Sagar Bhatt(14bit152)

Messaging Architecture used for long standing transaction.

Today web services are widely used due to SOAP, WS Messaging and WSS Security protocol standards. They also tend to provide synchronous mode of messaging communication between the client and the server. Almost of the modern technologies and vendors can call and consume web services easily.

Although web services have an advantage of having synchronous communication and easier way to be consumed they **lack asynchronous and one way messaging**. I am not sure about other technologies but WCF has a one-way asynchronous communication mechanism.

However in online transactions synchronous request-response messaging is essential and usage of web services is inevitable. Due to asynchronous processing nature of queues they should be avoided. Queues do have a request-response pattern as given below but the latency, client implementation of session management and reliability management (handshakes of messages) is an overhead.

In this post we will briefly look at request-response synchronous messaging via Queues and SOA technologies. I hope the conclusion would be interesting.

Sagar Bhatt(14bit152)

Reliability, atomicity and retry submit messages in

Web services

Reliability, retrying mechanism and atomicity is not possible in web services since they tend to be synchronous and stateless in nature. Once the request is processed there is no record of the message. This can be fragile and the messages can get lost. For example if the host process is restarted or got crashed and the messages were in-flight and were being processed by the web services. The transaction in the web services will be inconsistent in this case.

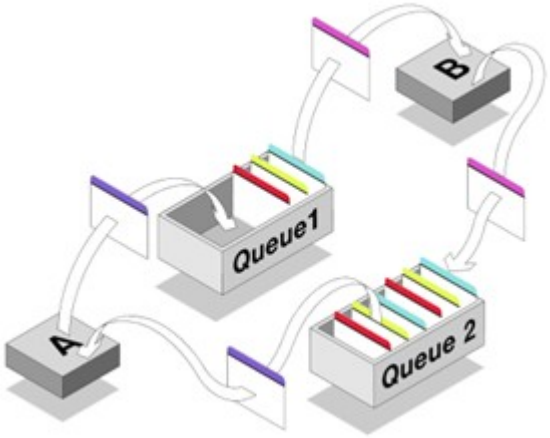
Queues

There are a lot of vendors for queues but we'll focus on **MSMQ**, **IBM MQ Series** and **JMS queues** request-response pattern. All of the queue technologies expose APIs where clients can access the queues and put messages in the queues for processing. This requires clients to handle session states, implement reliability mechanism by checking ACK/NAK messages etc. The advantages and disadvantages of queues are same regardless of different vendors and technologies. The queues can be transactional, allows messages to persist and retrying of message processing can be implemented.

The queues can be of two types.

1- Point-to-point: It means that each queue will be connected to only one subscriber/destination.

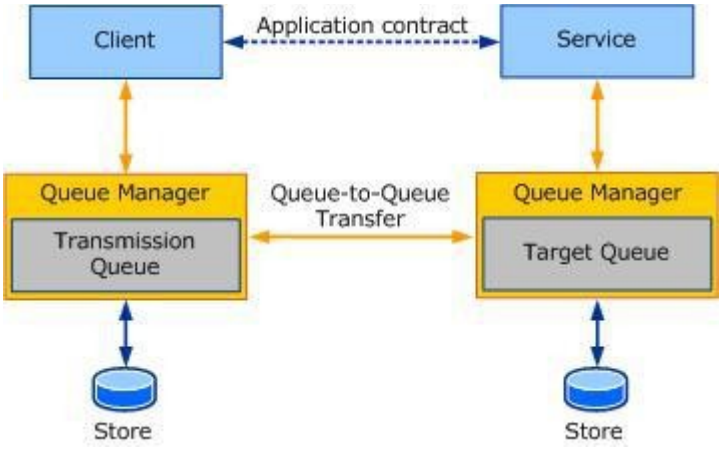
2- Publish-subscribe: It means that a queue can be connected to multiple subscribers but the publisher has to publish the message by topic and each subscriber gets the copy of message.



WCF- Reliable messaging via MSMQ

WCF offers the capability of reliable messaging via **WS-RM** protocol. It uses reliable sessions and MSMQ to queue messages. The **disadvantage** is latency, session state handling from client application and queues communication which is a different implementation from a client application than just calling a SOAP web service. However it will serve the purpose of reliability (messages are persisted) and

retrying. The following figure illustrates the structure of a WCF service and client using MSMQ as a transport.



WCF service and client using MSMQ as a transport

JMS Request-Response messaging via Queues

I will not go deep into JMS architecture and capabilities you can read here for more information. I am focusing on request/response synchronous pattern implementation via Queues.

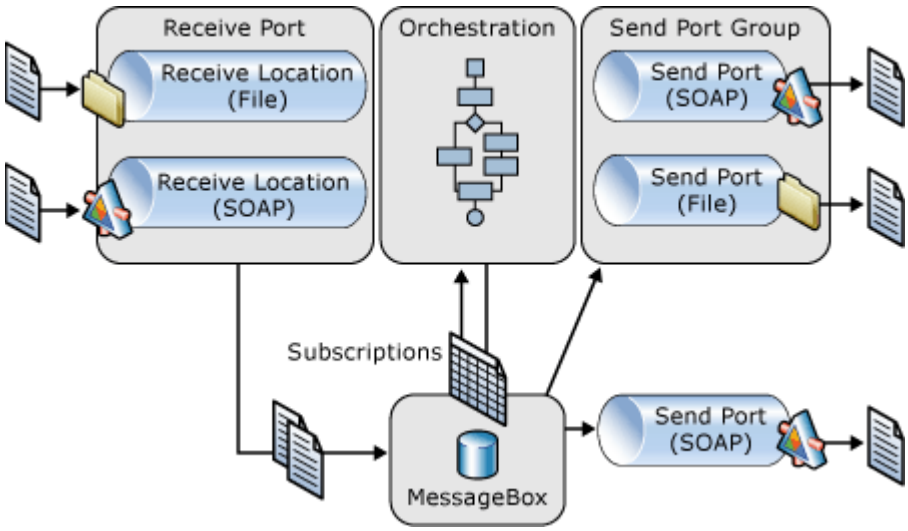
In the below diagram you can see that the publisher has to put message in the queue. The advantage is reliability but it comes at the cost of latency.

SOA products:

There are a number of SOA products in the market worth to mention is IBM Message broker, Oracle middleware fusion, TIBCO and Microsoft BizTalk Server. This genes of messaging system has a different set of features than the classical queuing systems like Orchestrations, Business Rules Engines, Business

Activity Monitoring, Enterprise Service Bus, Dynamic Routing, Transformation and Error Recovery capabilities. They are far more capable than the classical queues.

In this post we are looking at the request-response message pattern between web services, queues and SOA products. We are concerned about how the message flows between these three set of message processing and their reliability, resilience, error recovery, retrying and atomicity. Lets take a look how message flows between SOA products in request-response pattern then we can compare with Queues and Web services. The message flow is same in all the SOA products we'll look at the message flow in BizTalk server.



SOAP Request-Response Message flow along with an extra File message processing

In BizTalk server the client requests a SOAP message which is accepted by the receive location and then published into the Message Box. The subscribers (orchestrations or send ports) picks up the message. If

there are multiple subscribers they will get a copy of the message.

If the subscriber is orchestration it will be the processing unit of the message. Orchestrations has persistence points so the state of the transaction will also be persisted to the store. When the orchestration has finished processing it will again publish the message to the database to be picked up by another subscriber. In SOAP request-response pattern it will be routed back to the receive location.

If the subscriber is a send port then the message is probably meant to be sent to another system outside the middleware server. If there is any transmission failure in the message it will be stored in the message box awaiting for actions to be taken (retry submission or suspend).