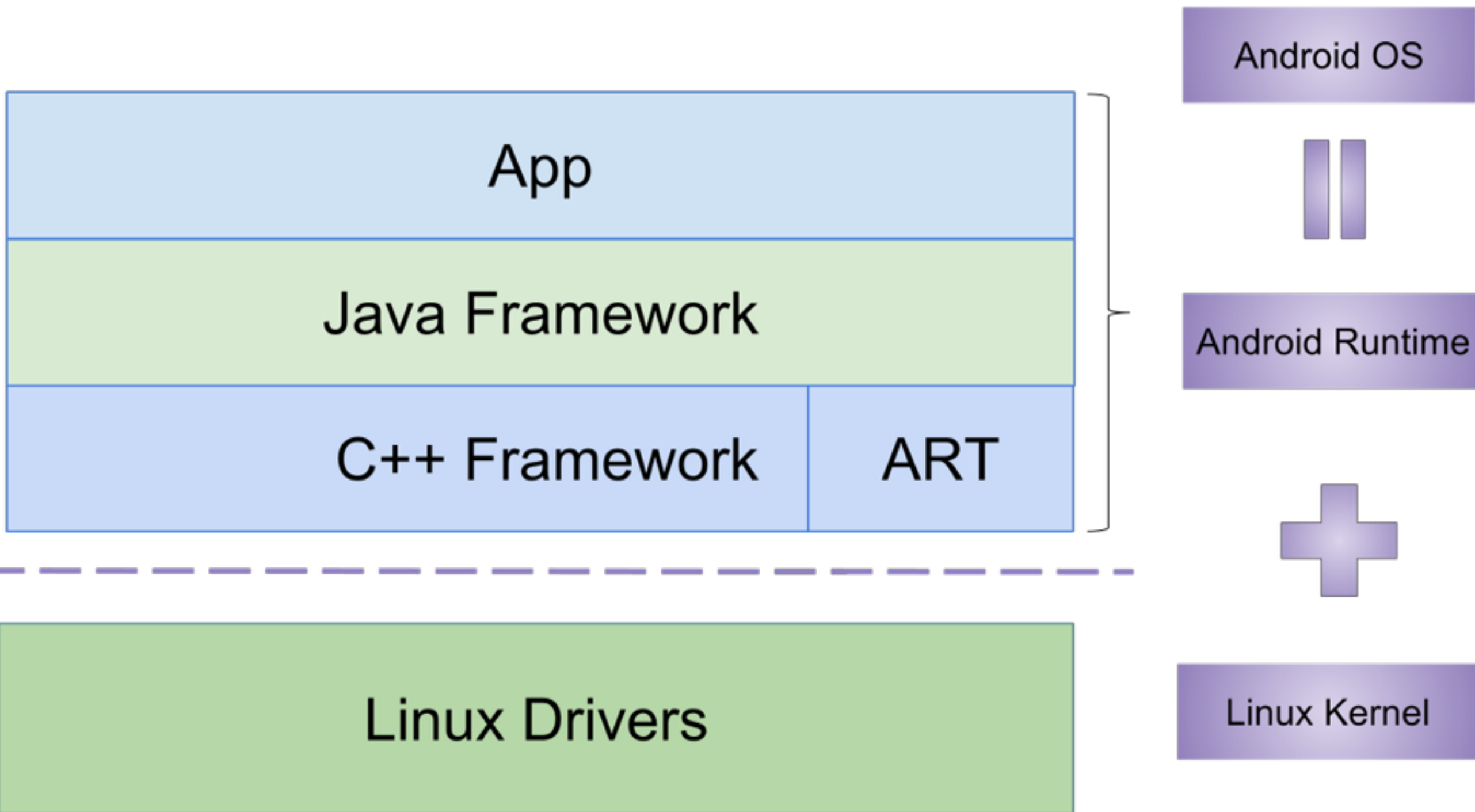
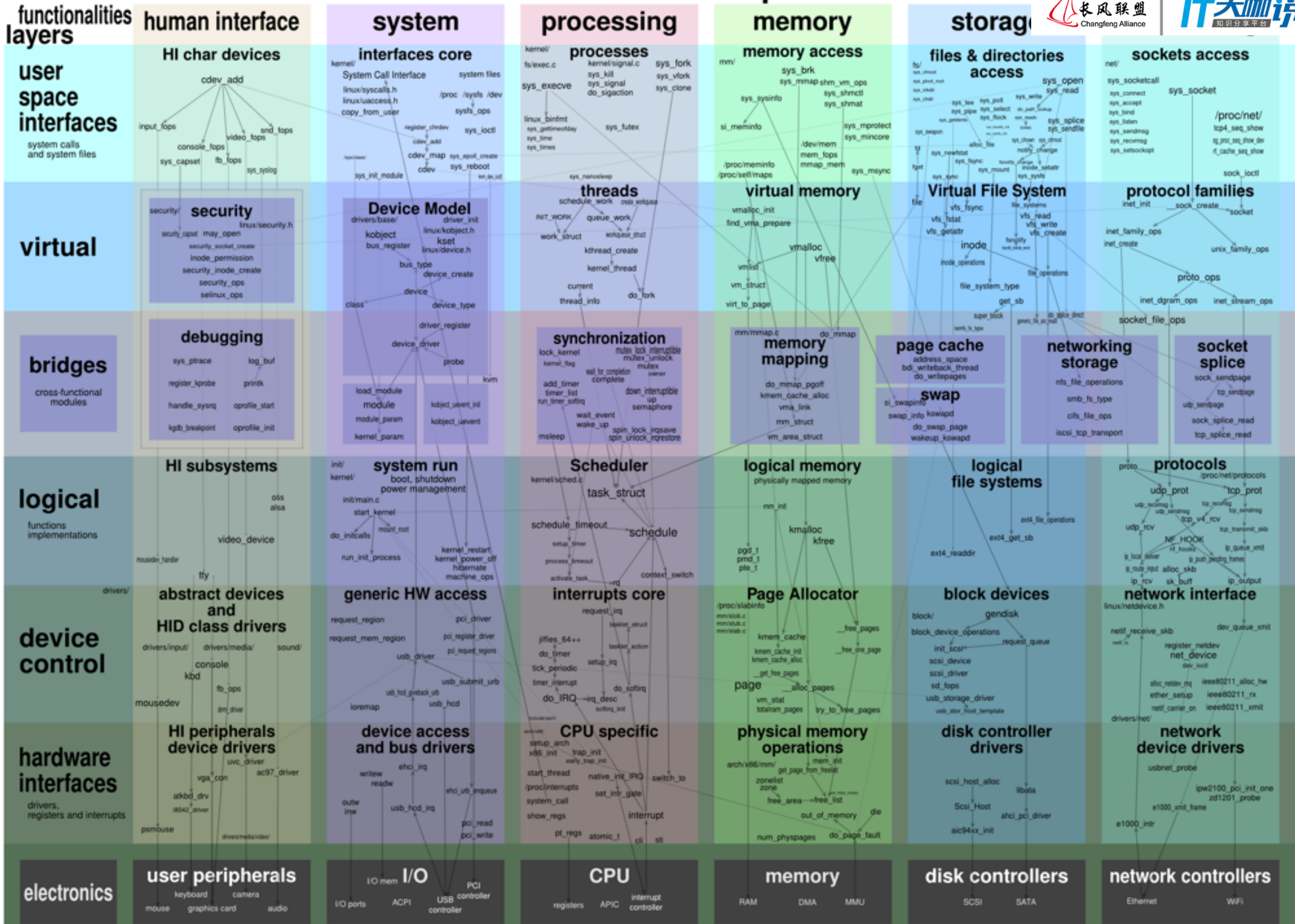


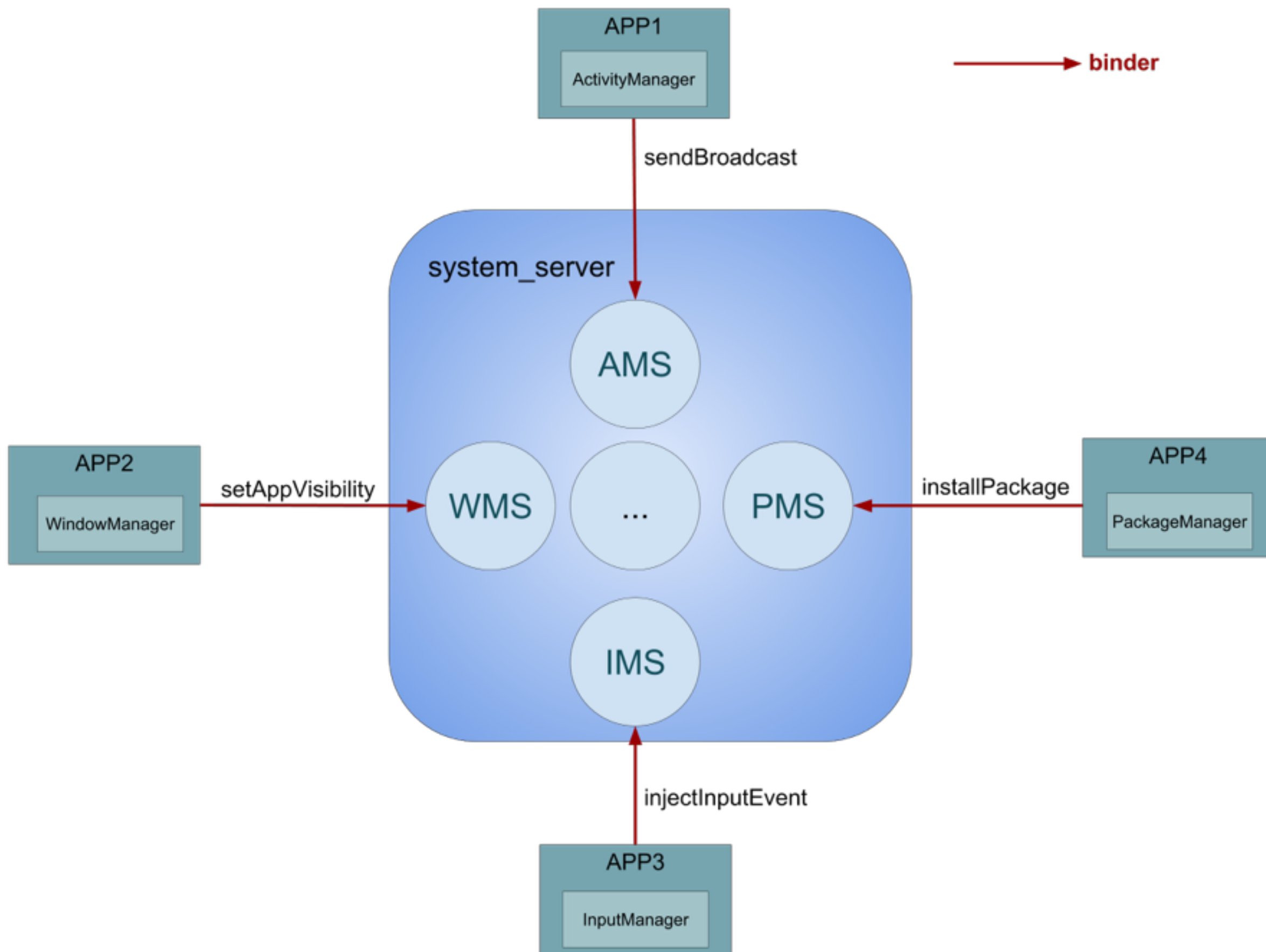
Android Binder设计之道

袁辉辉 GITYUAN



Linux kernel map





提纲

- ▶ Binder概述
- ▶ Binder设计思想
- ▶ Binder通信协议
- ▶ Binder死亡通知
- ▶ Binder线程池管理

提纲

- ▶ Binder概述
- ▶ Binder设计思想
- ▶ Binder通信协议
- ▶ Binder死亡通知
- ▶ Binder线程池管理

Binder 概述

1991, George Hoffman, OpenBinder



Later, Dinnie Hackborn, ParmOS

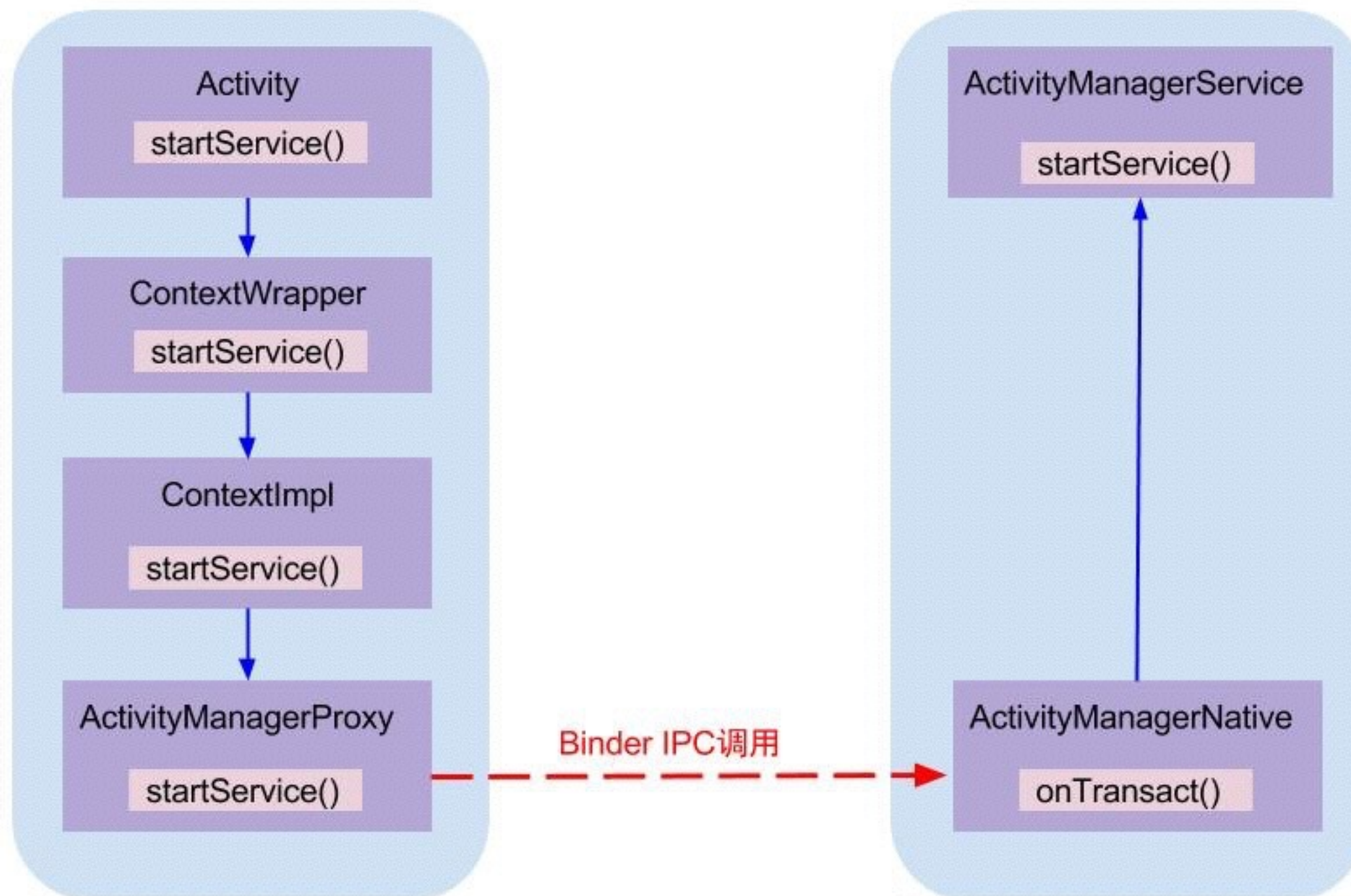


2003, Andy Rubin, Android OS

Binder 概述

- ▶ AIDL
- ▶ startService/ bindService
- ▶ send broadcast
- ▶ get contentProvider
- ▶ ...

例子: StartService(intent)



例子：StartService(intent)

```

public ComponentName startService(IApplicationThread caller, Intent service,
    String resolvedType, String callingPackage, int userId) throws RemoteException
{
    Parcel data = Parcel.obtain();
    Parcel reply = Parcel.obtain();
    data.writeInterfaceToken(IActivityManager.descriptor);
    data.writeStrongBinder(caller != null ? caller.asBinder() : null);
    service.writeToParcel(data, flags: 0);
    data.writeString(resolvedType);
    data.writeString(callingPackage);
    data.writeInt(userId);
    mRemote.transact(START_SERVICE_TRANSACTION, data, reply, flags: 0);
    reply.readException();
    ComponentName res = ComponentName.readFromParcel(reply);
    data.recycle();
    reply.recycle();
    return res;
}

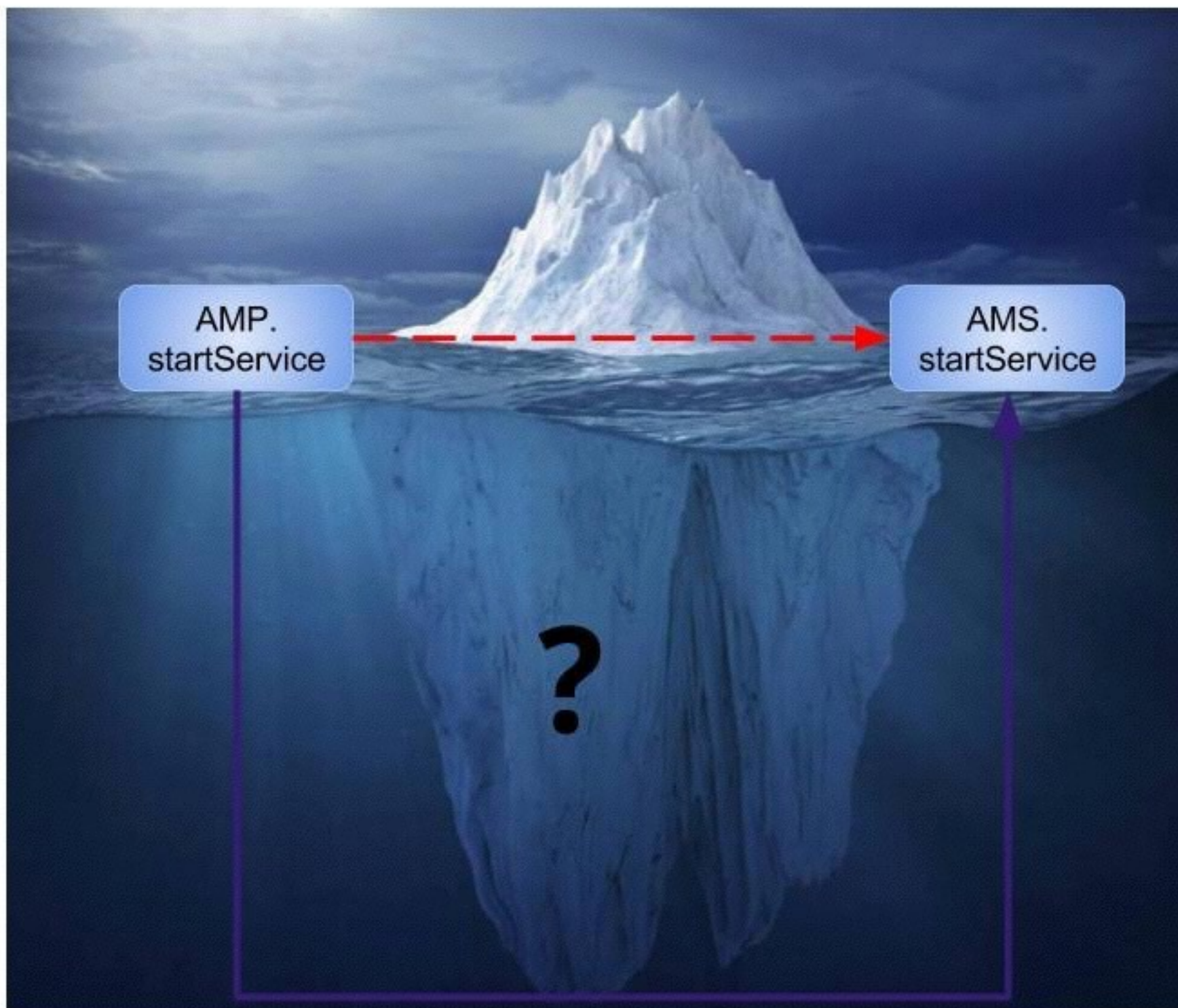
```

```

case START_SERVICE_TRANSACTION: {
    data.enforceInterface(IActivityManager.descriptor);
    IBinder b = data.readStrongBinder();
    IApplicationThread app = ApplicationThreadNative.asInterface(b);
    Intent service = Intent.CREATOR.createFromParcel(data);
    String resolvedType = data.readString();
    String callingPackage = data.readString();
    int userId = data.readInt();
    ComponentName cn = startService(app, service, resolvedType, callingPackage, userId);
    reply.writeNoException();
    ComponentName.writeToParcel(cn, reply);
    return true;
}

```


例子：StartService(intent)



相关概念

- ▶ BpBinder, BBinder , handle
- ▶ BinderProxy
- ▶ IPCThreadState, ProcessState
- ▶ binder_proc, binder_thread
- ▶ binder_node, binder_ref
- ▶ binder_buffer, binder_transaction

提纲

- ▶ Binder概述
- ▶ Binder设计思想
- ▶ Binder通信协议
- ▶ Binder死亡通知
- ▶ Binder线程池管理

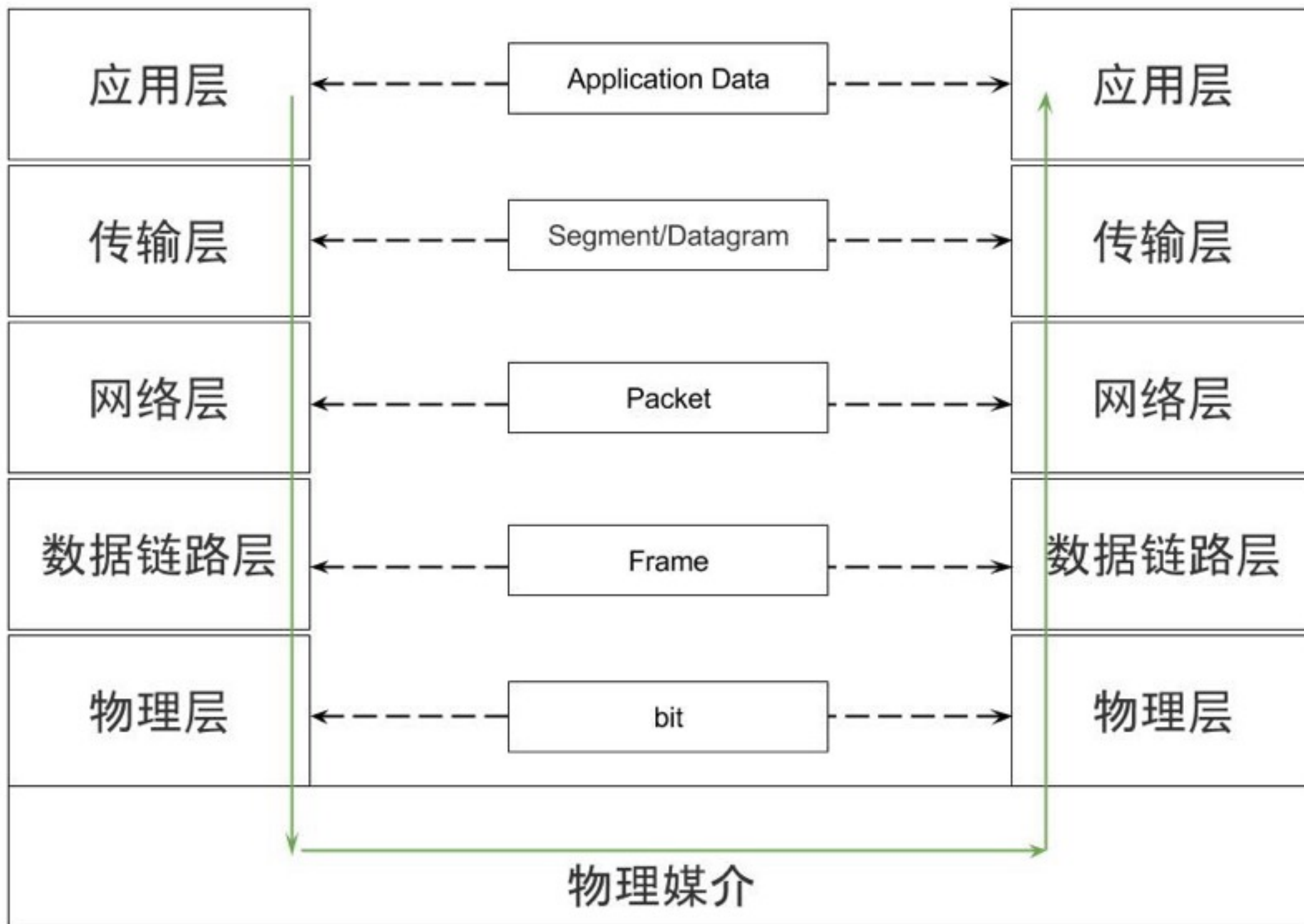
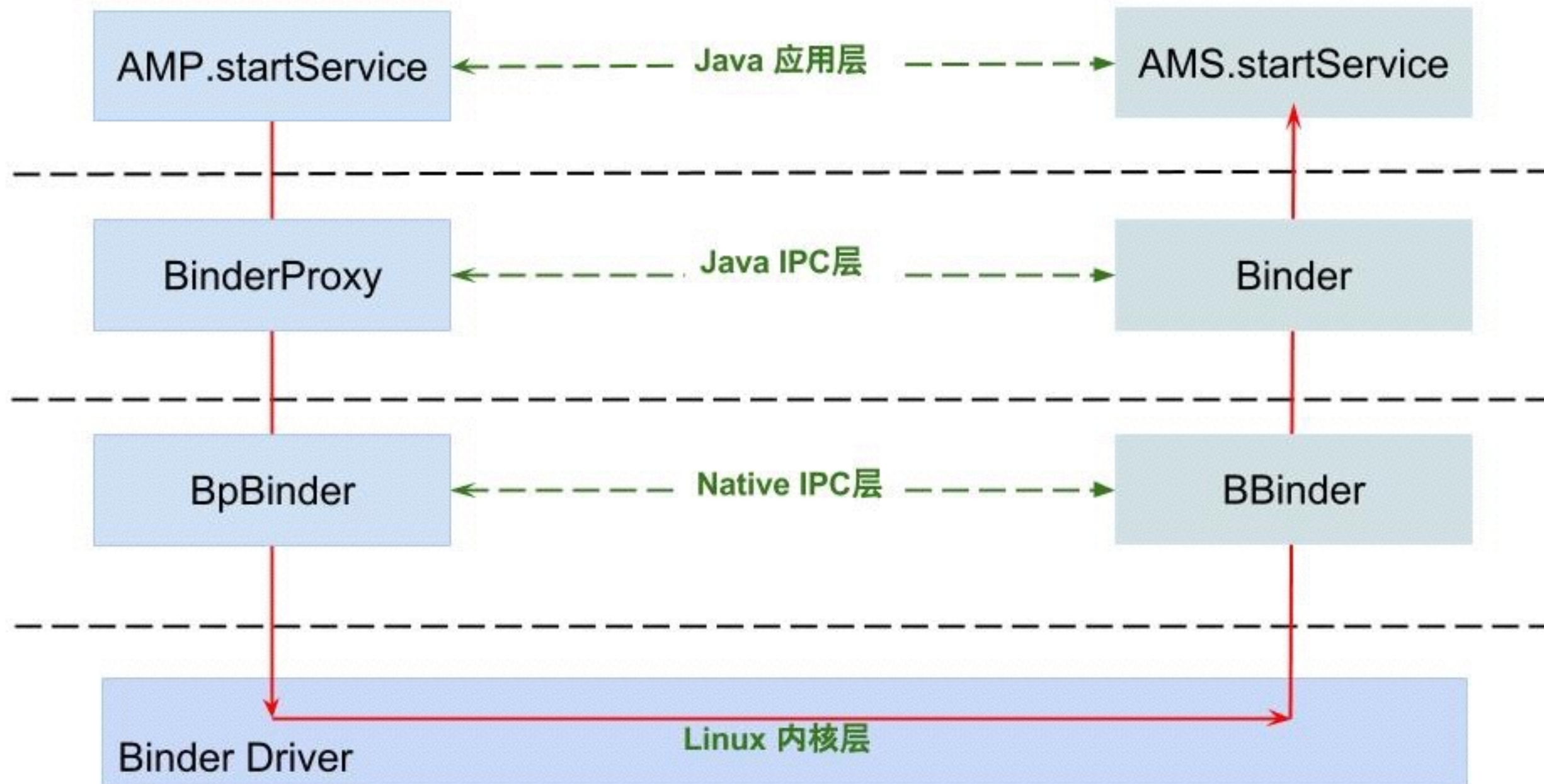


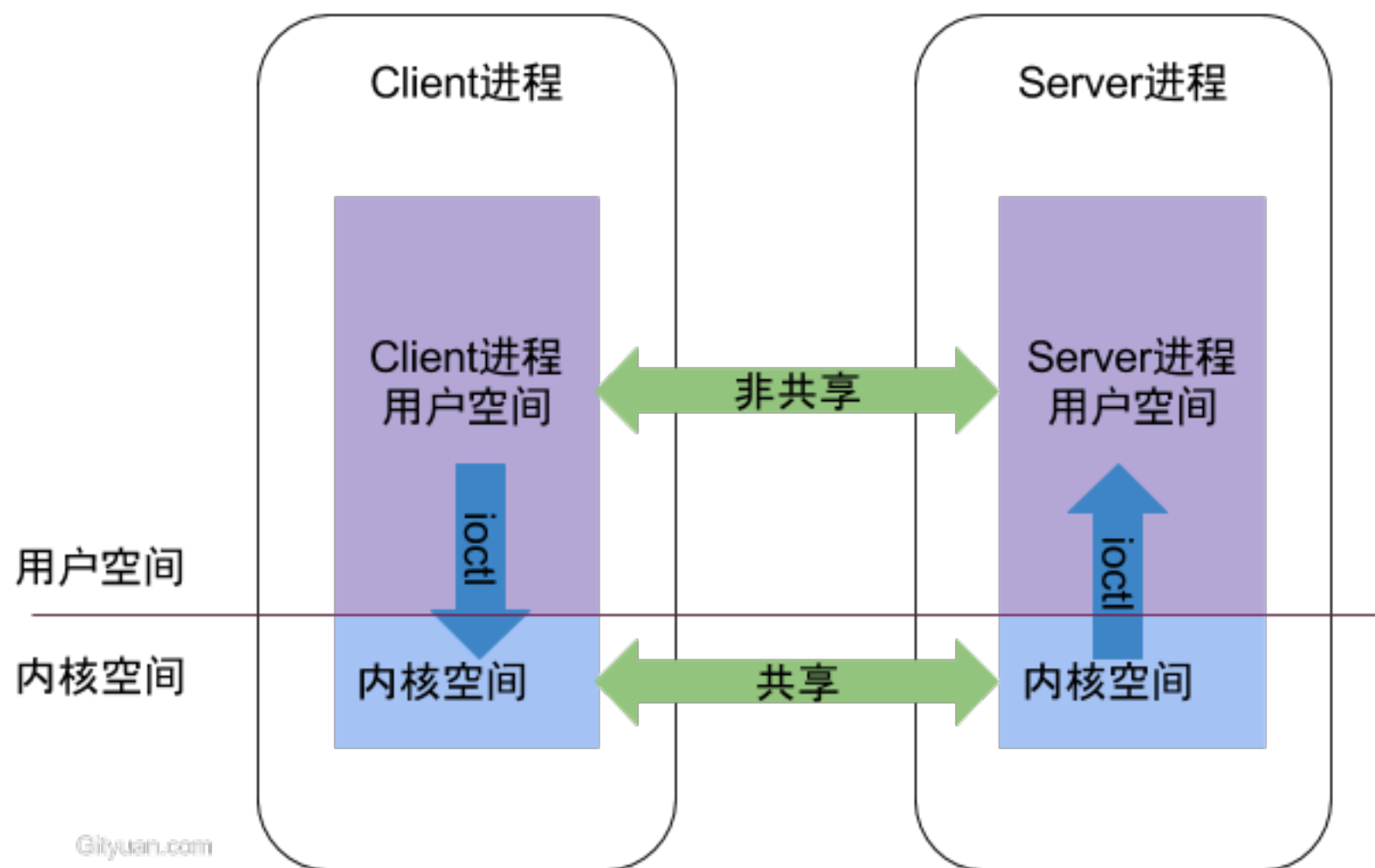
图: TCP/IP五层模型结构

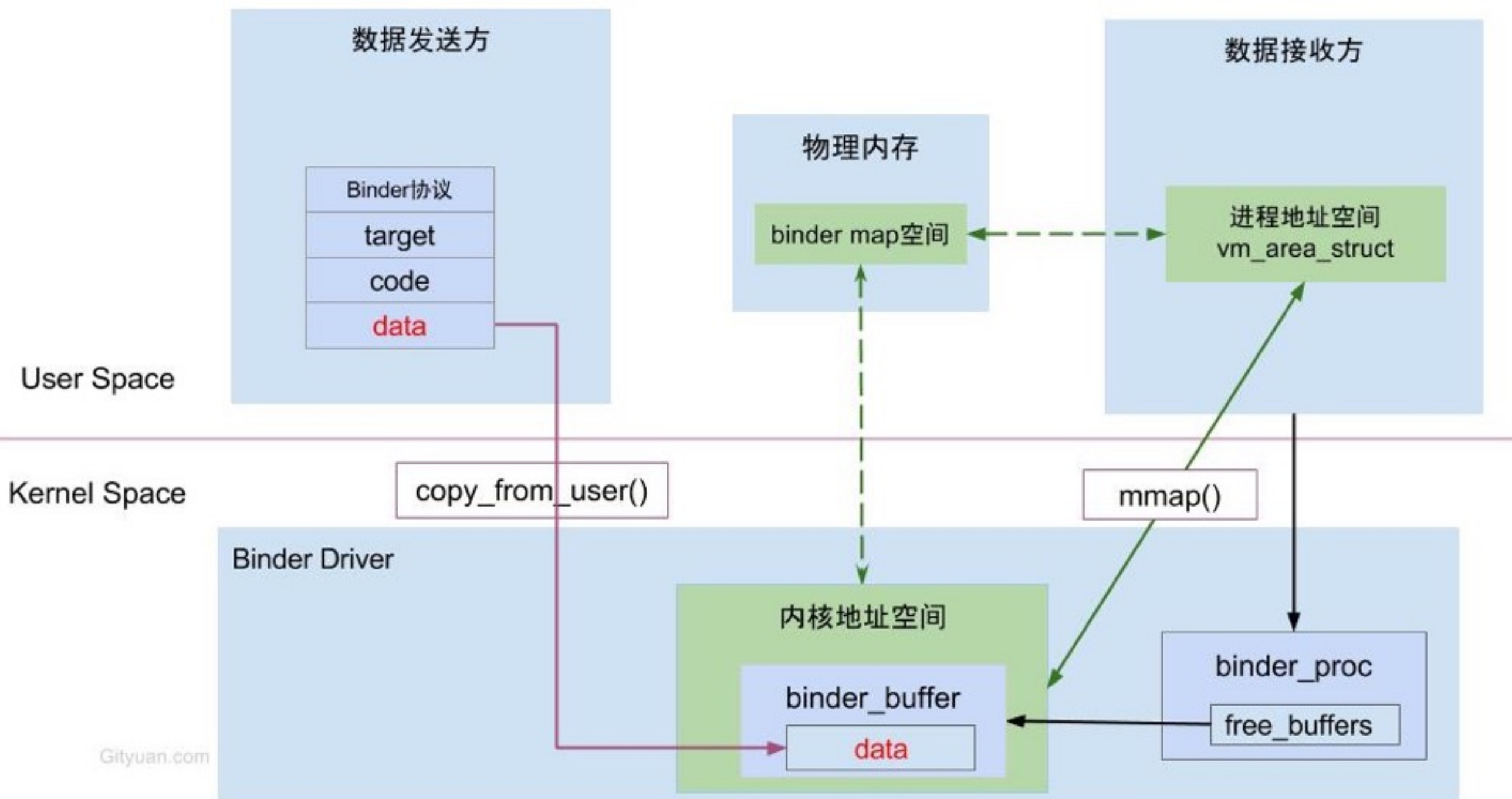


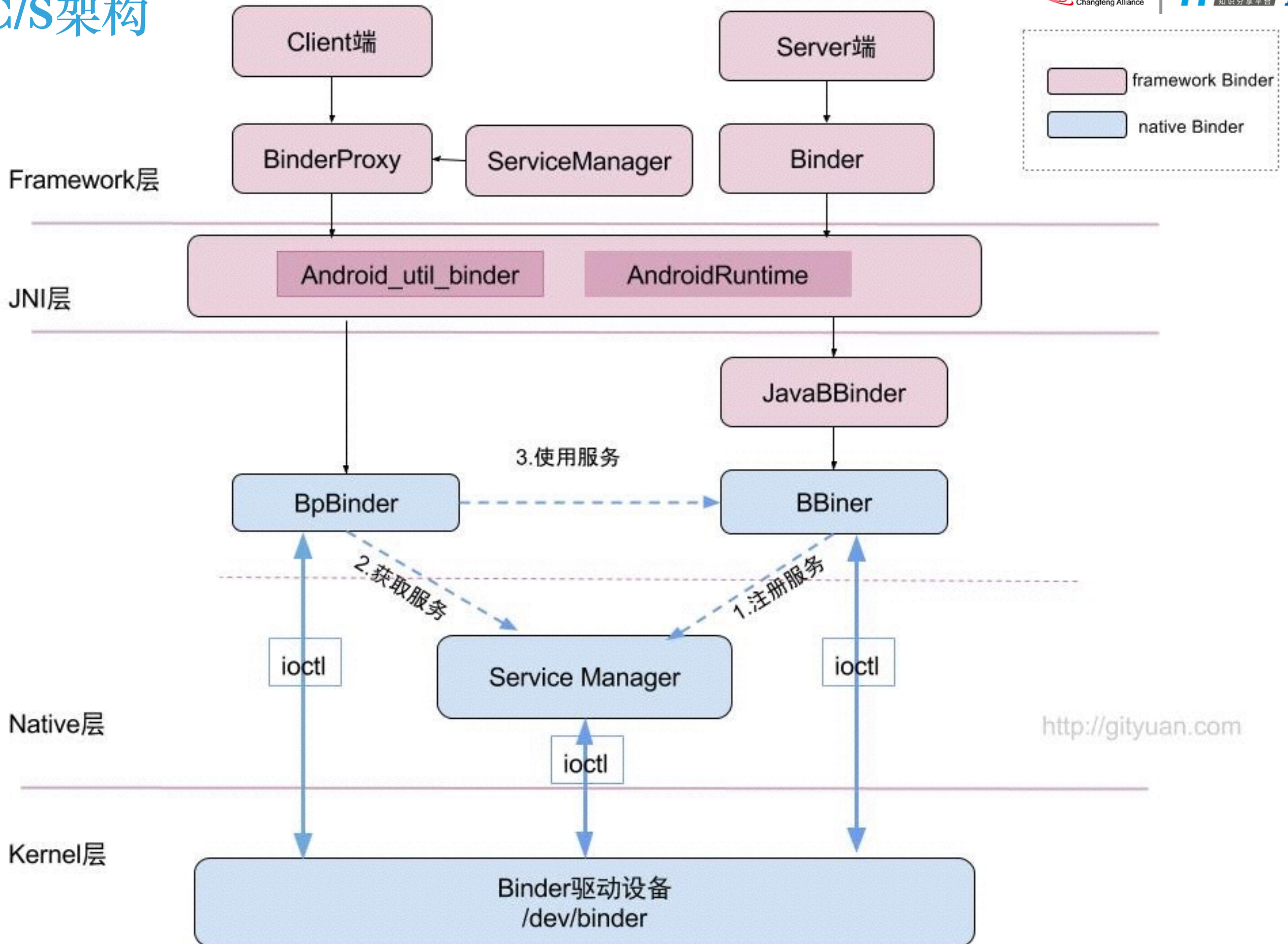
内存

IPC方式	数据拷贝次数
管道、消息队列、Socket	2
Binder	1
共享内存	0

内存





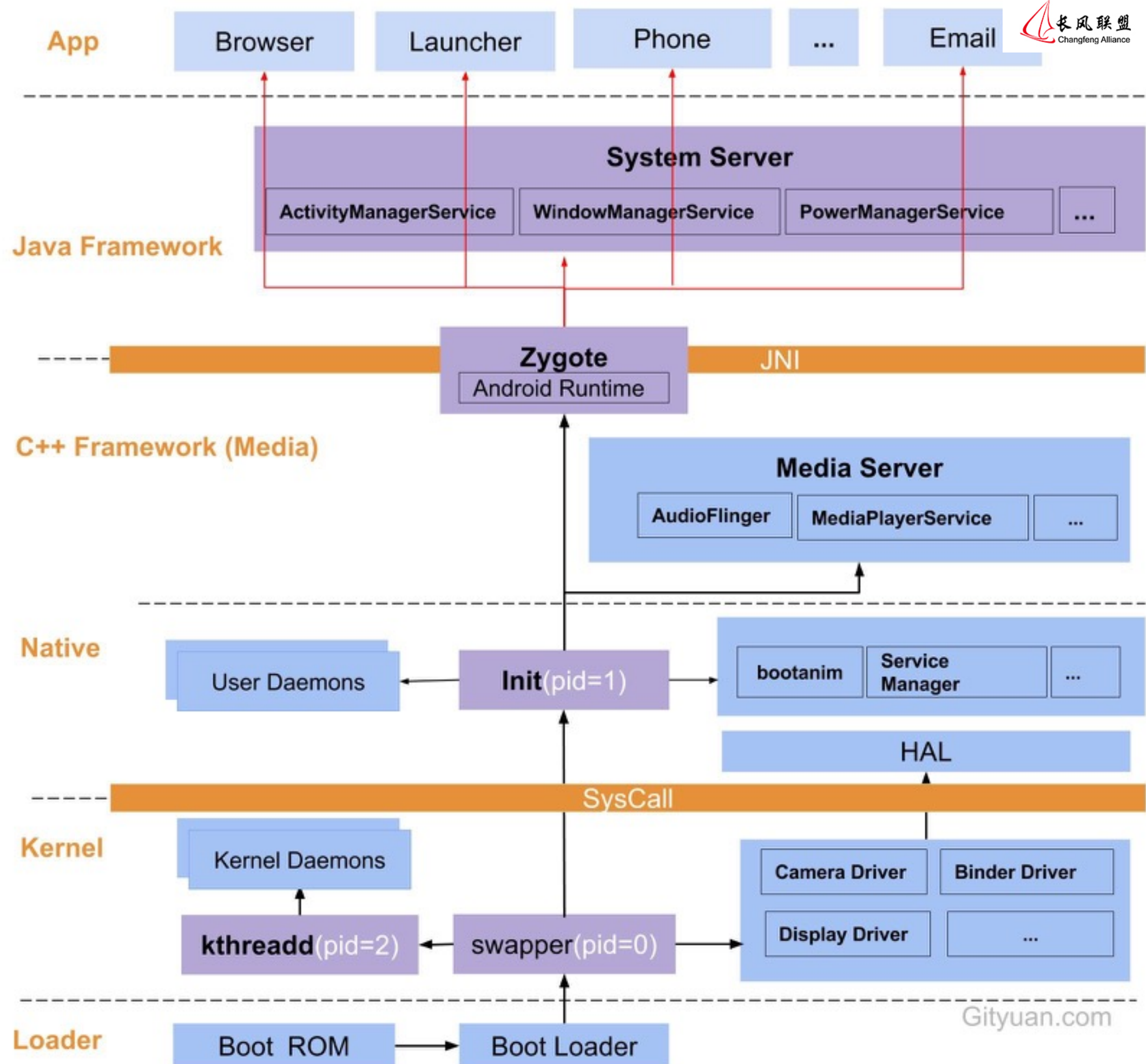


BINDER 安全性

- 传统IPC的接收方无法获得对方进程可靠的UID/PID，从而无法鉴别对方身份；
- Android在应用程序安装过程便已分配指定的UID，进程的UID是鉴别进程身份的重要标志；使用传统IPC只能由用户在数据包里填入UID/PID；
- Binder通信的身份信息PID/UID，是由Linux内核在进程通信过程填充，安全可靠。

设计角度

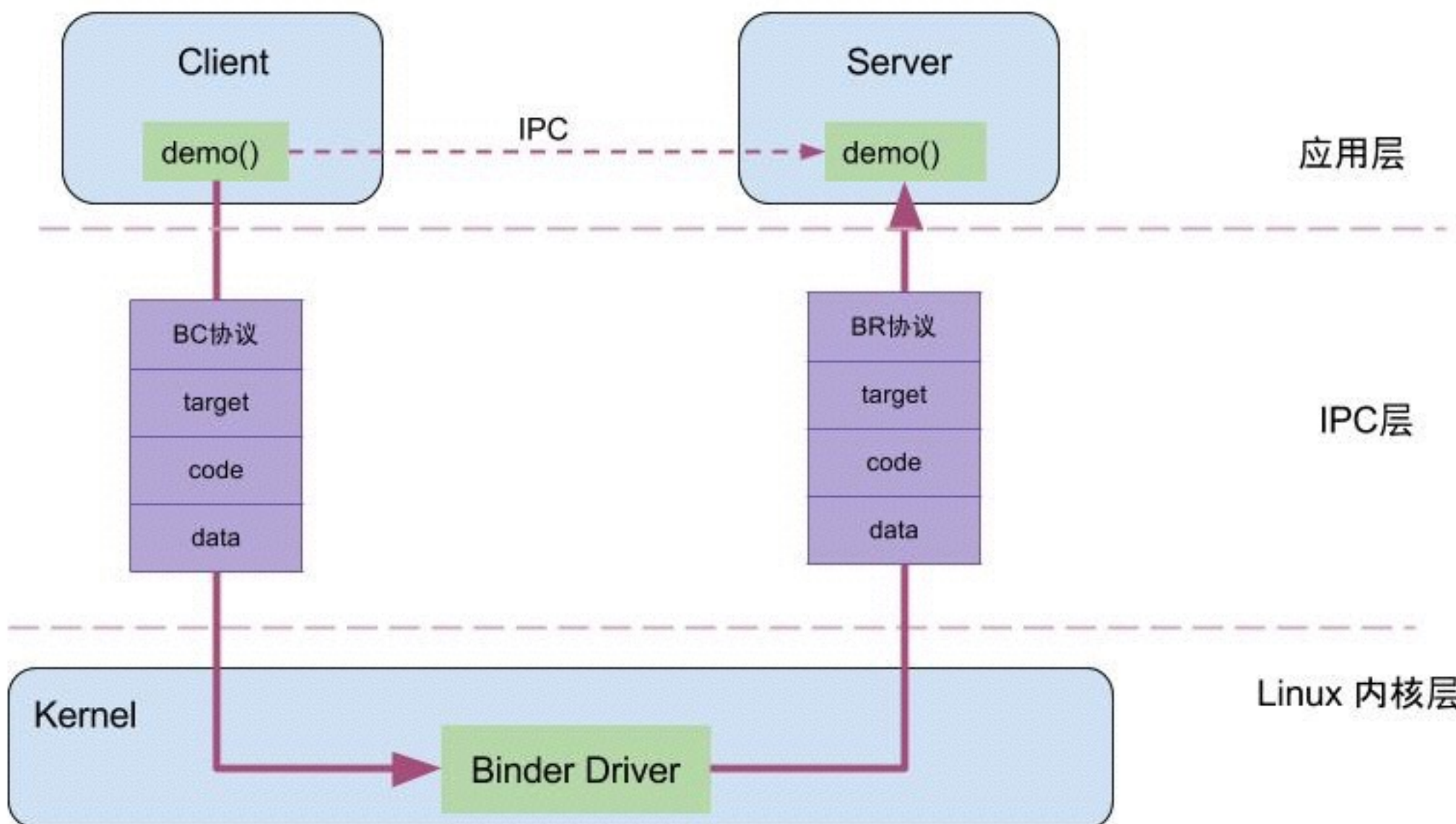
- ▶ 面向对象的设计体验
- ▶ 系统平台级的全方位支持，使用简单
- ▶ Binder模糊了进程边界，淡化了进程间通信过程，整个系统仿佛运行于同一个面向对象的程序之中



提纲

- ▶ Binder概述
- ▶ Binder设计思想
- ▶ Binder通信协议
- ▶ Binder死亡通知
- ▶ Binder线程池管理

BINDER 通信协议



实例

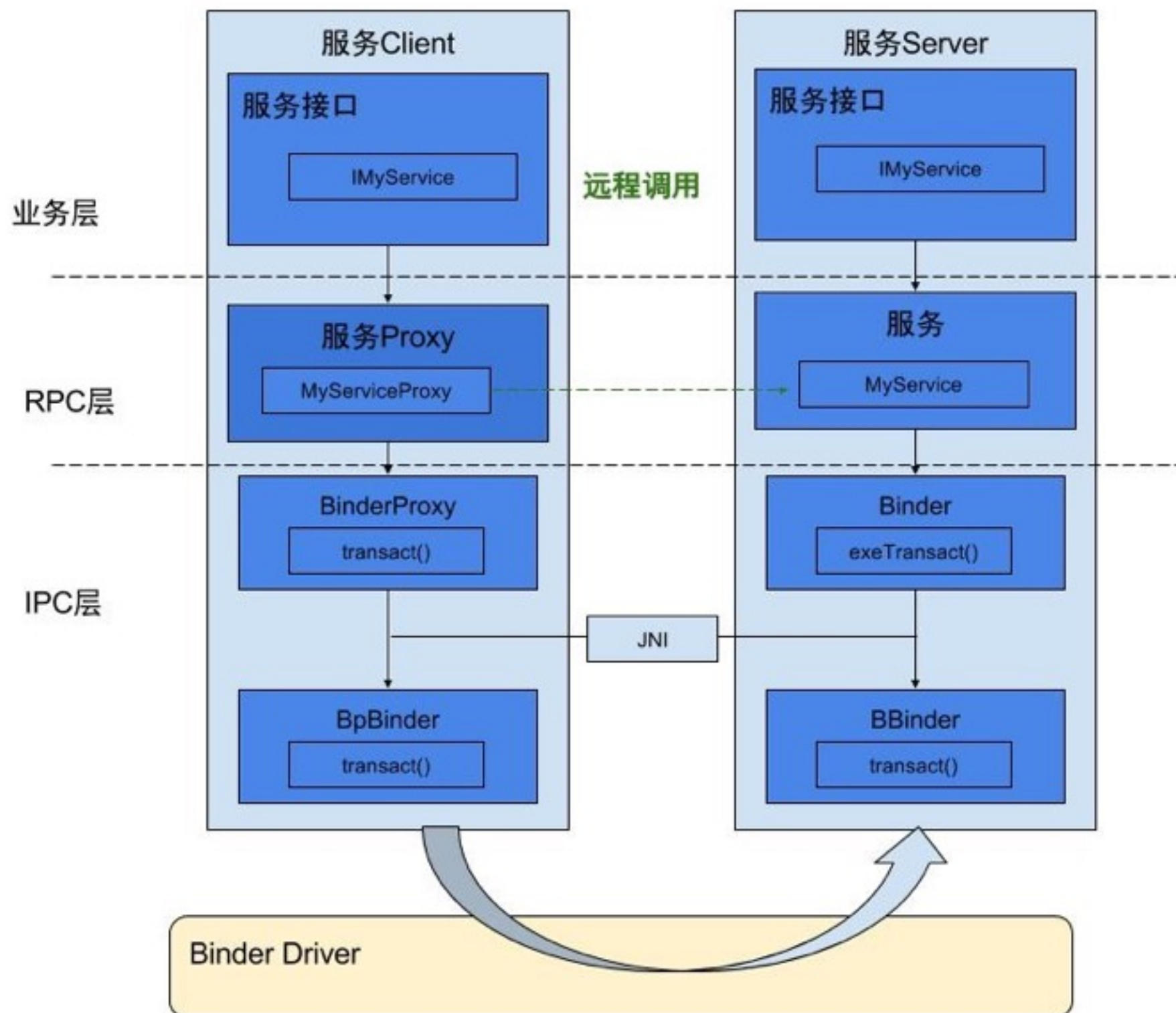
```
#include "IMyService.h"
int main() {
    //获取service manager引用
    sp < IServiceManager > sm = defaultServiceManager();
    //注册名为"service.myervice"的服务到service manager
    sm->addService(String16("service.myervice"), new BnMyService());
    ProcessState::self()->startThreadPool(); //启动线程池
    IPCThreadState::self()->joinThreadPool(); //把主线程加入线程池
    return 0;
}
```

注册服务

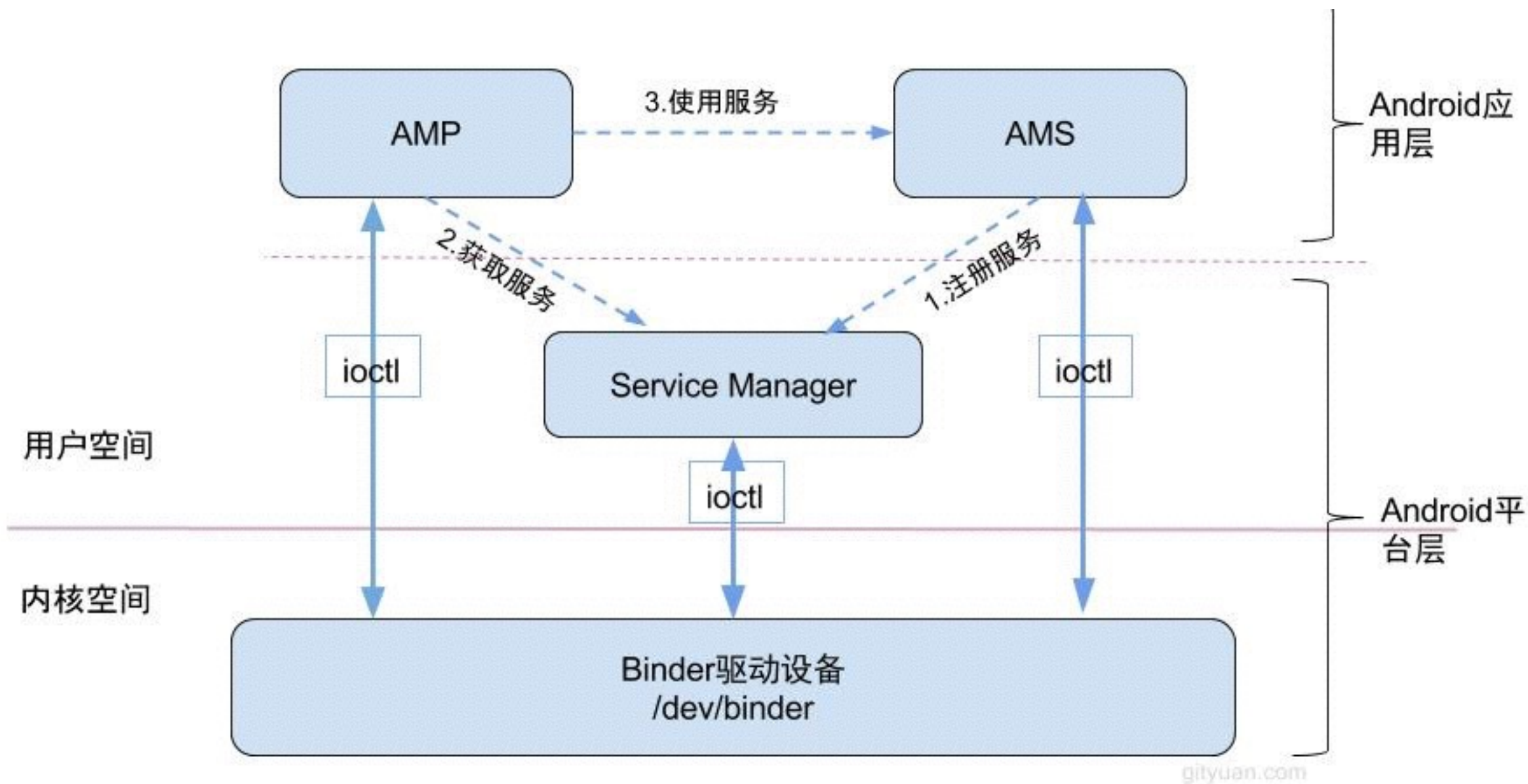
```
#include "IMyService.h"
int main() {
    //获取service manager引用
    sp < IServiceManager > sm = defaultServiceManager();
    //获取名为"service.myervice"的binder接口
    sp < IBinder > binder = sm->getService(String16("service.myervice"));
    //将binder对象转换为强引用类型的IMyService
    sp<IMyService> cs = interface_cast < IMyService > (binder);
    //利用binder引用调用远程sayHello()方法
    cs->sayHello();
    return 0;
}
```

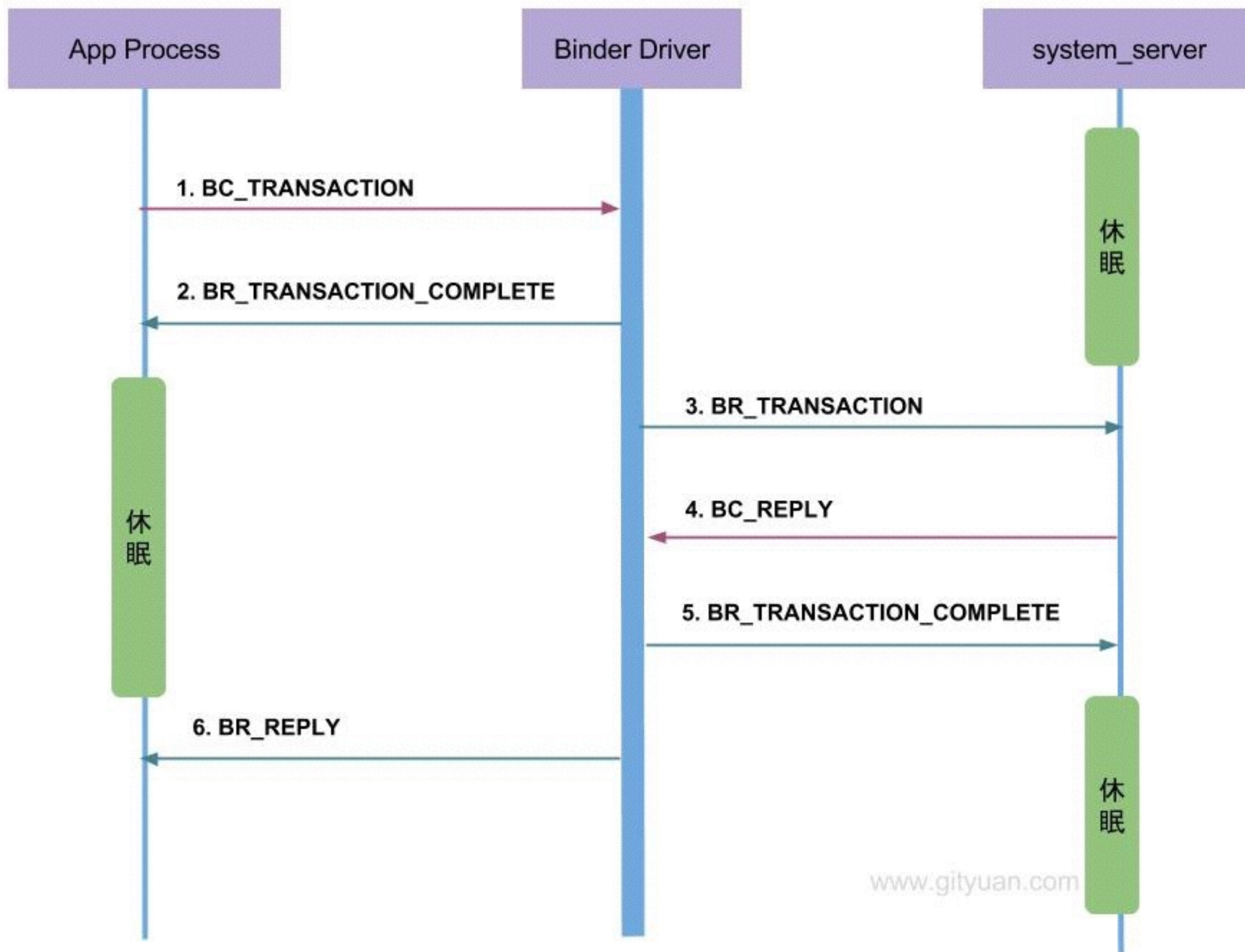
请求服务

实例

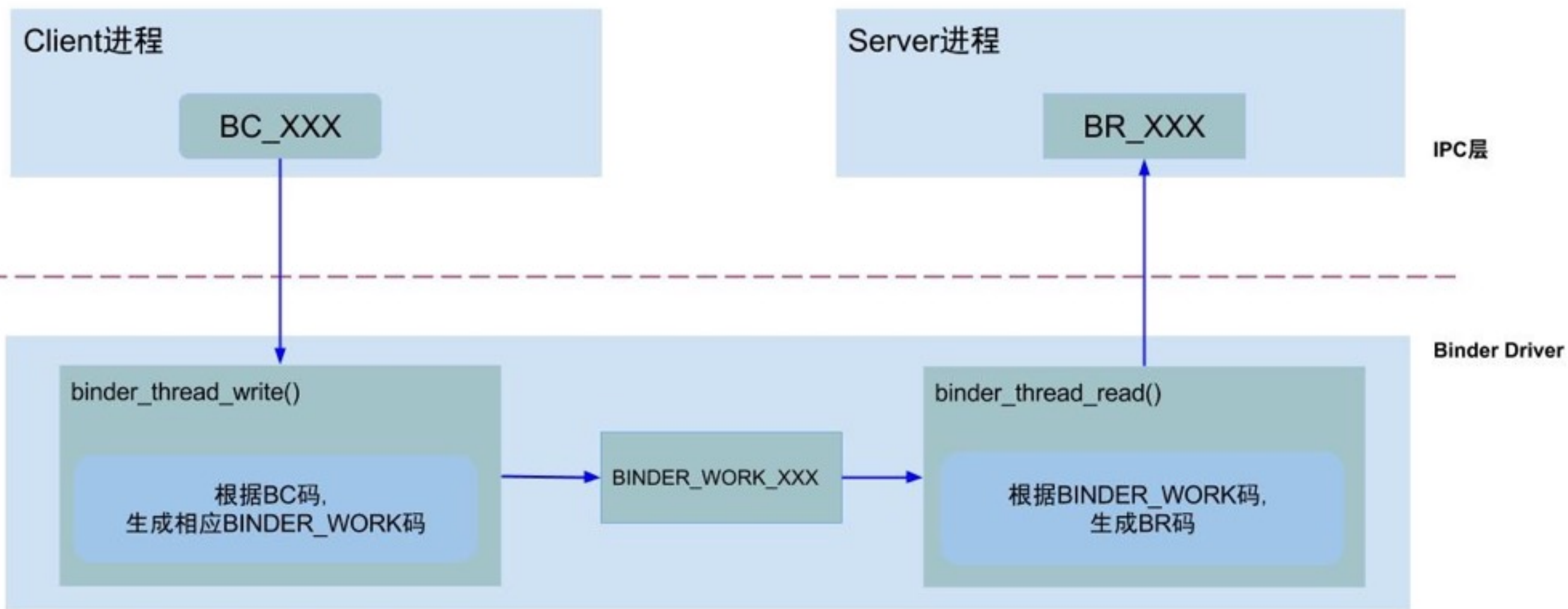


BINDER 通信协议





BINDER 通信协议



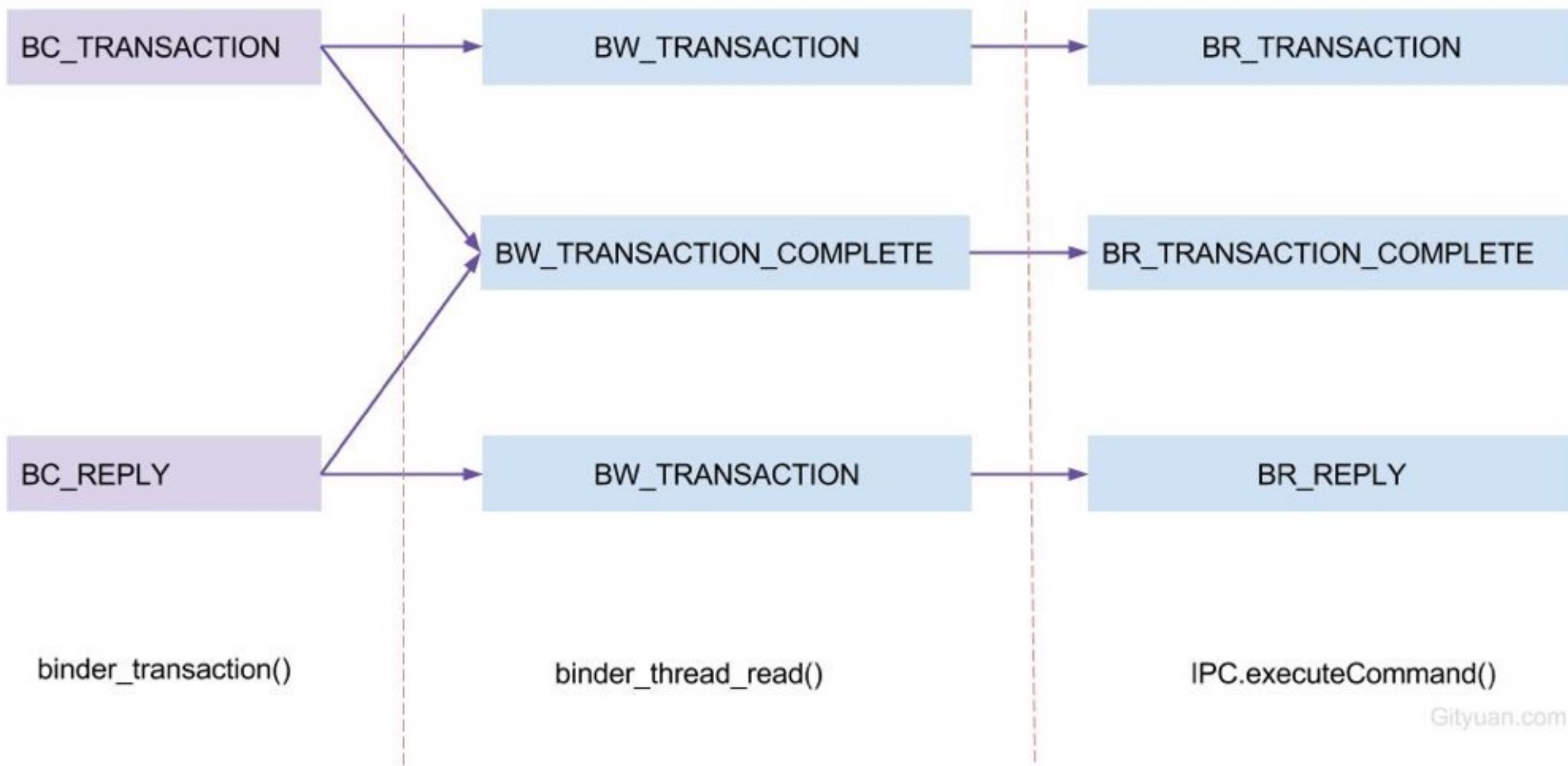
BC协议

BC协议	调用方法
BC_TRANSACTION	IPC.transact()
BC_REPLY	IPC.sendReply()
BC_FREE_BUFFER	IPC.freeBuffer()
BC_REQUEST_DEATH_NOTIFICATION	IPC.requestDeathNotification()
BC_CLEAR_DEATH_NOTIFICATION	IPC.clearDeathNotification()
BC_DEAD_BINDER_DONE	IPC.execute()

BR协议

BR协议	触发时机
BR_TRANSACTION	收到BINDER_WORK_TRANSACTION
BR_REPLY	收到BINDER_WORK_TRANSACTION
BR_TRANSACTION_COMPLETE	收到BINDER_WORK_TRANSACTION_COMPLETE
BR_DEAD_BINDER	收到BINDER_WORK_DEAD_BINDER或 BINDER_WORK_DEAD_BINDER_AND_CLEAR
BR_CLEAR_DEATH_NOTIFICATION_DONE	收到BINDER_WORK_CLEAR_DEATH_NOTIFICATION

协议转换

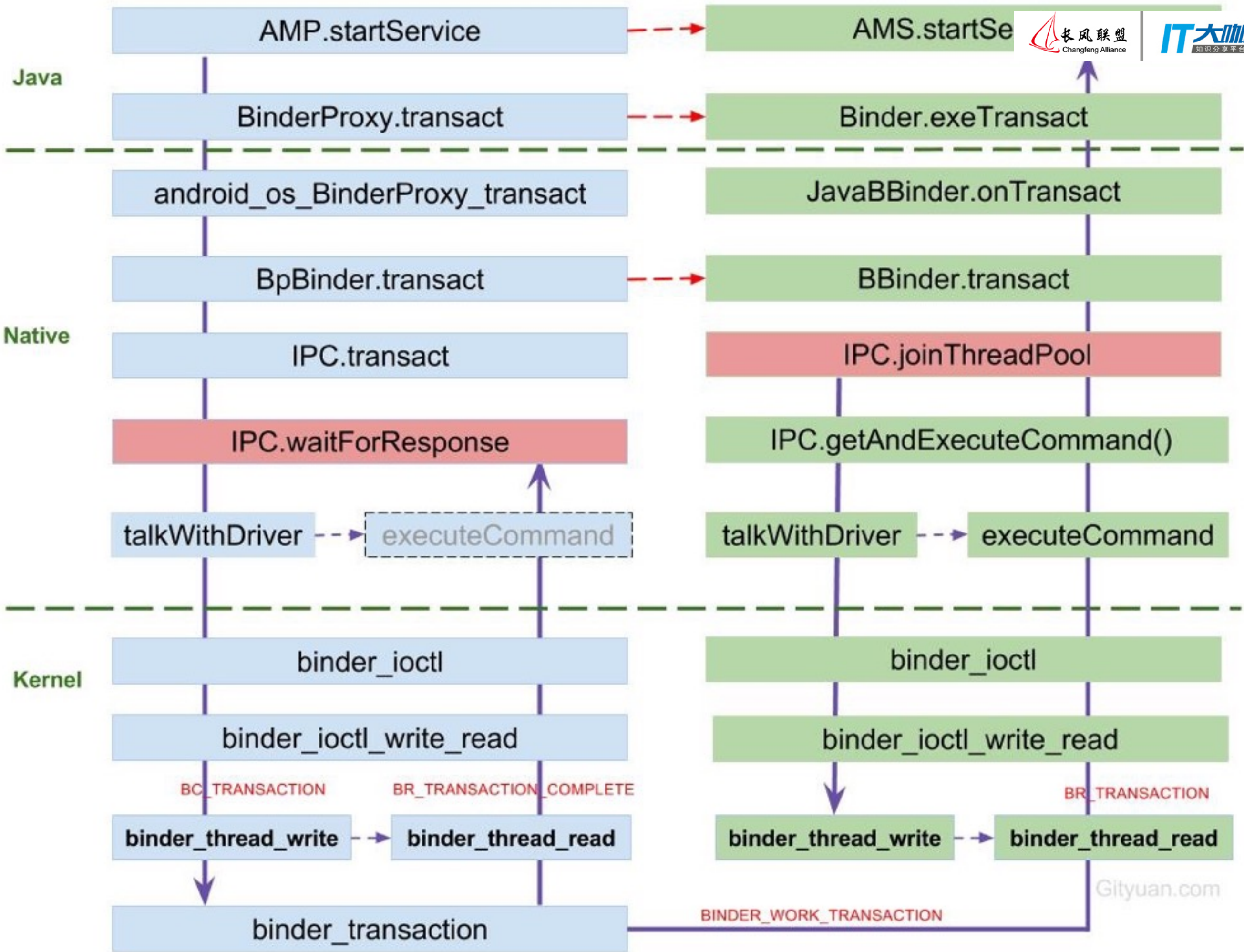


BINDER调用栈

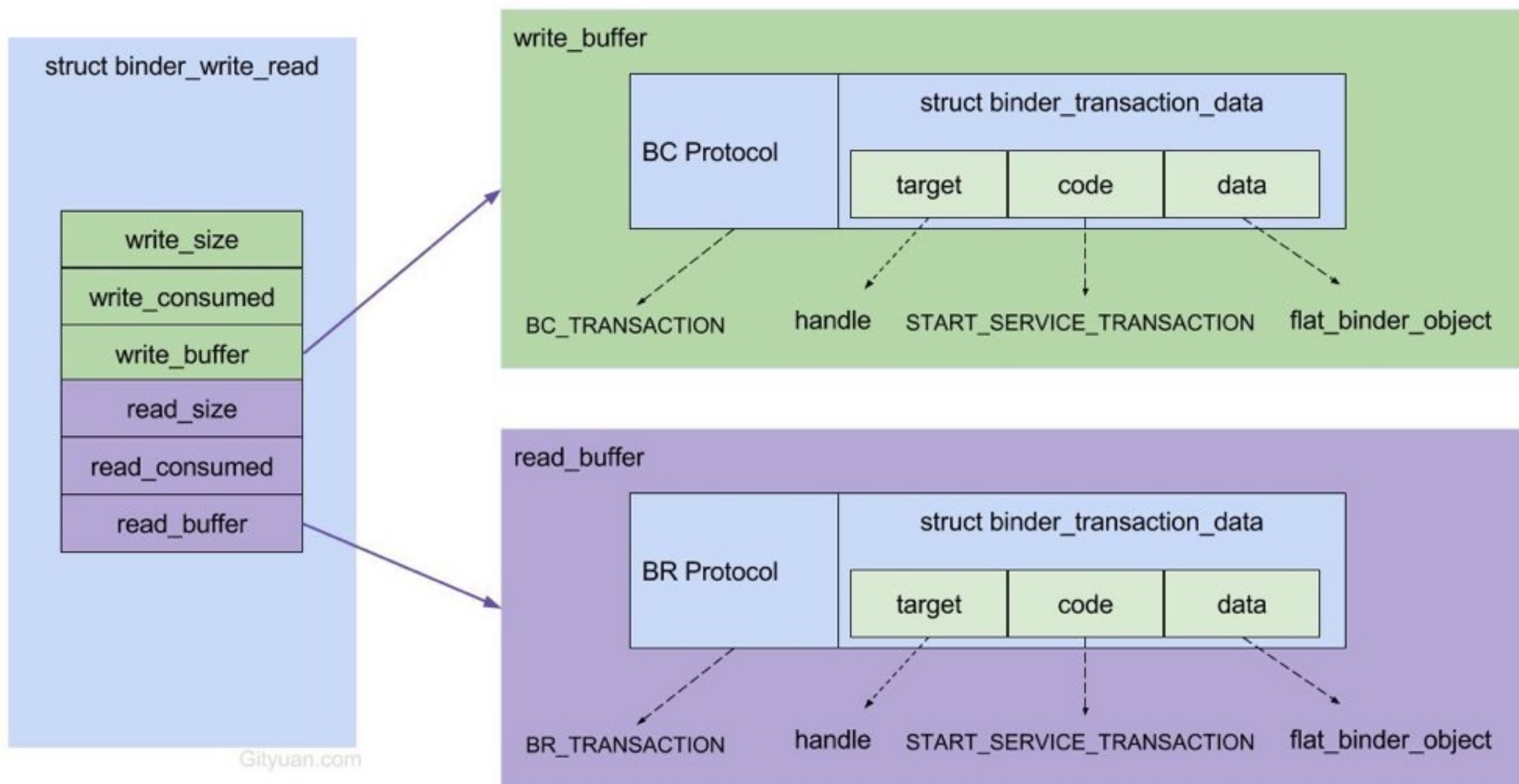
```

"Binder:1557_16" prio=5 tid=117 Blocked
  | group="main" sCount=1 dsCount=0 obj=0x153120d0 self=0x7f89b9ee00
  | sysTid=10642 nice=0 cgrp=default sched=0/0 handle=0x7f7352c450
  | state=S schedstat=( 0 0 0 ) utm=2748 stm=1377 core=1 HZ=100
  | stack=0x7f73432000-0x7f73434000 stackSize=1005KB
  | held mutexes=
  at com.android.server.am.ActivityManagerService.checkContentProviderAccess(ActivityManagerService.java:10786)
  - waiting to lock <0x0ef02a62> (a com.android.server.am.ActivityManagerService) held by thread 101
  at com.android.server.am.ActivityManagerService$LocalService.checkContentProviderAccess(ActivityManagerService.java:22502)
  at com.android.server.content.ContentService.registerContentObserver(ContentService.java:302)
  at android.content.IContentService$Stub.onTransact(IContentService.java:73)
  at com.android.server.content.ContentService.onTransact(ContentService.java:235)
  at android.os.Binder.execTransact(Binder.java:565)

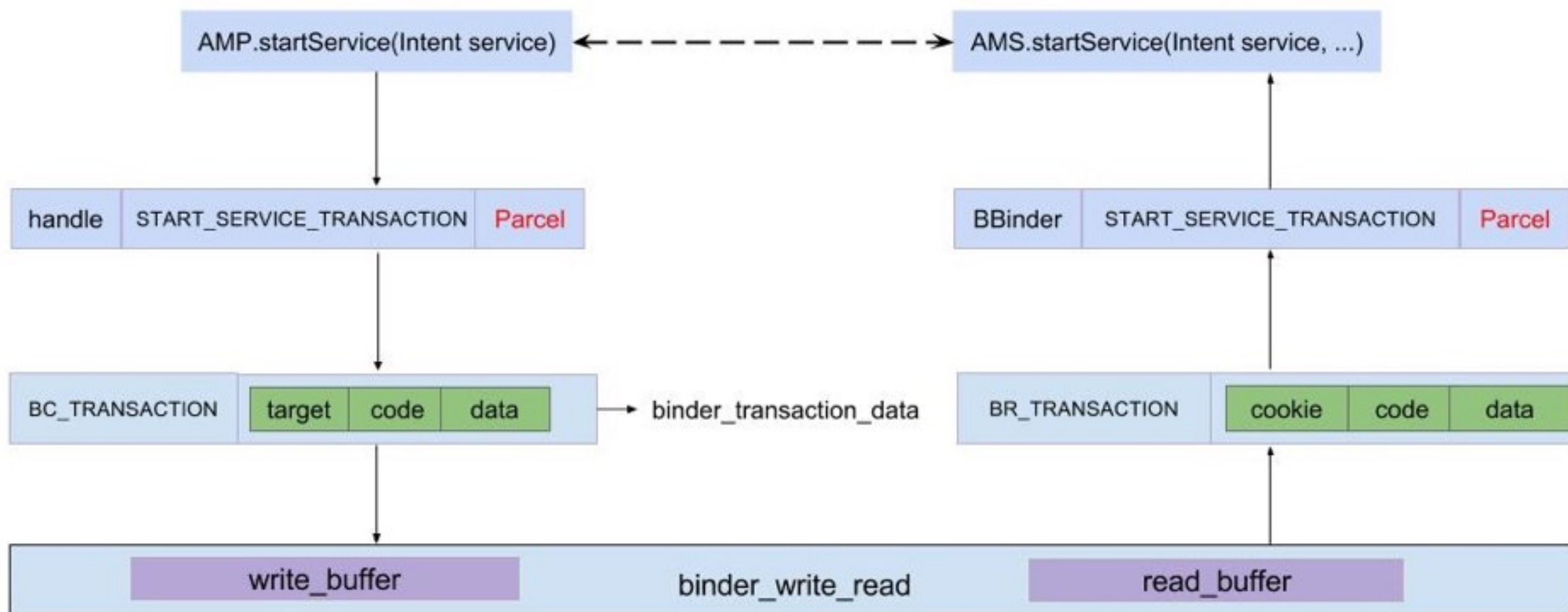
"Binder:1557_17" prio=5 tid=119 Native
  | group="main" sCount=1 dsCount=0 obj=0x15312a60 self=0x7f8a542600
  | sysTid=10643 nice=0 cgrp=default sched=0/0 handle=0x7f7342f450
  | state=S schedstat=( 0 0 0 ) utm=2781 stm=1306 core=1 HZ=100
  | stack=0x7f73335000-0x7f73337000 stackSize=1005KB
  | held mutexes=
  kernel: (couldn't read /proc/self/task/10643/stack)
  native: #00 pc 000000000007831c /system/lib64/libc.so (__ioctl+4)
  native: #01 pc 0000000000020020 /system/lib64/libc.so (ioctl+140)
  native: #02 pc 0000000000055b14 /system/lib64/libbinder.so (_ZN7android14IPCThreadState14talkWithDriverEb+256)
  native: #03 pc 0000000000055c7c /system/lib64/libbinder.so (_ZN7android14IPCThreadState20getAndExecuteCommandEv+24)
  native: #04 pc 00000000000563f8 /system/lib64/libbinder.so (_ZN7android14IPCThreadState14joinThreadPoolEb+128)
  native: #05 pc 00000000000747e4 /system/lib64/libbinder.so (???)
  native: #06 pc 0000000000012938 /system/lib64/libutils.so (_ZN7android6Thread11_threadLoopEPv+336)
  native: #07 pc 00000000000a0c88 /system/lib64/libandroid_runtime.so (_ZN7android14AndroidRuntime15javaThreadShellEPv+116)
  native: #08 pc 0000000000075c54 /system/lib64/libc.so (_ZL15__pthread_startPv+204)
  native: #09 pc 000000000001e0fc /system/lib64/libc.so (__start_thread+16)
  /...
  
```



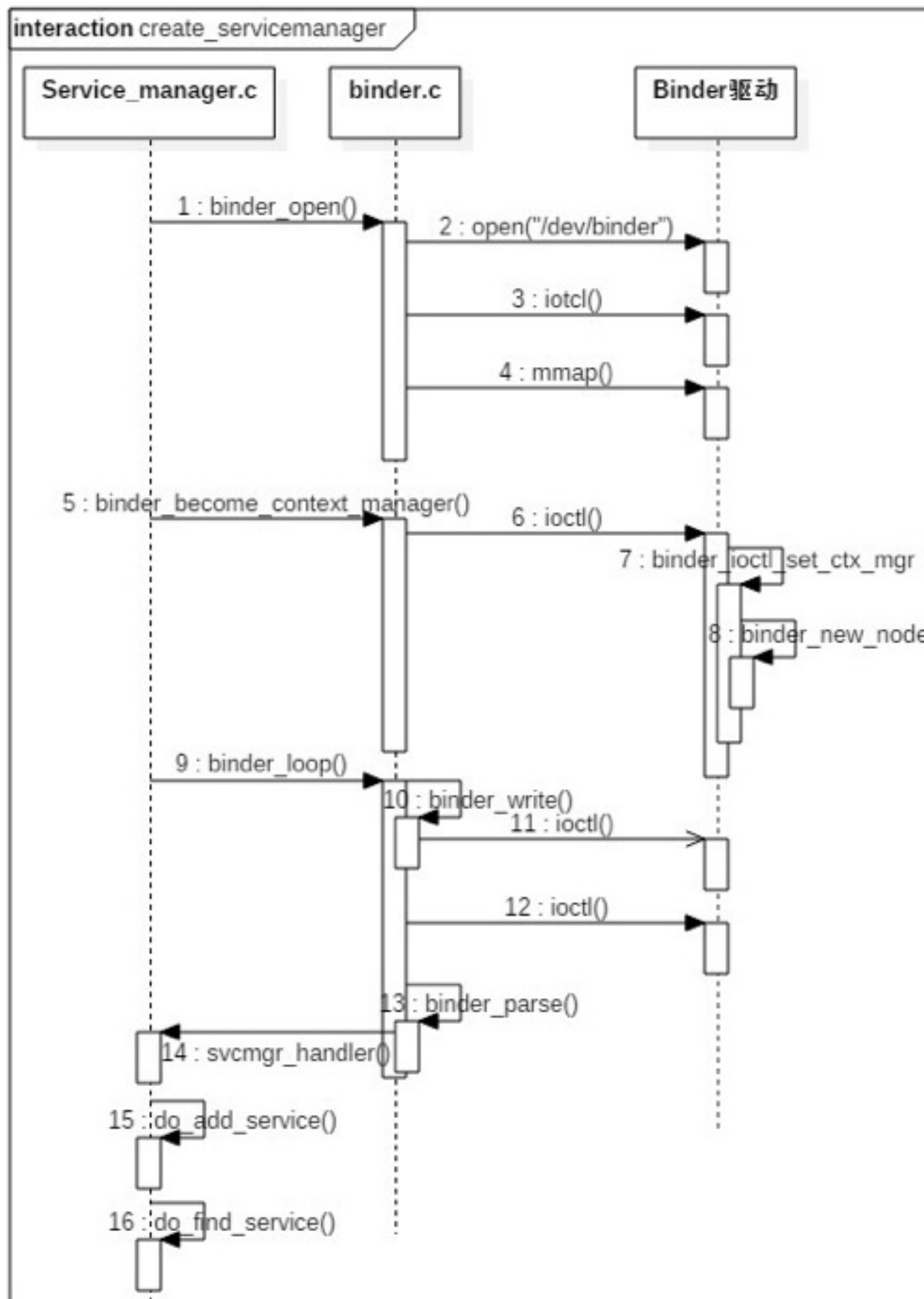
数据加工



数据加工



启动大管家



驱动加载

通过init(), 创建/dev/binder设备节点



通过open(), 获取Binder Driver的文件描述符

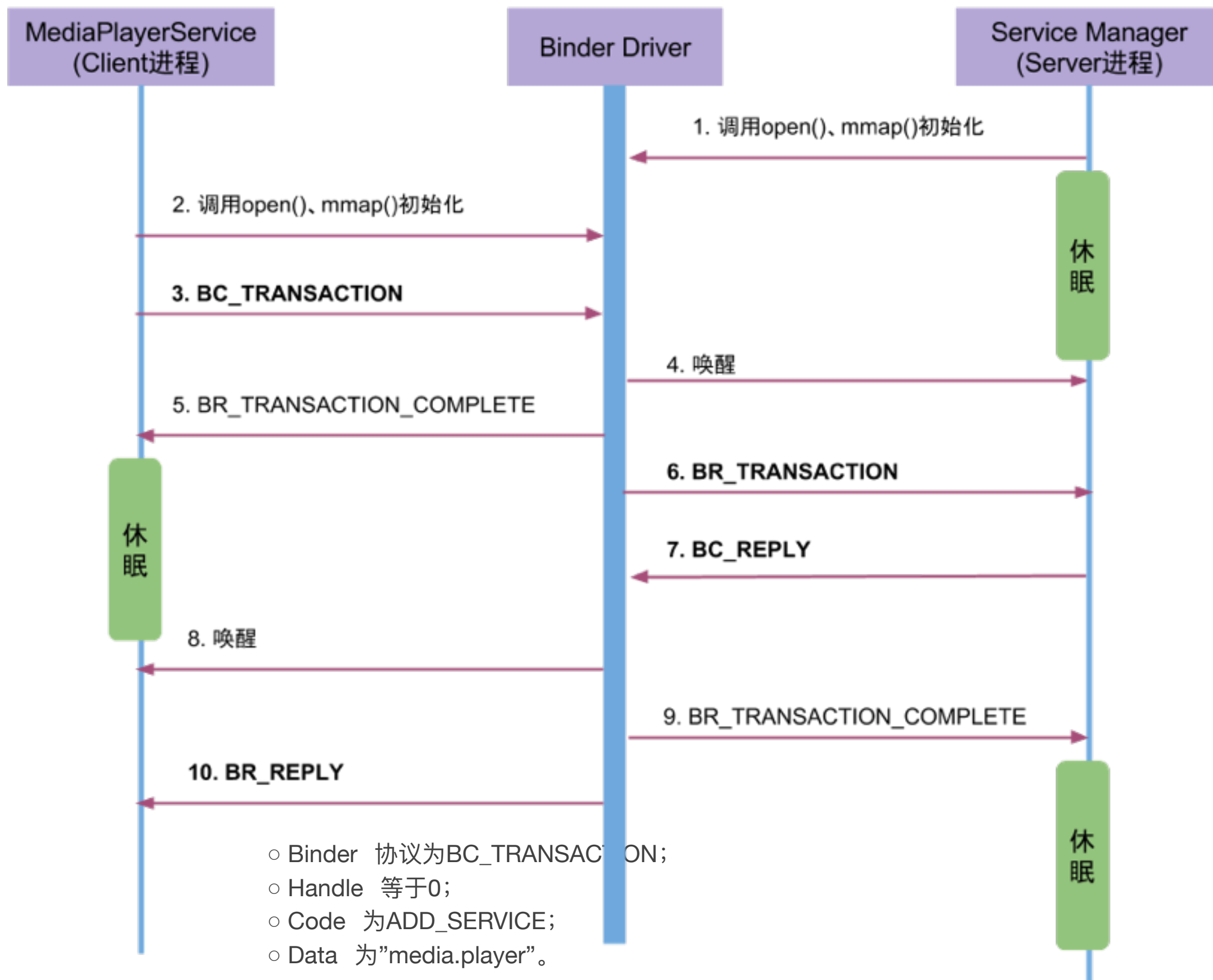


通过mmap(), 在内核分配一块内存, 用于存放数据



通过ioctl(), 将IPC数据作为参数传递给Binder Driver

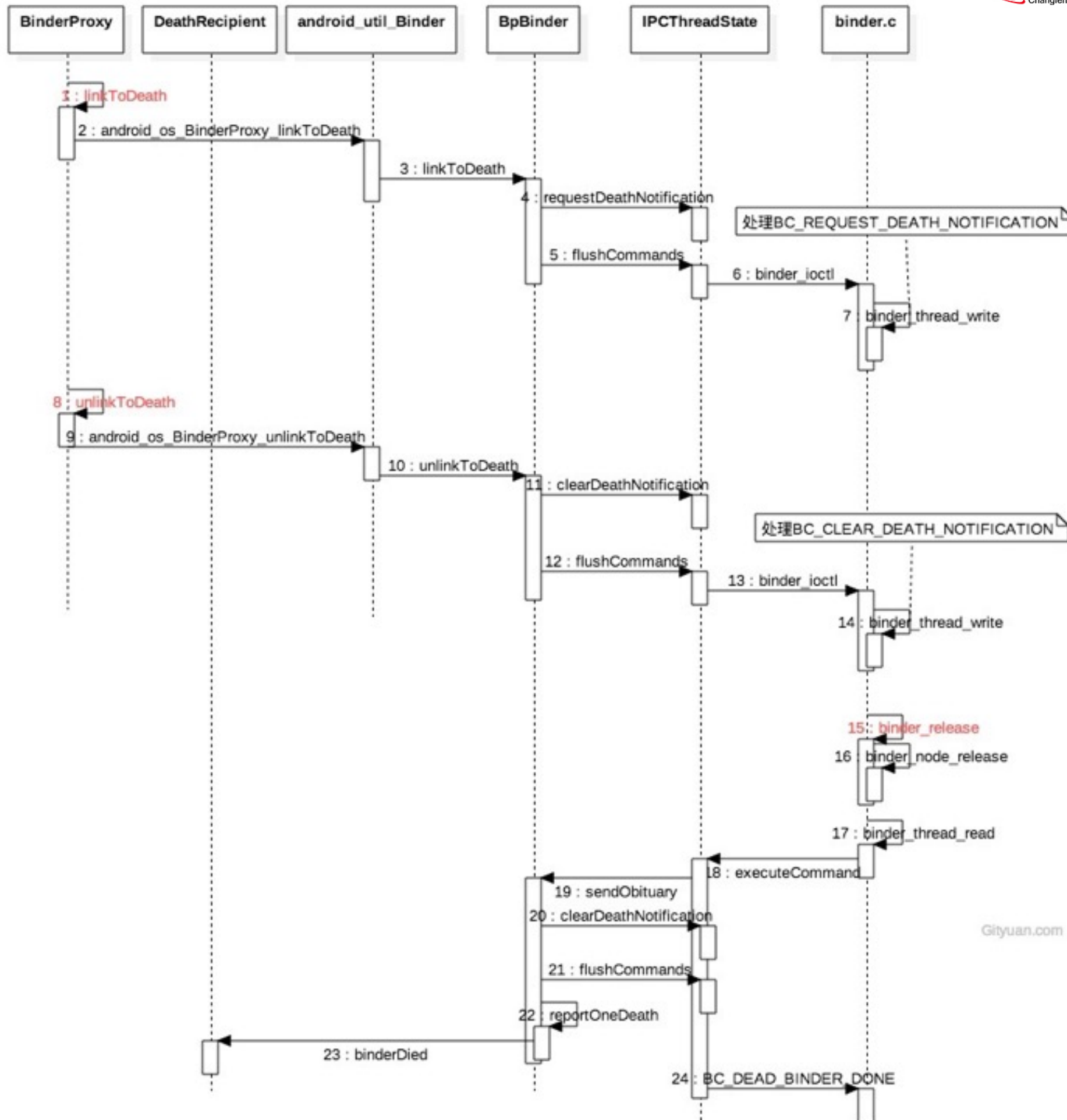
注册服务



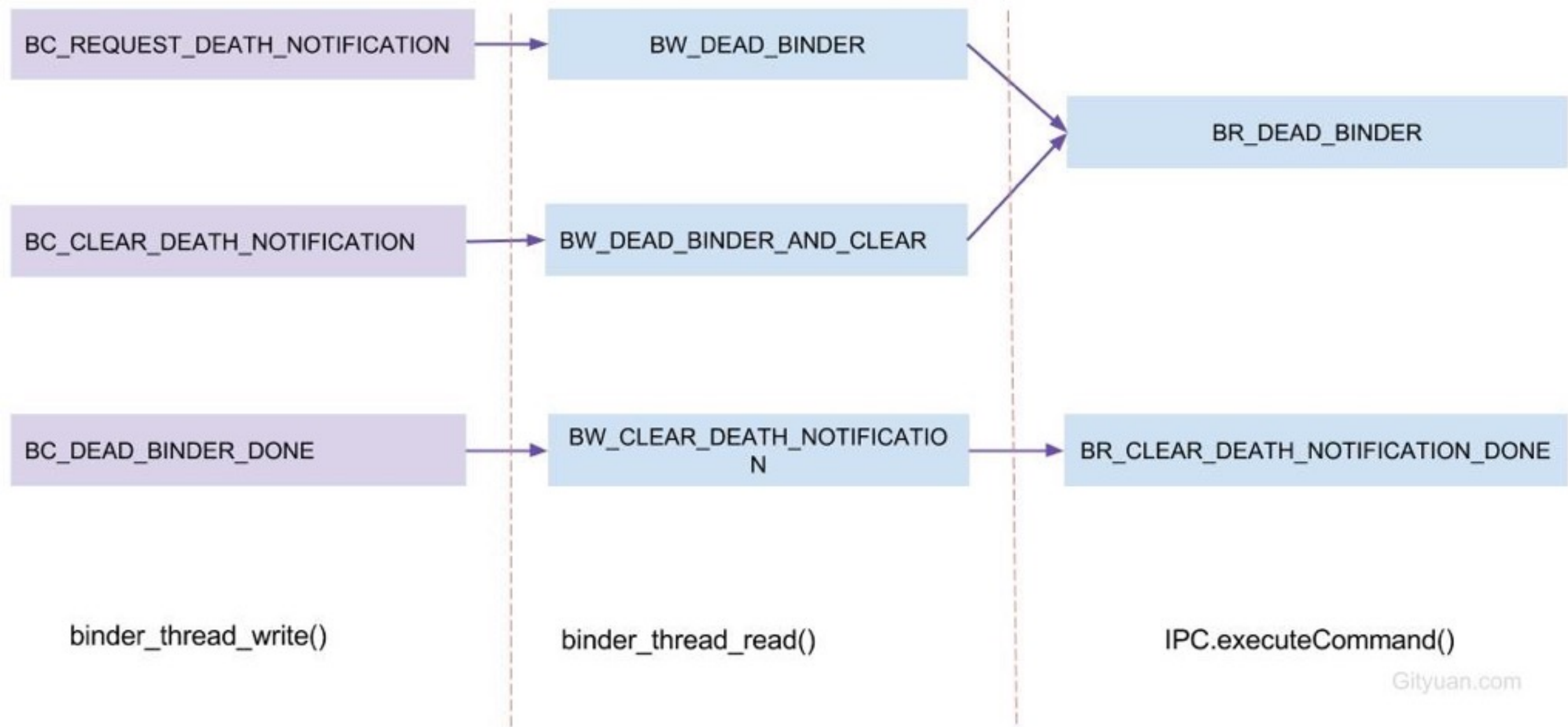
提纲

- ▶ Binder概述
- ▶ Binder设计思想
- ▶ Binder通信协议
- ▶ Binder死亡通知
- ▶ Binder线程池管理

死亡通知

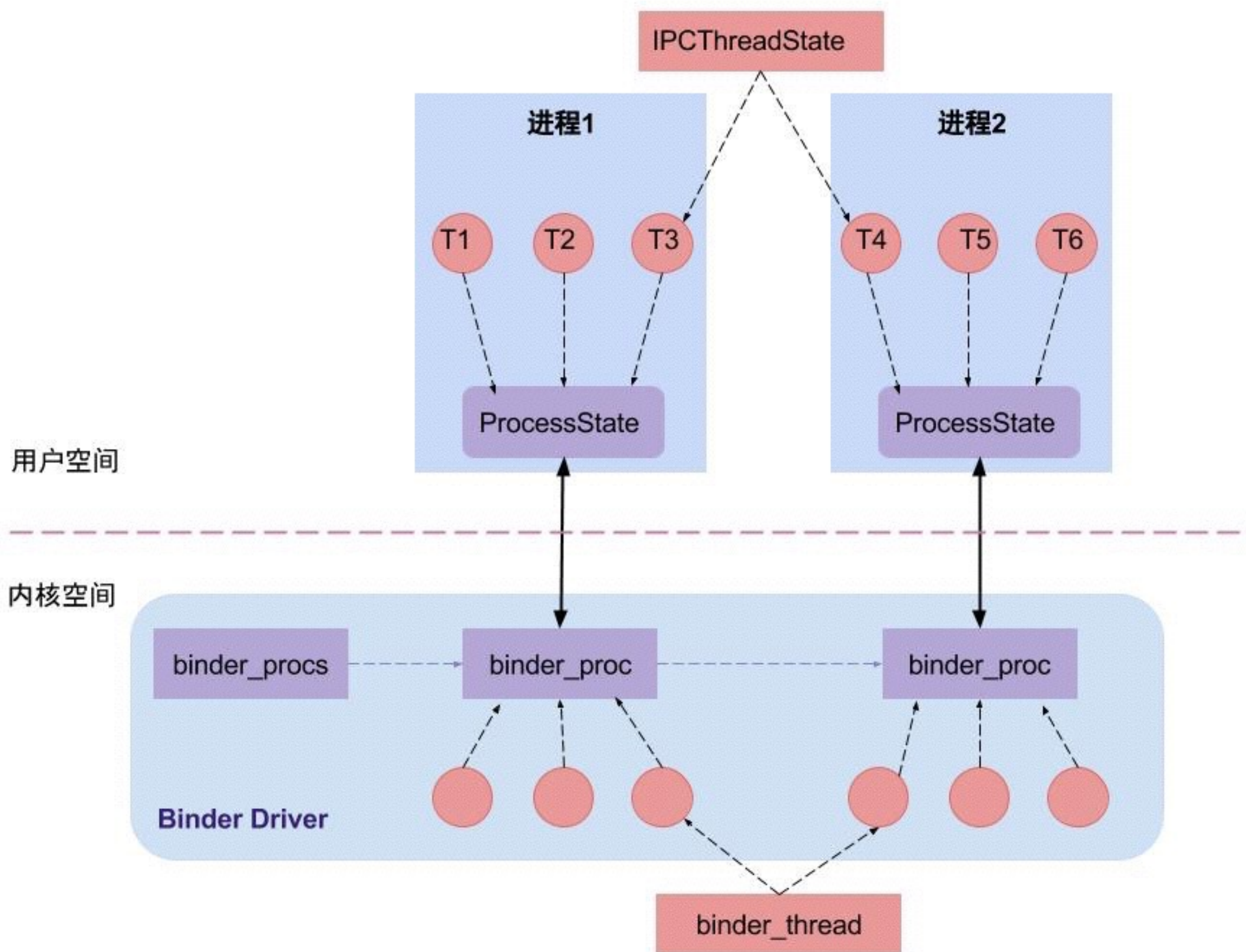


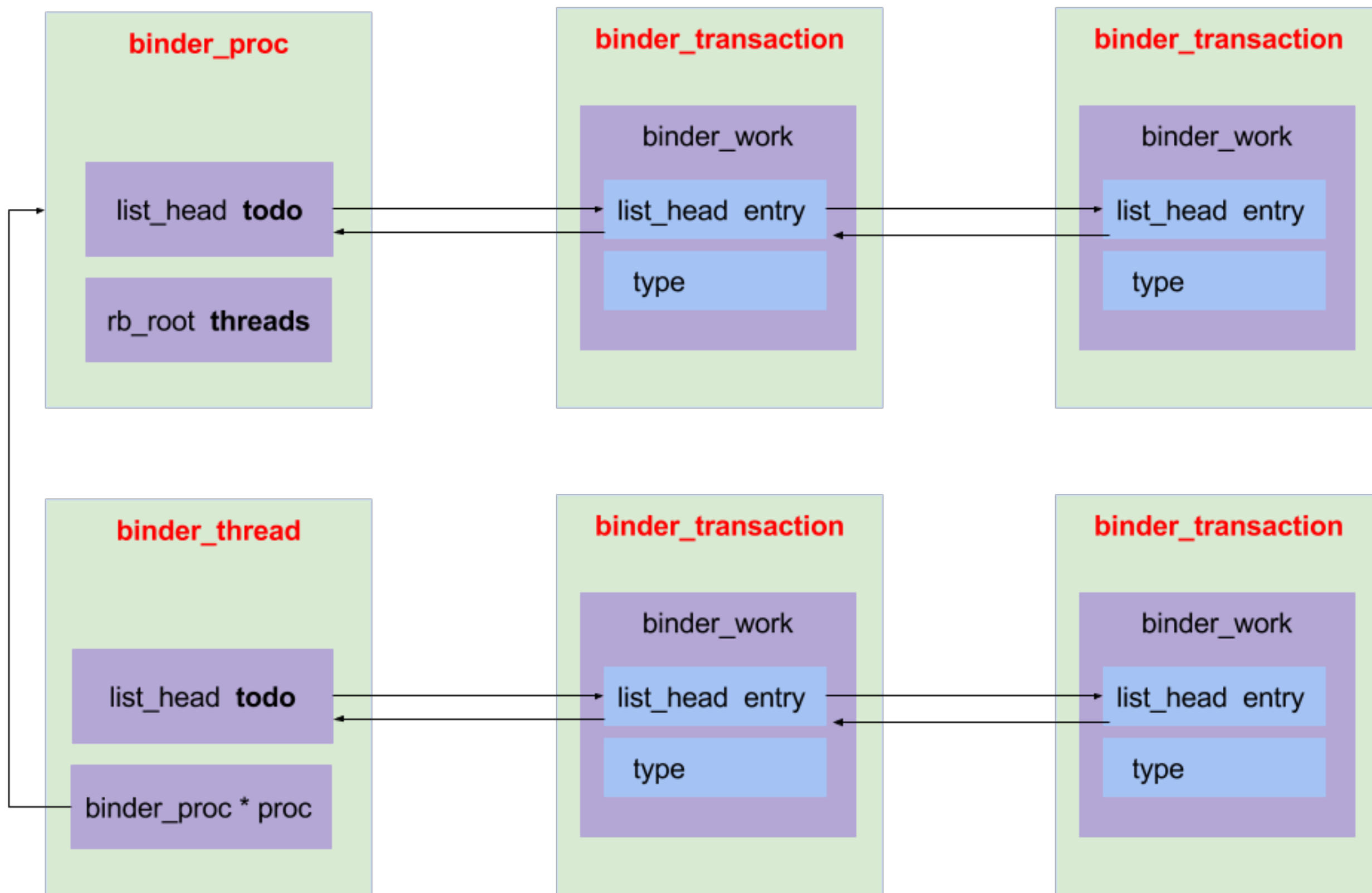
死亡通知

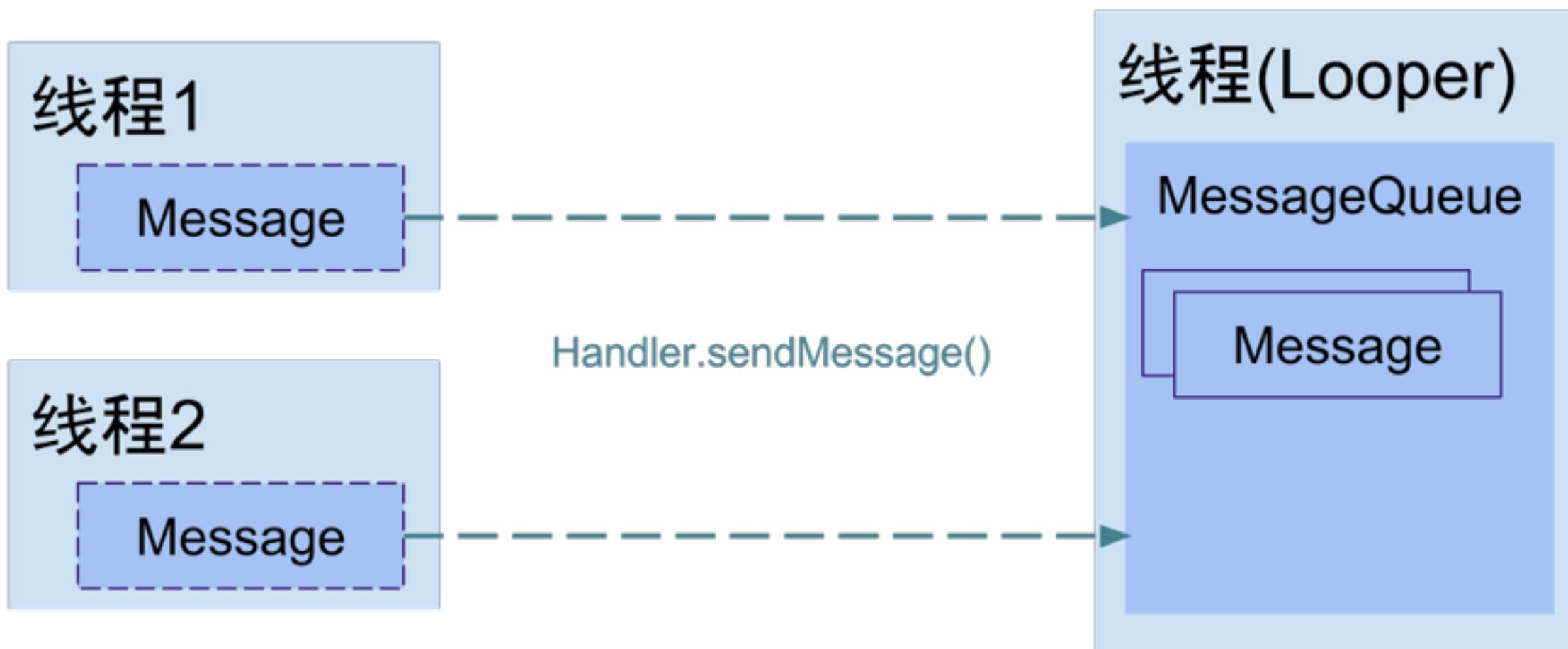


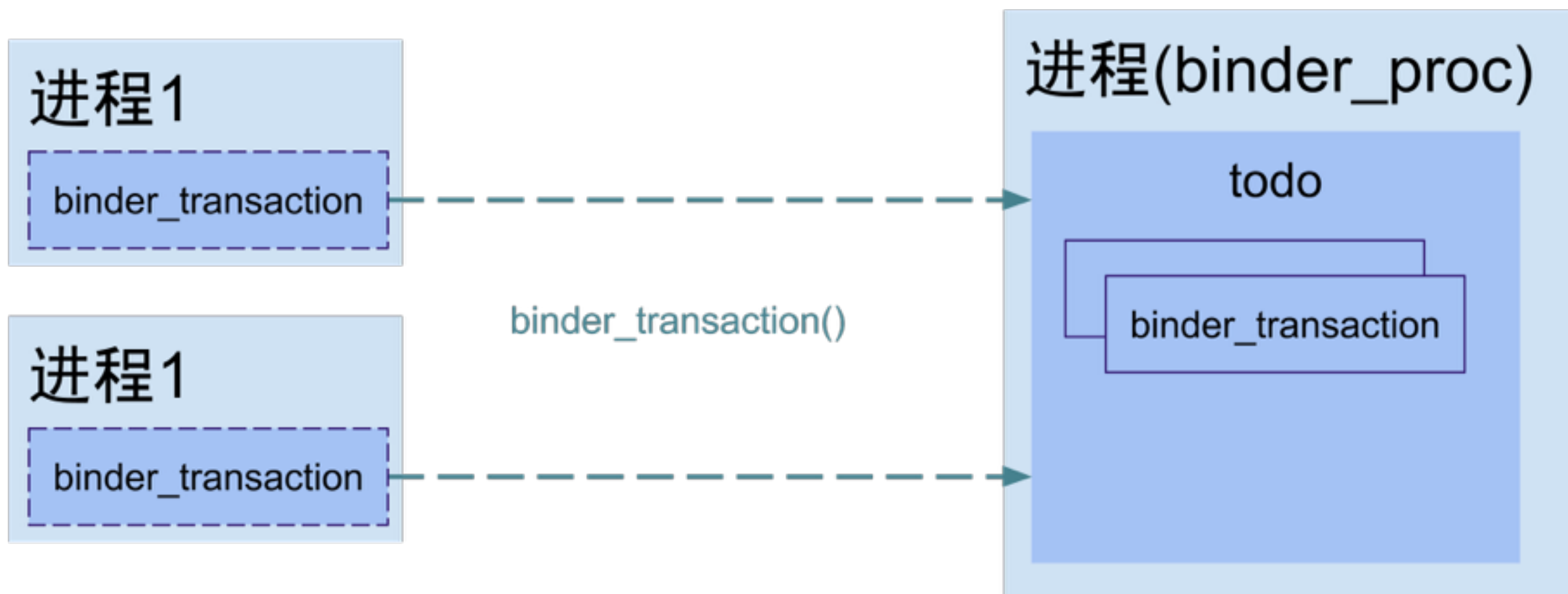
提纲

- ▶ Binder概述
- ▶ Binder设计思想
- ▶ Binder通信协议
- ▶ Binder死亡通知
- ▶ Binder线程池管理











Thank You