MICROOH 麦可网

Android-从程序员到架构师之路

出品人：Sundy
讲师：高焕堂（台湾）

http://www.microoh.com

E01_a

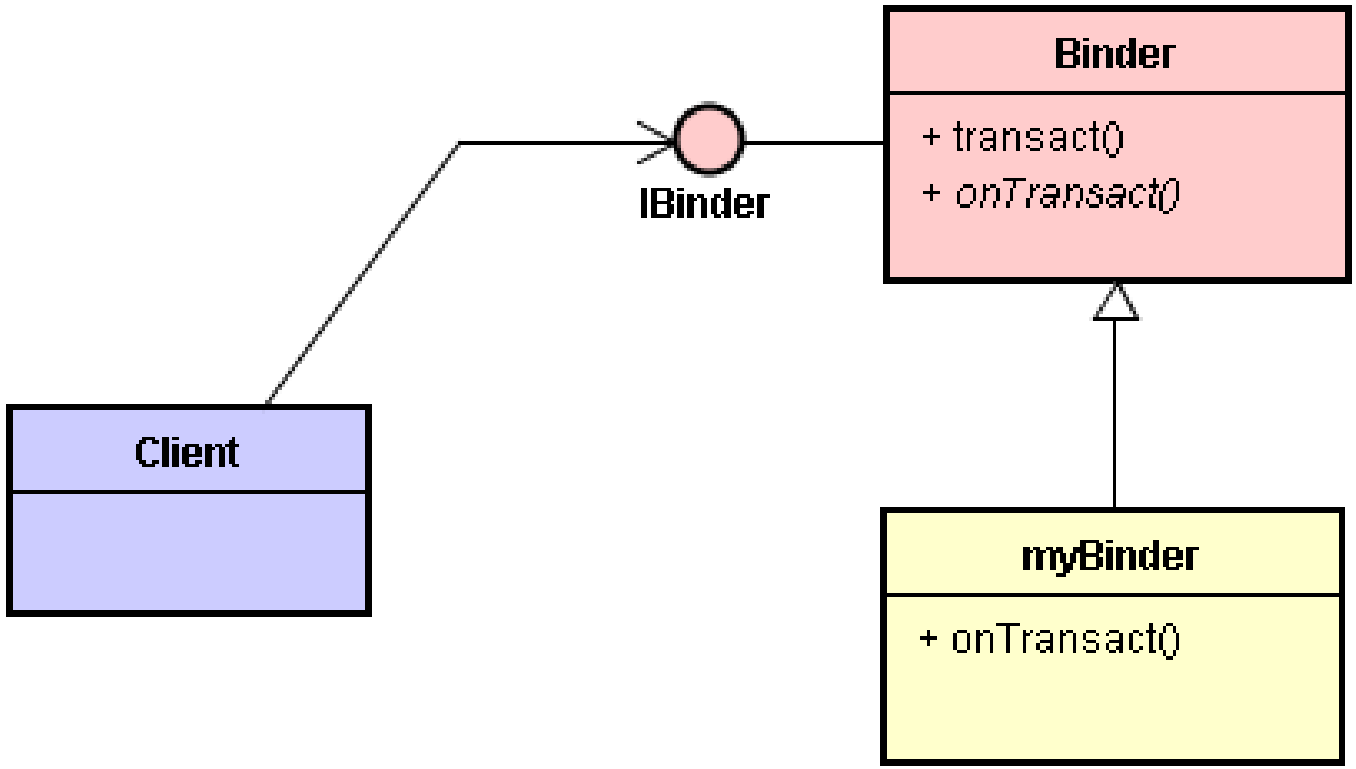
OOPC与HAL的美妙结合(a)

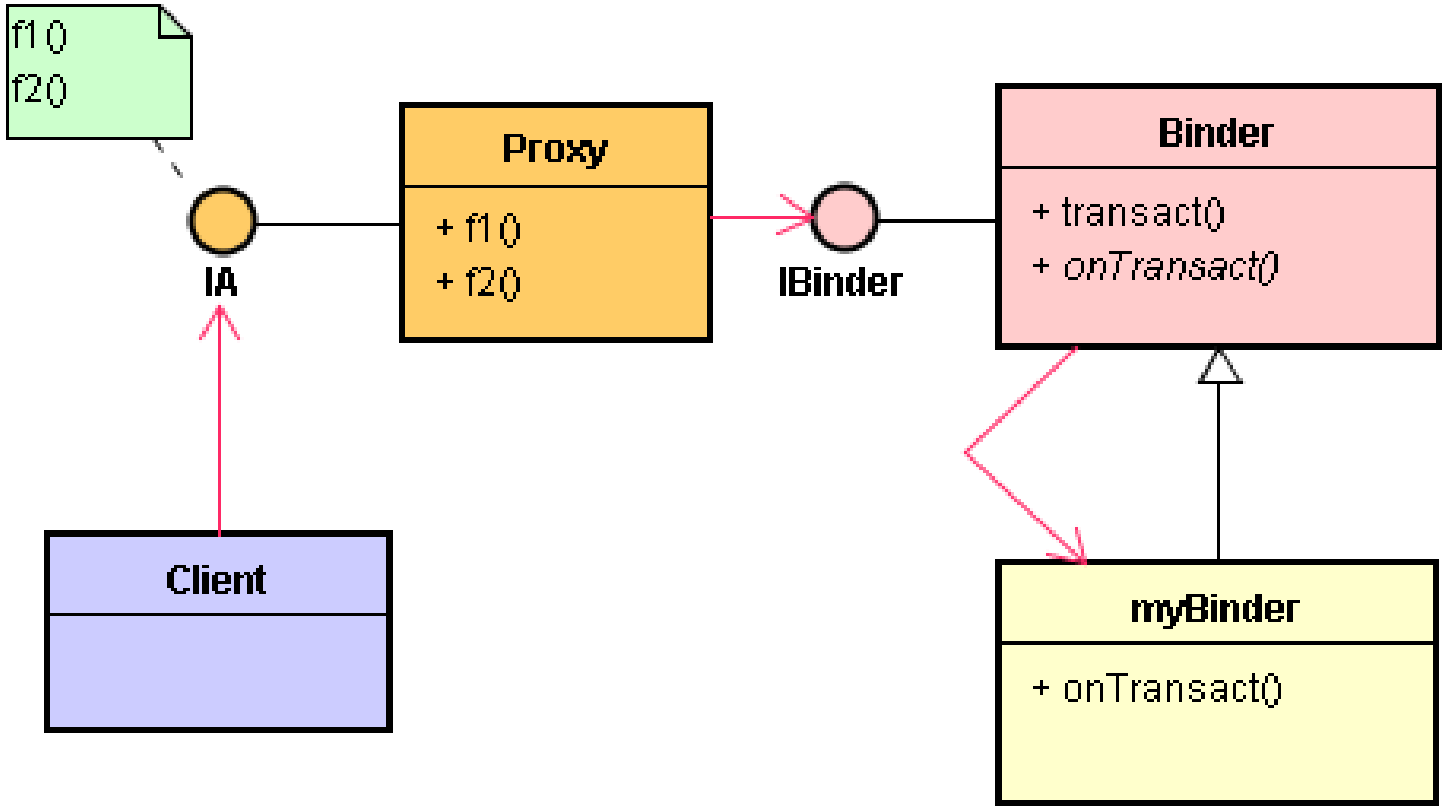
By 高煥堂

内容

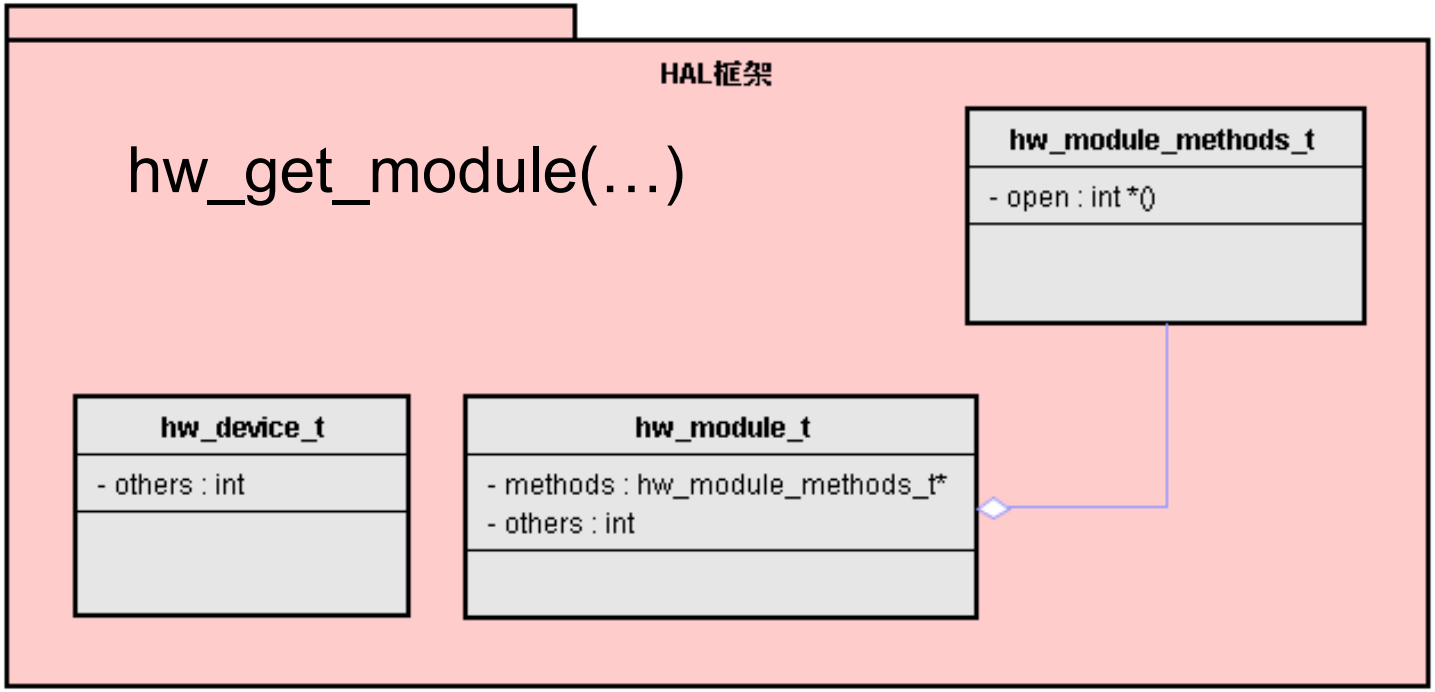
1. 设计：替HAL增添一个Proxy类
2. 基础：介绍LW_OOPC
3. 演练：<LW_OOPC + JNI>代码范例
4. 实践：以LW_OOPC撰写HAL的Proxy类
5. 讨论：HAL接口Proxy类的角色和意义

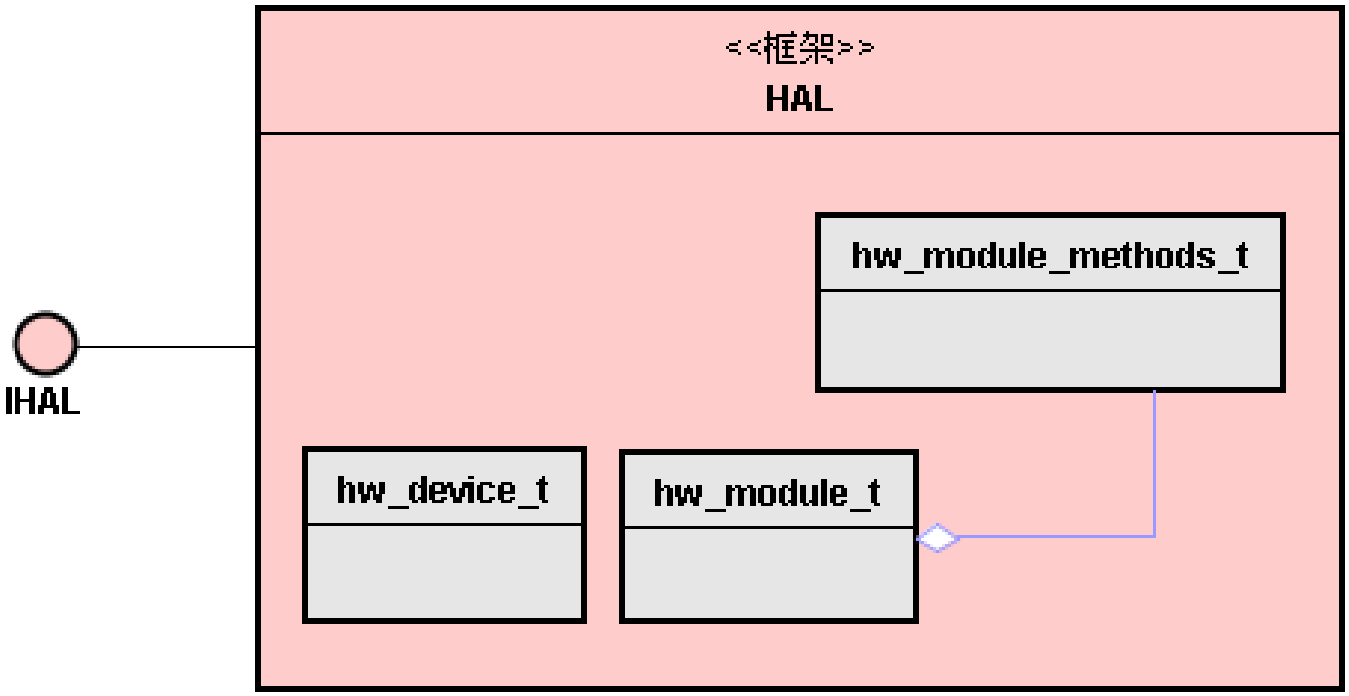
1、设计：替HAL
增添一个Proxy类

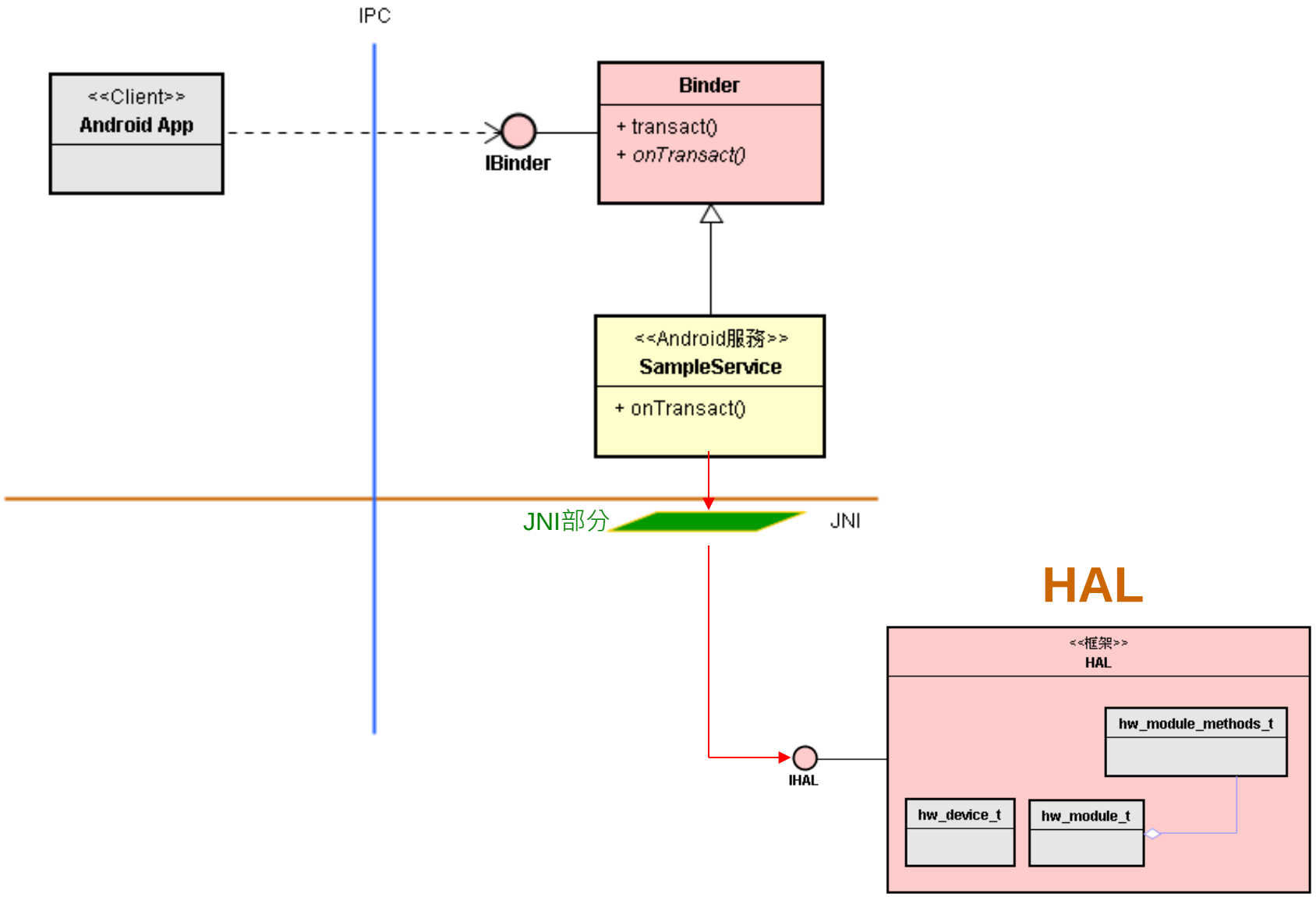

Binder
+ transact()
+ onTransact()
IBinder
Client
myBinder
+ onTransact()

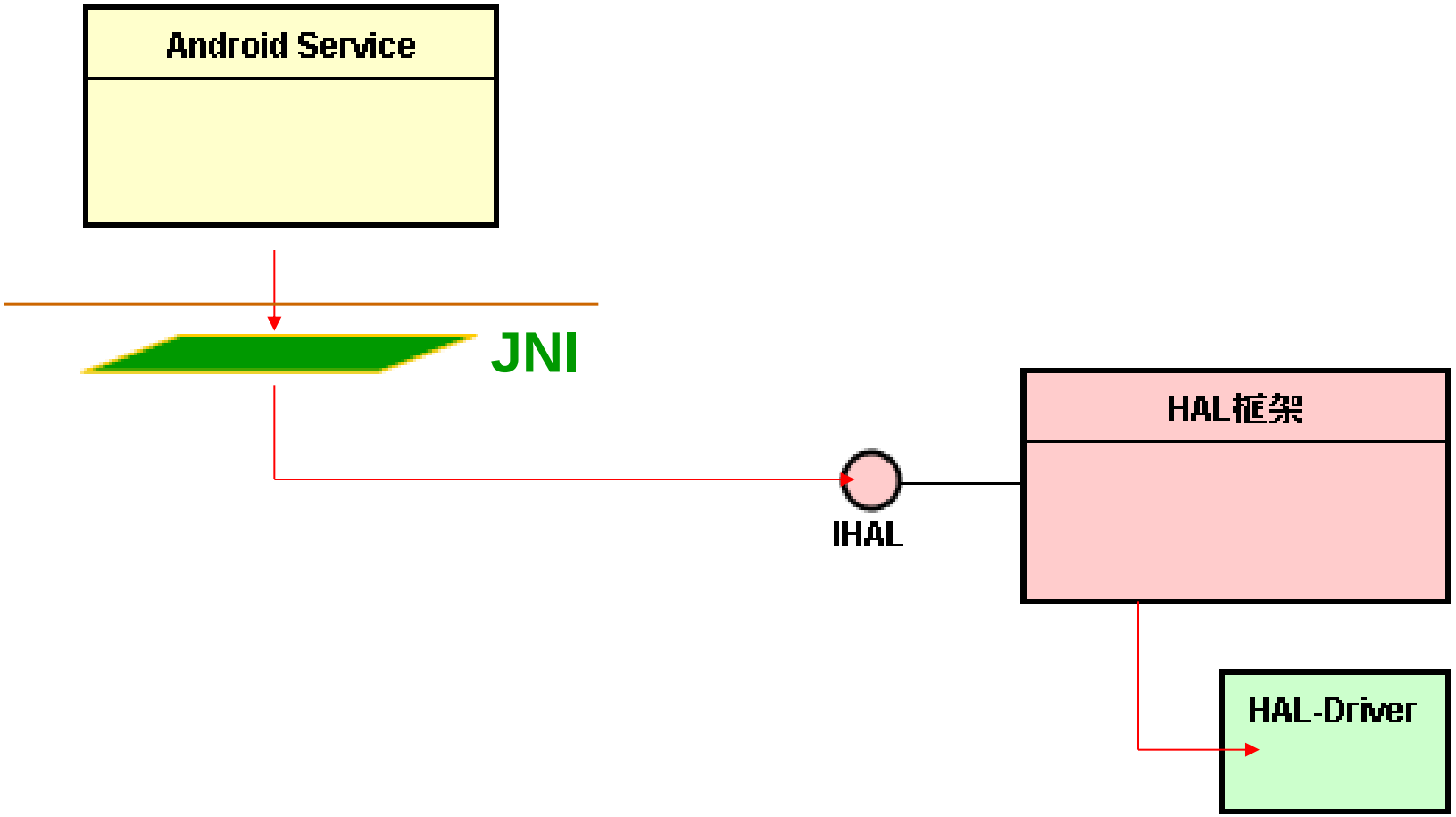

f1()
f2()
IA
Proxy
+ f1()
+ f2()
IBinder
Binder
+ transact()
+ onTransact()
Client
myBinder
+ onTransact()

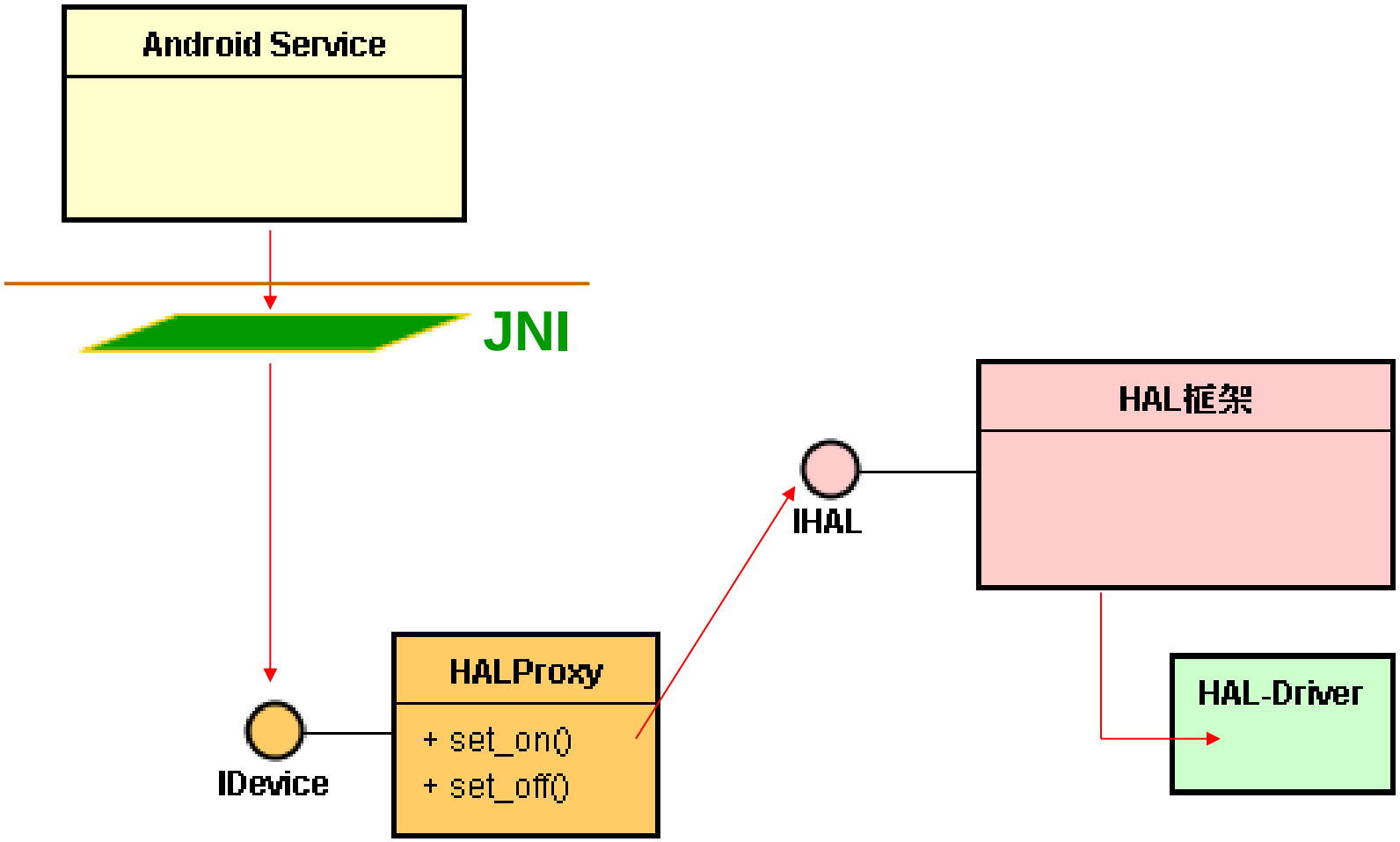
HAL的Proxy类

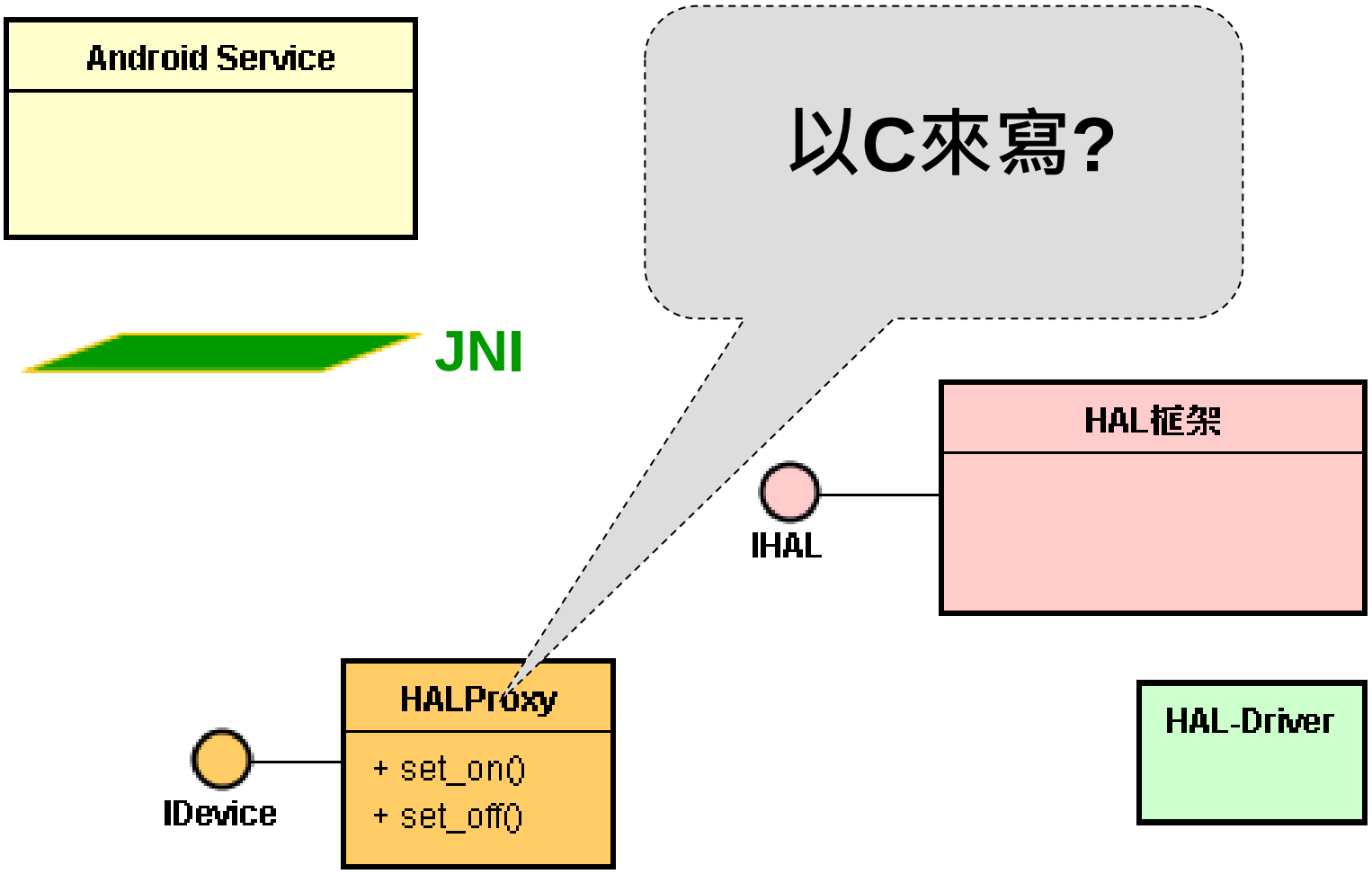

HAL框架
hw_get_module(…)
hw_module_methods_t
- open : int *()
hw_device_t
- others : int
hw_module_t
- methods : hw_module_methods_t*
- others : int

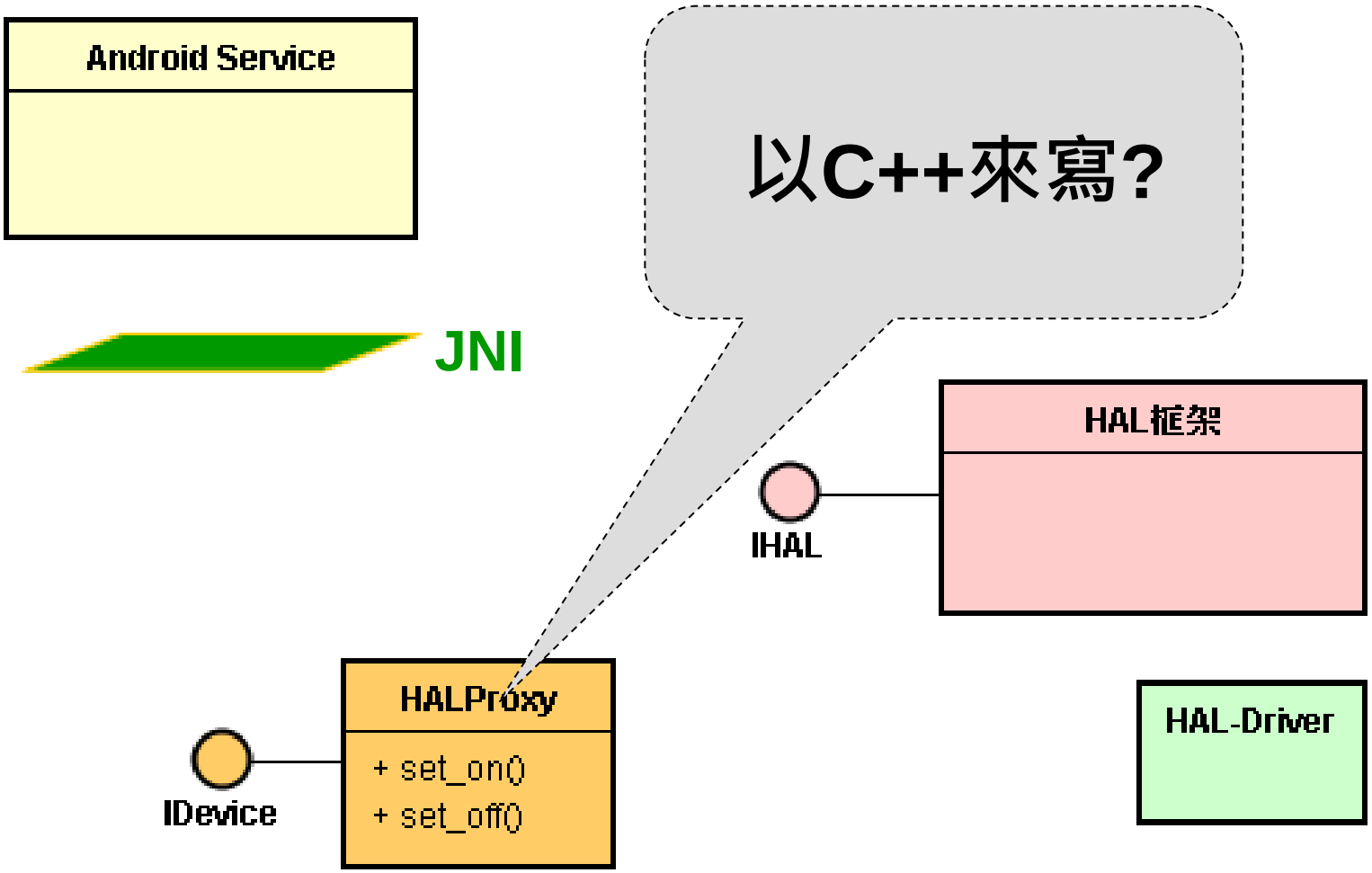

<<框架>>
HAL
hw_module_methods_t
IHAL
hw_device_t
hw_module_t

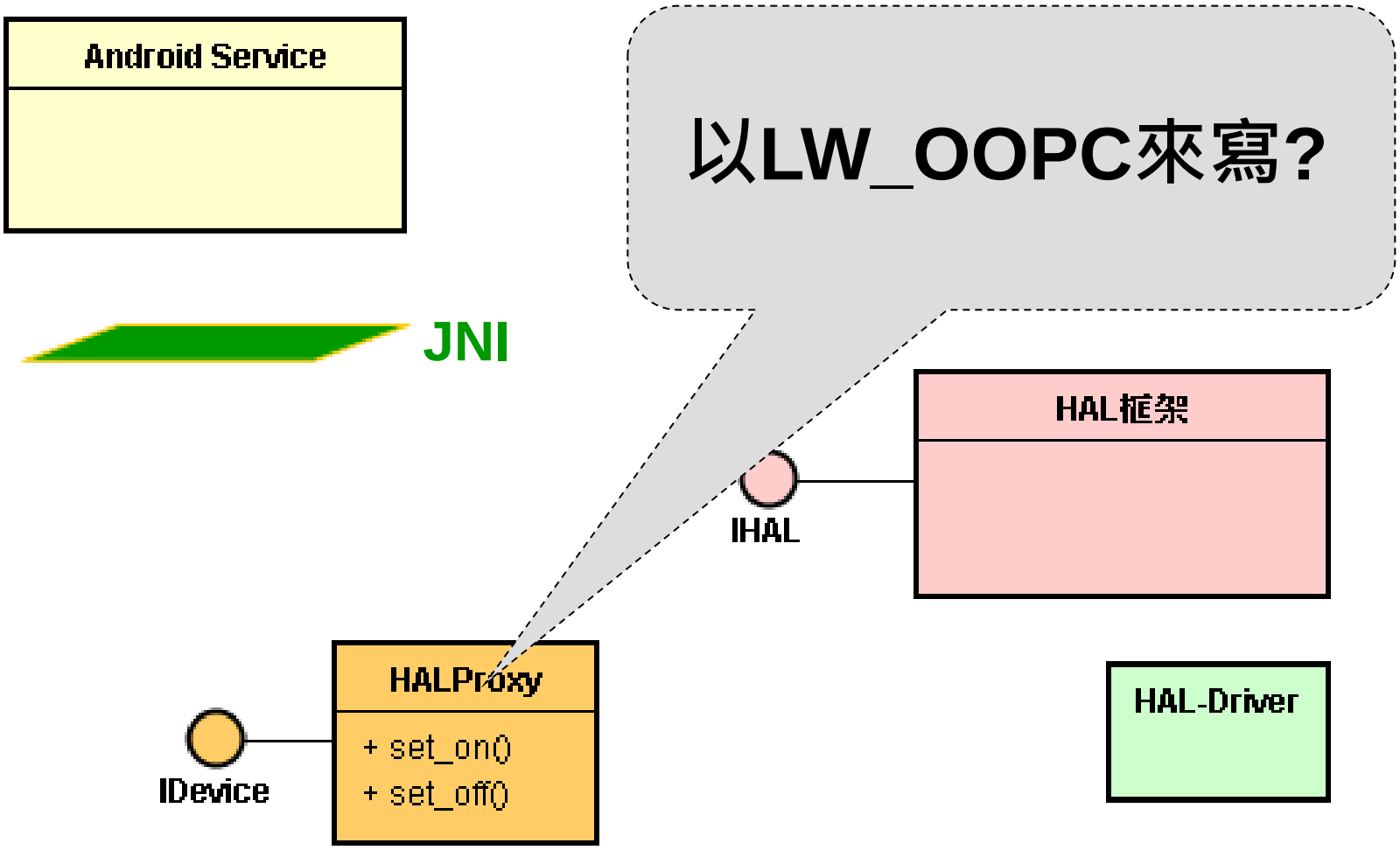

IPC
<<Client>>
Android App
IBinder
Binder
+ transact()
+ onTransact()
<<Android服務>>
SampleService
+ onTransact()
JNI部分
JNI
HAL
IHAL
<<框架>>
HAL
hw_module_methods_t
hw_device_t
hw_module_t


Android Service
JNI
IHAL
HAL框架
HAL-Driver


Android Service
JNI
HAL框架
IHAL
HALProxy
+ set_on()
+ set_off()
IDevice
HAL-Driver


Android Service
以C來寫?
JNI
HAL框架
IHAL
HALProxy
+ set_on()
+ set_off()
IDevice
HAL-Driver


Android Service
以C++來寫?
JNI
HAL框架
IHAL
HALProxy
+ set_on()
+ set_off()
IDevice
HAL-Driver


Android Service
以LW_OOPC來寫?
JNI
HAL框架
IHAL
HALProxy
+ set_on()
+ set_off()
IDevice
HAL-Driver

- 有了 Proxy类，对于JNI Native模块的代码，有何影响呢?

2、基础：介绍LW_OOPC

LW_OOPC是高老师的创意

- 高焕堂老师于2008年设计出LW_OOPC 初版，并出版<<UML+OOPC嵌入式C语言开发精讲>>一书。

<<UML+OOPC嵌入式C语言开发精讲>> 一书(博文视点出版)

- 后来，于2010年由金永华先生继续扩充，推出更新版本。目前为全球LGPL协议开源软件：
  http://sourceforge.net/projects/lwoopc/

金永华关于LW_OOPC的杂志文章:

http://blog.csdn.net/juniorhope/article/details/5719107

C语言 + 面向对象

- OOPC是指OOP(Object-Oriented Programming)与C语言的结合，藉由C语言的Macro指令定义出OOP概念的关键词(Key Word)，C 程序员就能运用这些关键词来表达OOP概念，如类别、对象、信息、继承、接口等等。

比C++更精简有力

- 虽然OOPC程序的语法不像C++那么简洁，但是OOPC也有其亮丽的特色，就是编译后的程序所占内存空间(Size)比C++程序来得小，较能满足像Embedded System等的内存限制，程序员也较能高效调整程序的瓶颈而提升其执行速度。

- OOPC是以C的宏写成的Header档案，可以任由C程序员去对它瘦身美容，删去不需要的部分，挑出自己所需要的OOP特性，以小而美的身材满足Embedded 系统开发的需要。

认识LW_OOPC

- LW_OOPC 是一种轻便又快速的对象导向C语言。在嵌入式程序员还是蛮青睐C语言的，只是C语言没有对象、类别等概念，程序很容易变成意大利面型的结构，维护上比较费力。

- 在1986年C++上市时，希望大家改用C++，但是C++的效率不如C，并不受嵌入式程序员的喜爱。
- 于是，**高煥堂老師** 设计一个轻便又高效率的OOPC语言。轻便的意思是：它只用了约20个C叙述而已，简单易学。其Macro如下：。

LW_OOPC初版的宏(Macro)

```
/*  lw_oopc.h */  /* 这就高焕堂老师团队所设计的C */
#include "malloc.h"
#ifndef LOOPC_H
#define LOOPC_H

#define CLASS(type)\
typedef struct type type; \
struct type

#define CTOR(type) \
void* type##New() \
{ \
 struct type *t; \
 t = (struct type *)malloc(sizeof(struct type));
/*  continued  */
```

```
#define CTOR2(type, type2) \
void* type2##New() \
{ \
 struct type *t; \
 t = (struct type *)malloc(sizeof(struct type));

#define END_CTOR return (void*)t;  };
#define FUNCTION_SETTING(f1, f2)  t->f1 = f2;
#define IMPLEMENTS(type) struct type type
#define INTERFACE(type) struct type
#endif
/*     end     */
```

- 于2010年由金永华先生继续扩充，推出更新版本。目前为全球LGPL协议开源软件，您可以下载各版本：

  http://sourceforge.net/projects/lwoopc/

Why, 高效率?

- 其高效率的意思是，它没提供类别继承，内部没有虚拟函数表(Virtual Function Table)，所以仍保持原来C语言的高效率。除了没有继承机制之外，它提供有类别、对象、信息传递、接口和接口多型等常用的机制。目前受到不少C程序员的喜爱。

~ Continued ~