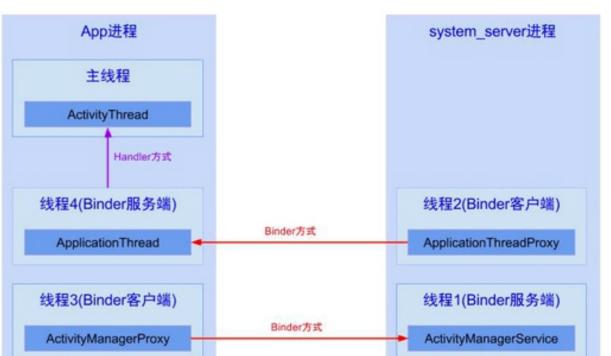
snalopainen ApplicationThread Launcher Activity ActivityManageProxy ActivityManagerService ActivityStack ApplicationThreadProxy ActivityThread Instrumentation 2.startActivity 1.startActivitySafely 3.startActivityForResult 4.execStartActivity 5.startActivity 6.startActivity 7.startActivityMayWait 8.startActivityLocked 9.startActivityUncheckedLocked 10.startActivityLocked 11.resumeTopActivityLocked 12.startPausingLocked 13.schedulePauseActivity 14.schedulePauseActivity 15.queueOrSendMessage 16.handleMessage 17.handlePauseActivity 18.activityPaused 19.activityPaused 20.activityPaused 21completePauseLocked 22.resumeTopActivityLocked 23.startSpecificActivityLocked 23.mService.startProcessLocked 25. startProcessLocked 26. main 27.attachApplication 28.attachApplication 29.attachApplicationLocked 30.realStartActivityLocked 31.scheduleLaunchActivity 32.scheduleLaunchActivity 3.queueOrSendMessage 34.handleMessage 35.handleLaunchActivity 36.performLaunchActivity

这个ApplicationThread实际上是一个Binder对象,是App所在的进程与 AMS所在进程system\_server通信的桥梁;

1.在Activity启动的过程中,App进程会频繁地与AMS进程进行通信:Ap p进程会委托AMS进程完成Activity生命周期的管理以及任务栈的管理;这 个通信过程AMS是Server端,App进程通过持有AMS的client代理Activit yManagerNative完成通信过程;

2.AMS进程完成生命周期管理以及任务栈管理后,会把控制权交给App进程,让App进程完成Activity类对象的创建,以及生命周期回调;这个通信过程也是通过Binder完成的,App所在server端的Binder对象存在于ActivityThread的内部类ApplicationThread;AMS所在client通过持有IApplicationThread的代理对象完成对于App进程的通信。



App进程内部的ApplicationThread server端内部有自己 的Binder线程池,它与App主线程的通信通过Handler完 成,这个Handler存在于ActivityThread类,它的名字很 简单就叫H