**JMS**

**Advantages of JMS over other technologies:**

FTP: OS overhead, No trace of the transfer(cost, effort,time),security

SFTP: OS overhead, No trace of the transfer(cost, effort,time)

Web services : Synchronous.. wait for response till the next req is processed

Messaging : Asynchronous

RMI(Remote Method Invocation) : You should know the actual method that resides on the other system.

JMS : Limited to Java

**Messaging size:**

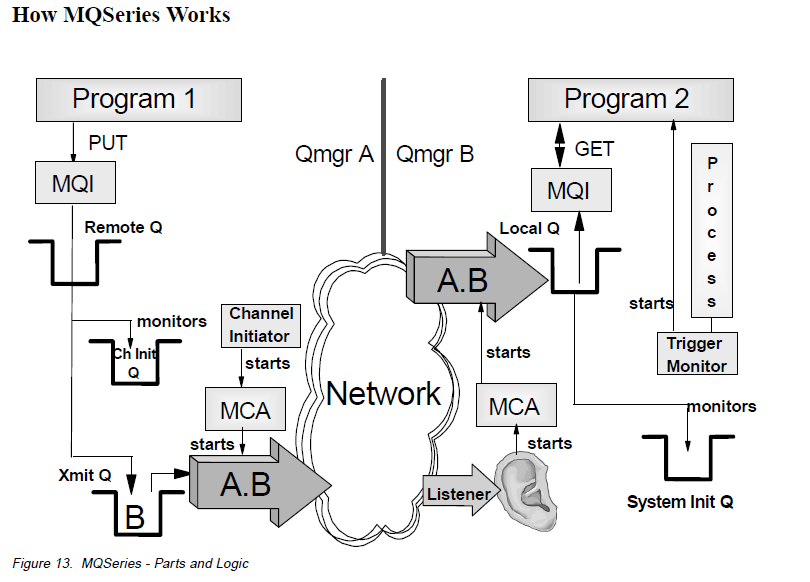
Message size : 4 MB std, but we can send upto 100 MB. If the message doesn’t fit into the queue, we can send it as segments(Segmentation). On the receiver end, they can be de-segmented based on the co-relation ids.

**Two phase commit:**

If there are 5 to 6 in a transaction, the transaction will be commited only if all the 5 or 6 steps are successful, otherwise it will be rolled back.

**Late Fan-out:**

You can send a message to more than one destination queue with one MQPUT call. This is done with a dynamic *distribution list.* A distribution list can be a file that is read at the time an application starts. It can be modified any time. It contains a list of queue names and the queue managers that own them. A message sent to multiple queues belonging to the same queue manager is sent over the network only once and so reduces network traffic. The receiving queue manager replicates the messages and puts them into the destination queues. This function is called *late fan-out.*

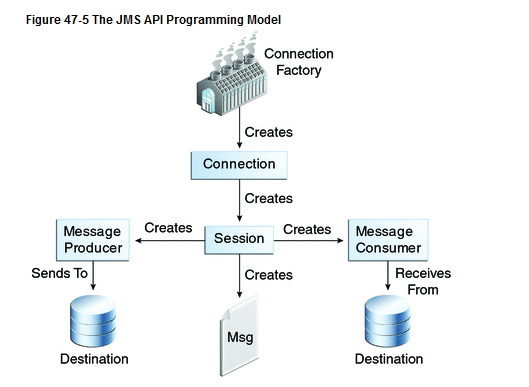
**

Step 1: Program 1 puts the message in Remote Q using MQPUT. It doesn’t know whether the Queue is remote or local. Only the Queue Manager knows the queue Q is not in Queue Manager A, but its in Queue Manager B.

Step 2: As soon as the message is in Q, two things occur. First a copy of this message will be put in transmit Queue (Xmit Q). Next Channel Initiator who is monitoring the Remote Q will see the message and create a trigger message in Channel Initiator Q[Ch Init Q]. As soon as it sees the trigger message in Ch Init Q, Channel Initiator starts the MCA[Message Channel Agent]. MCA transmits the message from A 🡪 B

Step 3: On the Listener side, once the message is seen, MCA starts and the message is put in Local Q. Once the message is in local Q, Trigger monitor will create a trigger message in System Init Q and it starts the Program 2 Process. This process will get the message from local Q using MQGET.

Note : Messages you see in Ch Init Q and System Init Q are not actual messages but they are trigger messages

**

The basic building blocks of a JMS application are:

* Administered objects: connection factories[A **connection factory** is the object a client uses to create a connection to a provider.] and destinations[A **destination** is the object a client uses to specify the target of messages it produces and the source of messages it consumes.]
* Connections[A **connection** encapsulates a virtual connection with a JMS provider.]
* Sessions[**session** is a single-threaded context for producing and consuming messages]
* Message producers[Created by Session to send messages]
* Message consumers[Created by Session to receive messages]
* Messages

**Messaging Models:**

Point to point

publisher subscriber model : Publisher to Topics[Like a Bulltin board], then to all subscribers.

Types of Queues: Persistent, Non-Persistent

Types of Acknowledgement : Auto, Client

Throttling – Controlling the inflow & outflow of messages

Message Broker – On top of Queueing infrastructure, it acts as transformation engine. It does Routing. You can queries using esql.

Message priorities – 0 to 9. 0 is the lowest; 9 is the highest

<http://docs.oracle.com/javaee/6/tutorial/doc/bnceh.html>