## MessageQueue

### Jms

#### 基本概念

JMS(JAVA Message Service)是java的消息服务，JMS的客户端之间可以通过JMS服务进行异步的消息传输

#### 消息模式

○ Point-to-Point(P2P)

○ Publish/Subscribe(Pub/Sub)

即点对点和发布订阅模型

##### P2P



特点：每个消息只有一个消费者（Consumer）(即一旦被消费，消息就不再在消息队列中)

发送者和接收者之间在时间上没有依赖性，也就是说当发送者发送了消息之后，不管接收者有没有正在运行，它不会影响到消息被发送到队列(持久性)

接收者在成功接收消息之后需向队列应答成功

##### Pub/Sub



特点：每个消息可以有多个消费者

发布者和订阅者之间有时间上的依赖性。针对某个主题（Topic）的订阅者，它必须创建一个订阅者之后，才能消费发布者的消息，而且为了消费消息，订阅者必须保持运行的状态(类似于广播)

##### durable subscription

### Activemq

Activemq是jms的一种具体实现

下载：<http://activemq.apache.org/download.html>

解压运行之后输入<http://127.0.0.1:8161/admin/>可进入后台管理页面，页面提示要密码，都是admin

Python环境测试：pip install paho-mqtt

#### 支持的protocol

AMQP协议：

即Advanced Message Queuing Protocol,一个提供统一消息服务的应用层标准高级消息队列协议,是应用层协议的一个开放标准,为面向消息的中间件设计。基于此协议的客户端与消息中间件可传递消息，并不受客户端/中间件不同产品，不同开发语言等条件的限制。

MQTT协议：

MQTT（Message Queuing Telemetry Transport，消息队列遥测传输）是IBM开发的一个即时通讯协议，有可能成为物联网的重要组成部分。该协议支持所有平台，几乎可以把所有联网物品和外部连接起来，被用来当做传感器和致动器（比如通过Twitter让房屋联网）的通信协议。

OpenWire协议：OpenWire协议在网上没有对应的介绍，似乎是activeMQ自己定义的一种协议，官方网站对其的介绍如下：

OpenWire is our cross language Wire Protocol to allow native access to ActiveMQ from a number of different languages and platforms. The Java OpenWire transport is the default transport in ActiveMQ 4.x or later. For other languages see the following...

stomp协议：STOMP，Streaming Text Orientated Message Protocol，是流文本定向消息协议，是一种为MOM(Message Oriented Middleware，面向消息的中间件)设计的简单文本协议。

ws协议：即websocket协议，基于h5

mqtt协议的实现最快，stomp的协议实现最慢。

## Zeromq

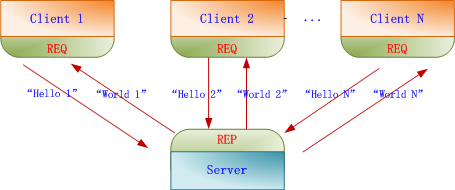
### 概念

ØMQ （也拼写作ZeroMQ，0MQ或ZMQ)是一个为可伸缩的分布式或并发应用程序设计的高性能异步消息库。它提供一个消息队列, 但是与面向消息的中间件不同，ZeroMQ的运行不需要专门的消息代理（message broker）

### 消息模式

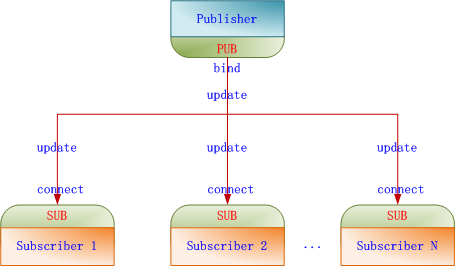
#### REQ-REP

俗称请求回应模式，由请求端发起请求，然后等待回应端应答。一个请求必须对应一个回应，从请求端的角度来看是发-收配对，从回应端的角度是收-发对。跟一对一结对模型的区别在于请求端可以是1~N个。该模型主要用于远程调用及任务分配等。Echo服务就是这种经典模型的应用



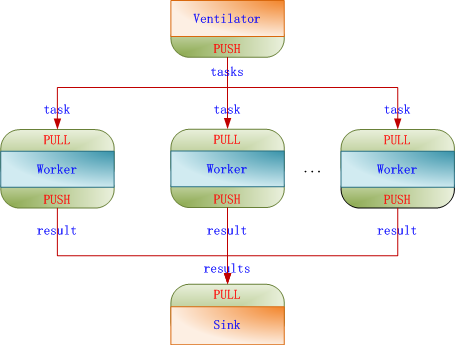
#### Publish-Subscribe

发布端单向分发数据，且不关心是否把全部信息发送给订阅端。如果发布端开始发布信息时，订阅端尚未连接上来，则这些信息会被直接丢弃。订阅端未连接导致信息丢失的问题，可以通过与请求回应模型组合来解决。订阅端只负责接收，而不能反馈，且在订阅端消费速度慢于发布端的情况下，会在订阅端堆积数据。该模型主要用于数据分发。天气预报、微博明星粉丝可以应用这种经典模型



#### PUSH-PULL

Server端作为Push端，而Client端作为Pull端，如果有多个Client端同时连接到Server端，则Server端会在内部做一个负载均衡，采用平均分配的算法，将所有消息均衡发布到Client端上。与发布订阅模型相比，推拉模型在没有消费者的情况下，发布的消息不会被消耗掉；在消费者能力不够的情况下，能够提供多消费者并行消费解决方案。该模型主要用于多任务并行（类似于发牌）



#### DEALER-ROUTER

代理模式（类似于一个异步的接线员，是一个N对接线员对M的链接拓扑关系）

优点是，如果客户端是同型的，服务端也是同型的，都可以动态加入或退出

