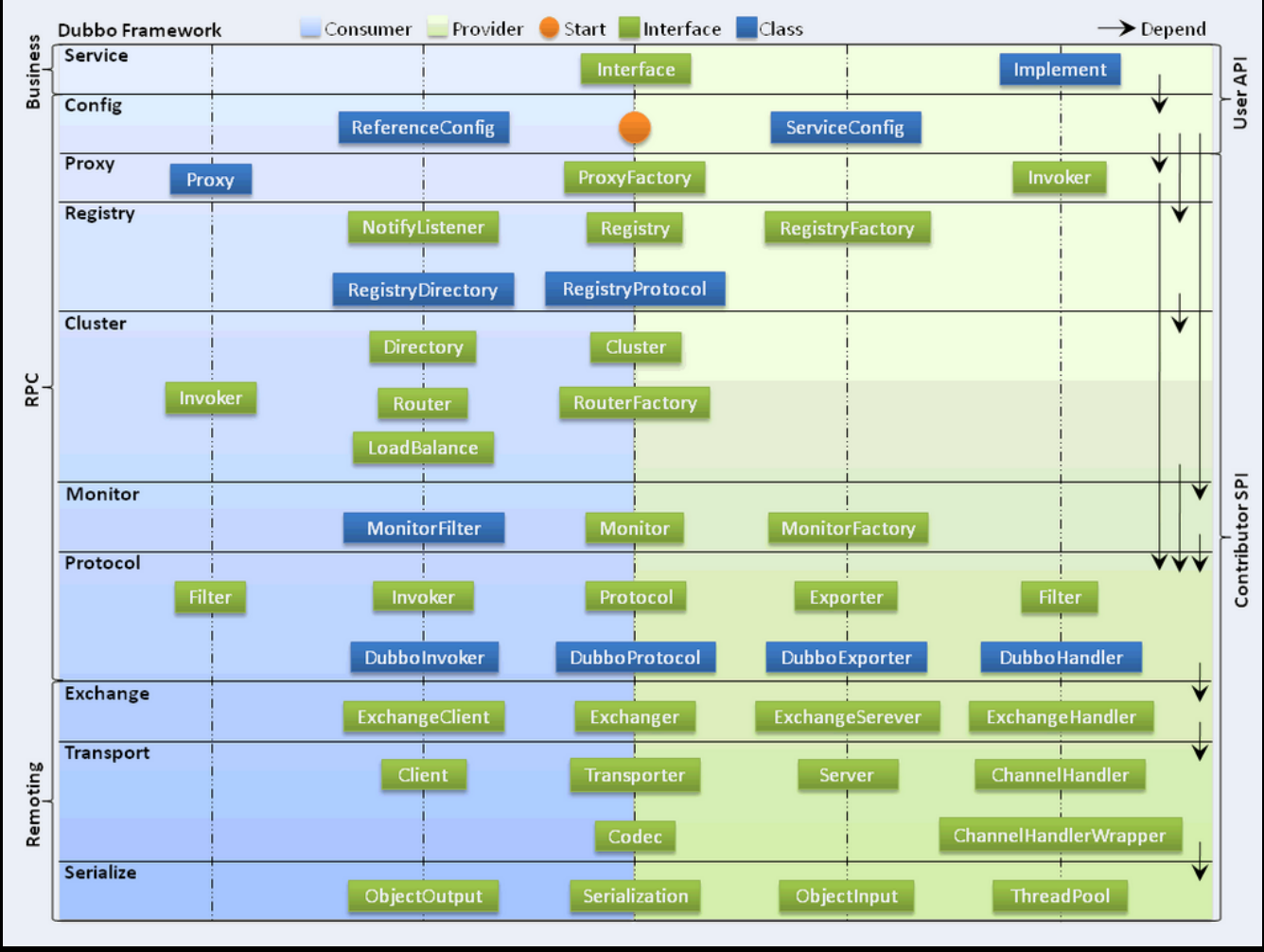
Dubbo架构设计详解

Dubbo是Alibaba开源的分布式服务框架，它最大的特点是按照分层的方式来架构，使用这种方式可以使各个层之间解耦合（或者最大限度地松耦合）。从服务模型的角度来看，Dubbo采用的是一种非常简单的模型，要么是提供方提供服务，要么是消费方消费服务，所以基于这一点可以抽象出服务提供方（Provider）和服务消费方（Consumer）两个角色。关于注册中心、协议支持、服务监控等内容，详见后面描述。

总体架构

Dubbo的总体架构，如图所示：



Dubbo框架设计一共划分了10个层，而最上面的Service层是留给实际想要使用Dubbo开发分布式服务的开发者实现业务逻辑的接口层。图中左边淡蓝背景的为服务消费方使用的接口，右边淡绿色背景的为服务提供方使用的接口， 位于中轴线上的为双方都用到的接口。
下面，结合Dubbo官方文档，我们分别理解一下框架分层架构中，各个层次的设计要点：

1. 服务接口层（Service）：该层是与实际业务逻辑相关的，根据服务提供方和服务消费方的业务设计对应的接口和实现。
2. 配置层（Config）：对外配置接口，以ServiceConfig和ReferenceConfig为中心，

可以直接new配置类，也可以通过spring解析配置生成配置类。

3. 服务代理层（Proxy）：服务接口透明代理，生成服务的客户端Stub和服务器端Skeleton，以ServiceProxy为中心，扩展接口为ProxyFactory。
4. 服务注册层（Registry）：封装服务地址的注册与发现，以服务URL为中心，扩展接口为RegistryFactory、Registry和RegistryService。可能没有服务注册中心，此时服务提供方直接暴露服务。
5. 集群层（Cluster）：封装多个提供者的路由及负载均衡，并桥接注册中心，以Invoker为中心，扩展接口为Cluster、Directory、Router和LoadBalance。将多个服务提供方组合为一个服务提供方，实现对服务消费方来透明，只需要与一个服务提供方进行交互。
6. 监控层（Monitor）：RPC调用次数和调用时间监控，以Statistics为中心，扩展接口为MonitorFactory、Monitor和MonitorService。
7. 远程调用层（Protocol）：封将RPC调用，以Invocation和Result为中心，扩展接口为Protocol、Invoker和Exporter。Protocol是服务域，它是Invoker暴露和引用的主功能入口，它负责Invoker的生命周期管理。Invoker是实体域，它是Dubbo的核心模型，其它模型都向它靠扰，或转换成它，它代表一个可执行体，可向它发起invoke调用，它有可能是一个本地的实现，也可能是一个远程的实现，也可能一个集群实现。
8. 信息交换层（Exchange）：封装请求响应模式，同步转异步，以Request和Response为中心，扩展接口为Exchanger、ExchangeChannel、ExchangeClient和ExchangeServer。
9. 网络传输层（Transport）：抽象mina和netty为统一接口，以Message为中心，扩展接口为Channel、Transporter、Client、Server和Codec。
10. 数据序列化层（Serialize）：可复用的一些工具，扩展接口为Serialization、ObjectInput、ObjectOutput和ThreadPool。

从上图可以看出，Dubbo对于服务提供方和服务消费方，从框架的10层中分别提供了各自需要关心和扩展的接口，构建整个服务生态系统（服务提供方和服务消费方本身就是一个以服务为中心的）。

根据官方提供的，对于上述各层之间关系的描述，如下所示：

- 在RPC中，Protocol是核心层，也就是只要有Protocol + Invoker + Exporter就可以完成非透明的RPC调用，然后在Invoker的主过程上Filter拦截点。
- 图中的Consumer和Provider是抽象概念，只是想让看图者更直观的了解哪些类分属于客户端与服务器端，不用Client和Server的原因是Dubbo在很多场景下都使用Provider、Consumer、Registry、Monitor划分逻辑拓普节点，保持统一概念。
- 而Cluster是外围概念，所以Cluster的目的是将多个Invoker伪装成一个Invoker，这样其它人只要关注Protocol层Invoker即可，加上Cluster或者去掉Cluster对其它层都不会造成影响，因为只有一个提供者时，是不需要Cluster的。
- Proxy层封装了所有接口的透明化代理，而在其它层都以Invoker为中心，只有到了暴露给用户使用时，才用Proxy将Invoker转成接口，或将接口实现转成Invoker，也就是去掉Proxy层RPC是可以Run的，只是不那么透明，不那么看起来像调本地服务一样调远程服务。
- 而Remoting实现是Dubbo协议的实现，如果你选择RMI协议，整个Remoting都不会用上，Remoting内部再划为Transport传输层和Exchange信息交换层，Transport层只负责单向消息传输，是对Mina、Netty、Grizzly的抽象，它也可以扩展UDP传输，而Exchange层是在传输层之上封装了Request-Response语义。
- Registry和Monitor实际上不算一层，而是一个独立的节点，只是为了全局概览，用层的方式画在一起。

从上面的架构图中，我们可以了解到，Dubbo作为一个分布式服务框架，主要具有如下几个核心的要点：

服务定义

服务是围绕服务提供方和服务消费方的，服务提供方实现服务，而服务消费方调用服务。

服务注册

对于服务提供方，它需要发布服务，而且由于应用系统的复杂性，服务的数量、类型也不断膨胀；对于服务消费方，它最关心如何获取到它所需要的服务，而面对复杂的应用系统，需要管理大量的服务调用。而且，对于服务提供方和服务消费方来说，他们还有可能兼具这两种角色，即既需要提供服务，有需要消费服务。

通过将服务统一管理起来，可以有效地优化内部应用对服务发布/使用的流程和管理。服务注册中心可以通过特定协议来完成服务对外的统一。Dubbo提供的注册中心有如下几种类型可供选择：

- Multicast注册中心
- Zookeeper注册中心
- Redis注册中心
- Simple注册中心

服务监控

无论是服务提供方，还是服务消费方，他们都需要对服务调用的实际状态进行有效的监控，从而改进服务质量。

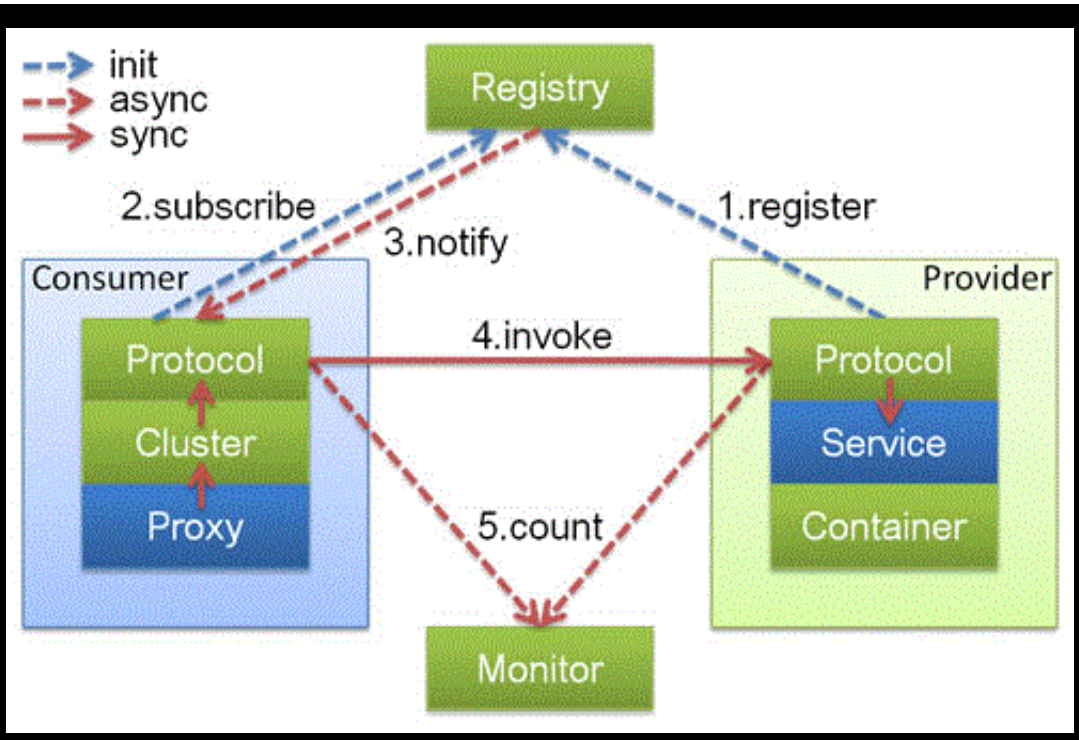
远程通信与信息交换

远程通信需要指定通信双方所约定的协议，在保证通信双方理解协议语义的基础上，还要保证高效、稳定的消息传输。Dubbo继承了当前主流的网络通信框架，主要包括如下几个：

- Mina
- Netty
- Grizzly

服务调用

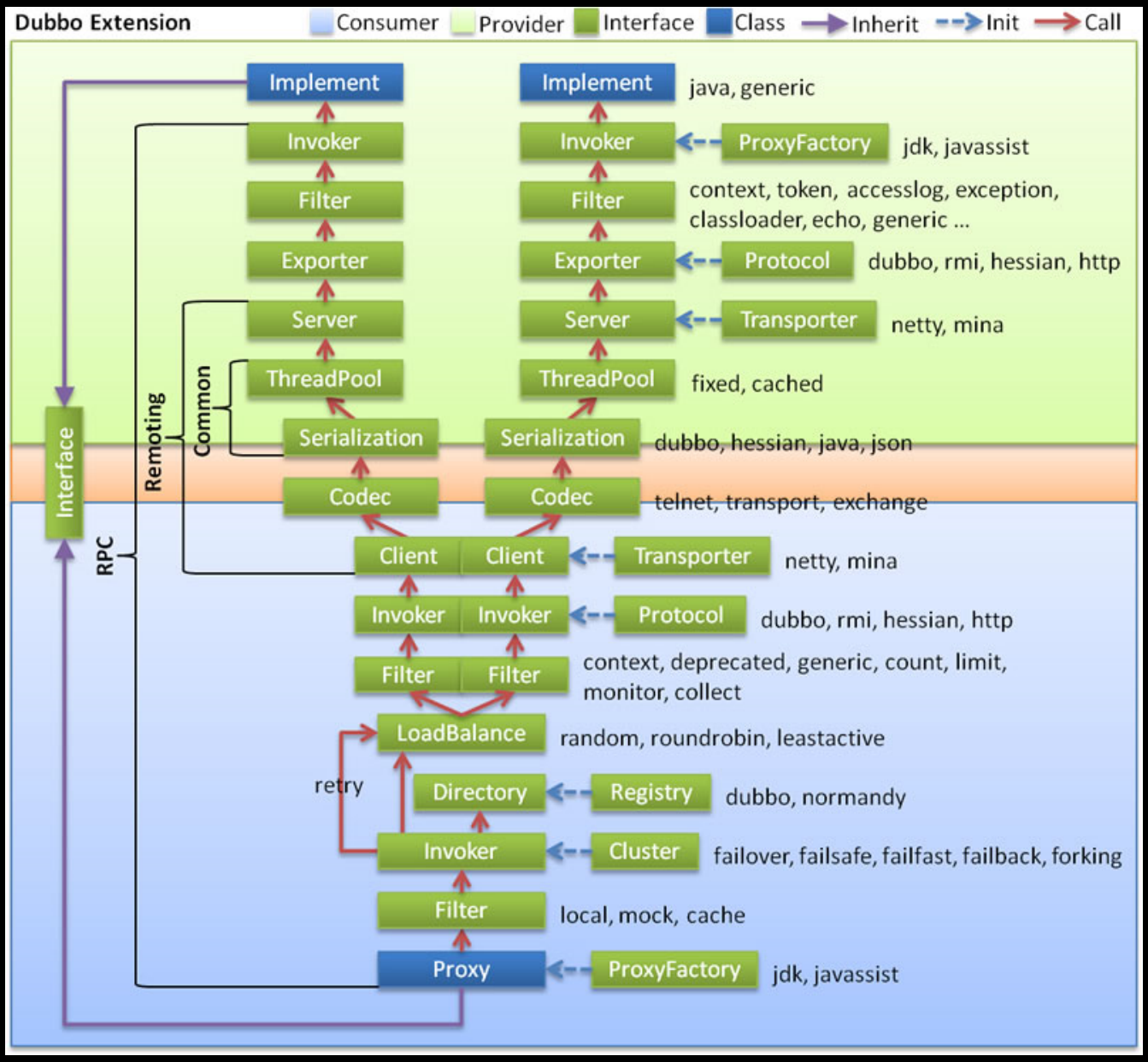
下面从Dubbo官网直接拿来，看一下基于RPC层，服务提供方和服务消费方之间的调用关系，如图所示：



上图中，蓝色的表示与业务有交互，绿色的表示只对Dubbo内部交互。上述图所描述的调用流程如下：

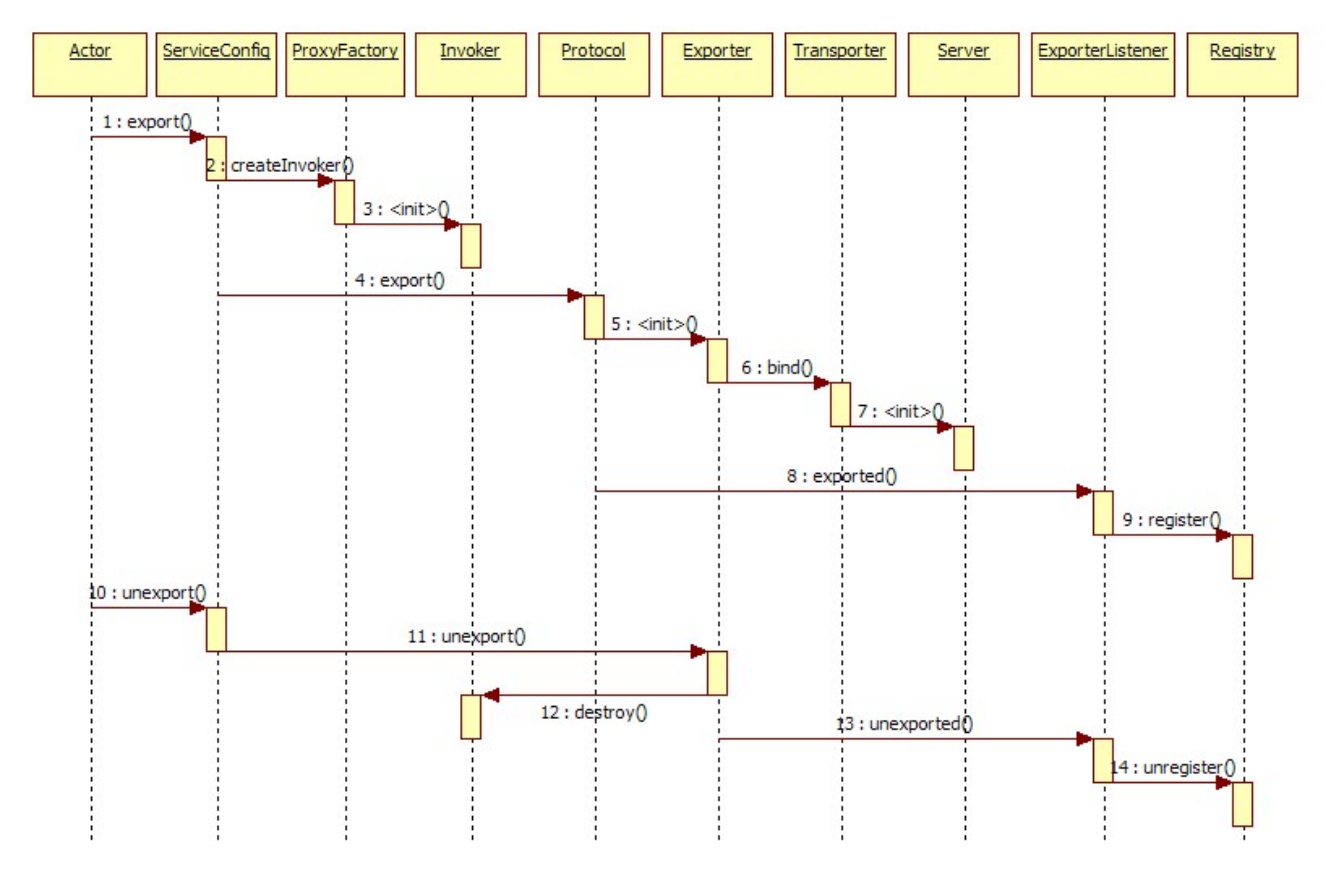
1. 服务提供方发布服务到服务注册中心；
2. 服务消费方从服务注册中心订阅服务；
3. 服务消费方调用已经注册的可用服务

接着，将上面抽象的调用流程图展开，详细如图所示：



注册/注销服务

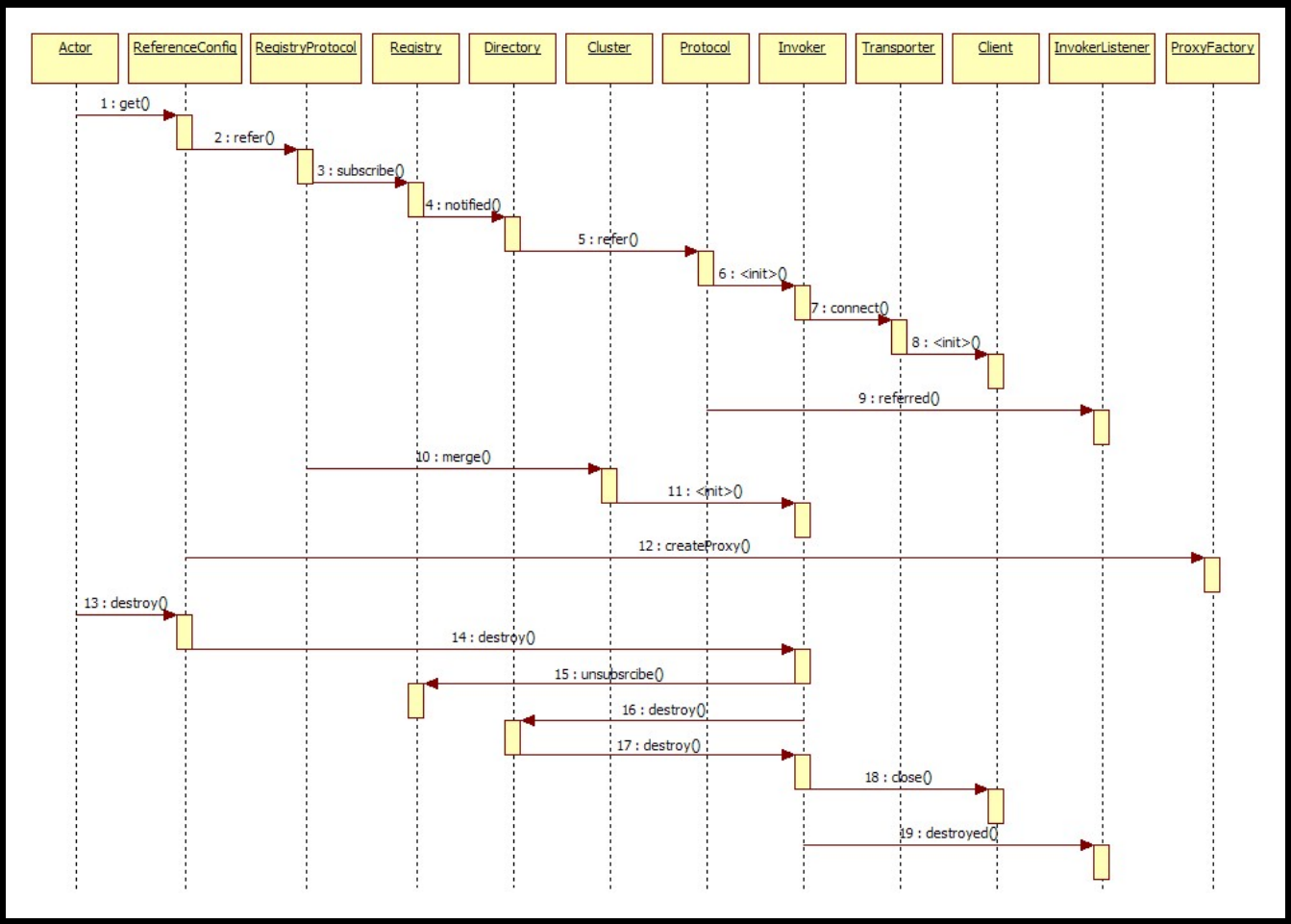
服务的注册与注销，是对服务提供方角色而言，那么注册服务与注销服务的时序图，如图所示：



服务订阅/取消

为了满足应用系统的需求，服务消费方的可能需要从服务注册中心订阅指定的有服务提供方发布的服务，在得到通知可以使用服务时，就可以直接调用服务。反过来，如果不需要某一个服

务了，可以取消该服务。下面看一下对应的时序图，如图所示：



协议支持

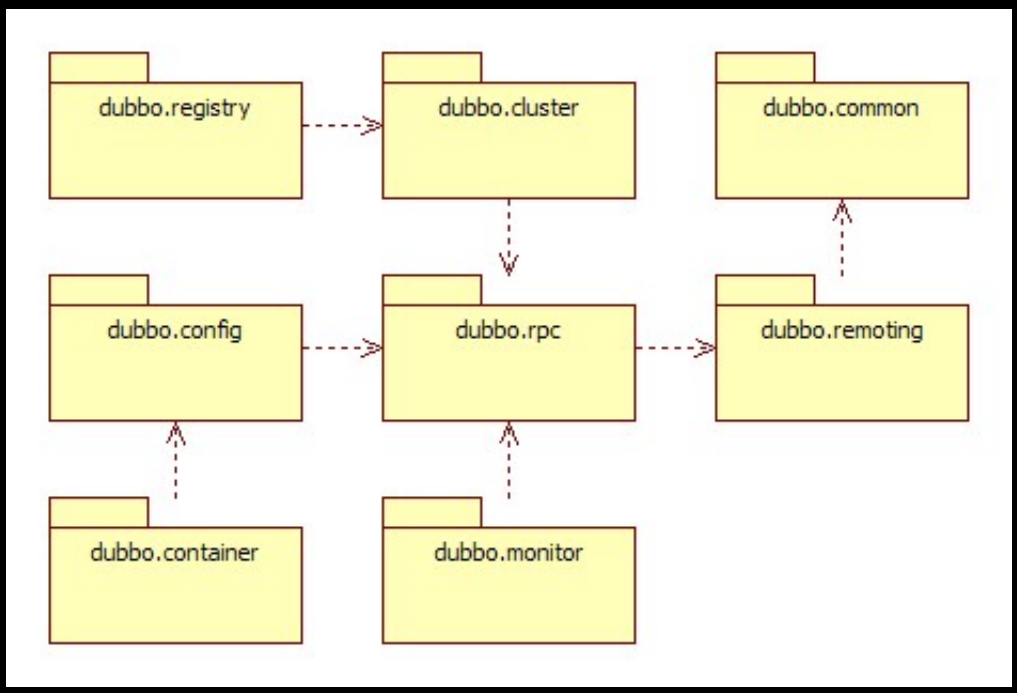
Dubbo支持多种协议，如下所示：

- Dubbo协议
- Hessian协议
- HTTP协议
- RMI协议
- WebService协议
- Thrift协议
- Memcached协议
- Redis协议

在通信过程中，不同的服务等级一般对应着不同的服务质量，那么选择合适的协议便是一件非常重要的事情。你可以根据你应用的创建来选择。例如，使用RMI协议，一般会受到防火墙的限制，所以对于外部与内部进行通信的场景，就不要使用RMI协议，而是基于HTTP协议或者Hessian协议。

参考补充

Dubbo以包结构来组织各个模块，各个模块及其关系，如图所示：



可以通过Dubbo的代码（使用Maven管理）组织，与上面的模块进行比较。简单说明各个包的情况：

- dubbo-common 公共逻辑模块，包括Util类和通用模型。
- dubbo-remoting 远程通讯模块，相当于Dubbo协议的实现，如果RPC用RMI协议则不需要使用此包。
- dubbo-rpc 远程调用模块，抽象各种协议，以及动态代理，只包含一对一的调用，不关心集群的管理。
- dubbo-cluster 集群模块，将多个服务提供方伪装为一个提供方，包括：负载均衡、容错、路由等，集群的地址列表可以是静态配置的，也可以是由注册中心下发。
- dubbo-registry 注册中心模块，基于注册中心下发地址的集群方式，以及对各种注册中心的抽象。
- dubbo-monitor 监控模块，统计服务调用次数，调用时间的，调用链跟踪的服务。
- dubbo-config 配置模块，是Dubbo对外的API，用户通过Config使用Dubbo，隐藏Dubbo所有细节。
- dubbo-container 容器模块，是一个Standalone的容器，以简单的Main加载Spring启动，因为服务通常不需要Tomcat/JBoss等Web容器的特性，没必要用Web容器去加载服务。

参考链接

- https://github.com/alibaba/dubbo (https://github.com/alibaba/dubbo)
- http://alibaba.github.io/dubbo-doc-static/Home-zh.htm (http://alibaba.github.io/dubbo-doc-static/Home-zh.htm)
- http://alibaba.github.io/dubbo-doc-static/User+Guide-zh.htm (http://alibaba.github.io/dubbo-doc-static/User+Guide-zh.htm)
- http://alibaba.github.io/dubbo-doc-static/Developer+Guide-zh.htm (http://alibaba.github.io/dubbo-doc-static/Developer+Guide-zh.htm)

- http://alibaba.github.io/dubbo-doc-static/Administrator+Guide-zh.htm (http://alibaba.github.io/dubbo-doc-static/Administrator+Guide-zh.htm)
- http://alibaba.github.io/dubbo-doc-static/FAQ-zh.htm (http://alibaba.github.io/dubbo-doc-static/FAQ-zh.htm)