Five different classes of failures can occur in RPC systems 

The client is unable to locate the server 

The request message from the client to server is lost

The reply message from the server to client is lost

The server crashes after receiving a request

The client crashes after sending a request

**RPC故障处理**

在RPC系统中可能出现5种不同类型的故障  
1、客户端无法找到服务器

使用特殊代码（如“-1”）作为过程的返回值来指示故障。

2、从客户端到服务器的请求消息丢失（超时重发）

内核在发送消息时启动一个计时器：  
如果定时器在回复或ACK返回之前到期：内核重新传输  
如果消息真的丢失了：服务器不会区分原来的和重传⇒一切都会正常工作  
如果有很多请求丢失：内核放弃并错误地断定服务器已经关闭⇒我们回到“无法找到服务器”

3、从服务器到客户端的回复消息丢失：

同理超时重发

4、服务器收到请求后崩溃

在这里做什么存在3个思想流派：

1等到服务器重新启动并再次尝试操作。 保证RPC至少执行一次（至少一次语义）

2立即放弃并报告失败。 保证RPC一次最多执行一次（最多一次语义）

3客户得不到帮助。 没有任何保证（RPC可能已经从0到大量的任何地方进行）。 易于实施。

5、发送请求后，客户端崩溃

在远程过程调用（RPC）过程中，如果客户端出现故障，请给出至少3种可能的解决方式。

客户端发送请求并在服务器回应之前崩溃：计算处于活动状态，没有父节点正在等待结果（孤立）

孤儿浪费CPU周期，可以锁定文件或绑定宝贵的资源

孤儿可能会引起混乱（客户端重新启动，并再次执行RPC，但孤儿的答复立即回来）

5.1可能的解决方案  
清除：在客户端存根发送RPC之前，它会在日志条目（安全存储）中告诉它将要执行的操作。 重新启动后，日志被检查，并孤立显式关闭。

5.2可能的解决方案（续）

轮回：将时间分成顺序编号的时代。 当客户端重新启动时，它会广播一条消息，声明新纪元的开始。 广播到来时，所有的远程计算都将被终止。 解决问题而不需要写入磁盘记录

5.3可能的解决方案（续）

到期：每个RPC都有一个标准的时间T来完成这项工作。 如果它不能完成，它必须明确要求另一个量子。

**动态绑定**

客户如何找到服务器？

1将服务器地址硬连线到客户端，快速但不灵活！

2更好的方法是动态绑定：

当服务器开始执行时，初始化主循环之外的调用将导出服务器接口

这意味着服务器发送一个消息到一个名为binder的程序，以使其存在已知。 这个过程被称为注册

为了注册，服务器给出了这个binder的名字，它的版本号，一个唯一的标识符（32位）和一个用来定位它的handle  
该handle取决于系统（例如，以太网地址，IP地址，X.500地址，...）

当客户首次调用其中一个远程过程时，说，读：  
1 客户端存根看到它还没有绑定到服务器，所以它发送一个消息给绑定器，要求导入服务器接口的版本x  
2 binder将检查一个或多个服务器是否已经使用此名称和版本号导出了一个接口。

如果当前正在运行的服务器不支持该接口，则读取调用失败

如果存在合适的服务器，则binder将其handle和唯一标识符提供给客户端存根

3 客户端存根使用handle作为发送请求消息的地址。 该消息包含参数和唯一标识符，服务器的内核用来将传入消息引导到正确的服务器

动态绑定的优点  
1，灵活性  
2，可以支持多个”拥有相同接口”的服务器，例如：

2.1，Binder可以将客户端随机地分布在服务器上，实现均匀加载

2.2，Binder可以定期轮询服务器，自动注销未能响应的服务器，达到一定的容错度

2.3，Binder可以协助认证：例如，一个服务器指定一个可以使用它的用户列表; Binder

将拒绝告诉用户不在名单上关于服务器

3，Binder可以验证客户端和服务器都使用相同版本的界面

动态绑定的缺点  
1，导出/导入界面的额外开销 花费时间  
2，绑定可能成为大型分布式系统的瓶颈