Amazon messaging POC

- Amazon message encryption
- Amazon SNS SQS POC
- Amazon SQS POC

This is a proof of concept to use Amazon SNS/SQS for application messaging.

SQS is for point to point messages. SNS is used as a topic which can send to multiple endpoints (in this case I am only interested in SQS queues, but could be HTTPS, etc).

If you send to SQS you get the message, if you send to SNS-SQS you get a JSON wrapped message. Also, if you want copies of messages, you could add a personal subscription to the SNS topic. For these reasons, you should probably always only send SNS-SQS even for point to point. To switch from only SQS to SNS-SQS would require programming on both sides unless there is a generic framework for it.

Note: there is security at Amazon to specify who is allowed to subscribe to a queue and who is allowed to subscribe to the topic, and who can publish to a queue or topic. The protocol is HTTPS web services and simple encryption signatures. There are a lot of clients available to make things even easier. http://aws.amazon.com/tools/ (sdk in java, .net, python, php, ruby, ios, android, node.js)

Amazon recently released "long-polling" which allows recipients to SQS to check for messages, and block for 20 seconds until a message is received. This makes receiving messages much more like JMS.

Receiving messages in SQS is a two step process so the messaging service knows a crash didnt happen before the message was processed. You receive the message, process it, and then when done, delete it. During that time window the message will not be delivered to another receiver.

Messages to SQS are not received in order. However, there is a design pattern to address this. If the sender has an index in the messages (or a timestamp), then the receiver can get all the messages available, make sure all the indexes are accounted for, or wait some time (10-20 seconds?) get all messages again, put them in order, process them and delete them. This assumes that the max time for a message to be delivered by SQS is ~10 seconds... the receiver could just keep blocking until you get the message index you are expecting.

Pricing

If you have an application which sends 10k messages per day to one receiver using SNS/SQS... then...

The sender pays 10k*31*0.06/100000 = $0.18 (18 cents)

The receiver needs to connect every 20 seconds: 3*60*24*31 = 134k. Then 10k messages to receive, and 10k messages to delete, for 31 days = 620k. You wont have this many requests since when you get a message, then you connect again, and not again until the next 20 seconds. You might get multiple messages from one request. So maybe it is 700k total.

This is (700k) * 0.01 / 10000 = (70 cents). Total price: $11 / year.

Not sure how this compares to hosting your own ActiveMQ. You have time to set it up, pay for servers / DB, backups, etc. And as you add more endpoints, you will need to scale it up. Not sure if you would have a support contract for ActiveMQ with a vendor.