1.RPC

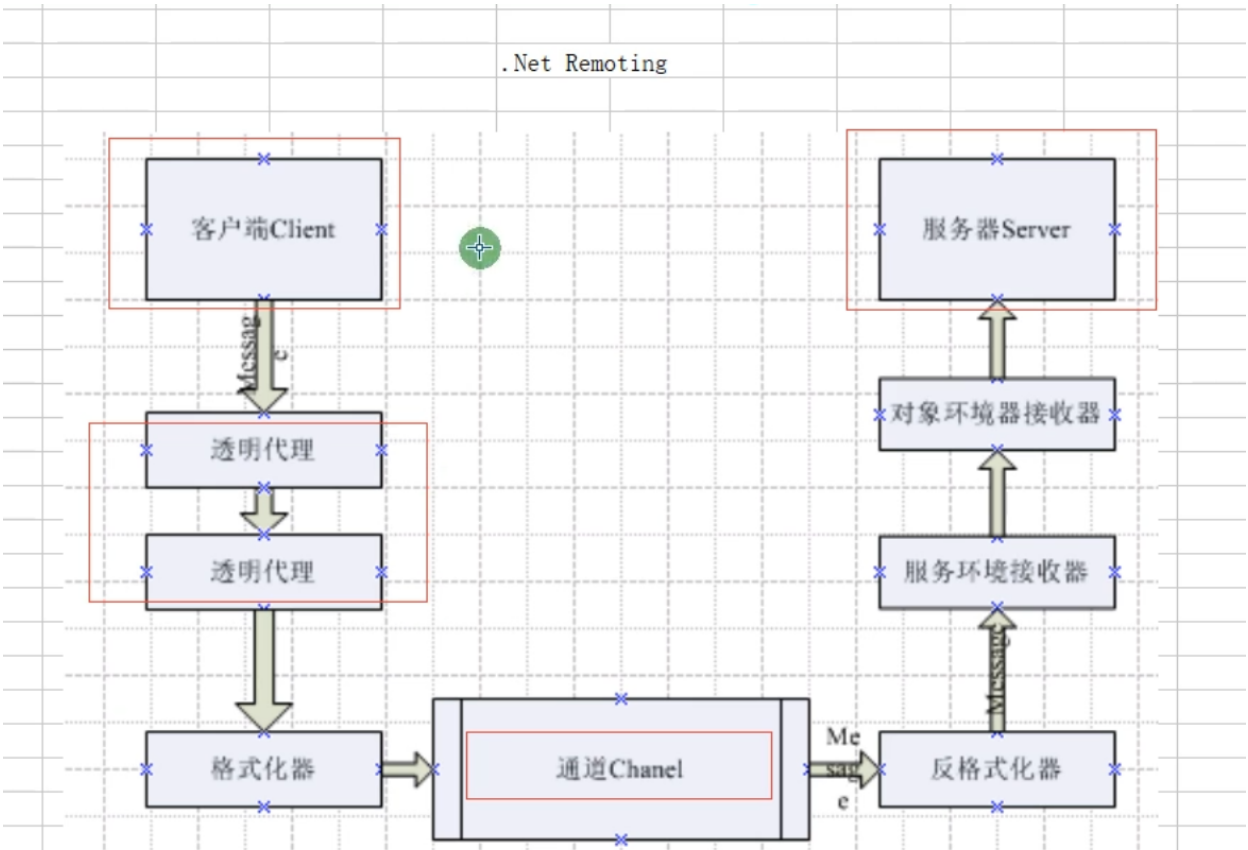
RPC（Remote Procedure Call Protocol）远程过程调用协议。一个通俗的描述是：客户端在不知道调用细节的情况下，调用存在于远程计算机上的某个对象，就像调用本地应用程序中的对象一样。比较正式的描述是：一种通过网络从远程计算机程序上请求服务，而不需要了解底层网络技术的协议。那么我们至少从这样的描述中挖掘出几个要点：

RPC是协议：既然是协议就只是一套规范，那么就需要有人遵循这套规范来进行实现。目前典型的RPC实现包括：Dubbo、Thrift、GRPC、Hetty等。这里要说明一下，目前技术的发展趋势来看，实现了RPC协议的应用工具往往都会附加其他重要功能，例如Dubbo还包括了服务管理、访问权限管理等功能。

网络协议和网络IO模型对其透明：既然RPC的客户端认为自己是在调用本地对象。那么传输层使用的是TCP/UDP还是HTTP协议，又或者是一些其他的网络协议它就不需要关心了。既然网络协议对其透明，那么调用过程中，使用的是哪一种网络IO模型调用者也不需要关心。

信息格式对其透明：我们知道在本地应用程序中，对于某个对象的调用需要传递一些参数，并且会返回一个调用结果。至于被调用的对象内部是如何使用这些参数，并计算出处理结果的，调用方是不需要关心的。那么对于远程调用来说，这些参数会以某种信息格式传递给网络上的另外一台计算机，这个信息格式是怎样构成的，调用方是不需要关心的。

应该有跨语言能力：为什么这样说呢？因为调用方实际上也不清楚远程服务器的应用程序是使用什么语言运行的。那么对于调用方来说，无论服务器方使用的是什么语言，本次调用都应该成功，并且返回值也应该按照调用方程序语言所能理解的形式进行描述。




通过代理的形式：可以把服务器端当作正常的代码来进行操作，同时也屏蔽了下层的协议，只用专注于自己的事情。

2.wsdl简介

1）是什么

web service description language 网络服务描述语言。即是用来描述服务的。

描述服务里面有那些方法，以及在什么地址用什么方式被调用。

2）结构讲解

```
definitions：根
    name：服务名称，类型+Service
    targetNamespace：http://倒置包名

    service：服务接口
      <soap:address location="http://172.16.14.254:9999/hello"/>服务在哪个地址能够被访问
    binding ：
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="docur
      transport：一什么样传输协议来调用。
      style="document"：表示我们wsdl文件遵循什么格式 document/rpc
    portType:接口类型
          描述了有哪些接口方法
    message:消息
          描述接口方法的参数和返回值消息
    types：描述消息的类型和名称-在document才有，而rpc没有
```

3.SOAP简介

1）是什么

simple object access protocol:简单对象访问协议

2）结构讲解

Envelope:信封
Header:信封头部，开发人员 用的，我们需要从客户端往服务端 传递一些数据的时候用---比如权限信息等
Body:主体，一般是系统自己用
对于请求而言，告诉服务端要调用方法名以及参数名和值
　　对于响应而言，告诉客户端响应的是那个方法，以及返回值的名称和值。
**所谓简单对象传输，即是用来传递xml**，类似http的请求/应答模型，来保证数据的交换。

4.URI和URL的区别和关联
统一资源标识符(uniform resource identifier URI)
和
统一资源定位符(uniform resource locator URL)
能定位就一定能标识，但是标识不代表能定位。北京XXX公司总经理”就是张三这个人的URI，但是还是不能定位张三。
说白了URI只是作为唯一标识的。

5.Restful风格和RPC风格
**RPC**
这是最常见的方式，RPC说的是本地调用远程的方法，面向的是过程。
RPC形式的API组织形态是类和方法，或者说领域和行为。
因此API的命名往往是一个动词，比如GetUserInfo和CreateUser。
因为URI会非常多而且往往没有一些约定规范，所以需要有详细的文档。
也是因为无拘无束，HTTP方法基本只用GET和POST，设计起来比较简单。
这里就不贴例子了，估计超过50%的API是这种风格的

**什么是restful，简称rest?**
REST ： Representational State Transfer，表现层状态转换；**是以 资源 为中心**，使用统一的接口URL，使用GET、POST、PUT、PATCH、DELETE等操作方法，来处理资源。

restful风格的原则条件:
C/S结构、无状态:
Web 应用程序最重要的 REST 原则是，客户端和服务器之间的交互在请求之间是无状态的。
从客户端到服务器的每个请求都必须包含理解请求所必需的信息。如果服务器在请求之间的任何时间点重启，客户端不会得到通知。
可以cache:
此外，无状态请求可以由任何可用服务器回答，这十分适合云计算之类的环境。客户端可以缓存数据以改进性能。

统一的接口：
在服务器端，应用程序状态和功能可以分为各种资源。
资源是一个有趣的概念实体，它向客户端公开。资源的例子有：应用程序对象、数据库记录、算法等等。每个资源都使用 URI (Universal Resource Identifier) 得到一个唯一的地址。所有资源都共享统一的接口，以便在客户端和服务器之间传输状态。使用的是标准的 HTTP 方法，比如 GET、PUT、POST 和 DELETE。Hypermedia 是应用程序状态的引擎，资源表示通过超链接互联。
分层系统：
另一个重要的 REST 原则是分层系统，这表示组件无法了解它与之交互的中间层以外的组件。通过将系统知识限制在单个层，可以限制整个系统的复杂性，促进了底层的独立性。
当 REST 架构的约束条件作为一个整体应用时，将生成一个可以扩展到大量客户端的应用程序。它还降低了客户端和服务器之间的交互延迟。统一界面简化了整个系统架构，改进了子系统之间交互的可见性。REST 简化了客户端和服务器的实现。