****

**《软件体系结构与设计》实习报告**

**题目： 第1次上机实验**

班级序号： 111171

学生姓名： 张洋铭

任课教师： 尚建嘎

**中国地质大学信息工程学院软件工程系**

**2019年9月**

**一、实验概况**

**实验时间**：2019年9月18（周三）晚上18:30-21:30，共4课时

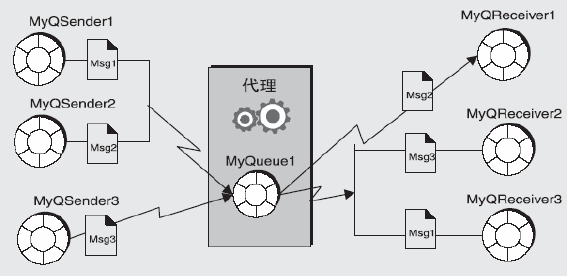
**实验地点**：未来城校区公教2-503

**实验目的**：掌握开发、测试、发布、调用进程间通信的基本方法、工具和流程，理解独立构件体系结构基本原理、结构和特点。掌握使用当今主流云平台来构建独立构件风格软件的相关开发技能。

**背景及要求：**

现今，越来越多的企业面临着各种各样的数据集成和系统整合的系统需求，掌握开发、测试、发布、调用进程间通信的基本方法、工具和流程显得很重要。在这样的系统需求之下，RPC中间件技术也应运而生，但由于采用RPC同步处理技术，在性能、健壮性、可扩展性上都存在诸多缺点。而基于消息的异步处理模型则采用非阻塞的调用特性，发送者将消息发送给消息服务器，消息服务器在合适的时候再将消息转发给接收者；发送和接收是异步的，发送者无需等待。使用消息中间件作为一个中间层的软件，掌握使用云计算技术来构建独立构件风格的相关技能。

1、基于AWS SQS（亚马逊云）或阿里云等简单队列服务的消息中间件，使用Java，C#或者其他语言分别编写一个发送程序和接收程序（构建两个进程或者程序，一个用于发送消息--发到云端队列，一个用于接收消息--从云端队列订阅下来），实现“点对点”的进程间通信功能。



**提示与思考**：

1. AWS 相关基本操作，在另外一个文档中，里面有基本的如何获取AWS key，以及如何建立AWS连接。
2. 前端页面简洁明了，用户体验较好，重点在后台通信机制。
3. 这种消息队列服务是基础性的，AWS 作为商业云平台提供了针对SQS的高可用性解决方案。如果你基于Kafka构建消息队列服务，如何确保其高可用性？
4. 相关链接：

AWS .NET API,你可以在该链接找到你想要的类及相关方法：

<https://docs.aws.amazon.com/sdkfornet/v3/apidocs/Index.html>

Java API:

<https://docs.aws.amazon.com/zh_cn/AWSJavaSDK/latest/javadoc/index.html>

SQS官方文档链接：

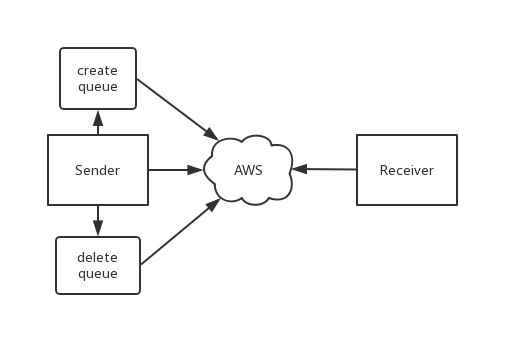
<https://docs.aws.amazon.com/sqs/index.html#lang/zh_cn>

**二、实验设计**

**2.1 系统需求**

构建两个进程或者程序，一个用于发送消息--发到云端队列，一个用于接收消息--从云端队列订阅下来，实现“点对点”的进程间通信功能。

**2.2 架构设计**



**2.3 接口设计**

Sender.py

Receiver.py

**三、实验过程**

**3.1 软件实现**

通过参考AWS SQS控制台中云端队列的操作流程。构建发送端Sender，完成在AWS云服务器中FIFO队列的创建和移除，并完成消息在指定群组中的发送。构建接收端Receiver，每30s从云端队列的指定群组中接收消息，并显示。

**3.2 实验环境**

AWS SQS

PyCharm

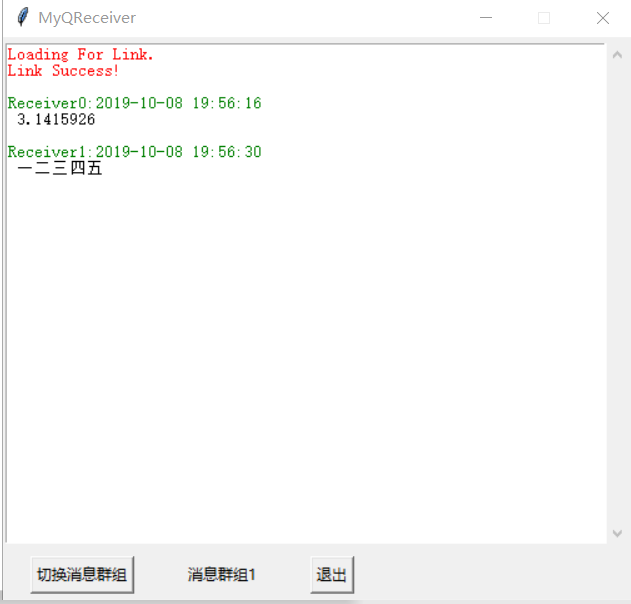
**3.3 实验步骤**

同时运行发送端和接收端文件，此时发送端完成在云端创建指定名字的FIFO队列。在发送端窗口中，发送按钮在按下后将消息框中的消息发送到当前群组的云端队列中；切换群组按钮，在按下后切换发送群组；退出按钮按下后，关闭发送端窗口，并且移除FIFO队列。在接收端窗口中，每30s从云端检查当前群组中的消息，若接收到消息，则显示在消息框中；切换群组按钮按下后，切换当前检查和接收的群组号；退出按钮按下后，关闭接收端窗口。

**四、实验评价**

**4.1 实验结果**





**4.2 结果分析**

基本完成了发送端和接收端的构建，在UI界面上还有优化的空间，并没有做到AWS SNS功能。

**五、总结**

在本次实习过程中，第一次接触了云，通过云基本上实现了“点对点”的进程间通信功能，虽然只是对AWS SQS控制台的仿照式程序，但还是感觉收获了很多。遗憾的是没有做到AWS SNS功能，且AWS凭证还需要在一段时间后手动从网页上获取，但还是很不容易的。

正如老师上课讲的，掌握使用当今主流云平台来构建独立构件风格软件的相关开发技能是很重要的，通过云来实现进程间点对点通信只是一个开始，我们要学的还有很多。