### dubbo 支持的协议以及注册中心

* 1. Dubbo 支持的协议：

dubbo、 rmi、hessian、http、webservice、thrift、memcached、redis、rest 协议；

* + 1. Dubbo ：

传输服务： mina、netty（默认）、grizzly

序列化： dubbo、hessian2（默认）、java、fastjson

连接描述： 单个长连接、NIO 异步传输

适用场景： 常规 RPC 调用、传输数据量少、提供者少于消费者

* + 1. Rmi ：

传输服务： java rmi 服务

序列化： java 原生二进制序列化

连接描述： 多个短连接，BIO 同步传输

适用场景： 常规 RPC调用、可传少量文件、不支持防火墙穿透

* + 1. hessian

传输服务： servlet 容器

序列化： hessian 二进制序列化

连接描述： 基于 http 协议传输、依赖 servlet 容器配置

适用场景： 提供者多于消费者、可以传大字段和文件、跨语言调用

* + 1. http

传输服务： servlet 容器

序列化： java 原生二进制序列化

连接描述： 依赖 servlet 容器配置

适用场景： 数据包大小混合

* + 1. Thrift：

连接描述： 长连接、 NIO 异步传输

* + 1. Webservice：

传输服务： servlet 容器

序列化： soap 文本序列化

连接描述： 基于 http 协议传输、

适用场景：系统集成、跨语言调用

* 1. 注册中心：

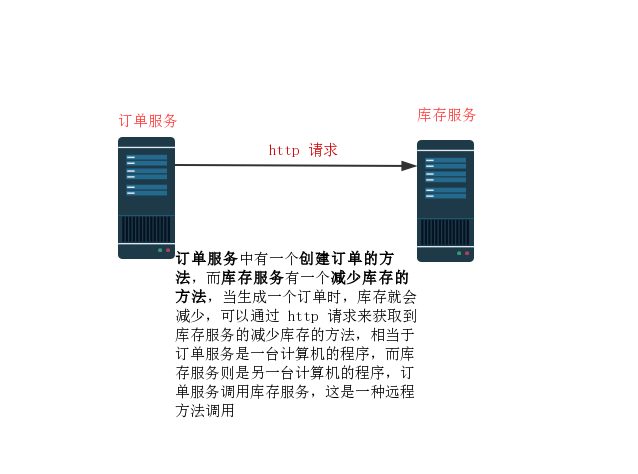
什么是注册中心，注册中心就是当服务端启动的时候，会把服务名和URL（ip 和 port）暴露给注册中心，注册中心会有一个存时的 List 集合存储数据，当服务器动态的添加或挂时，注册中心可以动态的感知到，并且客户端启动时，会发现注册中心中的服务，从而注册中心会把 URL 推送给客户端。

传统方式：

在以前的服务集群时，使用了 Nginx 来做代理的，多个服务端的 ip 配置到 Nginx 中，但有一个缺点，就是当服务端动态的添加或删除时，需要去 Nginx 中配置，这样做非常的耗费人力。所以当代出现了许多优秀的框架弥补了 Nginx 这个缺点比如： Dobbo、Spring Colud、Netfilx OSS、Service mesh

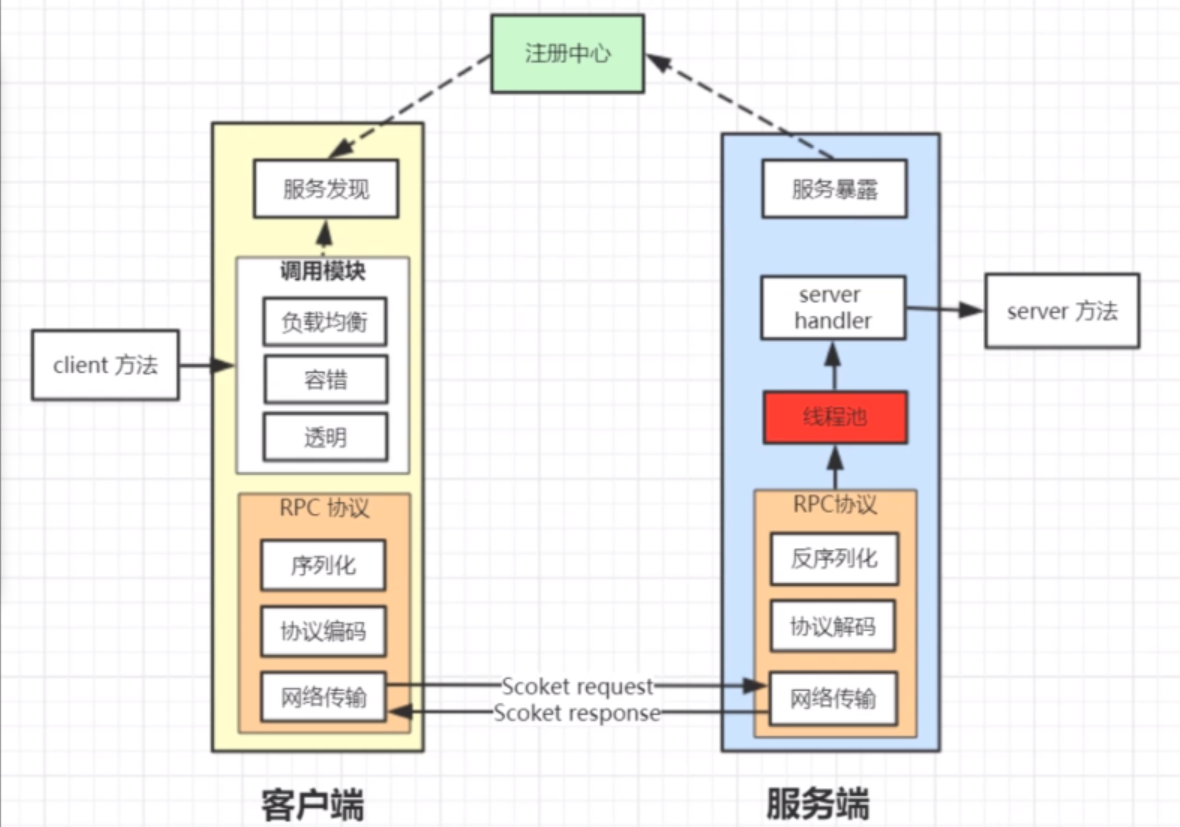
### RPC 底层原理

* 1. 什么是 RPC, RPC（Remote Procedure Call）远程过程调用，是一台计算机通信协议。该协议允许运行于一台计算机的程序调用另一台计算机的子程序，而程序员无需额外的为这个交互作用编程。如果涉及的软件采用面向对象编程（java），那么远程过程调用亦可称作远程调用或远程方法调用。如图 1.0.1解释



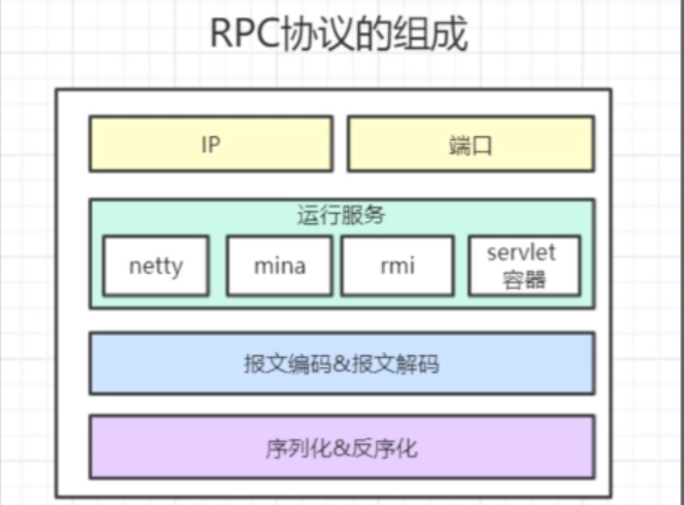
1.0.1 图

在一个典型的 RPC 使用场景中，包含了服务发现、负载均衡、容错、网络传输、序列化组件，其中 RPC 协议就指明了程序如果进行网络传输和序列化。也就是说一个RPC协议的实现就等于一个非透明的RPC调用，如何做到的的呢？如图：1.1.1 图



1.1.1 图

* + 1. 负载均衡：多个客户端和多个服务端、多个请求到服务端时怎么保证服务端的请求处理是负载均衡的？可以使用算法，一般使用轮询、随机、dubbo中的 一致性 Hash
    2. 容错：是服务端调用失败后，客户端要采用那些措施。
    3. 网络传输：客户端发起一个请求时，把参数通过二进制的方式传输到服务端，服务端做出响应返回结果。底层的传输是使用了 tcp/ ip 协议, 在 java 中叫 socket。
    4. 透明性：透明性是就是指框架对使用的动态代理的封装。比如：远程调用、序列化与反序列化、报文编码和报文解码等操作，都在框架的中使用动态代理的方式封装着。
    5. 解析 1.1.1图：
       1. 当服务端启动的时候，提供者会把自己的服务名和 URL注册到注册中心。
       2. 当客户端启动的时候，注册中心会把临时存储的数据推送给客户端。
       3. 当客户端发起一个请求时，调用了一个Service 方法，这个方法是通过动态代理透明化封装的，动态代理中还包含了负载均衡、容错、传输，负载的话，应该消费者调用那个提供者；当服务端调用失败的时候，客户端采取的策略是什么？称为”容错“。
       4. 客户端传输： RPC 协议中客户端发起请求市 params 方法传递的参数，做一个序列化，然后通过报文编码，最后在进行网络传输，可以是 socket、tcp/ip 进行传输。
       5. 服务端是进行接收：首先服务端要进行报文解码，并且对消息体进行一个反序列化，再通过线程池调用服务端相对应的 Service 方法。反之，当服务端返回时，跟客户端的请求一致，而客户端则才用了服务端原先的解码过程。
    6. RPC 协议的组成部分



1. IP: 服务提供者的地址
2. 端口：协议指定开发的端口
3. 报文编码： 协议报文编码， 分为请求头和请求体两部分
4. 序列化方法：将请求体序列化成对象
   1. Hessian2Serialization
   2. DubboSerialization
   3. JavaSerialization
   4. JsonSeralization
5. 运行服务：网络传输实现
   1. Netty
   2. mina
   3. RMI 服务
   4. Servlet 容器（jetty、tomcat、 jboss）