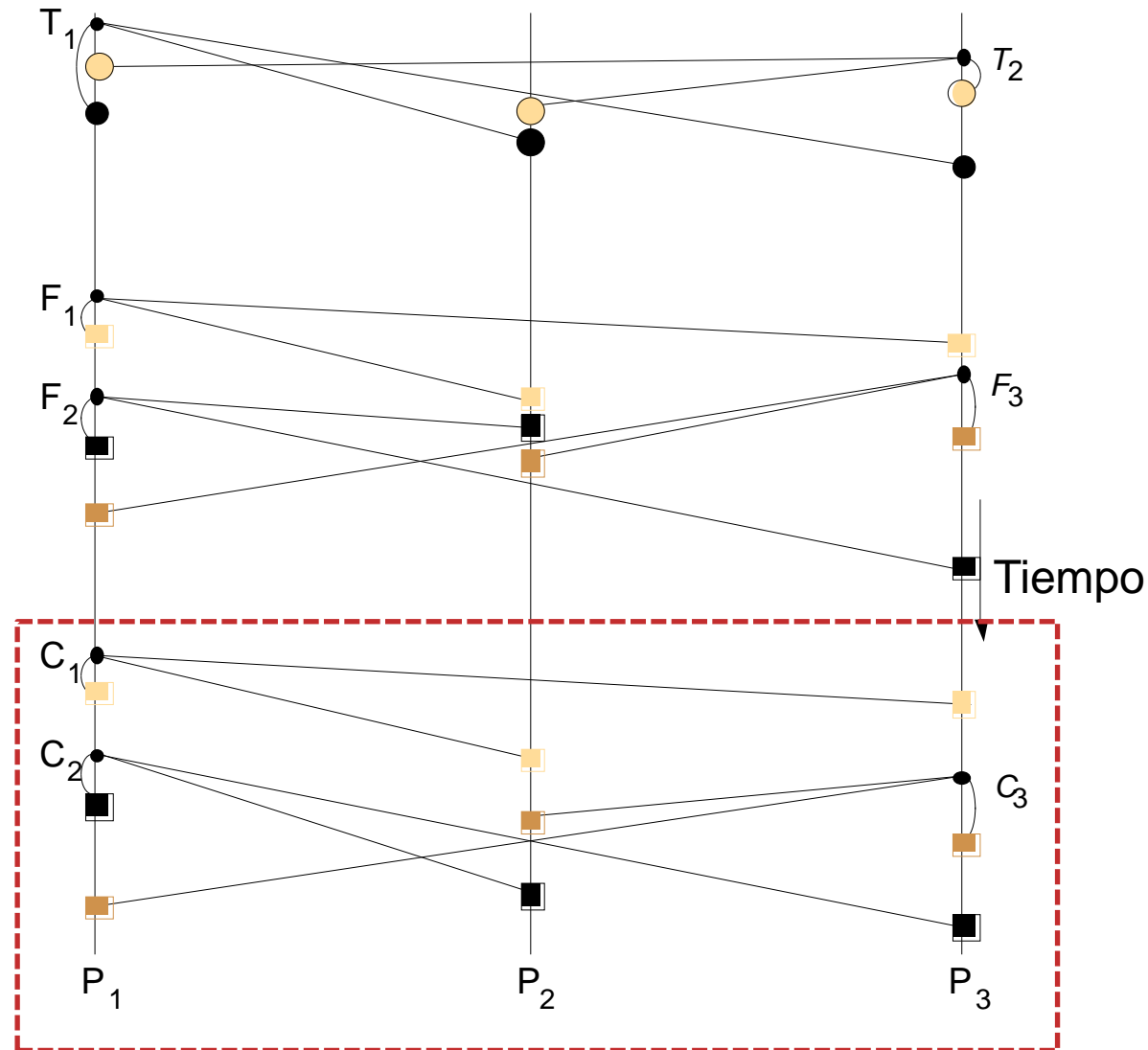
FIFO. Mensajes
de la misma
fuente llegan en
el orden que
fueron enviados



Comunicación grupal: Ordenamiento de mensajes

- **Causal.**

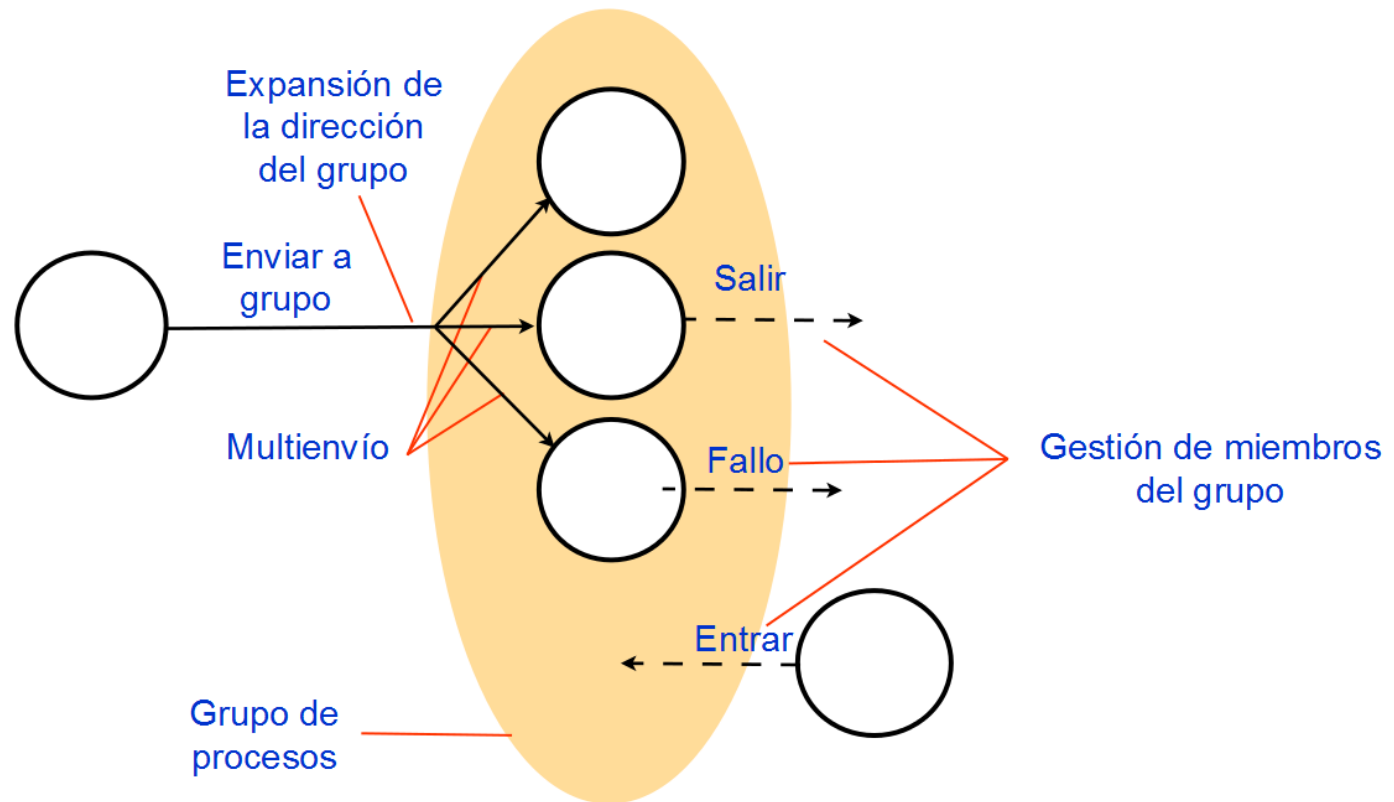
Generaliza la entrega de mensajes FIFO a un escenario en el que los mensajes se entregan en el orden de causalidad



Comunicación grupal

Manejo de membresía al grupo

- 1) Servicio de pertenencia y Manejo de grupos
- 2) Detector de fallos
- 3) Notificación de miembros
- 4) Expansión de direcciones



JGroups

A Toolkit for Reliable Multicast Communication



Comunicación publicación-inscripción



Comunicación publicación-inscripción

- Sistemas distribuidos basados en

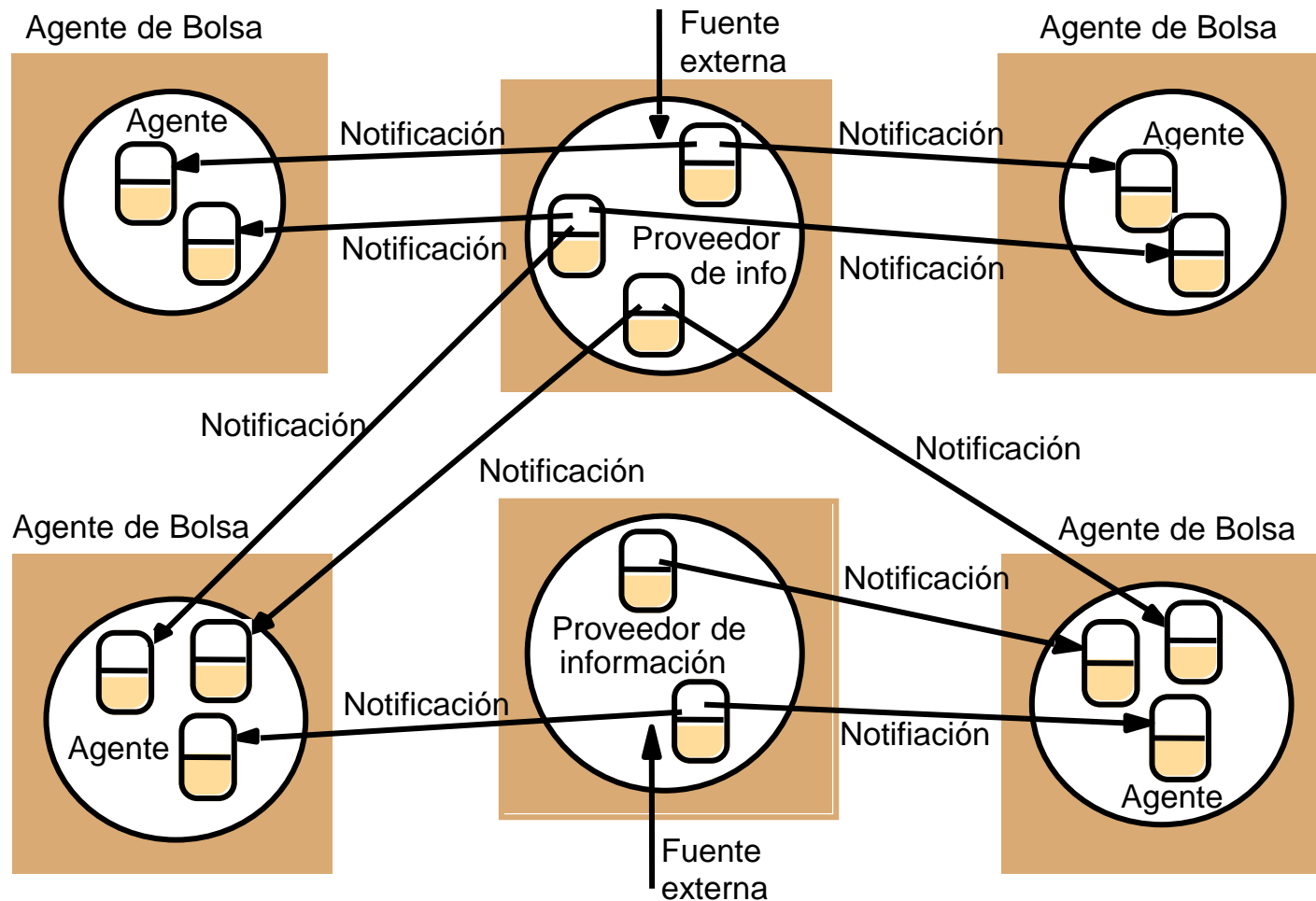


- Los editores publican eventos estructurados a un servicio
- Los suscriptores expresan interés por un tipo particular de eventos
- El servicio casa las suscripciones con los eventos publicados y asegura el envío correcto de notificaciones de eventos

- Un evento se envía a un conjunto de suscriptores



Comunicación publicación-inscripción



Propiedades de la comunicación publicación-inscripción

- **Heterogeneidad:**
 - Facilita que componentes variados trabajen juntos por medio de **interfaces** de **recepción** de **eventos**.



Propiedades de la comunicación publicación-inscripción

- **Asincronía:**
 - Las notificaciones se envían asíncronamente a los suscriptores, así se evita que estén acoplados.



Async, it's a good thing

Comunicación publicación-inscripción

Operaciones:

Publicar(evento)

Anunciar(filtro)

Notificar(evento)

Suscribir(filtro)

CancelarSuscripción(filtro)



Esquemas del filtro



- **Basados en Canales:**
 - Los sucesos se publican en un canal
 - Los suscriptores se suscriben a un canal
- **Basados en asunto o tema:**
 - Es uno de los campos del suceso/evento
 - La suscripción se define con respecto a este dato

```
MessageTemplate.MatchTopic(Topic);  
ACLMessage msg = receive(mt);
```



Esquemas del filtro



Basados en contenido:

- Condición lógica o “Query” sobre los valores de los atributos del evento.

```
MessageTemplate mt = MessageTemplate.and(  
    MessageTemplate.MatchPerformative( ACLMessage.INFORM ),  
    MessageTemplate.MatchSender( new AID( “myAgent”,  
        AID.ISLOCALNAME))) ;
```

```
ACLMessage msg = receive( mt );
```

Basados en tipos:

- Tipos de eventos compatibles con un tipo o subtipo de un filtro (objeto)
- Se pueden integrar elegantemente en lenguajes de programación



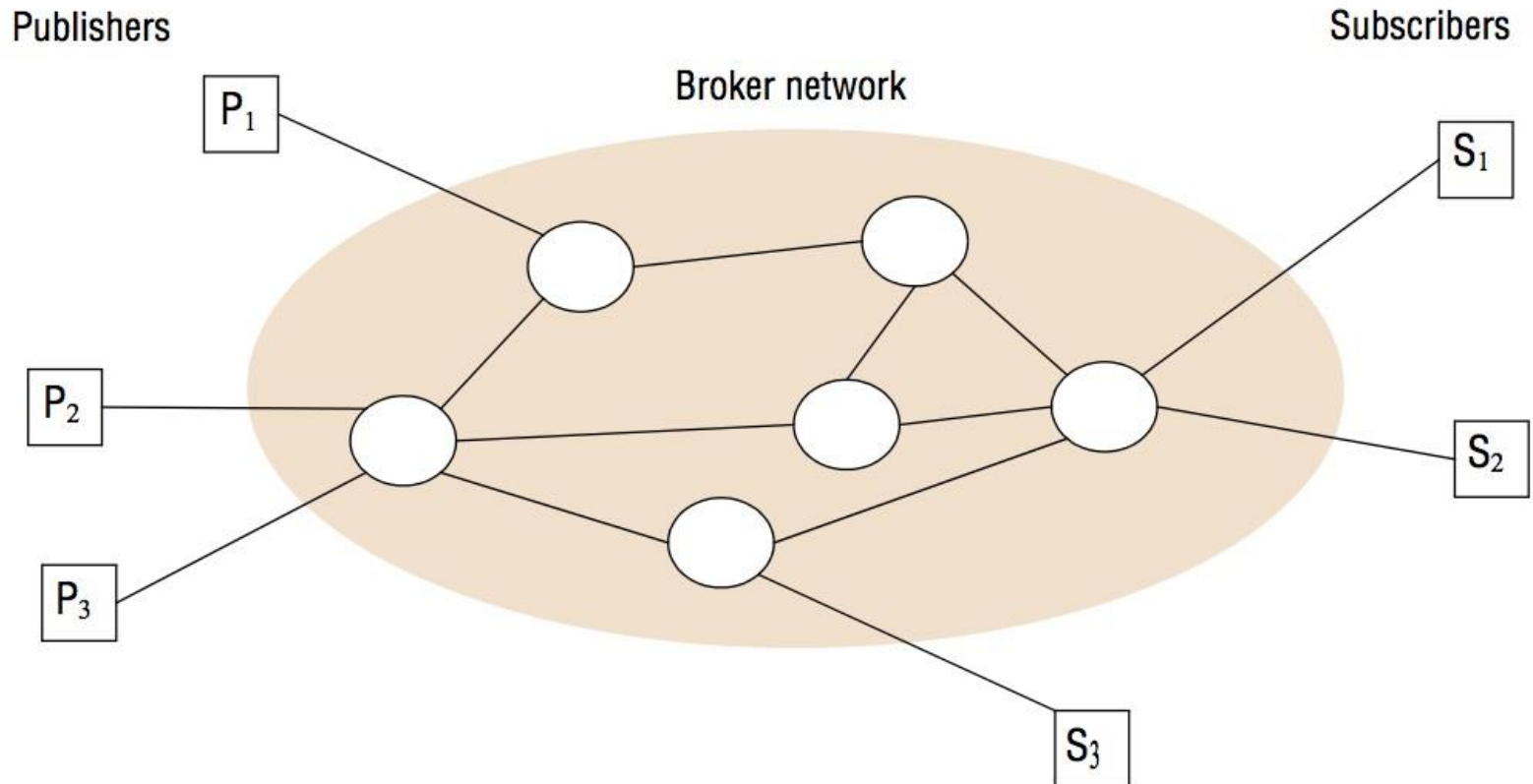
Comunicación publicación-inscripción

Centralizado Vs Distribuido



Comunicación publicación-inscripción

Centralizado Vs **Distribuido**

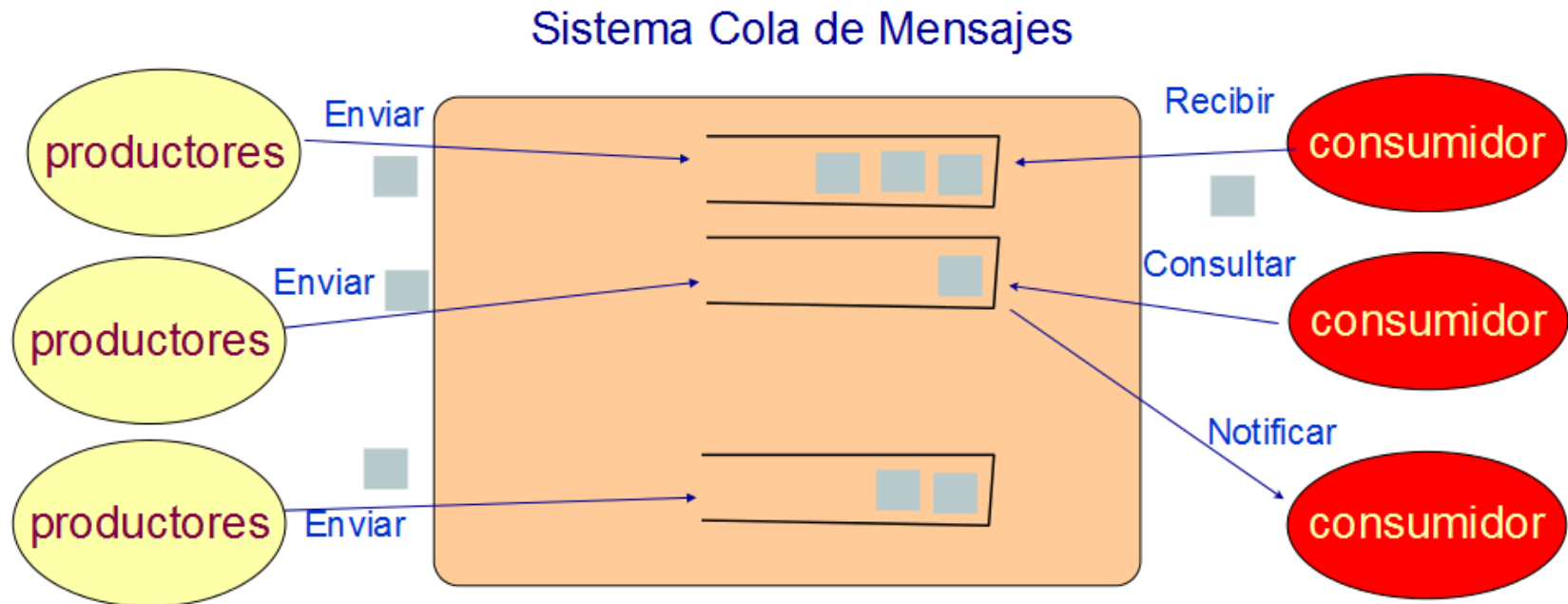




Colas de Mensajes

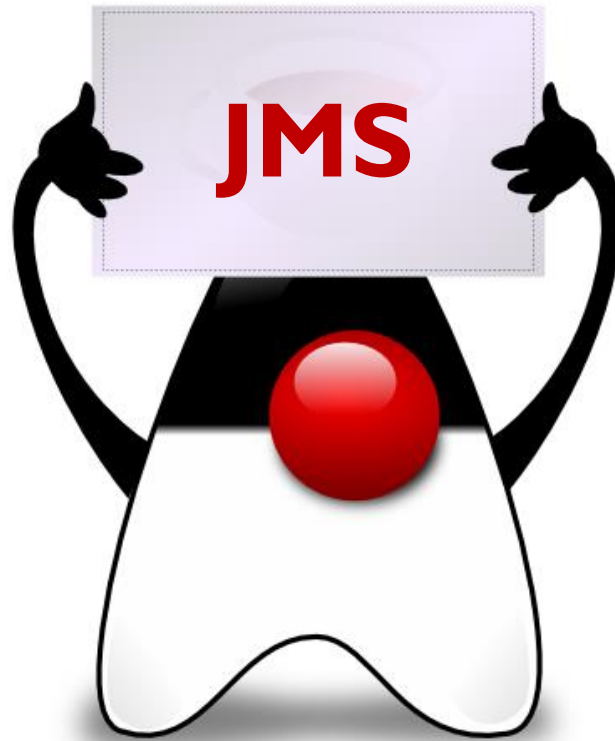
- El concepto de cola de mensajes fundamenta la **indirección**.
- Proporcionan **desacoplamiento espacial y temporal**
- Se usa para **integrar aplicaciones** o como base de **sistemas de procesamiento de transacciones**

Colas de Mensajes



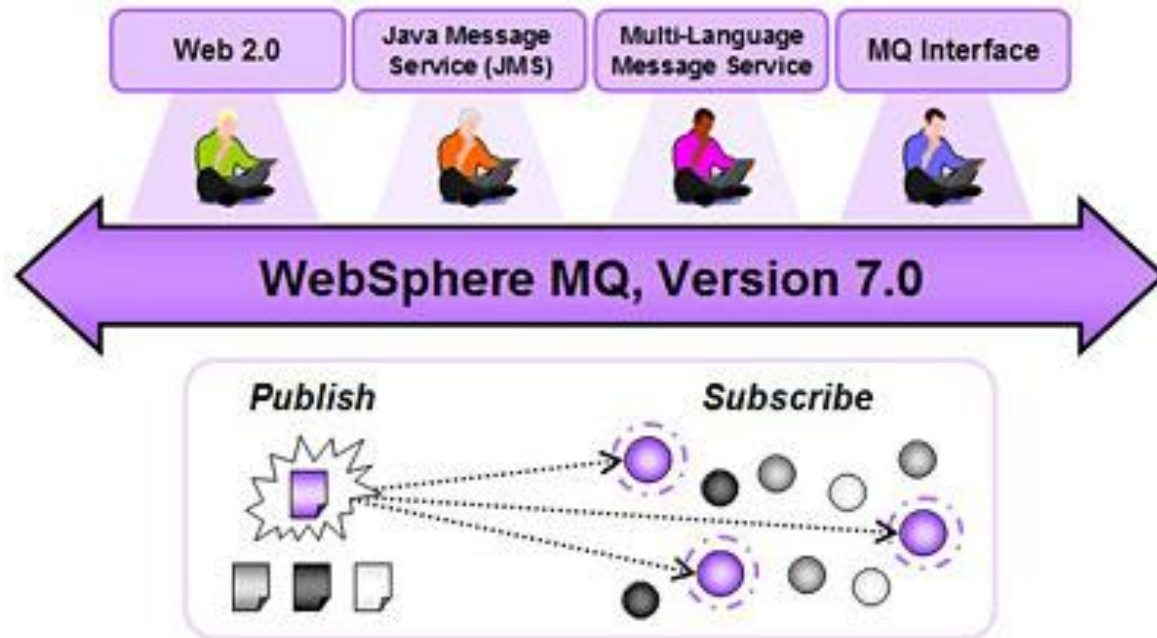
Colas de Mensajes

Java Message Service



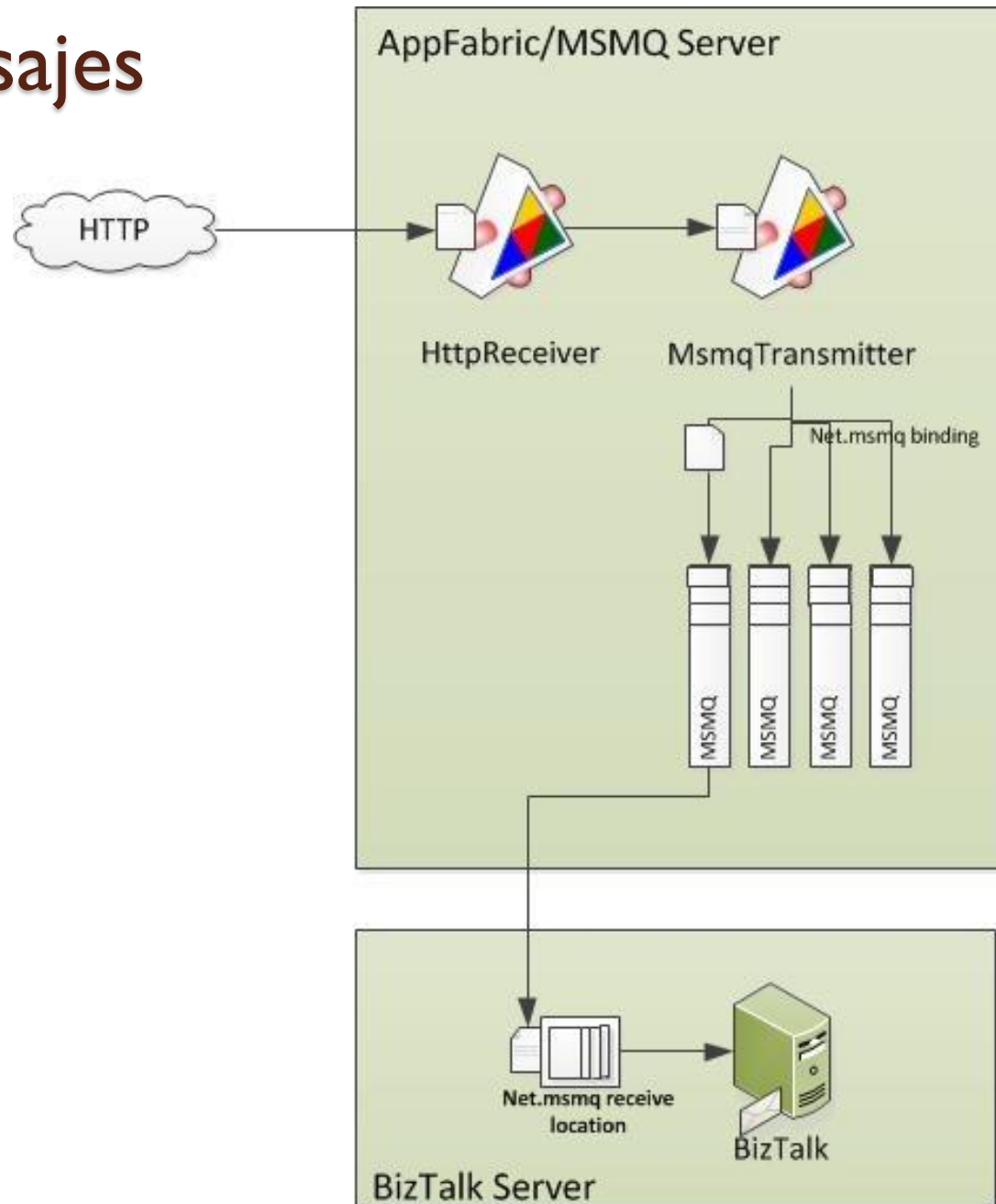
Colas de Mensajes

IBM WebSphere MQ



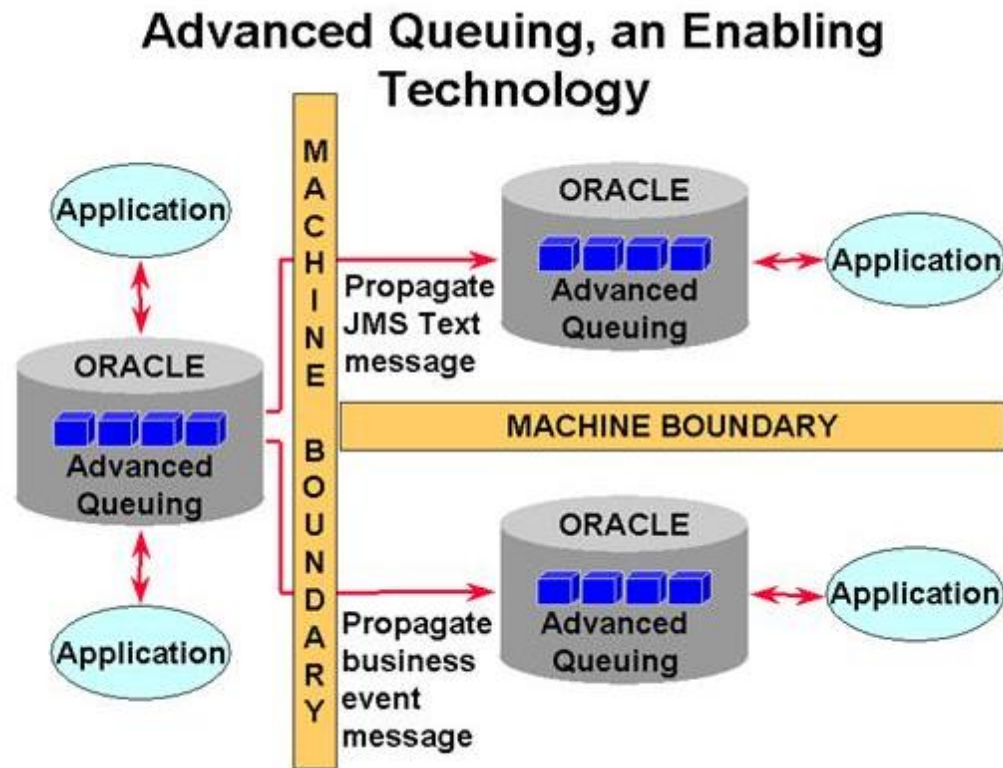
Colas de Mensajes

Microsoft
MSMQ



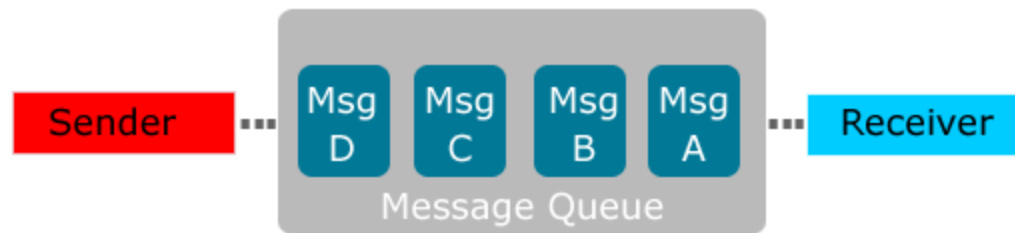
Colas de Mensajes

Oracle Streams **Advanced Queuing**



Colas de Mensajes

- El modelo de programación es sencillo:
 - Los **emisores** mandan **mensajes** a la cola
 - Los **receptores** leen mensajes de la cola



- Tres **estilos de receptores**:
 - Recepción **bloqueante**
 - Recepción **no bloqueante**
 - **Notificación**: se manda un evento cuando llega un mensaje

Colas de Mensajes

fifo



- Las colas tienen un orden **FIFO** o por **prioridades** para la entrega de mensajes.
- Se puede **seleccionar** un **mensaje** por sus características
- Los mensajes tienen un **destino**, **metadatos** y **cuerpo**

Colas de Mensajes



- Los mensajes son **persistentes**
 - Se almacenan indefinidamente, hasta que se leen.



- Se garantiza un envío **fiable**, pero **no cuando** se hace.

Colas de Mensajes

- Funcionalidades adicionales:



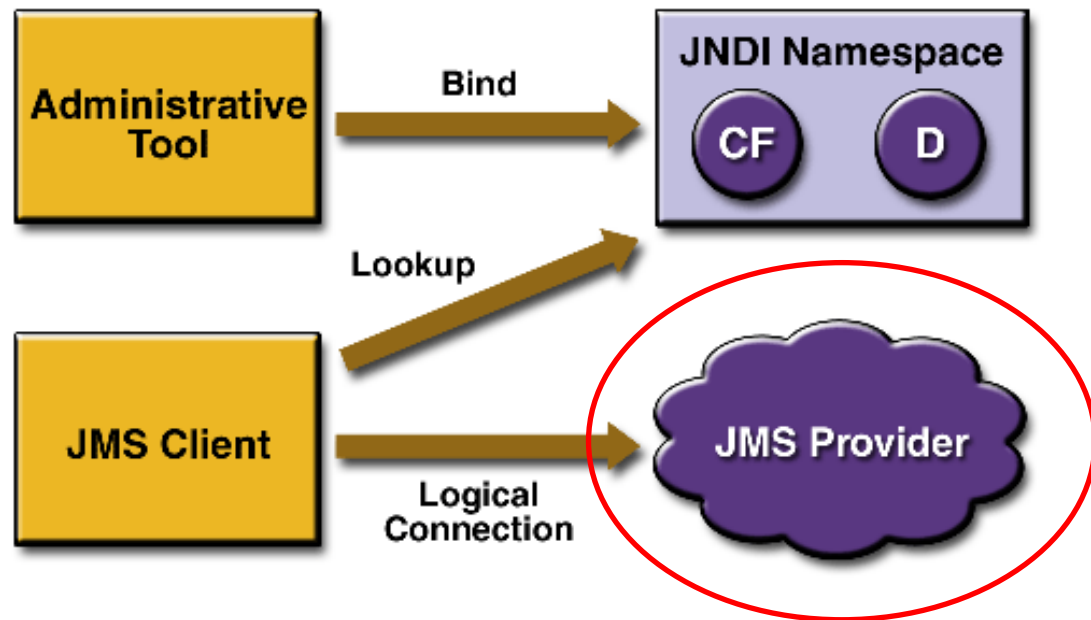
- Los **mensajes** pueden formar parte de una **transacción**: o se realizan todas las acciones o ninguna



- **Transformaciones** de mensajes al llegar a la cola.
- Se pueden realizar **cambios** para adaptar el **formato** de la información

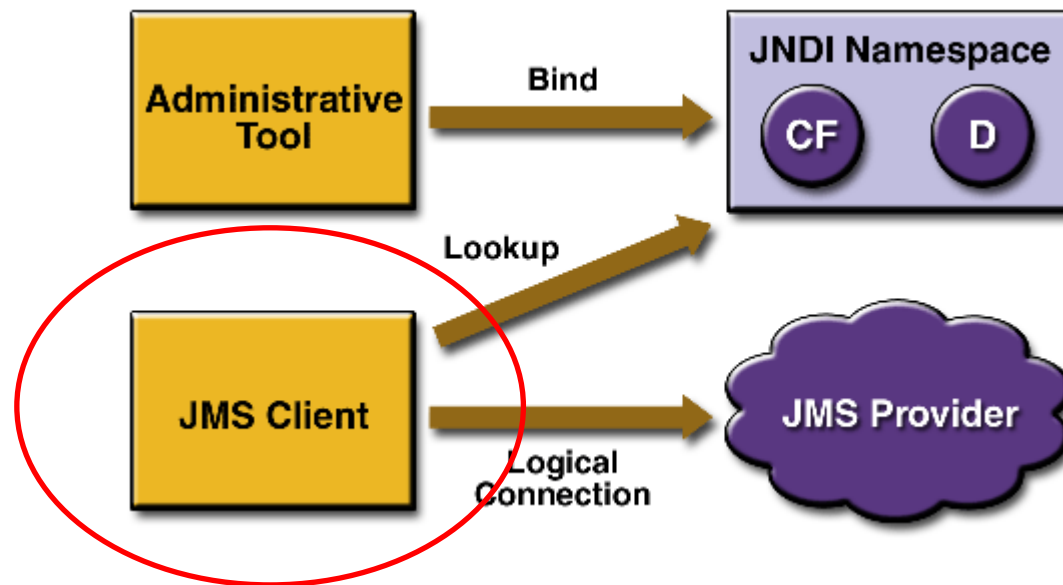
Java Message Service - JMS

- El proveedor JMS implementa la interfaces



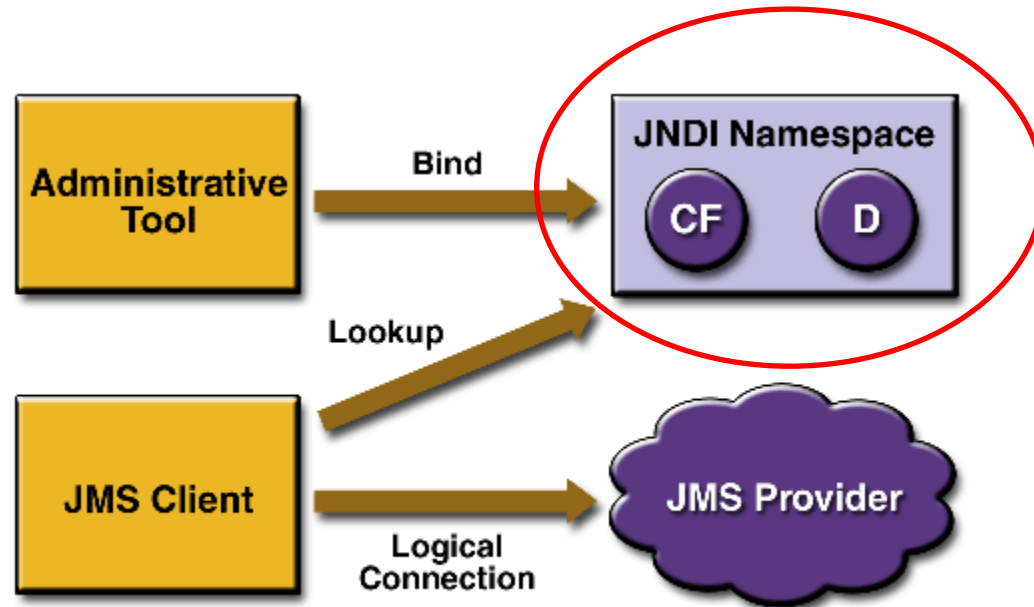
Java Message Service - JMS

- Clientes JMS producen y consumen mensajes



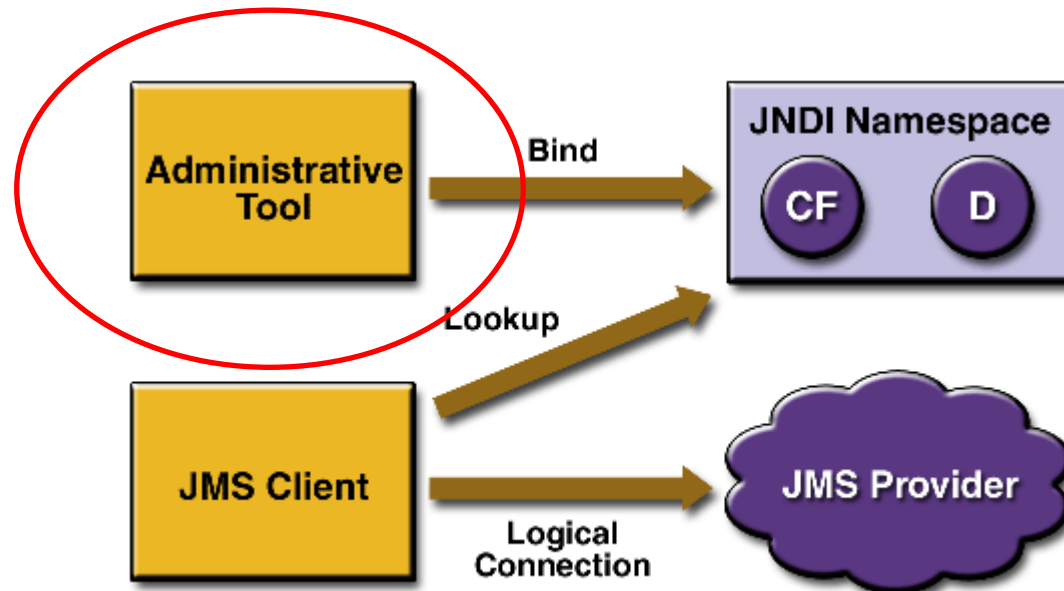
Java Message Service - JMS

- JNDI - Java Naming and Directory Interface



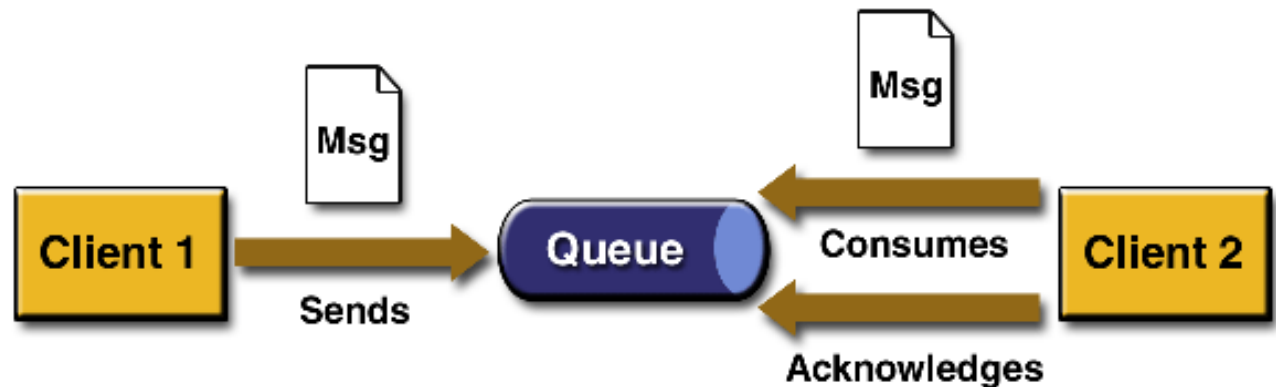
Java Message Service - JMS

- Las herramientas administrativas vinculan los destinos con las fábricas de conexión usando el directorio JNDI



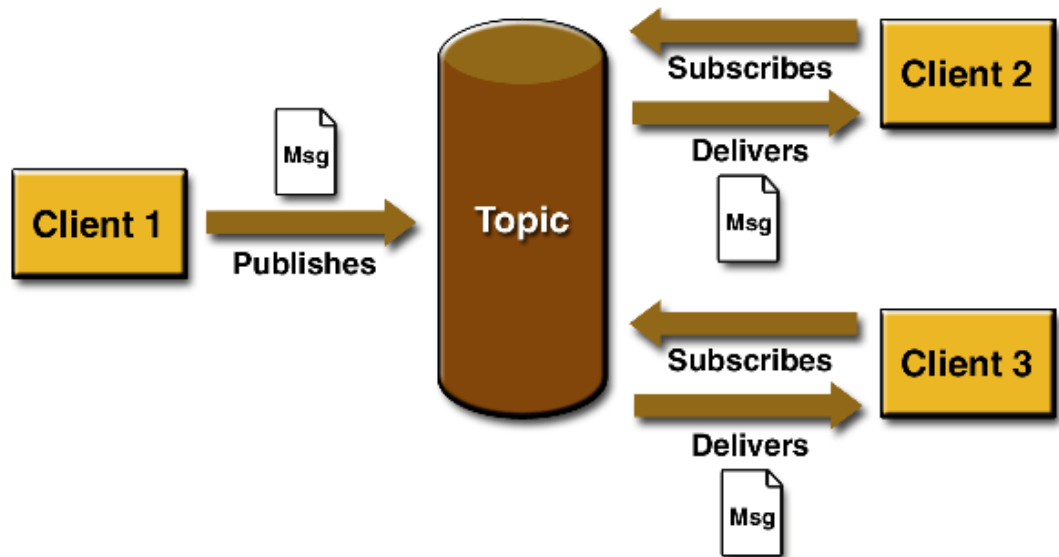
Java Message Service - JMS

- Mensajes **punto a punto**
 - Cada mensaje **solo** tiene un consumidor
 - **No dependencias** de tiempo entre emisor y receptor.
 - El receptor **confirma** la lectura del mensaje

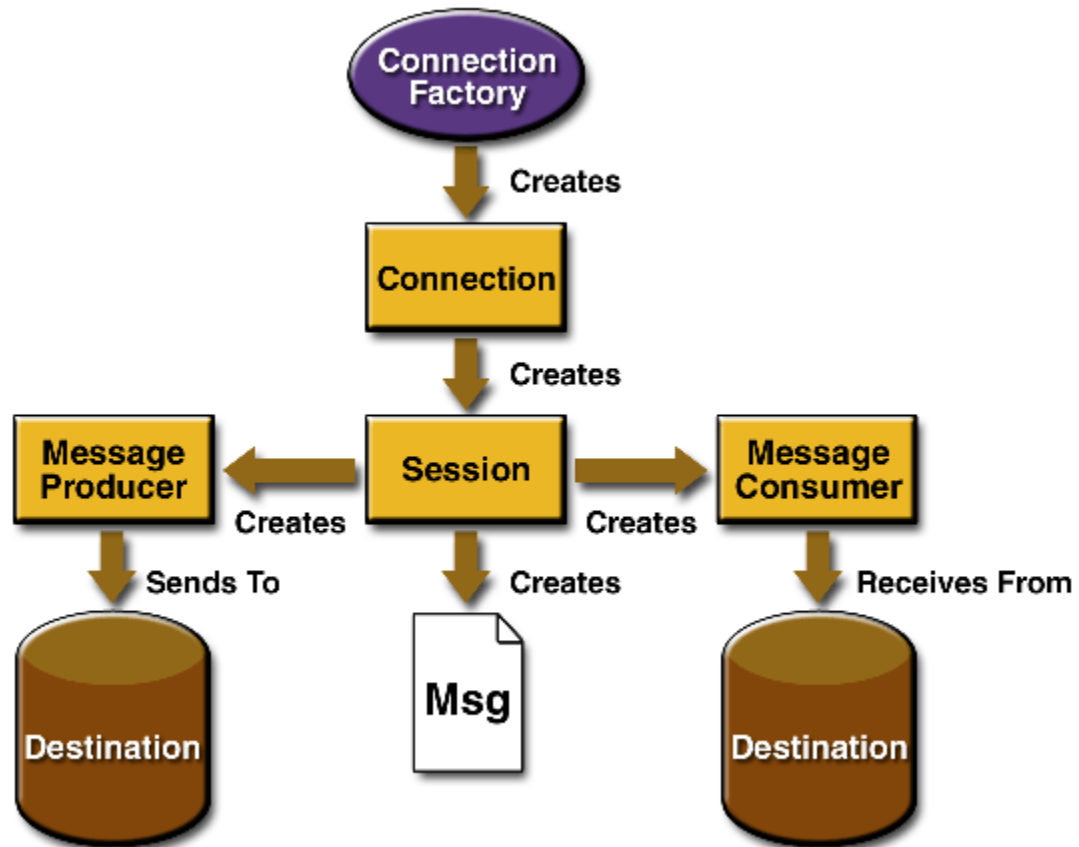


Java Message Service - JMS

- Comunicación publicación-inscripción
 - Cada mensaje puede tener múltiples consumidores
 - Productores y consumidores tienen dependencias de tiempo controladas por la suscripción.



Modelo de Programación JMS



JMS usando ActiveMQ



ACTIVEMQ

Descargar ActiveMQ



Ejecutar ActiveMQ

DIR_INSTALACIÓN/bin/activemq.bat start

Consola de administración

<http://localhost:8161>



Usuario y contraseña

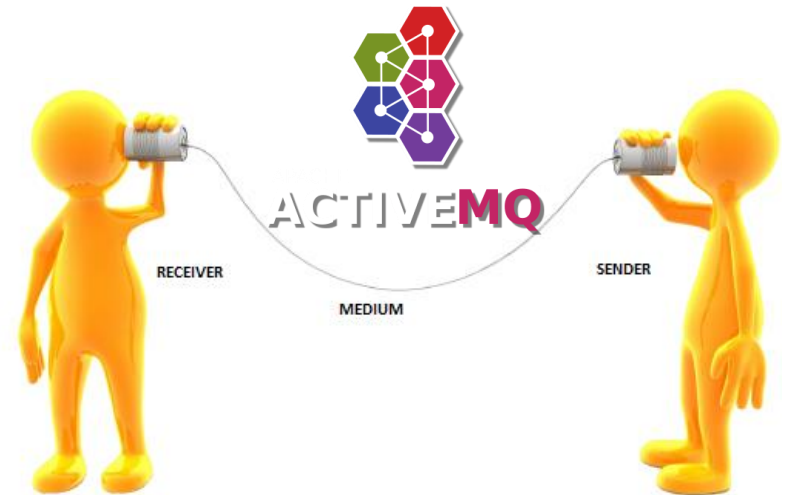
admin admin

Importar librerías

DIR_INSTALACIÓN/activemq-all-X.X.jar

JMS usando ActiveMQ

- MessageSender
- MessageReceiver
- MessageQueueBrowser
- AsynchMessReceiver



JMSQueueAMQ

JMSTopicAMQ

JMS usando ActiveMQ



ACTIVEMQ

- ¿Cómo se establecería la comunicación de los procesos usando **Topics – Publicación/Inscripción?**
- ¿Qué diferencias hay?

Java Message Service – JMS

Práctica de laboratorio de Topics

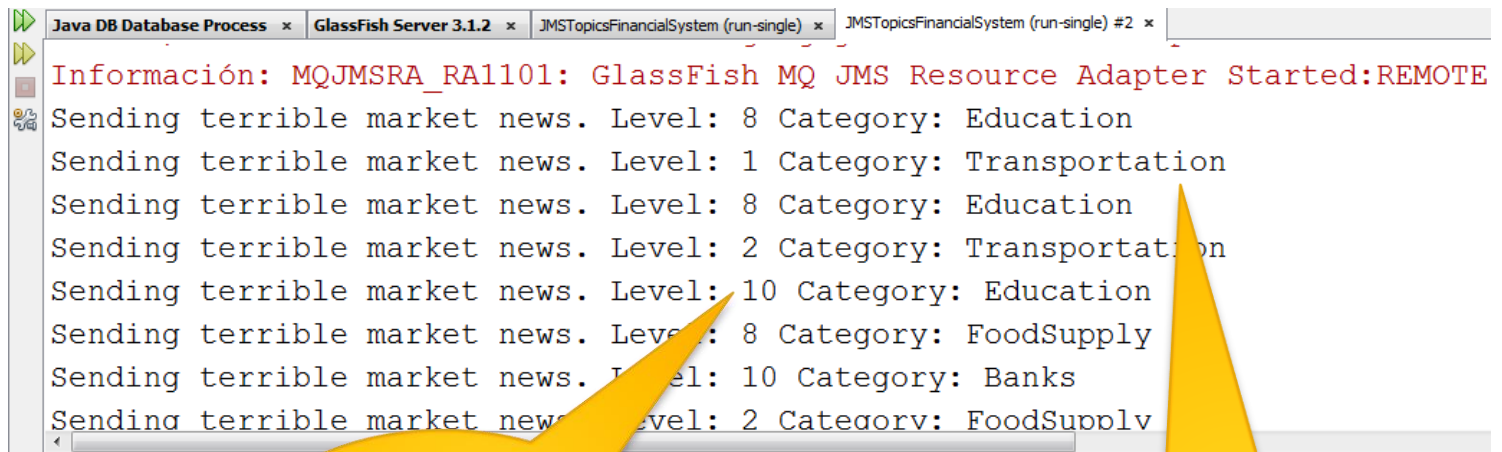
Sistema Financiero



Java Message Service – JMS

Práctica de laboratorio de Topics

Proveedor de Información del mercado



```
Java DB Database Process x GlassFish Server 3.1.2 x JMSTopicsFinancialSystem (run-single) x JMSTopicsFinancialSystem (run-single) #2 x
Información: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter Started:REMOTE
Sending terrible market news. Level: 8 Category: Education
Sending terrible market news. Level: 1 Category: Transportation
Sending terrible market news. Level: 8 Category: Education
Sending terrible market news. Level: 2 Category: Transportation
Sending terrible market news. Level: 10 Category: Education
Sending terrible market news. Level: 8 Category: FoodSupply
Sending terrible market news. Level: 10 Category: Banks
Sending terrible market news. Level: 2 Category: FoodSupply
```

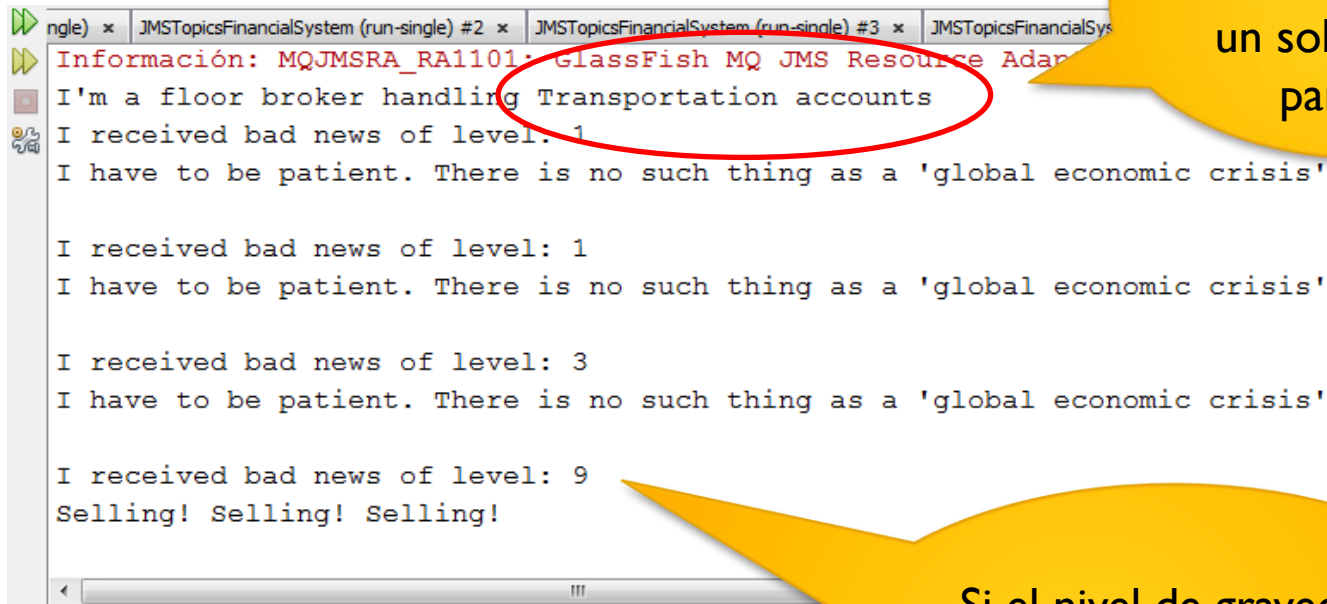
Envía
información
negativa a los
agentes de
bolsa

La información es
distribuida por
categorías (Topics)

Java Message Service – JMS

Práctica de laboratorio de Topics

Agente de bolsa



```
ngle) x JMSTopicsFinancialSystem (run-single) #2 x JMSTopicsFinancialSystem (run-single) #3 x JMSTopicsFinancialSys
Información: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter
I'm a floor broker handling Transportation accounts
I received bad news of level: 1
I have to be patient. There is no such thing as a 'global economic crisis'

I received bad news of level: 1
I have to be patient. There is no such thing as a 'global economic crisis'

I received bad news of level: 3
I have to be patient. There is no such thing as a 'global economic crisis'

I received bad news of level: 9
Selling! Selling! Selling!
```

Recibe
información de
un solo topic en
particular

Si el nivel de gravedad
de la información es
mayor a 5, entonces se
venden acciones

Java Message Service – JMS

Práctica de laboratorio de Topics

- Hay un solo proveedor de información

- Provee información de **5 categorías**

- Telecommunications
 - Banks
 - Transportation
 - FoodSupply
 - Education



- Se debe de poder indicar un **número de noticias a transmitir.**

- Cuando todas las noticias se han enviado, el proveedor de información debe de enviar una notificación de **fin de sesión financiera** a todos los agentes de bolsa.

- La categoría de la noticia y su nivel debe de ser determinada **aleatoriamente.**

Java Message Service – JMS

Práctica de laboratorio de Topics

- Puede haber N **agentes de bolsa**
 - Un agente de bolsa solo se puede registrar a **una categoría** a la vez. La cual será determinada aleatoriamente.
 - Debe **terminar** su proceso cuando reciba una notificación de fin de sesión financiera.



Java Message Service – JMS


Práctica de laboratorio



Java Message Service – JMS

Práctica de laboratorio



- Objeto **Papa**
 - Identificador
 - Tiempo (aleatorio) para caerse
- Objeto **Jugador**
 - Avienta la papa a otro jugador vía una cola de mensajes
 - Verifica en una cola de mensajes si otro jugador le ha aventado una papa
 - Si le aventaron una papa, verifica si se le ha caído (tiempo=0), si sí, pierde el juego, sino regresa la papa vía una cola de mensajes restándole un tiempo.
- **Juego**
 - 2 jugadores, cada jugador avienta su papa con diferente tiempo para caerse. El primer jugador al que se le caiga la papa pierde y termina el juego.
- Algunas **cuestiones técnicas** a considerar
 - `session.createTextMessage()`  `session.createObjectMessage()`
 - `setText()` `setObject()`
 - `ActiveMQConnectionFactory connection = new ActiveMQConnectionFactory();`
 - `connection.setTrustAllPackages(true);`