命名式服务部署

开发说明

[文档副标题]

目录

[1 服务端 2](#_Toc471772457)

[1.1 接口实现（IServer） 2](#_Toc471772458)

[1.1.1 发布服务 2](#_Toc471772459)

[2 客户端 2](#_Toc471772460)

[2.1 使用服务 2](#_Toc471772461)

[2.2 负载均衡 3](#_Toc471772462)

[2.2.1 算法 3](#_Toc471772463)

[2.2.2 情况 3](#_Toc471772464)

[3 缓存 3](#_Toc471772465)

[3.1 服务端数据缓存 3](#_Toc471772466)

[4 轻量消息实时传递 3](#_Toc471772467)

[4.1 架设服务端 3](#_Toc471772468)

[5 通讯 3](#_Toc471772469)

[5.1 通讯方式 3](#_Toc471772470)

[5.2 通讯设计 4](#_Toc471772471)

[5.2.1 通讯实现 4](#_Toc471772472)

[6 下一步修改 4](#_Toc471772473)

[6.1 负载均衡 4](#_Toc471772474)

[6.2 通讯 4](#_Toc471772475)

命名式服务部署

# 服务端

## 接口实现（IServer）

服务端实现该接口，则将作为服务发布；

### 发布服务

继承IServer的类则是服务；实现一个实例，ServiceRegistry.AddServers方法将实例添加注册，同时按照格式：服务名称：-h IP –p 端口 –t 通讯类型 组成URL传入，传入服务启动初始化的参数，则启动了服务

例如：

DataCenter.*call*=**new** ProcessData();

IServer obj=**new** RecData();

String url="Test:-h 127.0.0.1 -p 4444 -t udp";

HashMap<String, String> hamp=**new** HashMap<String, String>();

ServiceRegistry.*AddServers*(obj, url, hamp);

# 客户端

客户端使用服务端功能

## 使用服务

客户端通过客户端代理，返回服务端接口使用；即可向服务端传递数据

例如：

StringBuilder error=**new** StringBuilder();

IPoxyObj proxy= ProxyClient.*CastObj*("Test", error);

**byte**[]bytes=new byte[10];

proxy.SetData(bytes);

## 负载均衡

### 算法

使用HASH一致性算法

### 情况

当服务端出现多个同名的服务时，客户端视为同一个服务多个部署，自动启用负载均衡

# 缓存

## 服务端数据缓存

考虑服务端数据问题，正在研究服务端缓存方式，由以前的内存队列优化，改成了本地数据库缓存，已经写了一些数据封装使用，任然需要测试；主要是Sqlite, BerkeleyDB,Redis等；提供数据库连接池

# 轻量消息实时传递

## 架设服务端

为满足一般回话式通讯，提供简单的服务通讯回话及开发聊天室式的程序，添加了xmpp协议客户端；该客户端当前还没有启用；服务端已经有成熟的软件，建议方案Openfire服务端

# 通讯

程序开发考虑所有的内容都进行隔离

## 通讯方式

当前测试使用的TCP,UDP

## 通讯设计

顶部封装通用接口，由服务端在发布时确认自己的通讯方式，客户端只要有对应的通讯方式则可以与服务端建立通讯

### 通讯实现

实现接口IDDS\_Protocol；并用@ProtocolType给通讯加上一个名称；根据该名称创建对应的通讯实例；

其它第三方通讯，例如ZMQ,DDS等成熟组件也可以直接封装作为通讯方式，自己可以灵活实现定义

# 下一步修改

## 负载均衡

现在的负载均衡是自动使用的，而且是放在客户端，这样可能会有问题；下一步是隔离出来，通过配置方式实现使用，采用现在的一般的设计方式，使用代理中心，确认；

在客户端可能会有两种情况：1.负载均衡，2重复部署（属于错误）3容错；需要根据需求来确认服务

## 通讯

服务发布信息，我打算研究dds通讯，使用其作为底层；