**分布式模式之Broker模式**

**问题来源：**

创建一个游戏系统，其将运行在互联网的环境中。客户端通过WWW服务或特定的客户端软件连接到游戏服务器，随着流量的增加，系统不断的膨胀，最终后台数据、业务逻辑被分布式的部署。然而相比中心化的系统，复杂度被无可避免的增大了，该如何降低各个组件之间的耦合度。

**挑战：**

需要保证可伸缩性、可维护性、可更新性，需要将服务划分为各个相对独立的组件，组件被分布式的部署，它们之间通过进程间通信方式实现交互。服务的增加、删除、改变都应该被支持。理想情况，以开发者的角度看，集中化的系统和分布式的系统在中心逻辑上没有什么不同。为实现这个目标：

可以远程的访问服务，而对于访问者，服务的位置应该是透明的。

提供服务的组件可以增加、删除、改变，而且这些在运行期同样应该被支持。

访问服务的客户端不应该关心服务的实现细节。

**解决方案：**

引入一个Broker组件，解耦客户端和服务端。服务端注册自己到Broker，通过暴露接口的方式允许客户端接入服务。客户端是通过Broker发送请求的，Broker转发请求道服务端，并将请求的结果或异常回发给客户端。通过使用Broker模式，应用可以通过发送消息访问远程的服务。

这一架构模式允许动态的改变、添加、删除服务端，从客户端的角度，这些都是透明的。

**结构：**

Broker模式定义了6中类：Client，Server，Client\_Proxy，Server\_Proxy，Broker，Bridge。

**Server** **：**

责任：处理特定领域的问题，实现服务的细节，注册自己到Broker，处理请求并返回结果或异常。

协作类：Server\_Proxy，Broker

**Client：**

Client是需要访问远程服务的应用程序，为此，Client发送请求到Broker，并从Broker上接收响应或异常。Client和Server只是逻辑上相关而已，实际上Client并不知道Server的确切位置。

责任：1. 实现用户端功能，2. 发送请求到Broker，3. 接收相应和异常。

协作类：Broker，Client\_Proxy

**Broker：**

Broker可以被看成消息转发器。Broker也负责一些控制和管理操作。它能够定位服务端的位置，若发生异常，能够将异常捕获传给Client。Broker需要提供注册服务的接口给Server。如果请求来自其他的Broker，本地的Broker需要转发请求并最终将结果或异常回应给相应的远程Broker。Broker提供的服务和name service非常相像（如DNS、LDAP）。

责任：1. 注册服务。2. 提供服务API。3. 转发消息。4. 容错处理。5. 与其他Broker的交互。6。 定位服务。

协作类：Client\_Proxy,Server\_Proxy,Bridge

**Client\_Proxy：**

连系Client和Broker，这一层保证了通讯的透明性，使Client调用远程服务就像调用本地的服务一样。

责任：1. 封装特定的系统调用。2. 封装通讯的参数、控制信息等。

协作类：Client,Broker。

**Server\_Proxy：**

Server\_proxy是与Client\_Proxy相对应的，它接受请求，解包消息，解析出参数并调用服务的实现接口。

责任：1. 封装特定的系统调用。2. 封装通讯的参数、控制信息等。3. 调用server的服务接口。

协作类：Server,Broker。

**Bridge：**

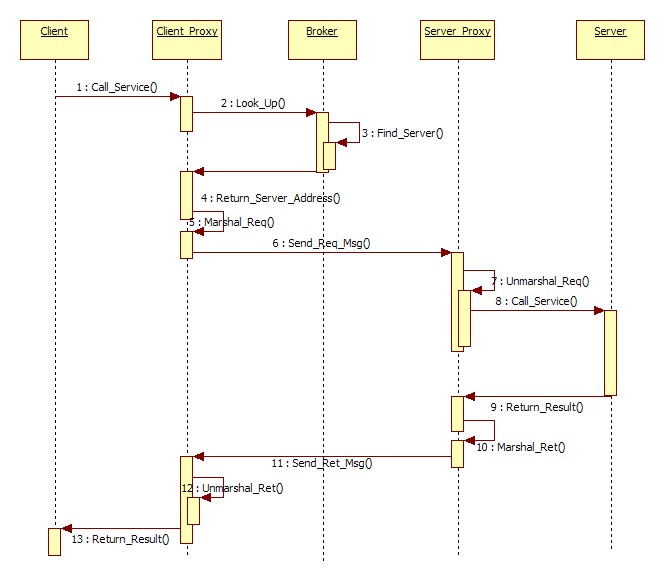
Bridge用来连接各个Broker，一般这个组件是可选的。当系统是发杂的网络组成时，有可能需要这一角色。

责任：1. 封装特定的网络特性。2. 传递Broker之间的通讯。

协作类：Broker。

**应用场景一：**

直接通讯方式。Client和Server相互理解他们之间的通讯协议。Broker主要完成Client和Server之间的握手。之后所有的消息、异常都是由Client与Server直接交互。（想象DNS）。简单对象交互如图：

[](http://blog.chinaunix.net/attachment/201101/16/23093301_1295175004flGk.jpg)

**应用场景二：**

Broker启动，完成自身的初始化，之后进入事件循环，等待消息到来。

Server启动，首先执行自身的初始化，然后注册自己到Broker。

Broker接收Server的注册请求，将其加入到可使用服务的列表，并回应Ack给Server。

Server接收Ack，进入事件监听循环，等待消息到来。

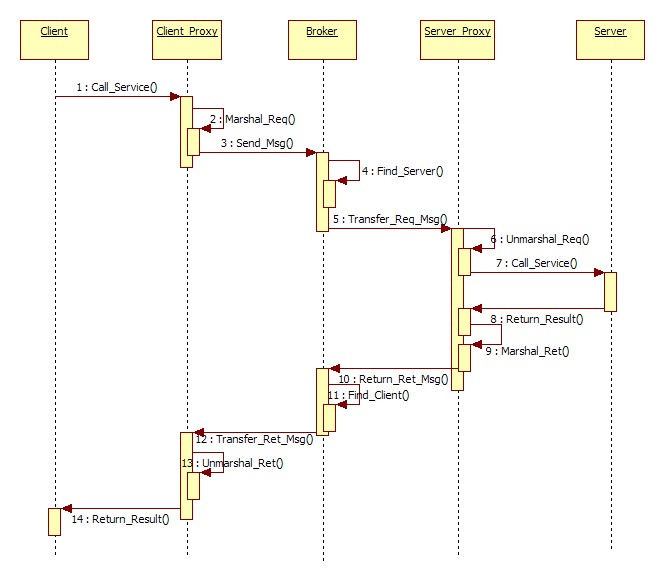
Client调用远程服务对象的方法，Client\_Proxy封装消息请其发送给Broker。

Broker查询可使用的Server，将请求转发给Server。

Server\_Proxy解析消息，分离出参数和控制信息，并调用特定的Server实现接口。Server处理完的结果通过Server\_proxy封装成消息转发到Server。

Broker将相应消息转发给正确的Client\_Proxy，Client受到响应继续其他逻辑。

简单对象交互如图：

[](http://blog.chinaunix.net/attachment/201101/16/23093301_12951751188mhH.jpg)

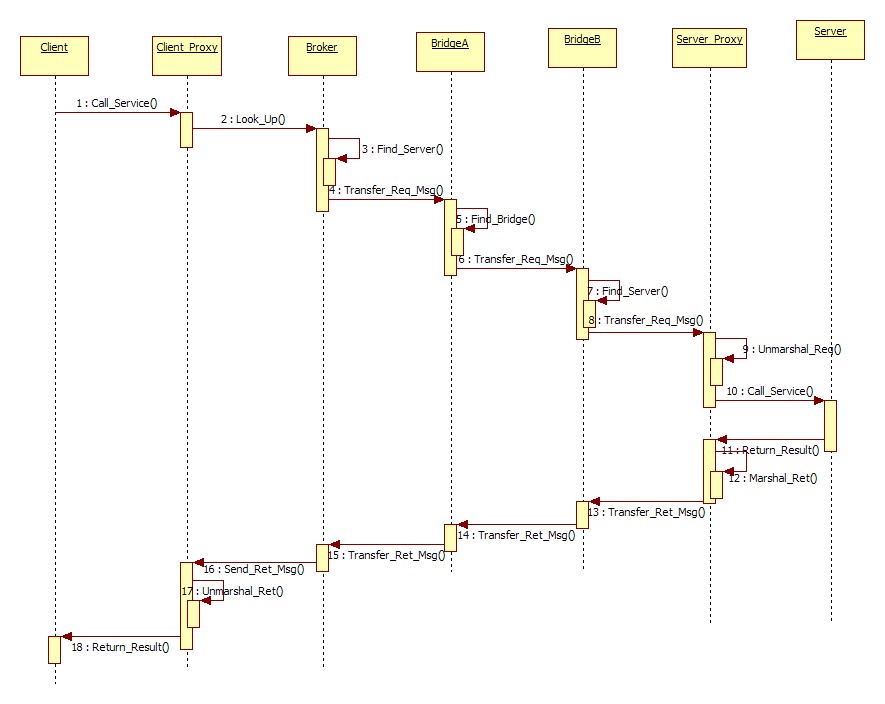
**应用场景三：**

Broker A接收到请求，交由Server处理，但是发现该Server位于其他的网络节点。

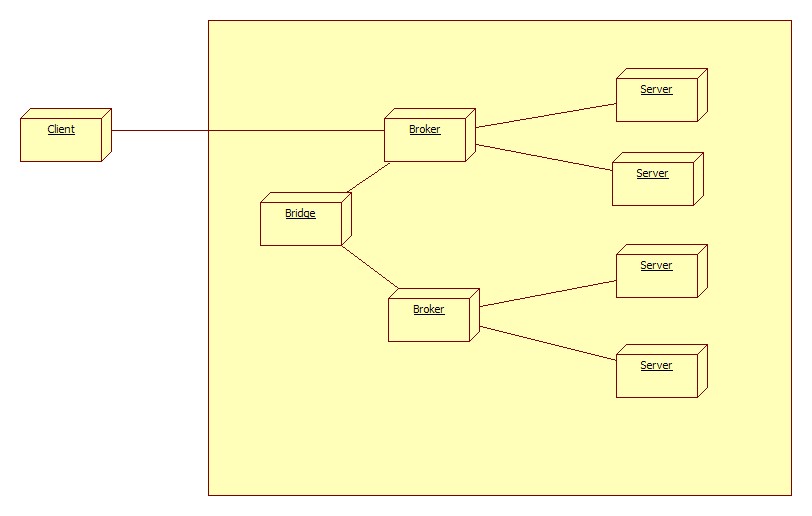
Broker A将请求转发给Bridge A，Bridge A将请求进行必要的格式化，传送给Bridge B。

Bridge B将请求进行必要的格式化，转化成Broker B可以理解的格式，并转发给Broker B。Broker B执行场景二中的过程，处理的结果按如上逆序返回。

简单对象交互如图：

[](http://blog.chinaunix.net/attachment/201101/16/23093301_12951751690Sws.jpg)

**部署示意图：**

**[](http://blog.chinaunix.net/attachment/201101/16/23093301_1295175265jHj8.jpg)**

**总结：**

**优点：**

**1. 服务的位置透明性。**

**2. 组件的可变性及扩展性。由于Server是注册到Broker上的，所以Server可以动态的增加、删除、改变。**

**3. Broker之间可交互。**

**4. 可重用性。**

**5. 由于组件的耦合度较小，调试和测试的工作也是可控的。**

**缺点:**

**1. 效率；增加了一层Broker的消息转发，效率有所降低。**

**2. 容错能力必须要特别考虑。**

**3. 调试和测试的工作加大。**