# 第二篇 Binder机制

1. 了解Binder

在Android系统中，每一个应用程序都运行在独立的进程中，这也保证了当其中一个程序出现异常而不会影响另一个应用程序的正常运转。在许多情况下，我们activity都会与各种系统的service打交道，很显然，我们写的程序中activity与系统service肯定不是同一个进程，但是它们之间是怎样实现通信的呢？

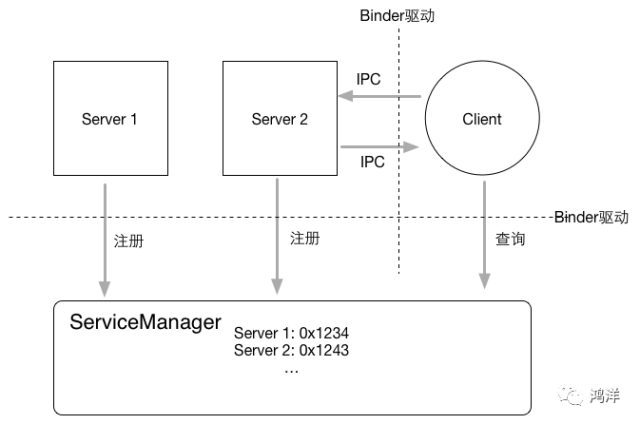
所以Binder是android中一种实现进程间通信（IPC）的方式之一。

1. .首先，Binder分为Client和Server两个进程。

注意，Client和Server是相对的。谁发消息，谁就是Client，谁接收消息，谁就是Server。

举个例子，两个进程A和B之间使用Binder通信，进程A发消息给进程B，那么这时候A是Binder Client，B是Binder Server；进程B发消息给进程A，那么这时候B是Binder Client，A是Binder Server——其实这么说虽然简单了，但还是不太严谨，我们先这么理解着。

2).其次，我们看下面这个图（摘自田维术的博客），基本说明白了Binder的组成解构：



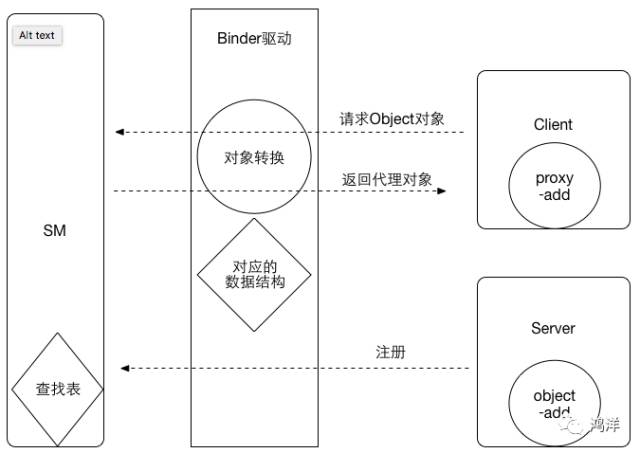
图中的IPC就是进程间通信的意思。

图中的ServiceManager，负责把Binder Server注册到一个容器中

有人把ServiceManager比喻成电话局，存储着每个住宅的座机电话，还是很恰当的。张三给李四打电话，拨打电话号码，会先转接到电话局，电话局的接线员查到这个电话号码的地址，因为李四的电话号码之前在电话局注册过，所以就能拨通；没注册，就会提示该号码不存在。

对照着Android Binder机制，对着上面这图，张三就是Binder Client，李四就是Binder Server，电话局就是ServiceManager，电话局的接线员在这个过程中做了很多事情，对应着图中的Binder驱动.

3).接下来我们看Binder通信的过程，还是摘自田维术博客的一张图：



注：图中的SM也就是ServiceManager。

我们看到，Client想要直接调用Server的add方法，是不可以的，因为它们在不同的进程中，这时候就需要Binder来帮忙了。

首先是Server在SM这个容器中注册。

其次，Client想要调用Server的add方法，就需要先获取Server对象， 但是SM不会把真正的Server对象返回给Client，而是把Server的一个代理对象返回给Client，也就是Proxy。

然后，Client调用Proxy的add方法，SM会帮他去调用Server的add方法，并把结果返回给Client。

以上这3步，Binder驱动出了很多力，但我们不需要知道Binder驱动的底层实现，涉及到C++的代码了——把有限的时间去做更有意义的事情。

2.为什么android选用Binder来实现进程间通信？

1).可靠性。在移动设备上，通常采用基于Client-Server的通信方式来实现互联网与设备间的内部通信。目前linux支持IPC包括传统的管道，System V IPC，即消息队列/共享内存/信号量，以及socket中只有socket支持Client-Server的通信方式。Android系统为开发者提供了丰富进程间通信的功能接口，媒体播放，传感器，无线传输。

这些功能都由不同的server来管理。开发都只关心将自己应用程序的client与server的通信建立起来便可以使用这个服务。毫无疑问，如若在底层架设一套协议来实现Client-Server通信，增加了系统的复杂性。在资源有限的手机 上来实现这种复杂的环境，可靠性难以保证。

2).传输性能。socket主要用于跨网络的进程间通信和本机上进程间的通信，但传输效率低，开销大。消息队列和管道采用存储-转发方式，即数据先从发送方缓存区拷贝到内核开辟的一块缓存区中，然后从内核缓存区拷贝到接收方缓存区，其过程至少有两次拷贝。虽然共享内存无需拷贝，但控制复杂。比较各种IPC方式的数据拷贝次数。共享内存：0次。Binder：1次。Socket/管道/消息队列：2次。

3).安全性。Android是一个开放式的平台，所以确保应用程序安全是很重要的。Android对每一个安装应用都分配了UID/PID,其中进程的UID是可用来鉴别进程身份。传统的只能由用户在数据包里填写UID/PID，这样不可靠，容易被恶意程序利用。而我们要求由内核来添加可靠的UID。

所以，出于可靠性、传输性、安全性。android建立了一套新的进程间通信方式。