Messaging Services

Messaging

Messaging Systems are middleware components that facilitate passing messages between two applications.

These messages can be anything but usually contain business transactions or to inform applications that some event has occurred.

The middleware can also contain network and routing information to efficiently send a message.

Advantages of Messaging

Heterogeneous Integration

A messaging system is built independent of platform. Each platform builds interfaces to the messaging framework much like JDBC drivers are used today to communicate with databases.

Reduce System Bottlenecks

Instead of having a single system send a message to many clients synchronously one after the other, the sender sends a single message and the messaging system handles distribution.

Increase Scalability

In the same way messaging reduce bottlenecks it can also increase throughput. When queue times build up additional message listeners are created to handle messages concurrently.

Increase End User Productivity

Clients no longer wait for long running operations to complete. Messaging allows notification of work completed.

Examples of Messaging Middleware

IBM Websphere MQ

SonicMQ

MSMQ

TIBCO Rendezvous

ActiveMQ (Open source)

Architectures

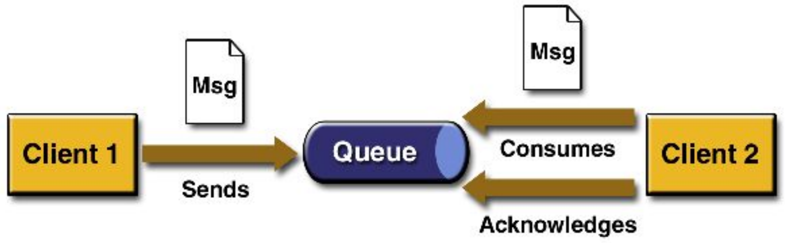
- In a centralized architecture a message server or cluster is responsible for queuing and sending messages.
- In decentralized architecture IP multicast is used at the network level.

Messaging Models

Point to Point

Message is consumed by one and only one client.

Supports both sync and async operations.



Publish and Subscribe

- Messages are published to channels called *topics*.
- Message can be consumed by multiple clients.
- Messages are pushed (no polling).

P2P API

- *QueueConnectionFactory*
- *Queue*
- *QueueConnection*
- *QueueSession*
- *Message*
- *Queue Sender*
- *Queue Receiver*

Publish-Subscribe API

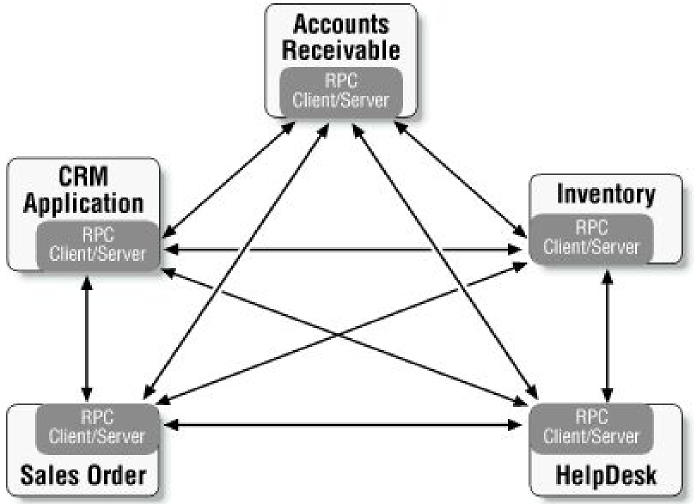
- *TopicConnectionFactory*
- *Topic*
- *TopicConnection*
- *TopicSession*
- *Message*
- *TopicPublisher*
- *TopicSubscriber*

Messaging vs RPC

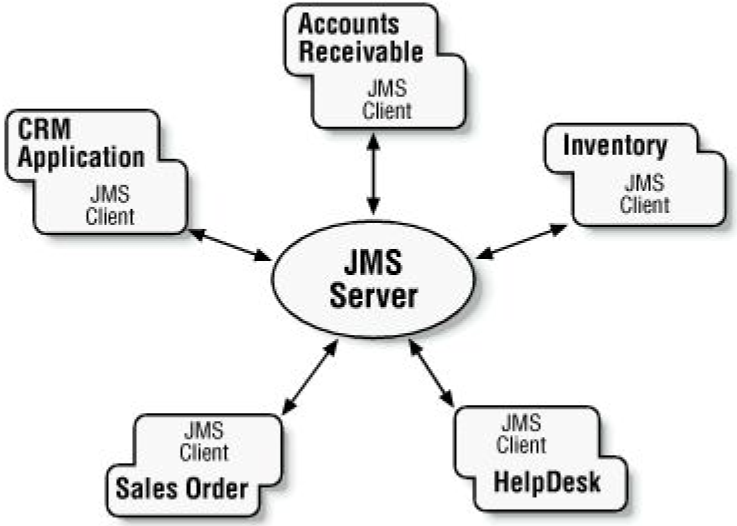
Tightly coupled nature of RPC created highly interdependent systems.

Generally calls are synchronous.

Case Study: RPC



Case Study: Java Messaging



JNDI

Stands for Java Naming and Directory Interface, it is a standard API for accessing resources on your system by name. Resources include LDAP, NDS, CORBA, and even JMS.