Android Graphics

Jim Huang (黃敬群)

Developer & Co-Founder, 0xlab

jserv@0xlab.org

Sep 24, 2011 / Study-Area

Rights to copy

http://0xlab.org/

contact@0xlab.org

creative commons
COMMONS DEED

Attribution – ShareAlike 3.0

Corrections, suggestions, contributions and translations are welcome!

Latest update: Nov 20, 2011

You are free

- to copy, distribute, display, and perform the work
- to make derivative works
- to make commercial use of the work

Under the following conditions

Attribution. You must give the original author credit.

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

License text: http://creativecommons.org/licenses/by-sa/3.0/legalcode

Agenda

(1) Binder IPC

(2) Android Graphics

(3) 2D and Accelerations

(4) OpenGL|ES

Notice:

Before looking into Android Graphics, you should be aware of the design of Binder IPC, otherwise you would get lost!

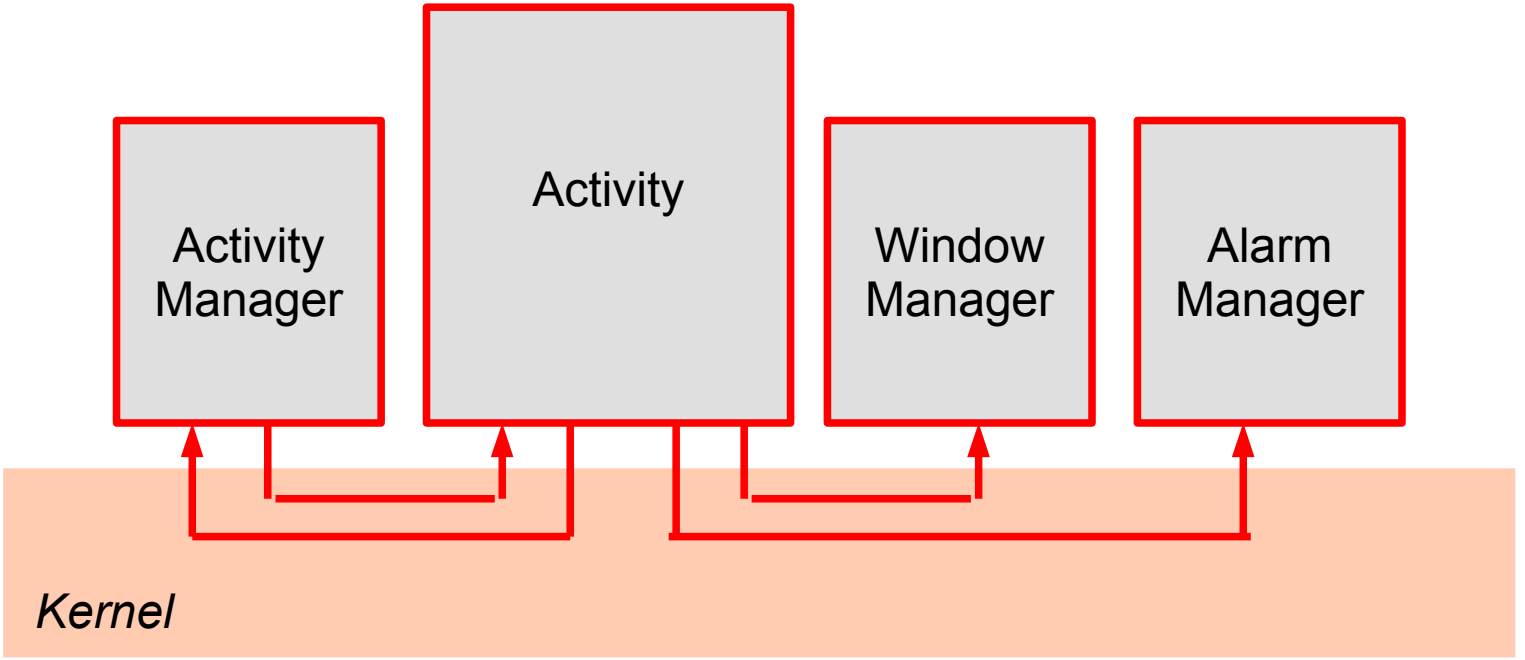
Binder IPC

Processes running on Android

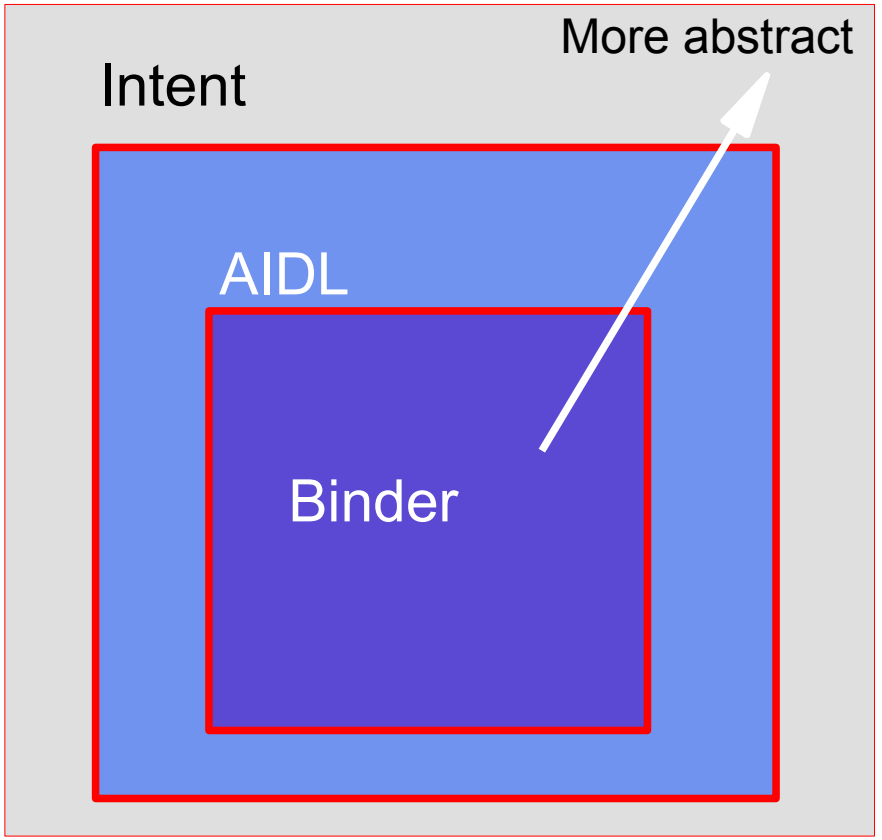
```
$ ps
    ...
root      37    1     248    156   c00aef2c 0000875c S /sbin/ueventd
system    42    1     768    260   c022950c afd0b6fc S /system/bin/servicemanager
root      43    1     3824   564   ffffffff afd0bdac S /system/bin/vold
root      44    1     3796   560   ffffffff afd0bdac S /system/bin/netd
root      45    1     628    264   c02588c0 afd0c0cc S /system/bin/debuggerd
radio     46    1     4336   672   ffffffff afd0bdac S /system/bin/rild
root      47    1     62224  27576 c00aef2c afd0b844 S zygote
media     48    1     16828  3736  ffffffff afd0b6fc S /system/bin/mediaserver
bluetooth 49    1     1216   572   c00aef2c afd0c59c S /system/bin/dbus-daemon
root      50    1     776    316   c02a8424 afd0b45c S /system/bin/installd
keystore  51    1     1704   432   c02588c0 afd0c0cc S /system/bin/keystore
shell     52    1     696    336   c0050934 afd0c3ac S /system/bin/sh
root      53    1     3356   160   ffffffff 00008294 S /sbin/adbd
system    67    47    172464 32596 ffffffff afd0b6fc S system_server
system    115   47    80028  20728 ffffffff afd0c51c S com.android.systemui
app_24    124   47    80732  20720 ffffffff afd0c51c S com.android.inputmethod.latin
radio     135   47    87848  20324 ffffffff afd0c51c S com.android.phone
app_18    144   47    89136  24160 ffffffff afd0c51c S com.android.launcher
app_7     165   47    86136  22736 ffffffff afd0c51c S android.process.acore
app_0     197   47    73996  17472 ffffffff afd0c51c S com.android.deskclock
app_14    208   47    75000  18464 ffffffff afd0c51c S android.process.media
app_3     219   47    72228  17652 ffffffff afd0c51c S com.android.bluetooth
app_25    234   47    85336  17836 ffffffff afd0c51c S com.android.mms
app_26    254   47    74656  19080 ffffffff afd0c51c S com.android.email
app_27    266   47    74912  18100 ffffffff afd0c51c S com.android.providers.calendar
app_1     285   47    71616  16280 ffffffff afd0c51c S com.android.protips
app_19    293   47    72184  16572 ffffffff afd0c51c S com.android.music
app_21    301   47    74728  17208 ffffffff afd0c51c S com.android.quicksearchbox
app_28    311   47    75408  18040 ffffffff afd0c51c S com.cooliris.media
shell     323   52    856    316   00000000 afd0b45c R ps
$
```

More than 30 processes (200+ threads).

IPC = Inter-Process Communication

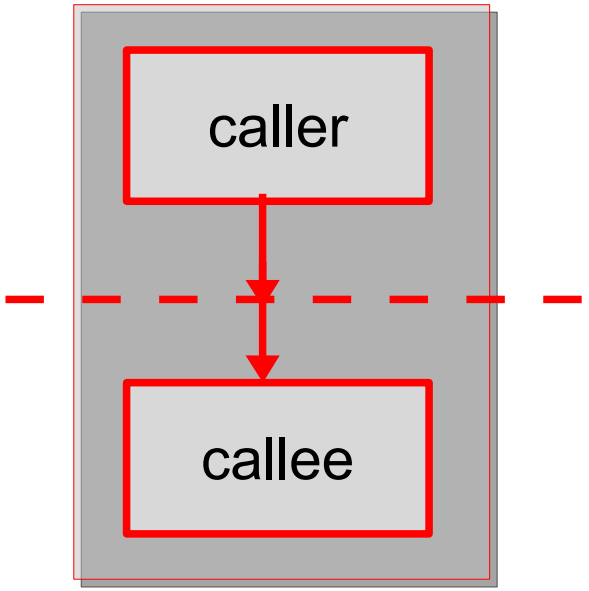


IPC Abstraction



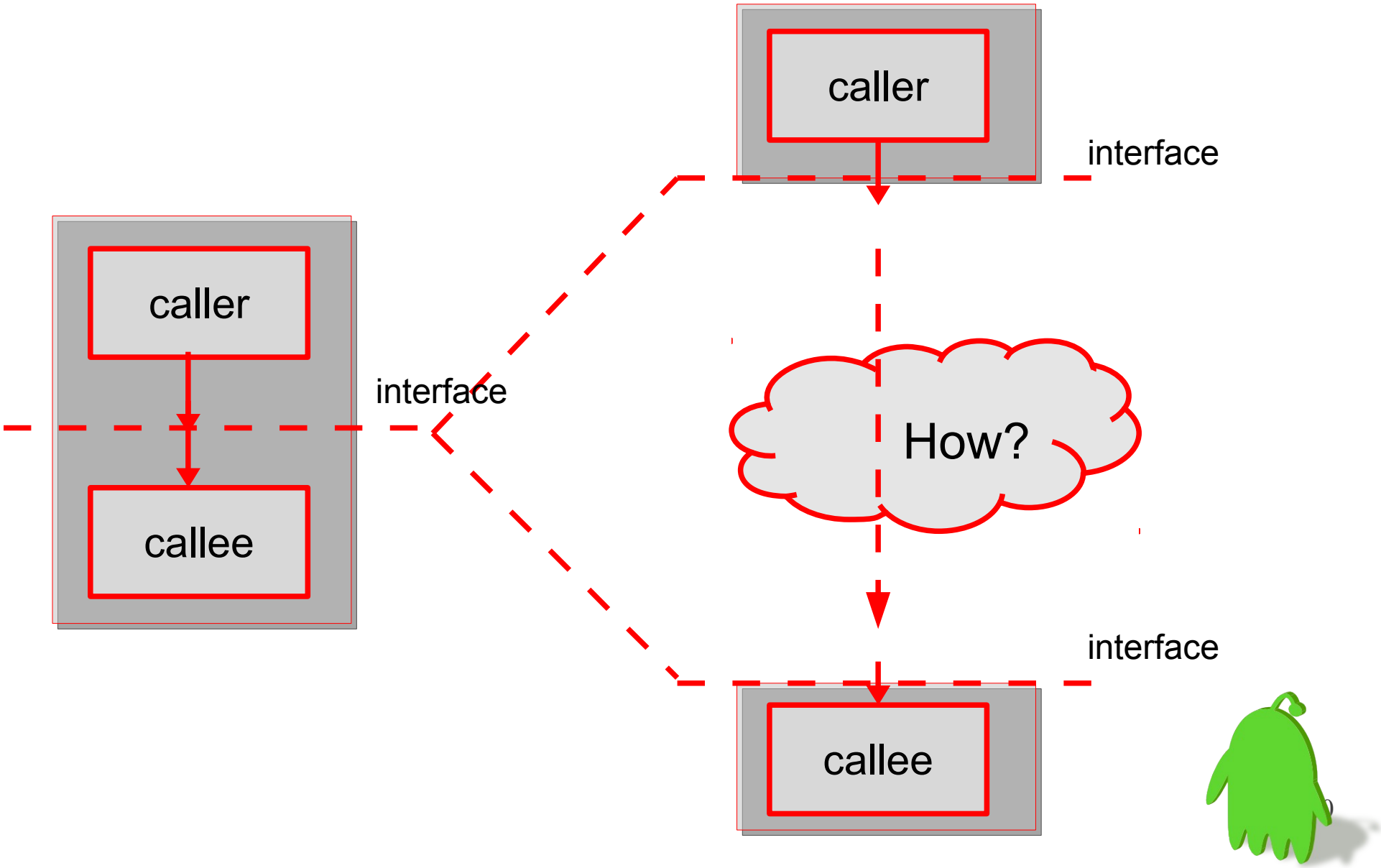
- Intent
  - The highest level abstraction
- Inter process method invocation
  - **AIDL**: Android Interface Definition Language
- binder: kernel driver
- ashmem: shared memory

Method invocation

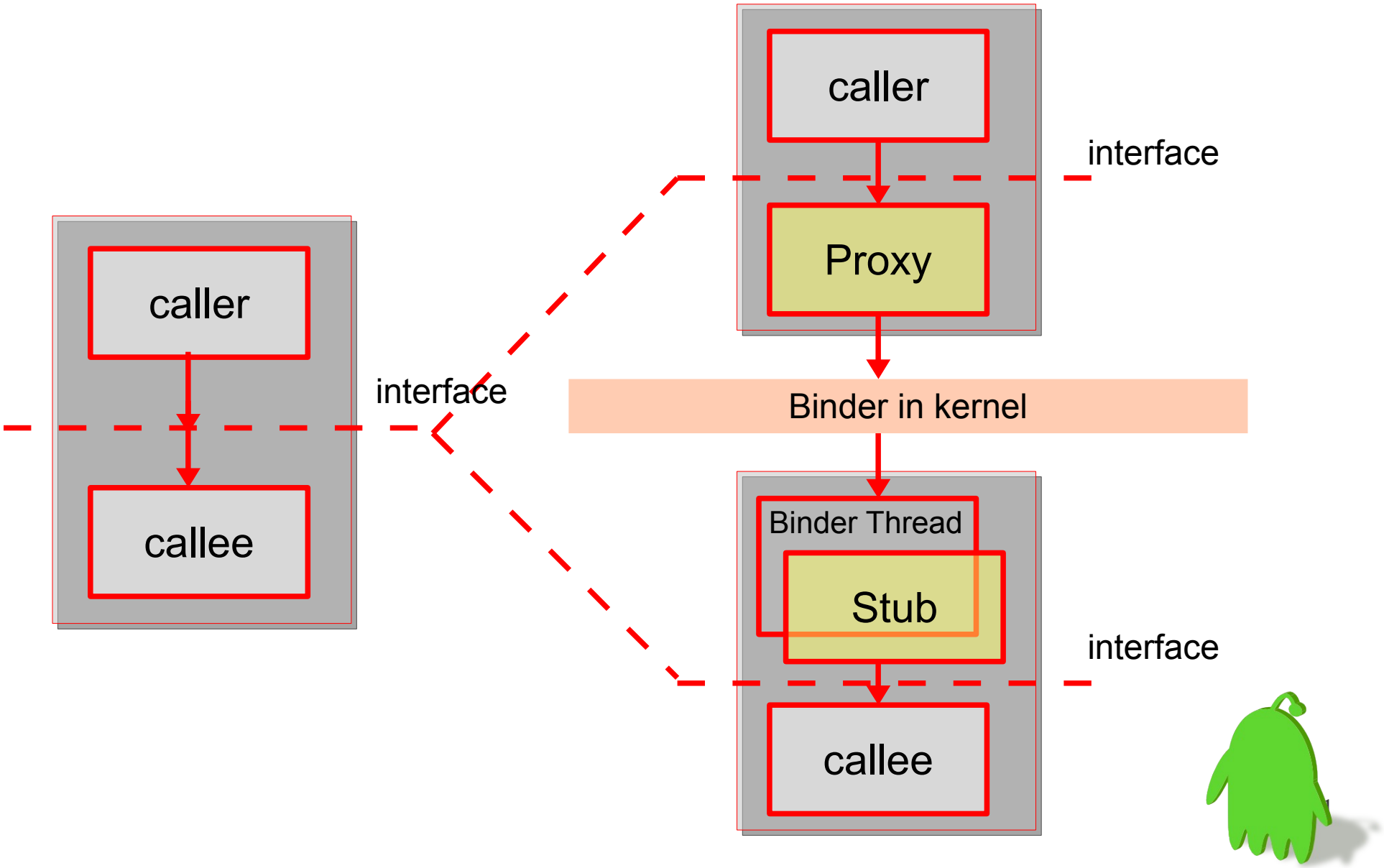


In the same process

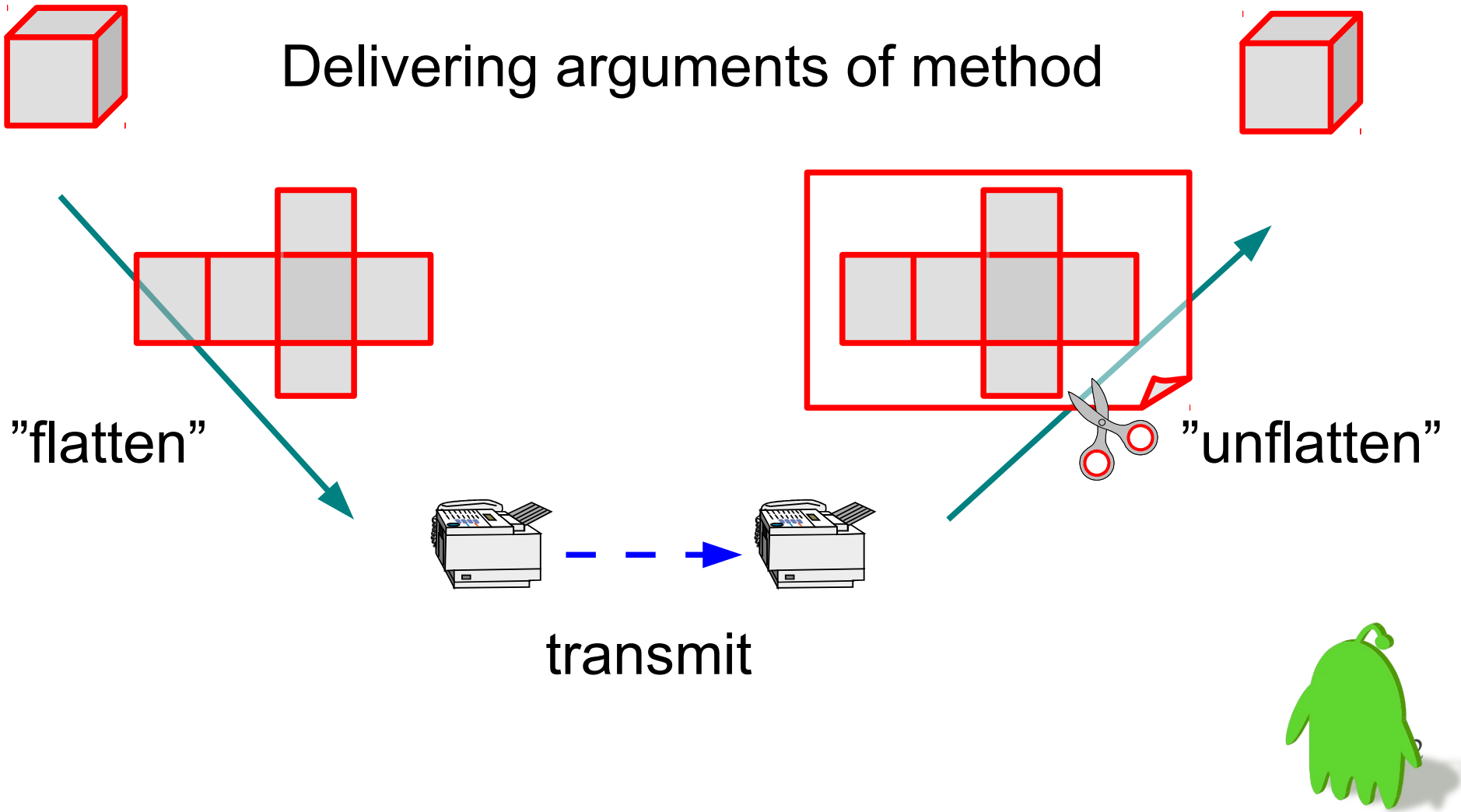
Inter-process method invocation



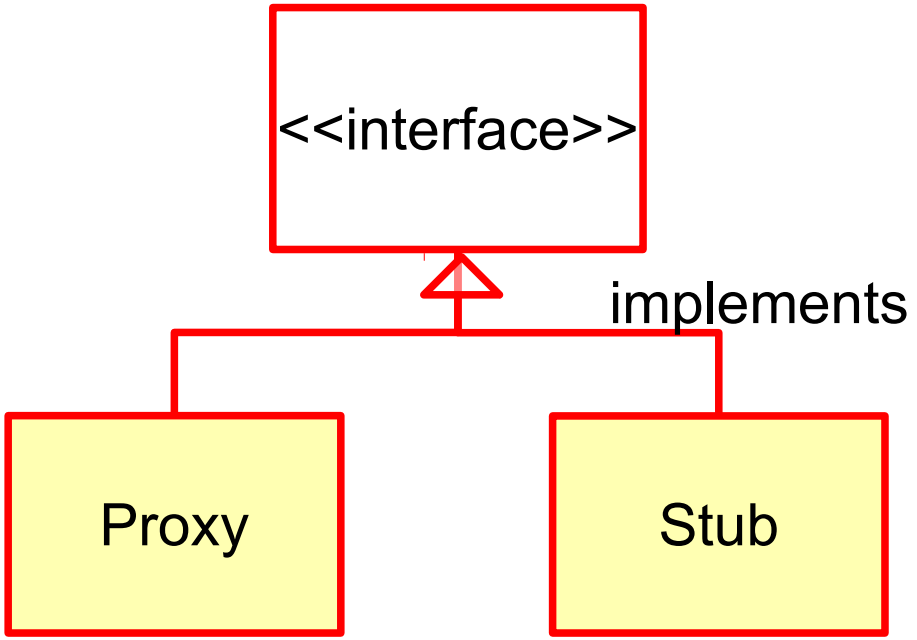
Inter-process method invocation



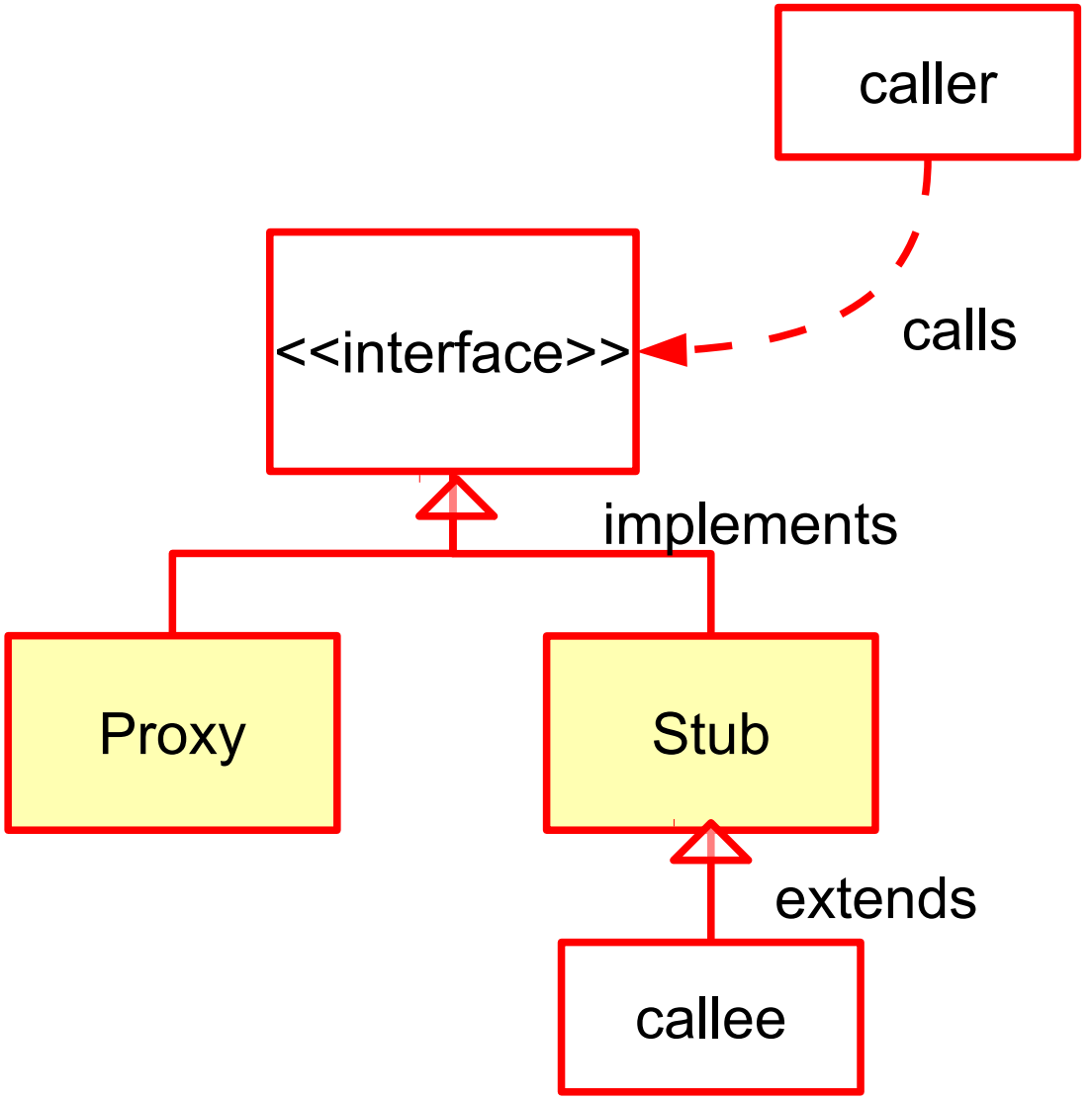
android.os.Parcel



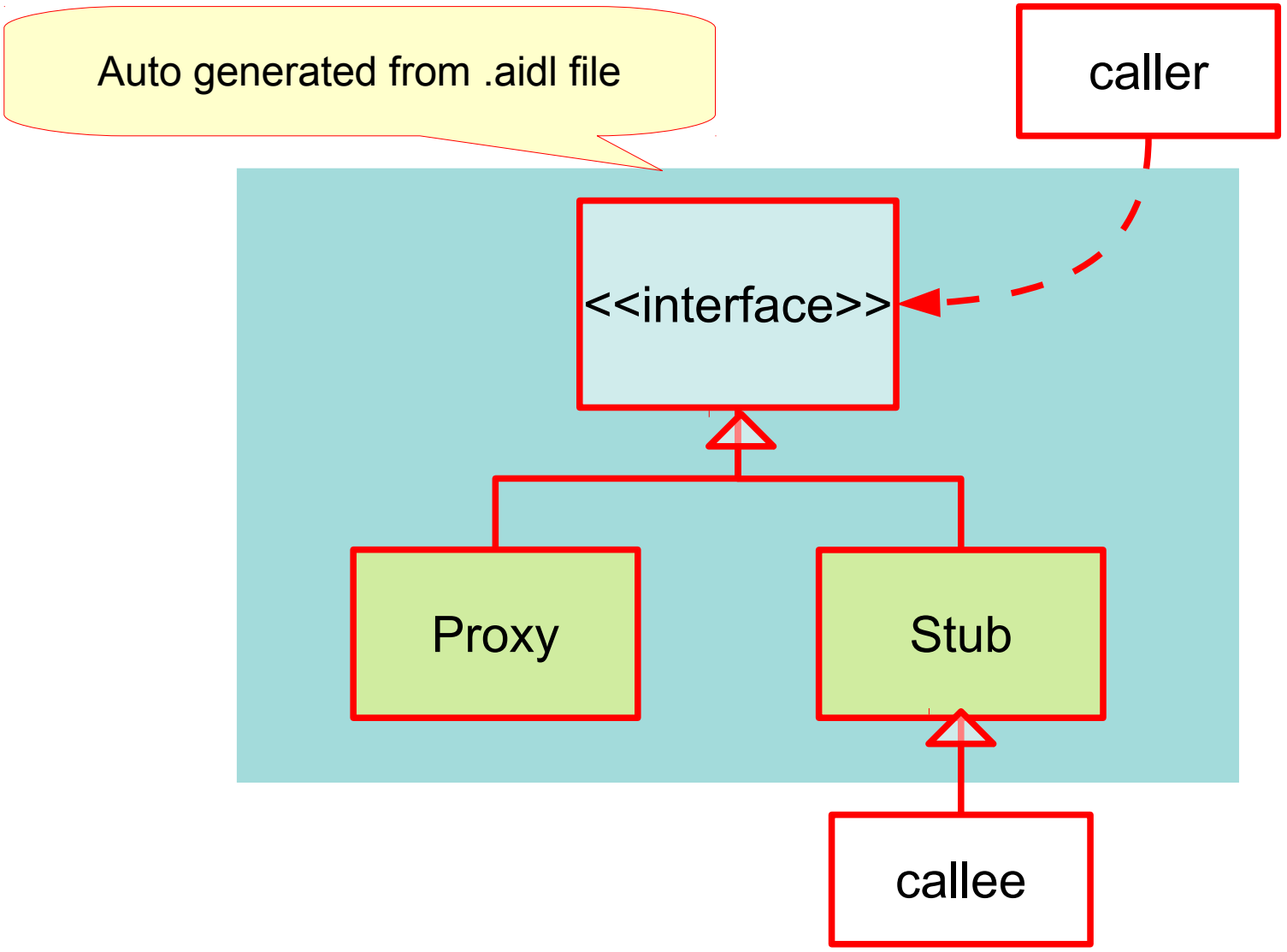
UML Representation



UML Representation

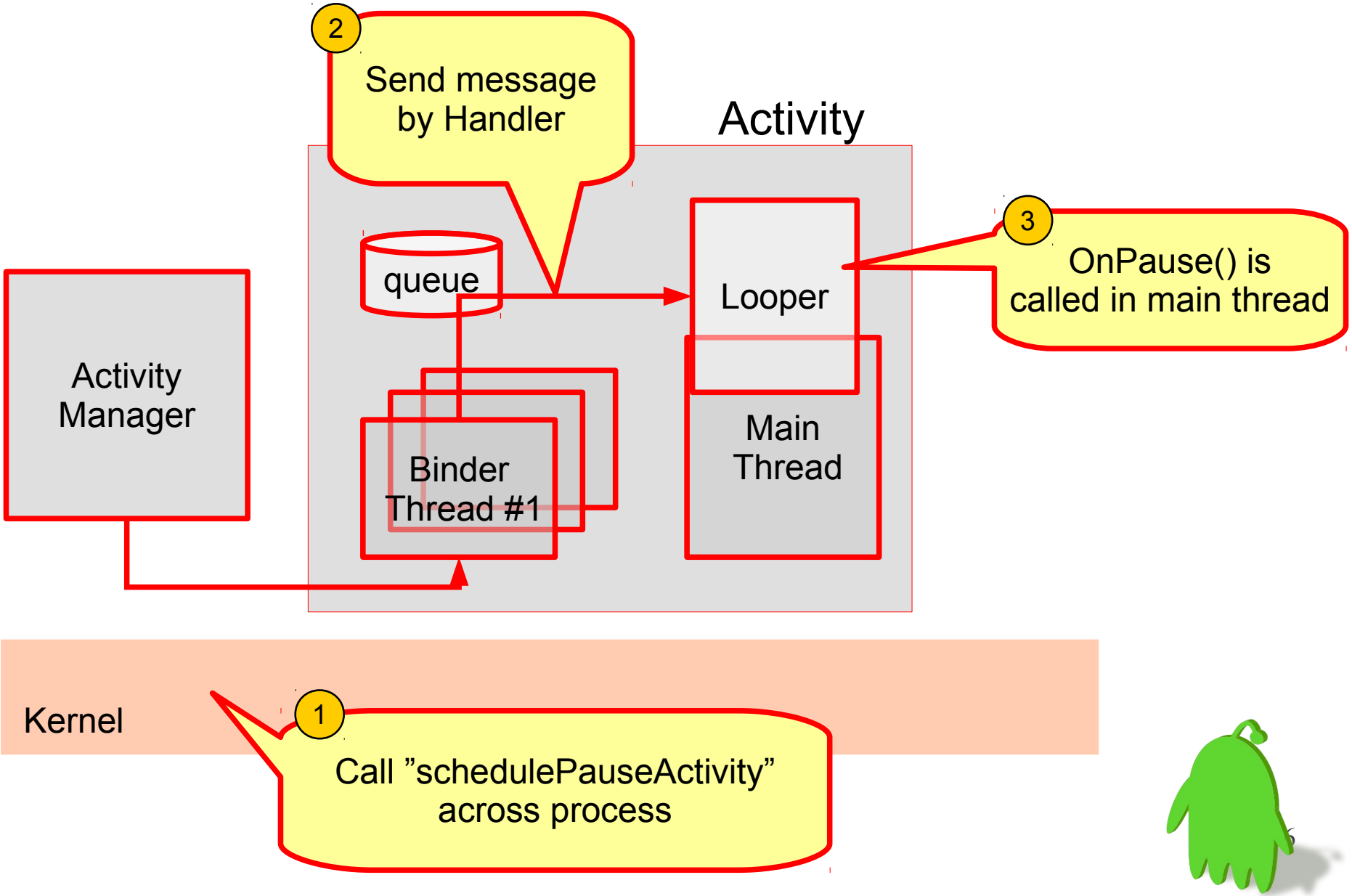


AIDL

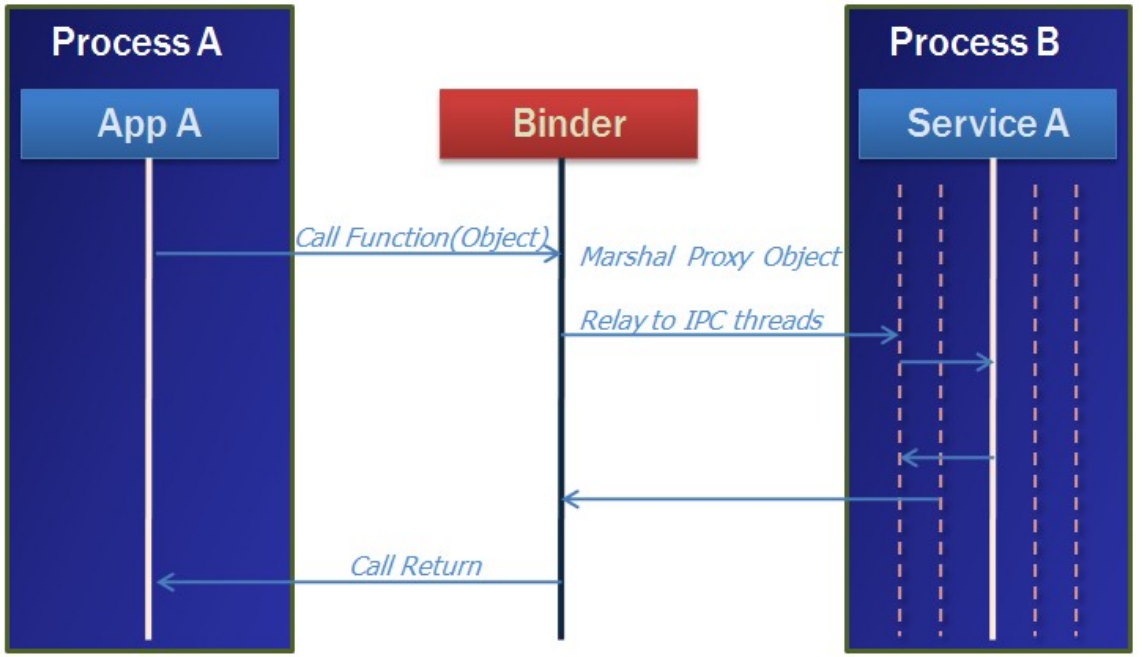

Auto generated from .aidl file
caller
<<interface>>
Proxy
Stub
callee

Use Case:

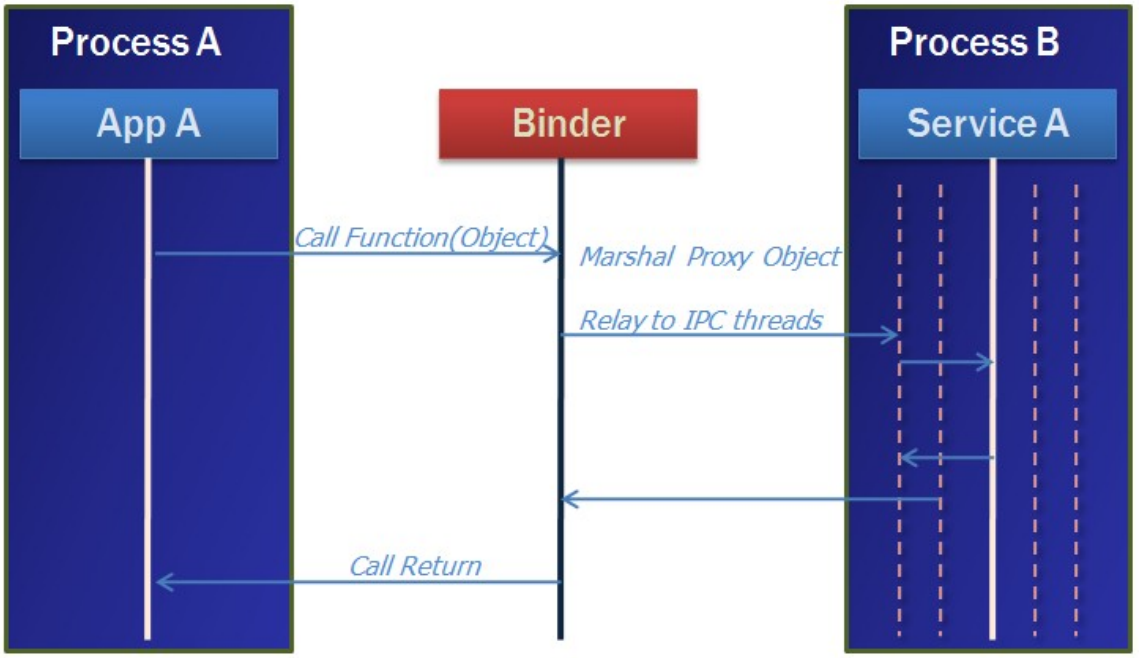
Who calls onPause() in Activity?



- Multi-thread aware
  - Have internal status per thead
  - Compare to UNIX socket: sockets have internal status per file descriptor (FD)

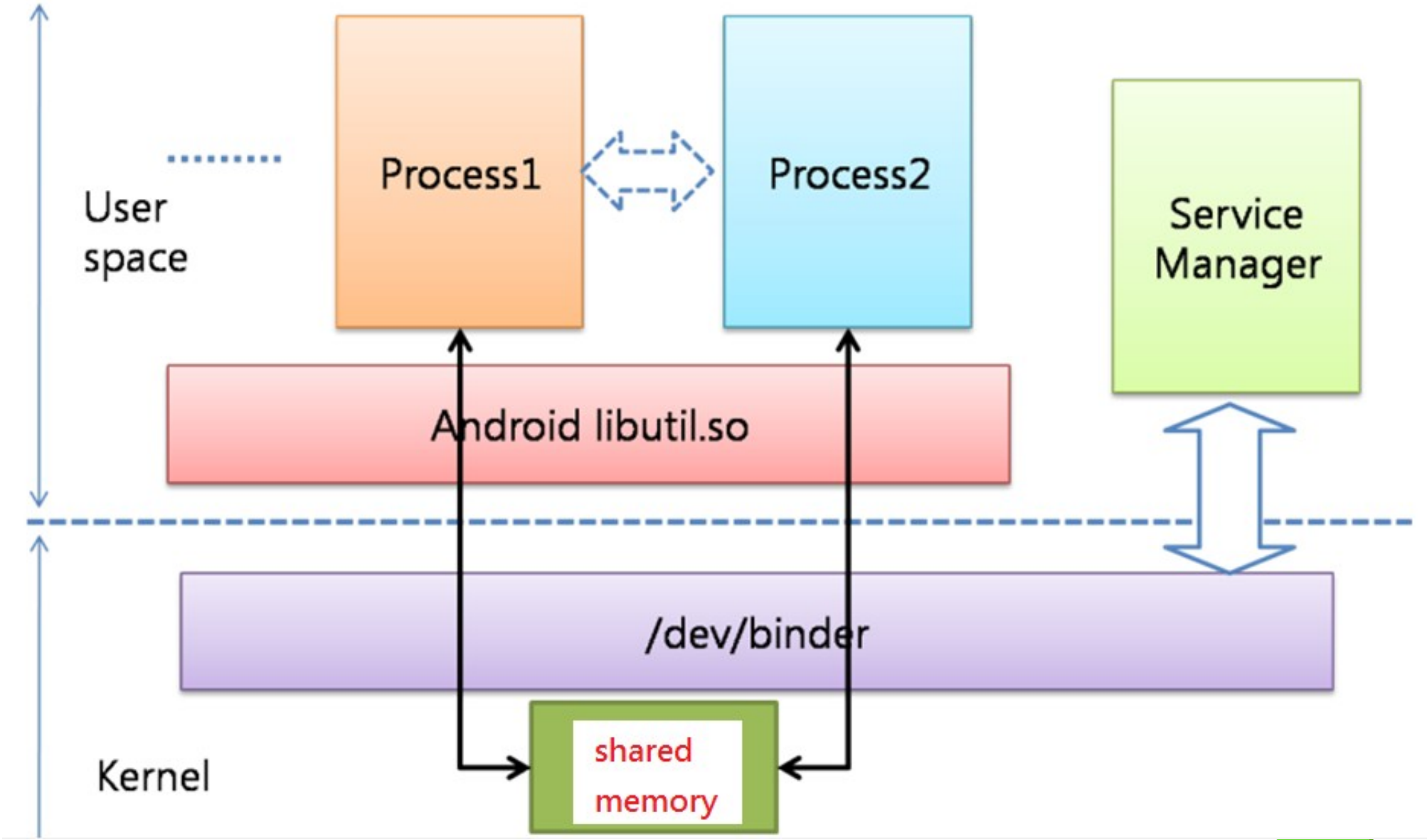


Binder



- ✓ A pool of threads is associated to each service application to process incoming IPC (Inter-Process Communication).
- ✓ Binder performs mapping of object between two processes.
- ✓ Binder uses an object reference as an address in a process's memory space.
- ✓ Synchronous call, reference couting

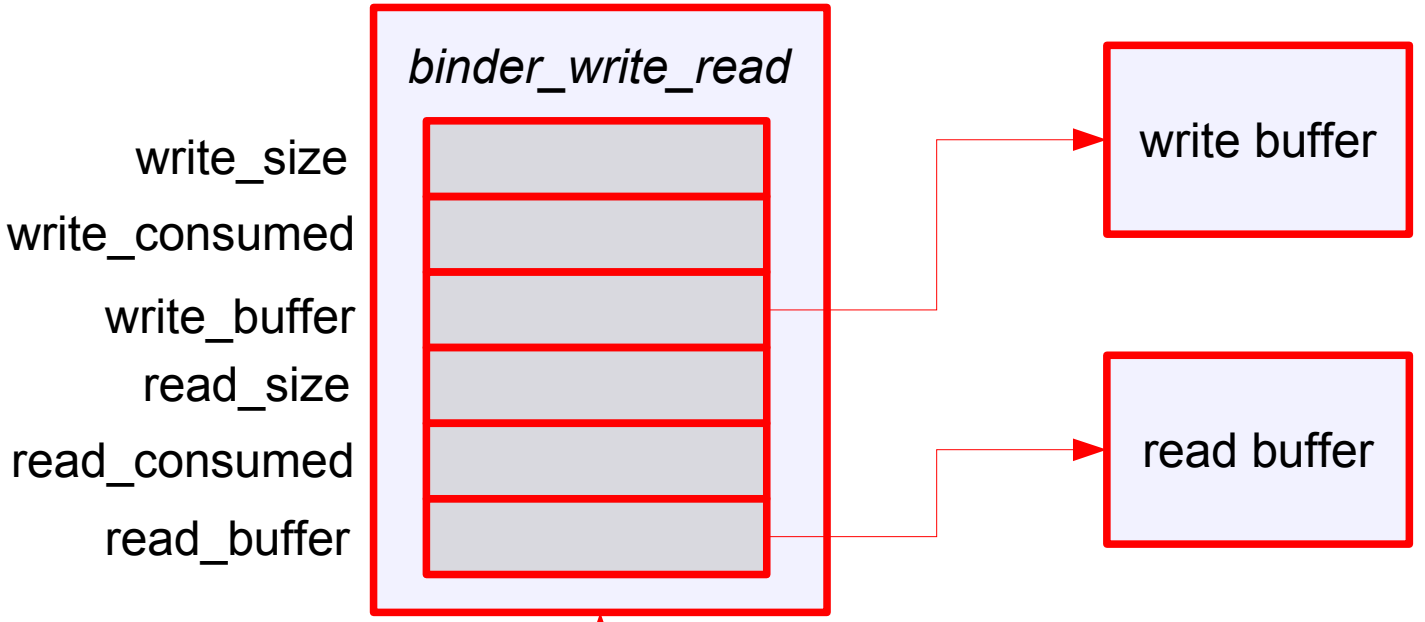
Binder


User space
Process1
Process2
Service Manager
Android libutil.so
/dev/binder
shared memory
Kernel

Binder is different from UNIX socket

	socket	binder
internal status	associated to FD	associated to PID (FD can be shared among threads in the same process)
read & write operation	stream I/O	done at once by **ioctl**
network transparency	Yes	No expected local only

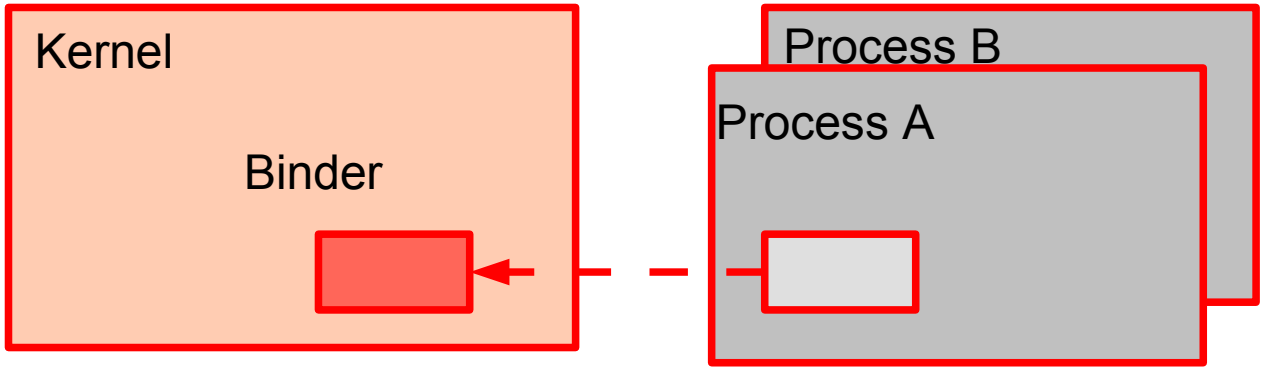
Transaction of Binder




```
if (ioctl(fd, BINDER_WRITE_READ, &bwt) >= 0)
    err = NO_ERROR;
else
    err = -errno;
```

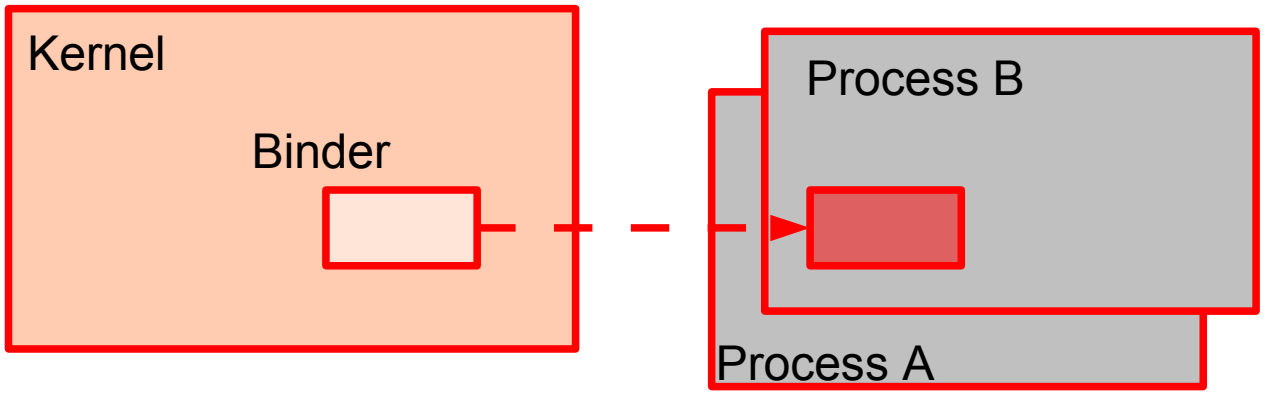
Transaction of Binder

Process A and B have different memory space.
They can not see each other.



Copy memory by **copy_from _user**

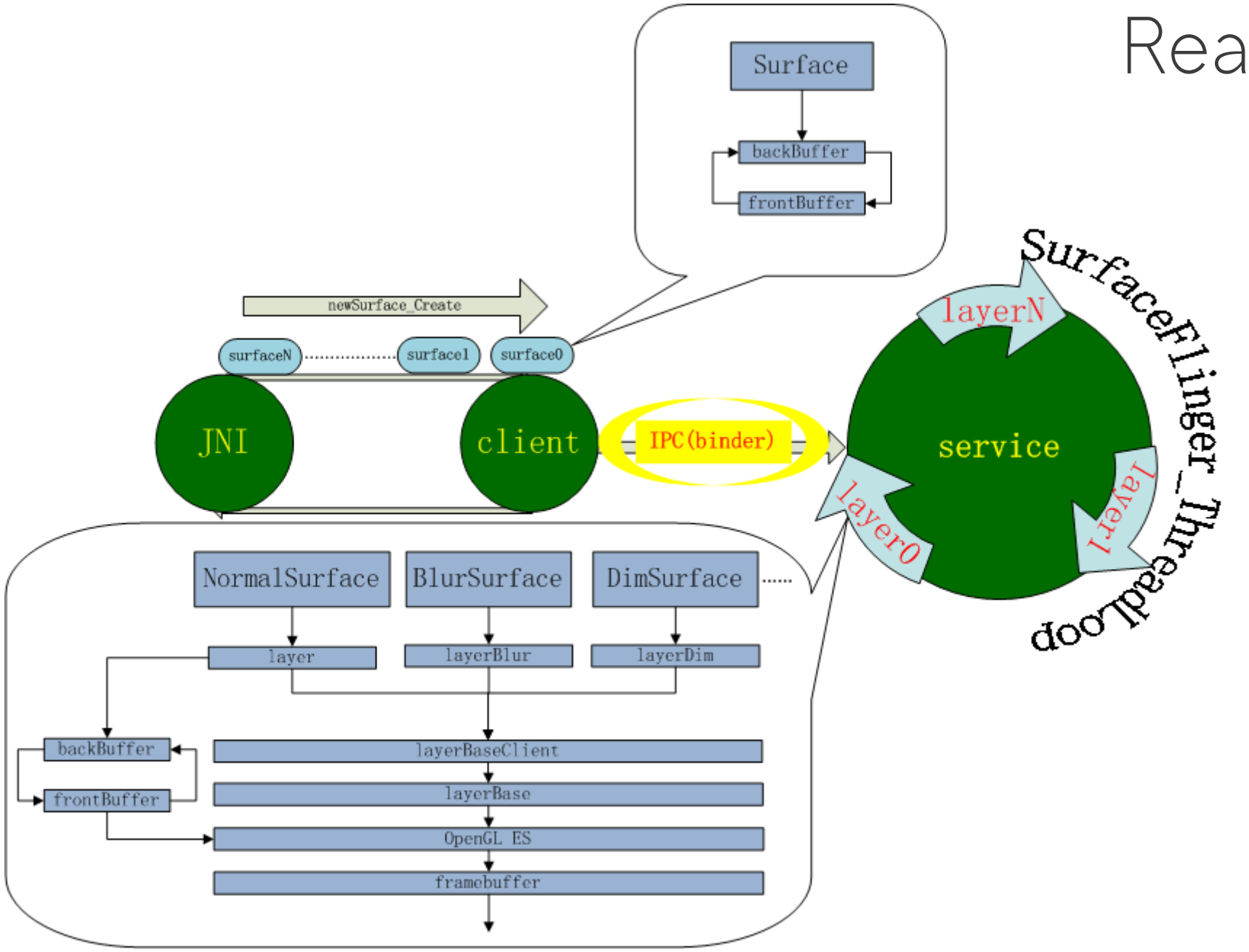
Then, wake up process B



Copy memory by **copy_to_user**

Internally, Android uses Binder for graphics data transaction across processes.
It is fairly efficient.

Real Case



Binder IPC is used for communicating between Graphics client and server.
Taken from http://www.cnblogs.com/xl19862005/archive/2011/11/17/2215363.html

Ashmem

- Android / Anonymous SHared MEMory subsystem
  - system/core/cutils/ashmem.h
  - int **ashmem_create_region**(const char *name, size_t size)
  - int **ashmem_set_prot_region**(int fd, int prot)
  - int **ashmem_pin_region**(int fd, size_t offset, size_t len)
  - int **ashmem_unpin_region**(int fd, size_t offset, size_t len)
- a named memory block shared between processes that the kernel is allowed to free.
  - **This is notable as the kernel is not allowed to free standard shared memory.**
- Similar to weak reference of Java. Useful to implement cache.
- Used in android.os.MemoryFile (Java), 2D memory allocator, etc.

Android Graphics

Surface

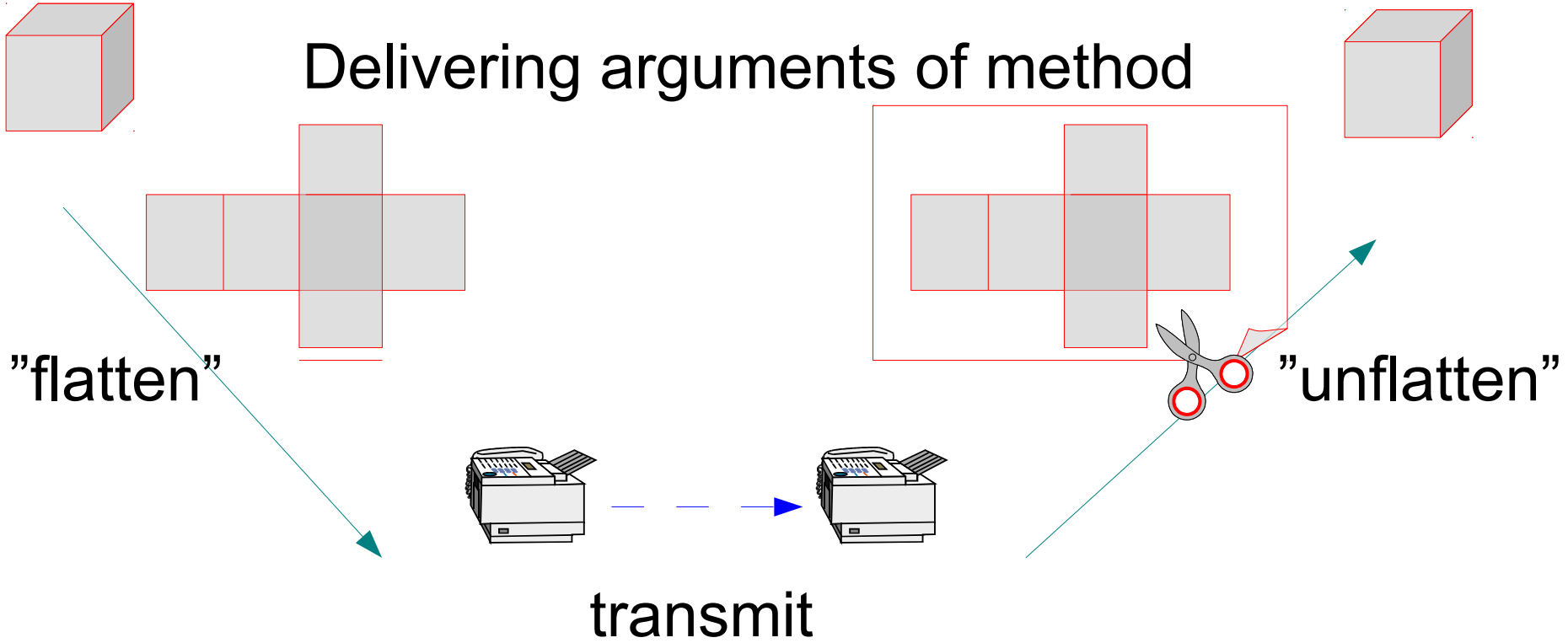
Source: frameworks/base/core/java/android/view/Surface.java

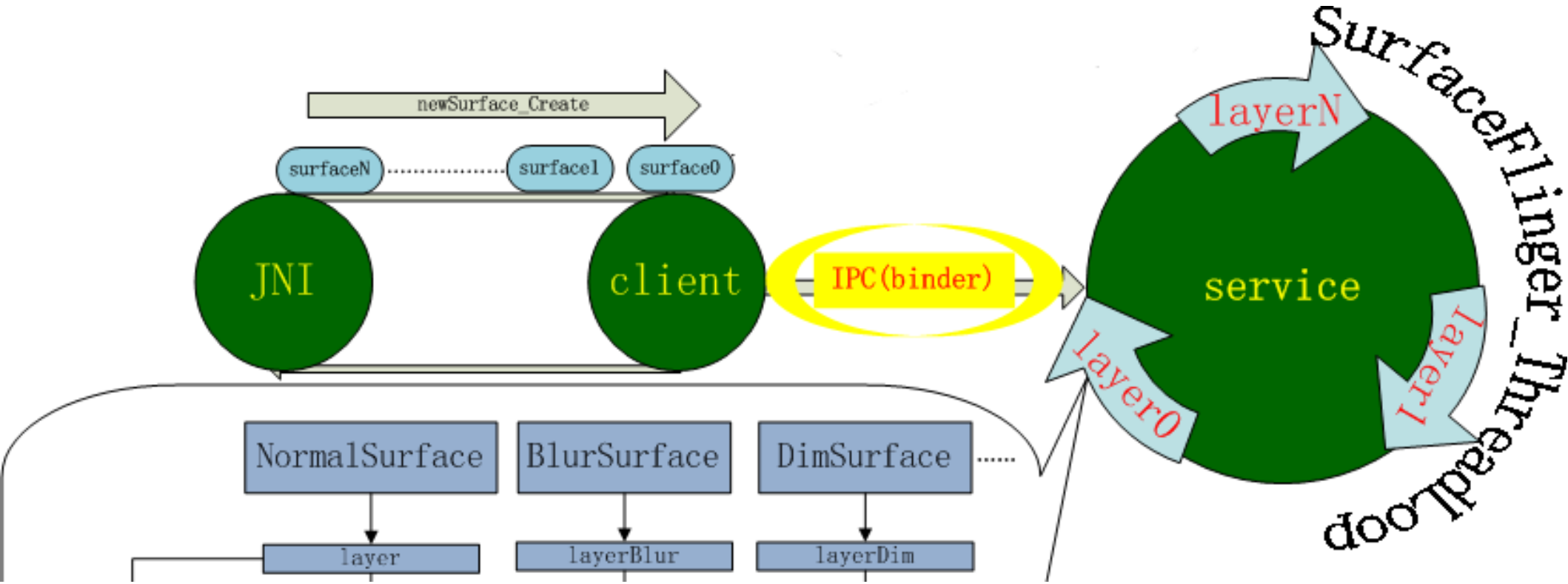
- /* **Handle on to a raw buffer that is being managed by the screen compositor** */

```
public class Surface implements Parcelable {
    public Surface() {
        mCanvas = new CompatibleCanvas();
    }
    private class CompatibleCanvas
                extends Canvas { /* ... */ }
}
```

Surface instances can be written to and restored from a Parcel.

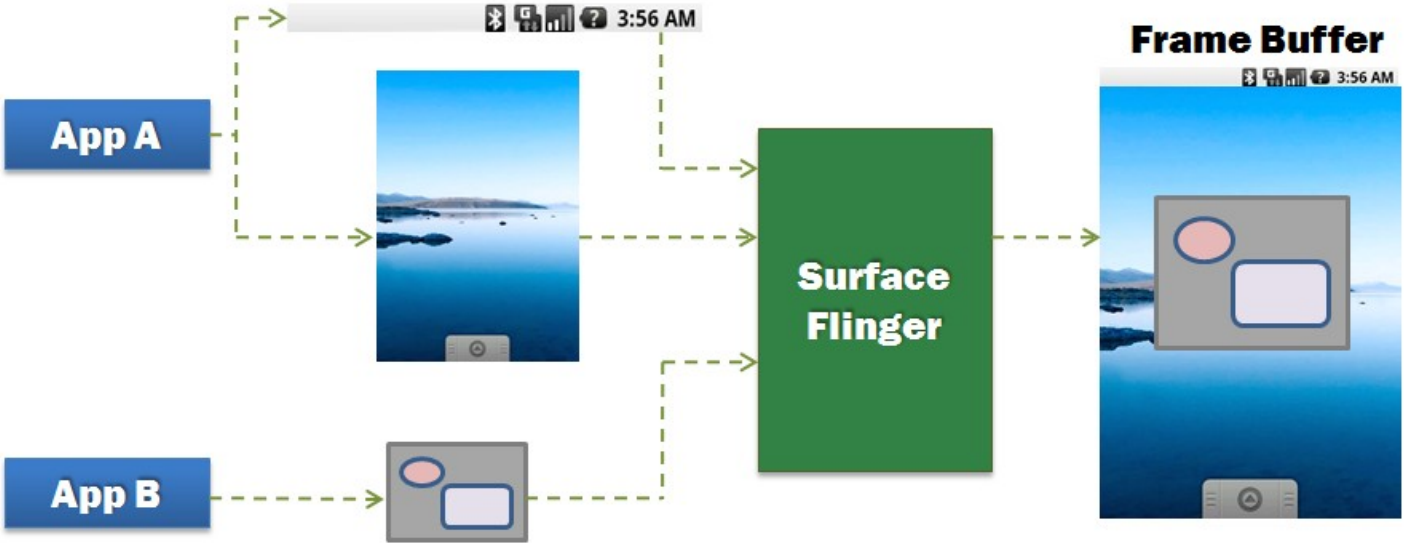
Delivering arguments of method

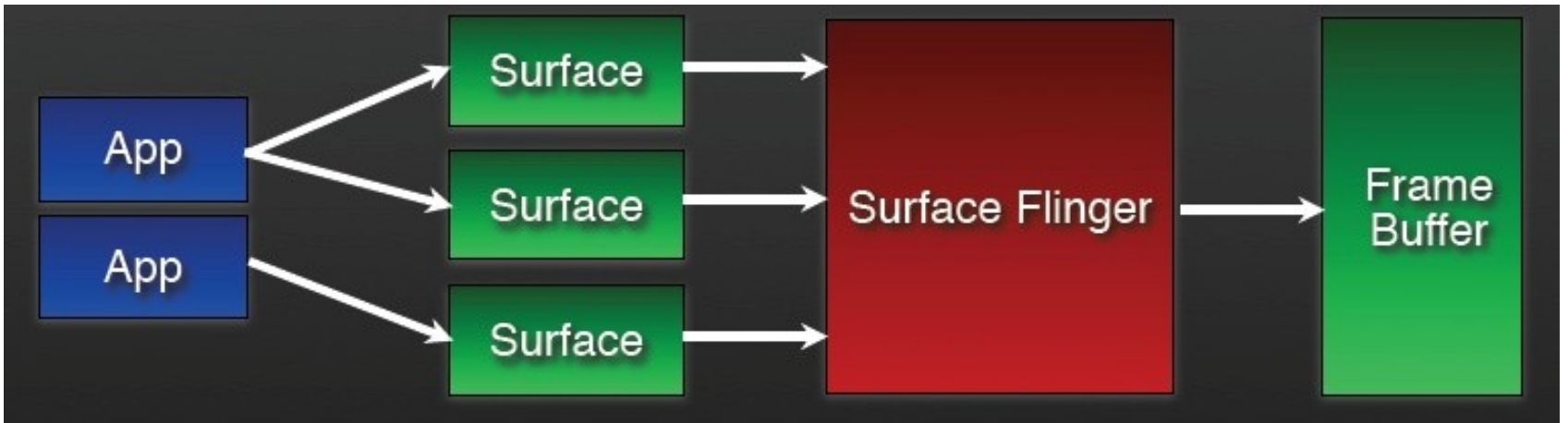




Android SurfaceFlinger

- Properties
  - Can combine 2D/3D surfaces and surfaces from multiple applications
  - Surfaces passed as buffers via Binder IPC calls
  - Can use OpenGL ES and 2D hardware accelerator for its compositions
    - Double-buffering using page-flip

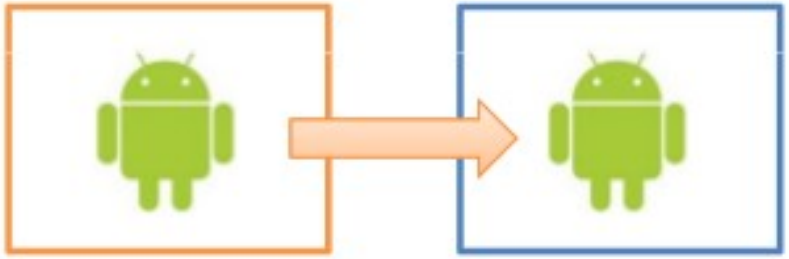




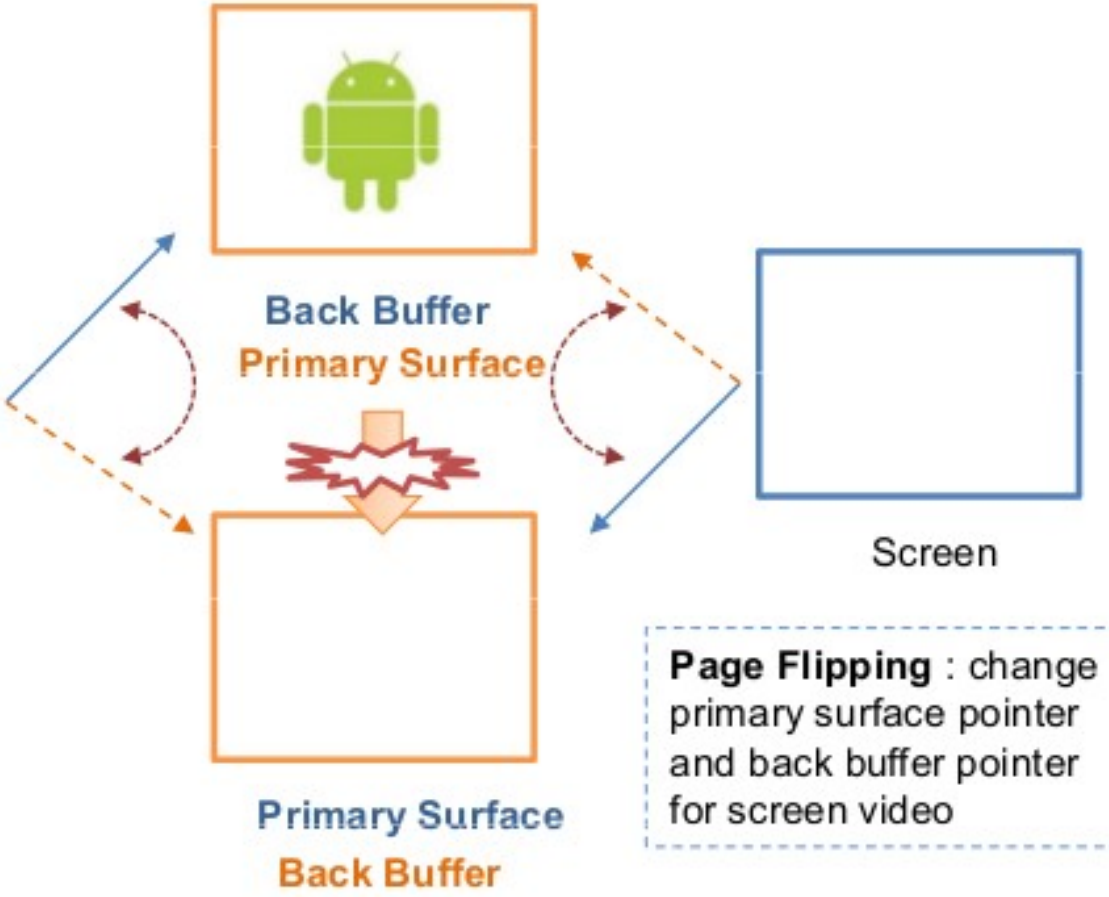
Double Buffering

Draw

Copy (BLT : Block Line Transfer)

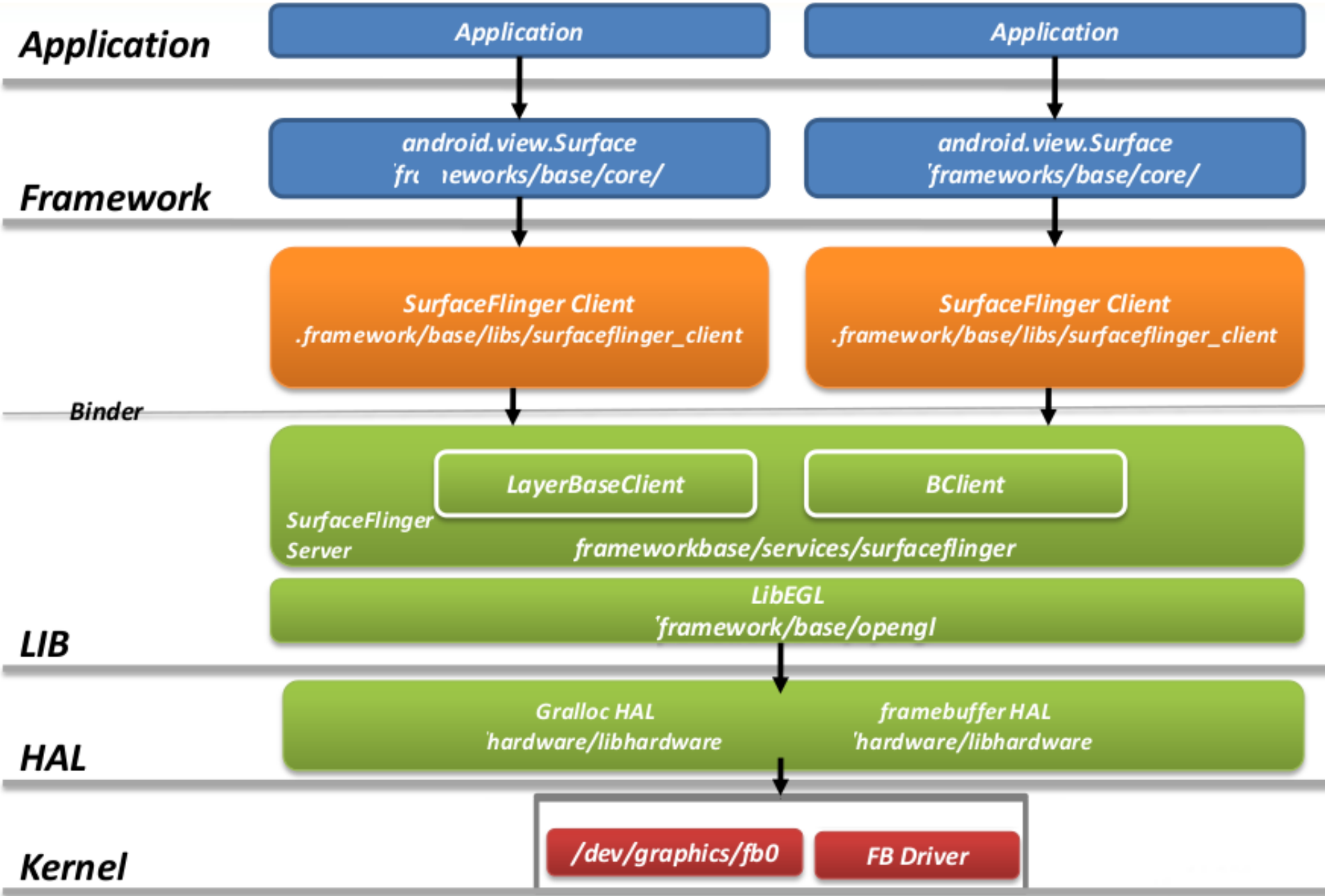


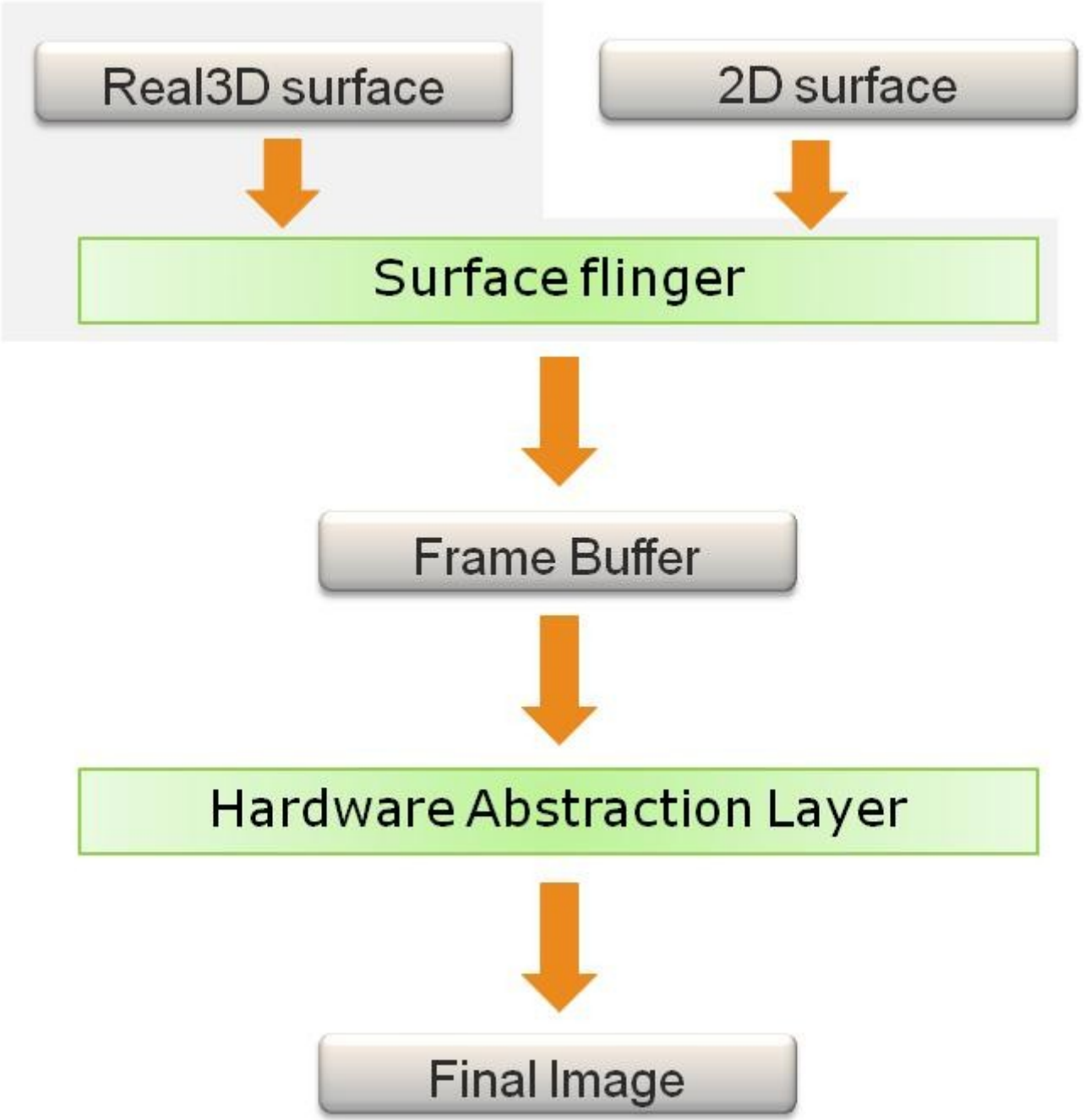
Page Flipping

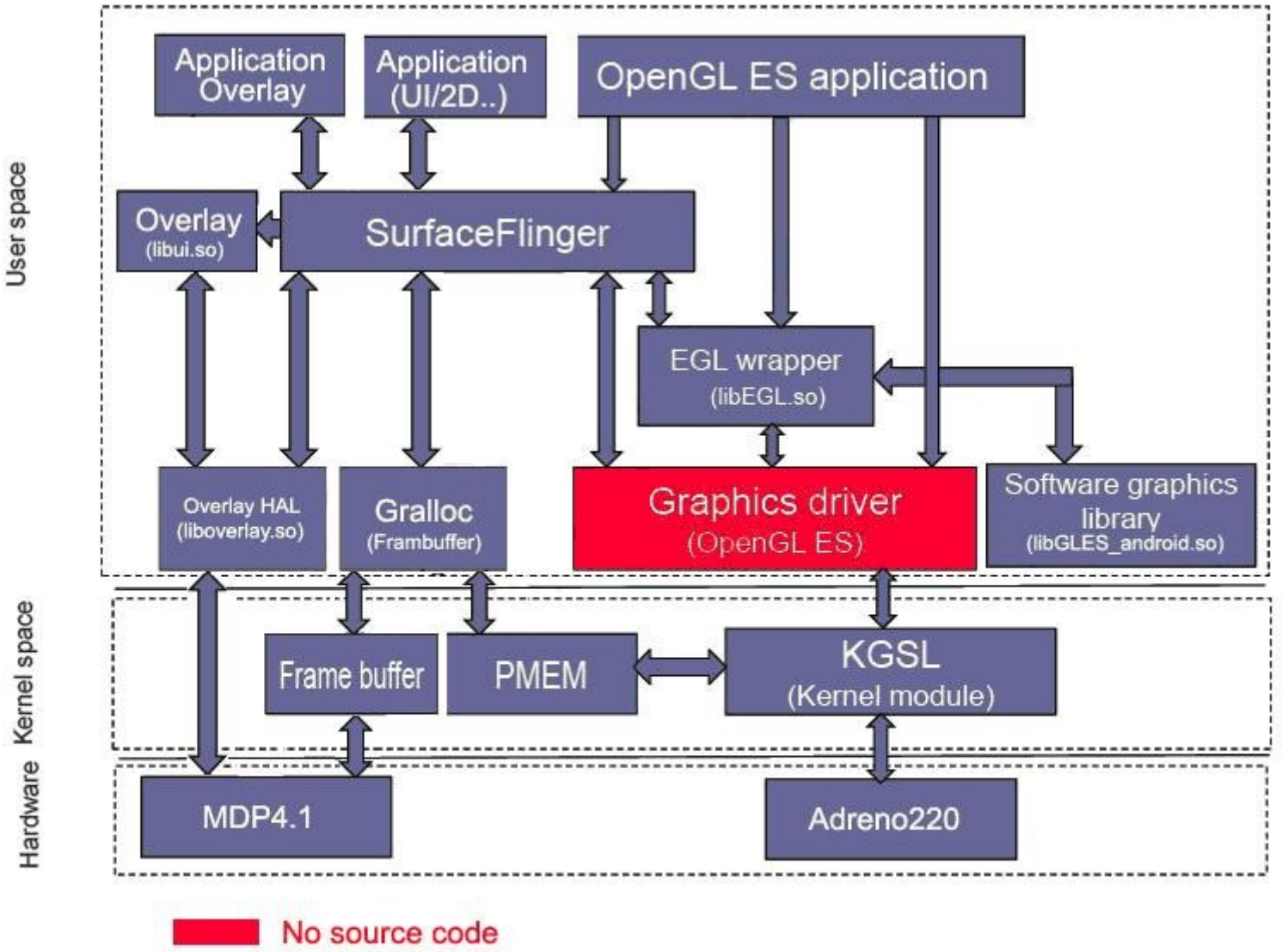


Page Flipping : change primary surface pointer and back buffer pointer for screen video

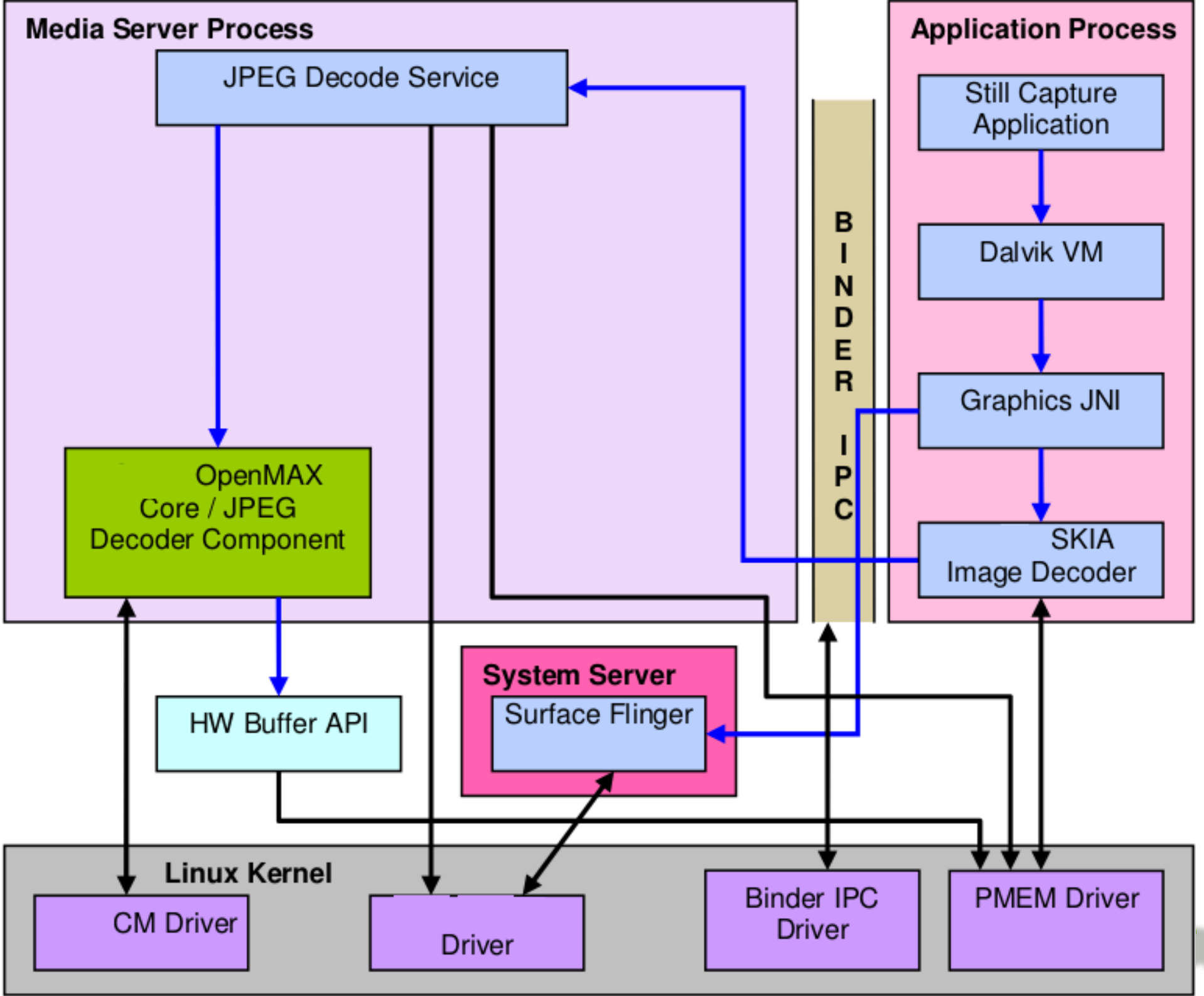
from SurfaceFlinger to Framebuffer

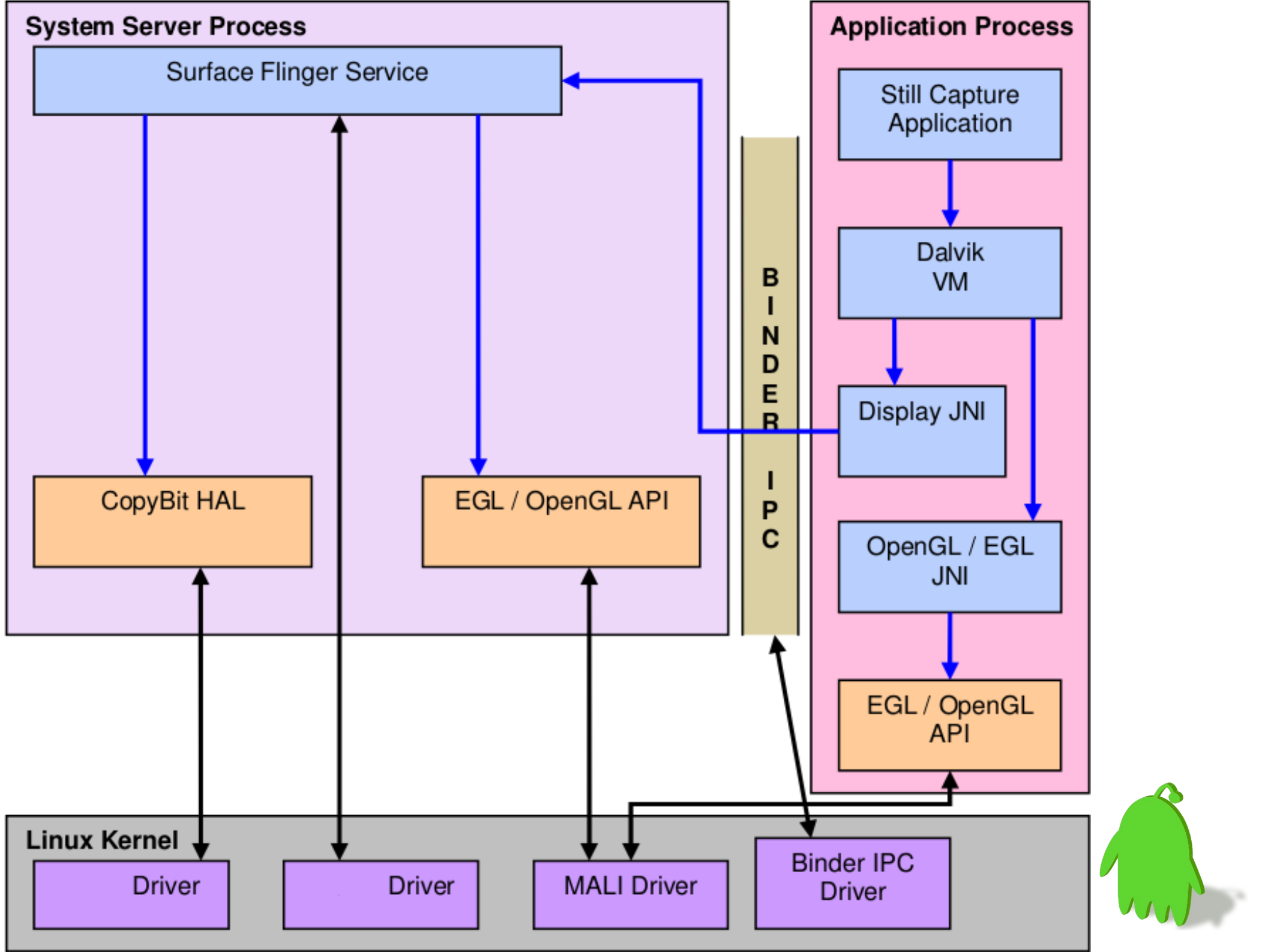


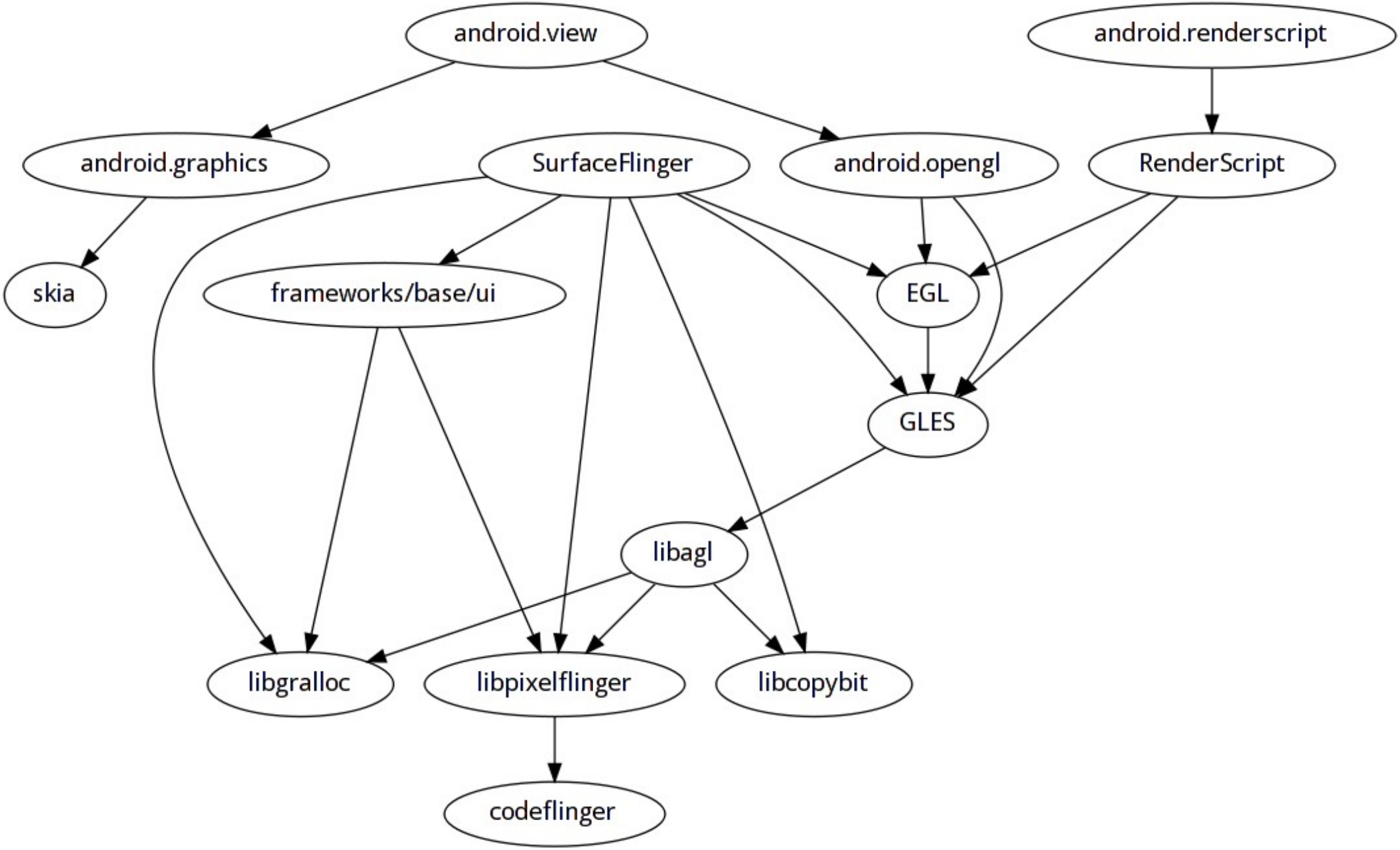

Real3D surface
2D surface
Surface flinger
Frame Buffer
Hardware Abstraction Layer
Final Image



Qualcomm 8x60 platform android(Froyo) system display architecture


Media Server Process
JPEG Decode Service
OpenMAX
Core / JPEG
Decoder Component
HW Buffer API
Application Process
Still Capture
Application
Dalvik VM
Graphics JNI
SKIA
Image Decoder
BINDER IPC
System Server
Surface Flinger
Linux Kernel
CM Driver
Driver
Binder IPC
Driver
PMEM Driver

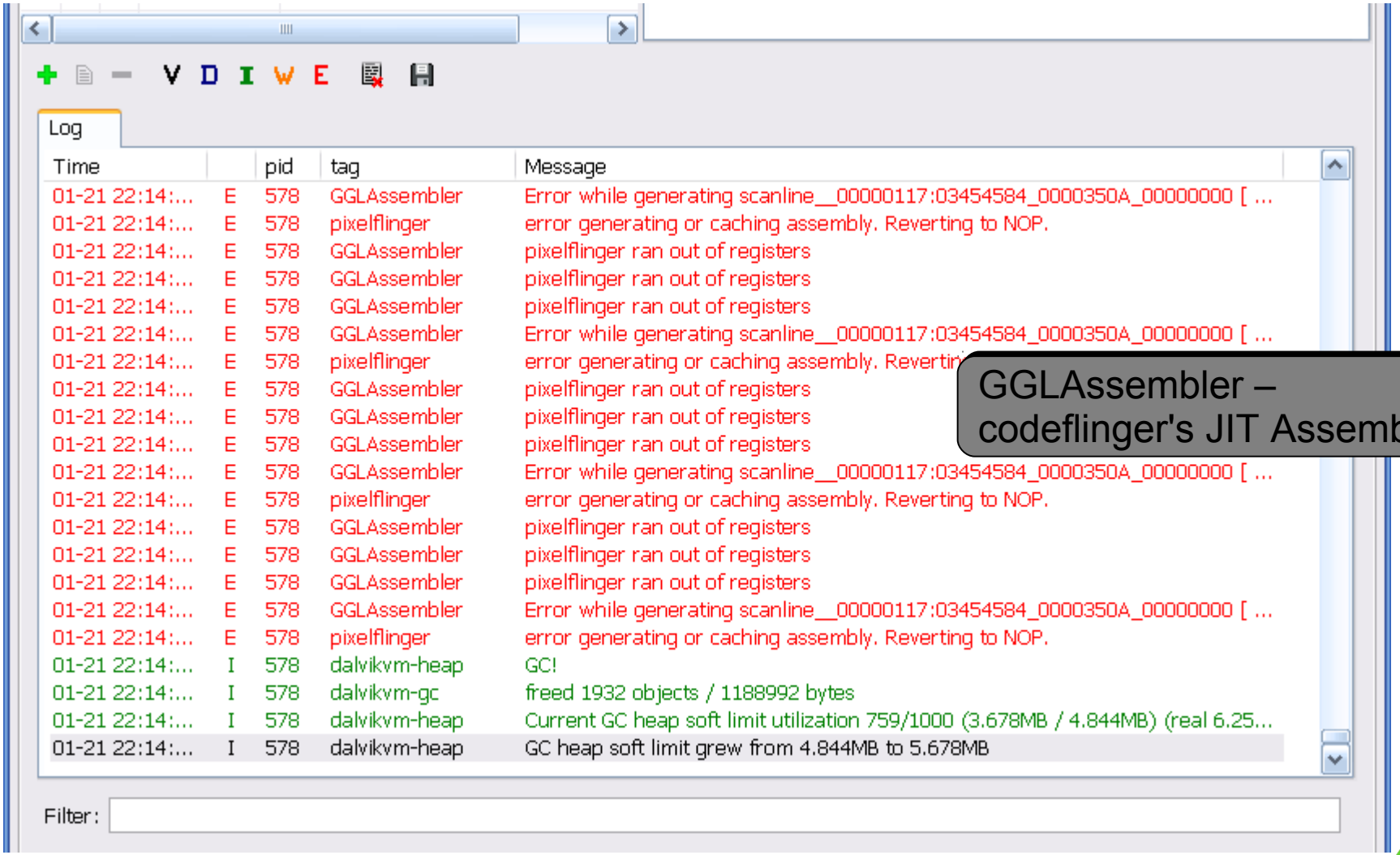

System Server Process
Surface Flinger Service
CopyBit HAL
EGL / OpenGL API
BINDER IPC
Application Process
Still Capture Application
Dalvik VM
Display JNI
OpenGL / EGL JNI
EGL / OpenGL API
Linux Kernel
Driver
Driver
MALI Driver
Binder IPC Driver


android.view
android.renderscript
android.graphics
SurfaceFlinger
android.opengl
RenderScript
skia
frameworks/base/ui
EGL
GLES
libagl
libgralloc
libpixelflinger
libcopybit
codeflinger

PixelFlinger : software renderer

- Render functions: pointx, linex, recti, trianglex
- Texture and color buffer: activeTexture, bindTexture, colorBuffer, readBuffer, depthBuffer, BindTextureLod
- …
- Device framebuffer functions: copyPixels, rasterPos2x, rasterPos2i
- Optimizer: codeflinger (JIT assembler)

```
I/SurfaceFlinger( 1931): OpenGL informations:
I/SurfaceFlinger( 1931): vendor    : Android
I/SurfaceFlinger( 1931): renderer : Android PixelFlinger 1.2
I/SurfaceFlinger( 1931): version   : OpenGL ES-CM 1.0
```



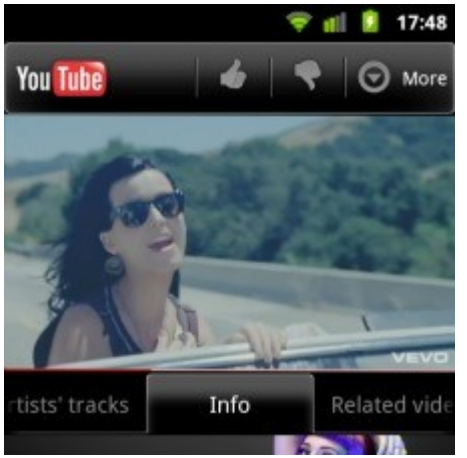
GGLAssembler – codeflinger's JIT Assembler

2D and Accelerator

Skia

- SkBitmap
- SkBitmap::Allocator
- GPU integration in Android 4.0

Case Study: WebKit rendering



Fill rectangle

FillRectangles()

Set clip region

Show glyphs

BatchBlit()

Acquire font cache

Blit(), memcpy()

Allocate font cache

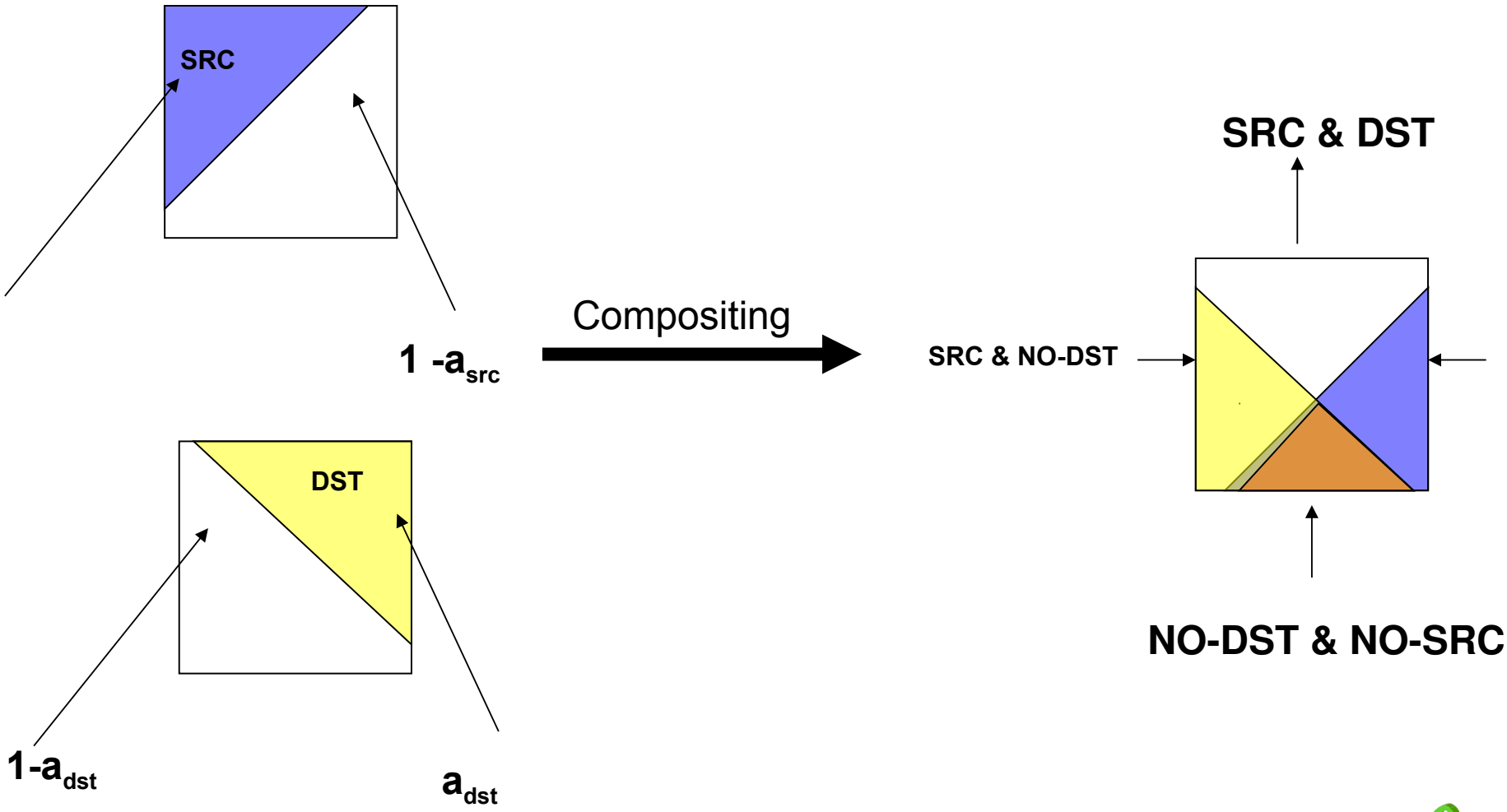
Surface composite

Blit()
TileBlit()

Prepare / finish composite

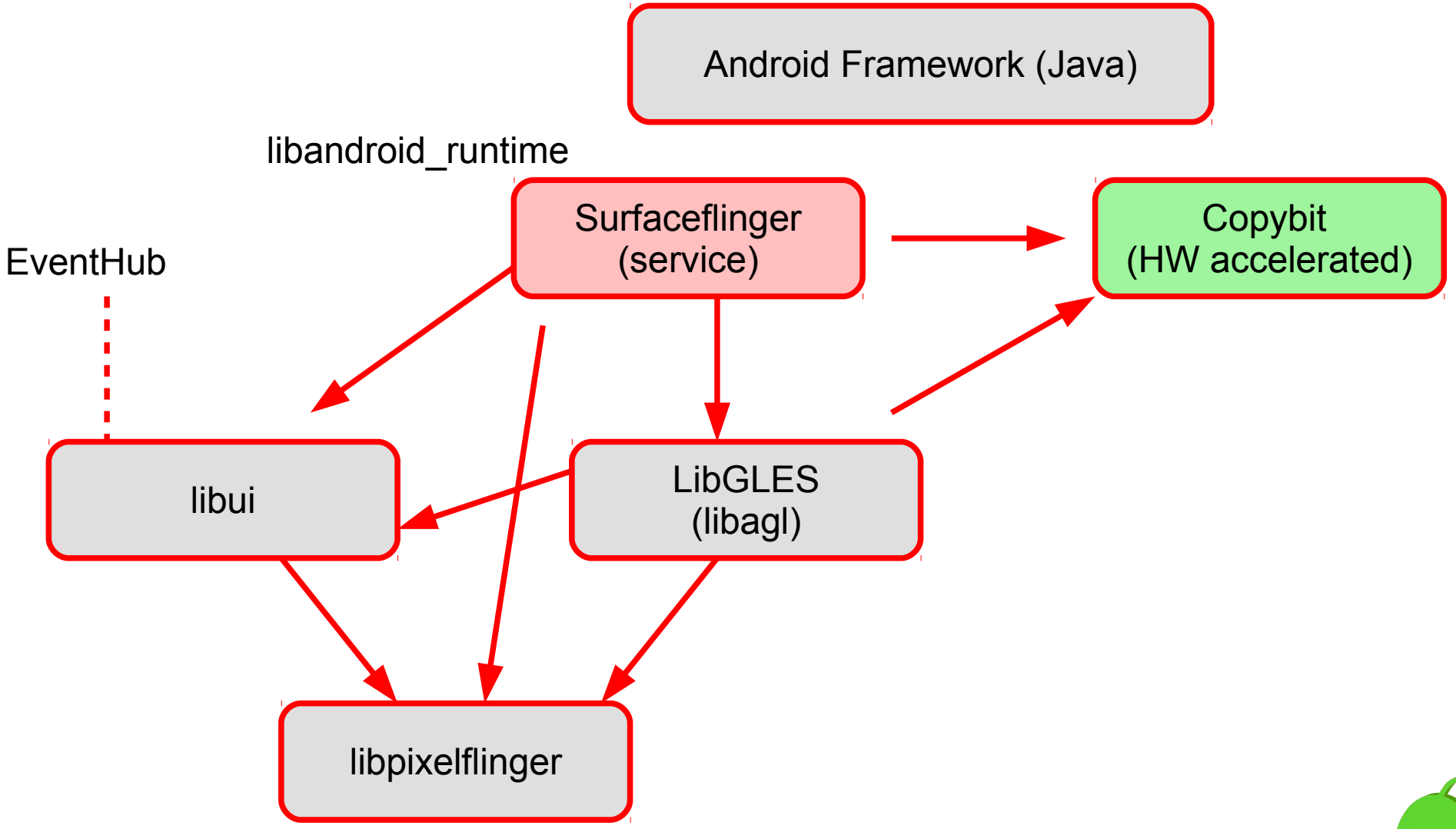
Clear()

2D Compositing



Enable 2D Accelerator

Android Graphics (HAL view)



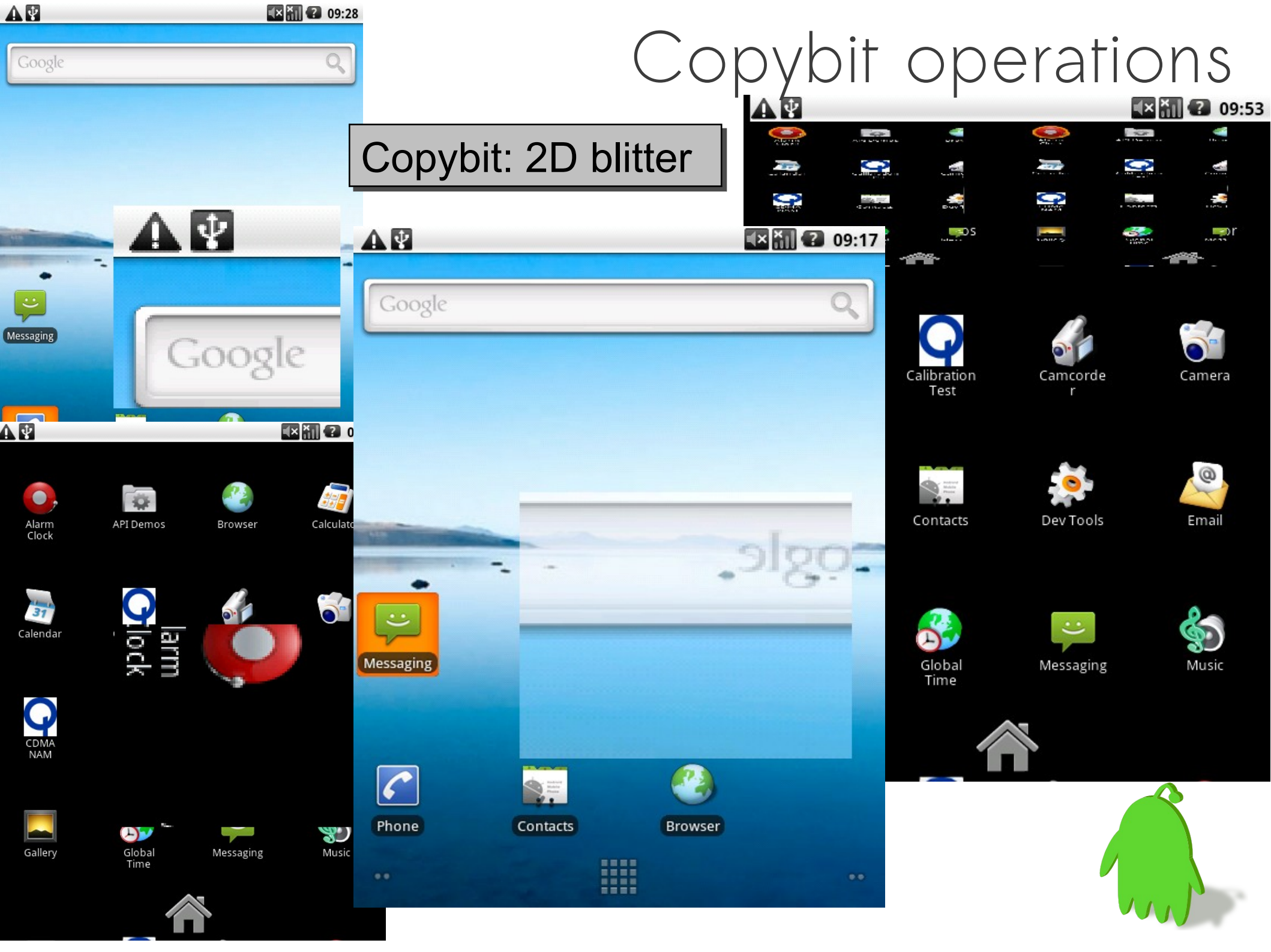
2D Accelerator for Android Graphics

- lib**copybit** provides hareware bitblit operations which includes moving, scaling, rotation, mirroring, and more effects, like blending, dithering, bluring, etc.
- Removed since Android 2.3
- Android has two copybit interfaces:
  - **Blit: moving / blending**
  - **Stretch: scaling besides moving**
- libcopybit is called by libagl which can do swapBuffers to do the framebuffer page flipping that can also be accelerated by libcopybit.

Copybit could improve the performance of page flipping

Copybit operations

Copybit: 2D blitter

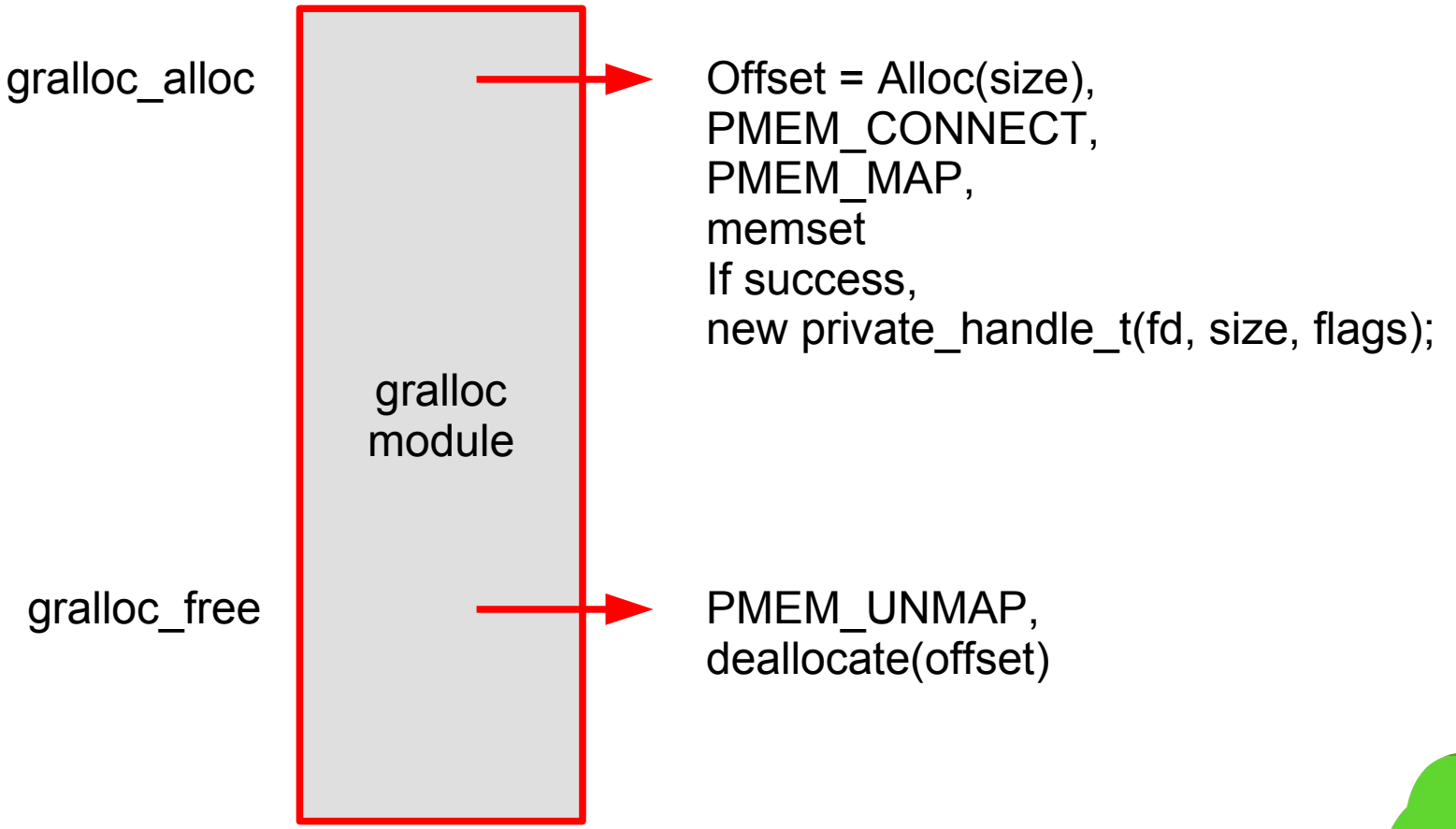


PMEM: Manipulate physically continuous memory

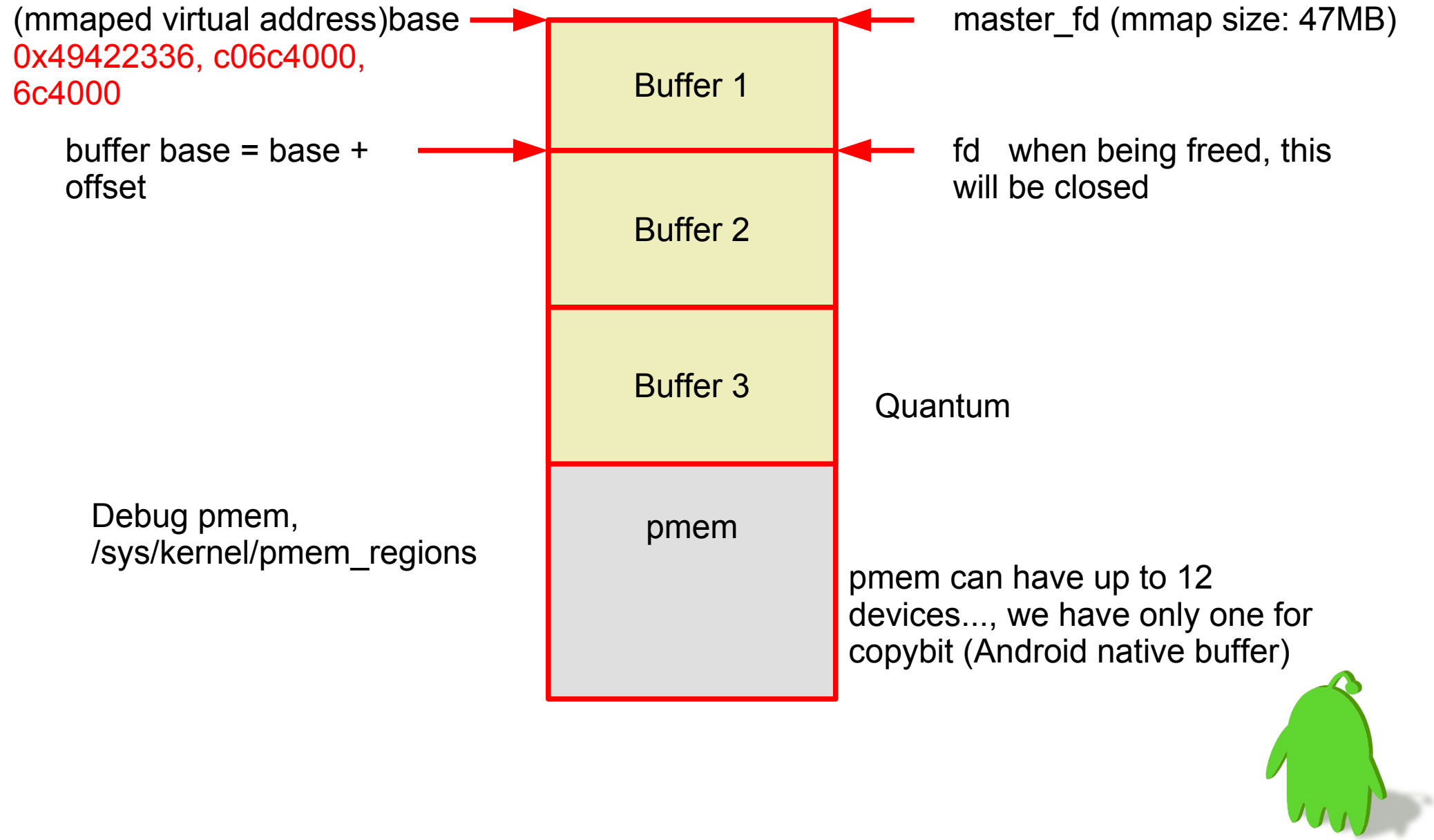
- Hardware graphics/bitblt operations (blitter) needs physical continuous memory to manipulate.
- Android use libgralloc to allocate pmem (physical continuous memory) for android native buffer. pmem driver can support up to 12 devices, we have only one for copybit (Android, android native buffer)
- While running 0xbench, the peak size of the pmem allocated (mapped) is 25194496 bytes.

Take Qualcomm MSM7x25 for example:
- /dev/pmem
- /dev/pmem_adsp
  - For multimedia codec, audio, video, camera

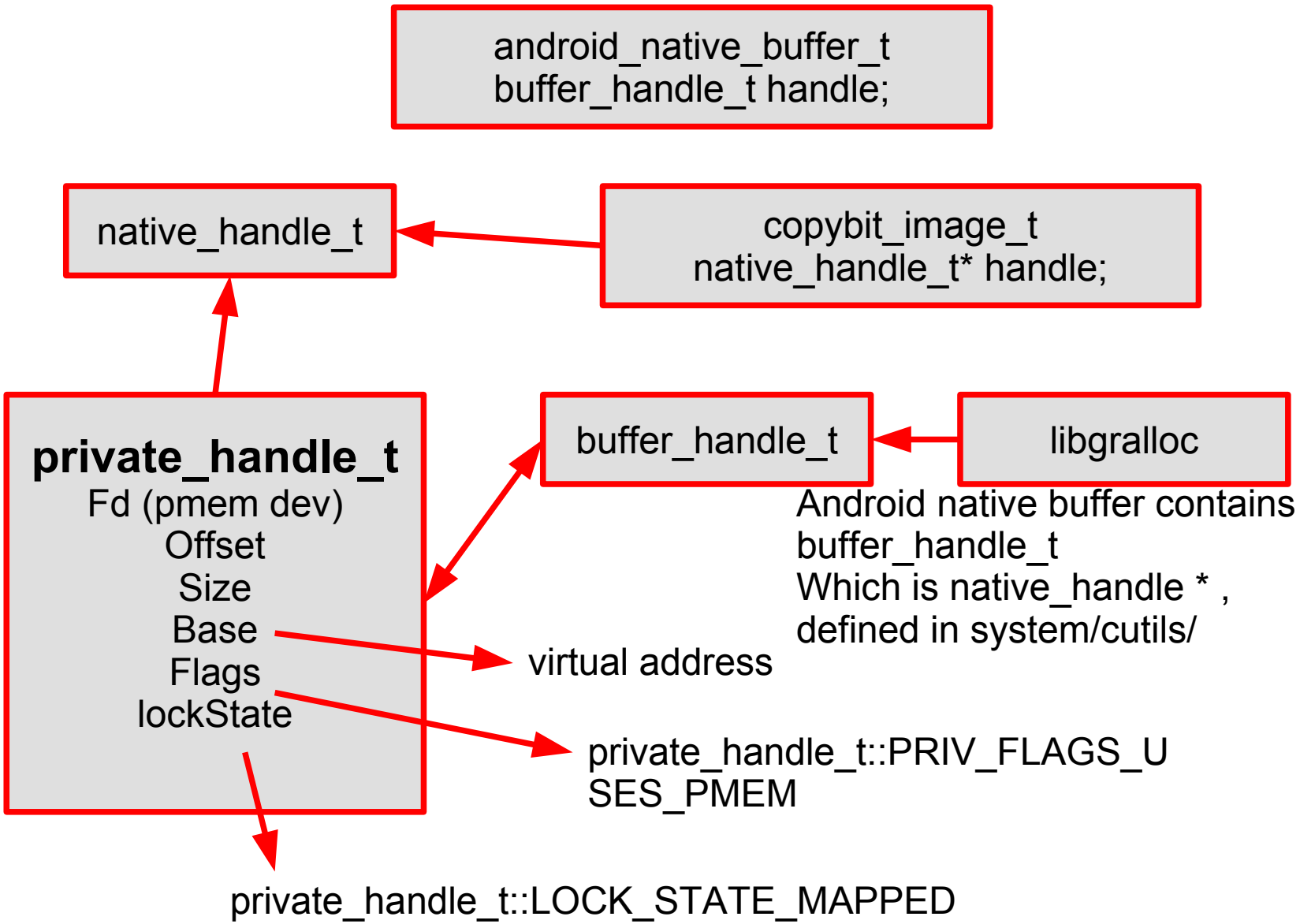
gralloc & pmem



pmem memory


(mmaped virtual address)base
0x49422336, c06c4000,
6c4000
buffer base = base +
offset
Buffer 1
Buffer 2
Buffer 3
pmem
master_fd (mmap size: 47MB)
fd when being freed, this
will be closed
Quantum
Debug pmem,
/sys/kernel/pmem_regions
pmem can have up to 12
devices..., we have only one for
copybit (Android native buffer)

gralloc/copybit structure



Functions in copybit HAL

- BLIT (moving)
- Stretch (scaling)
- Alpha-blending
  - **Unused in real case**
- Rotate
  - **Unused in real case**

OpenGL|ES

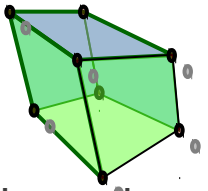
Key Concepts

- Android is moving to OpenGL|ES accelerated rendering since version 2.x
- Window systems already comprehend z-order
- 3D != 2D with depth
- OpenGL is object based
  - **Describes a scene using its components and properties. Output quality is dependent on the renderer, not source**

OpenGL Terminology

- OpenGL

  An API from Khronos (from SGI), for constructing a 3D object, doing operations on the object, and displaying it



- Primitives

  Triangles, Lines, Points, that can be specified through vertices to define an arbitrary shape

- Texture

  Small (!) bitmap to make objects more realistic

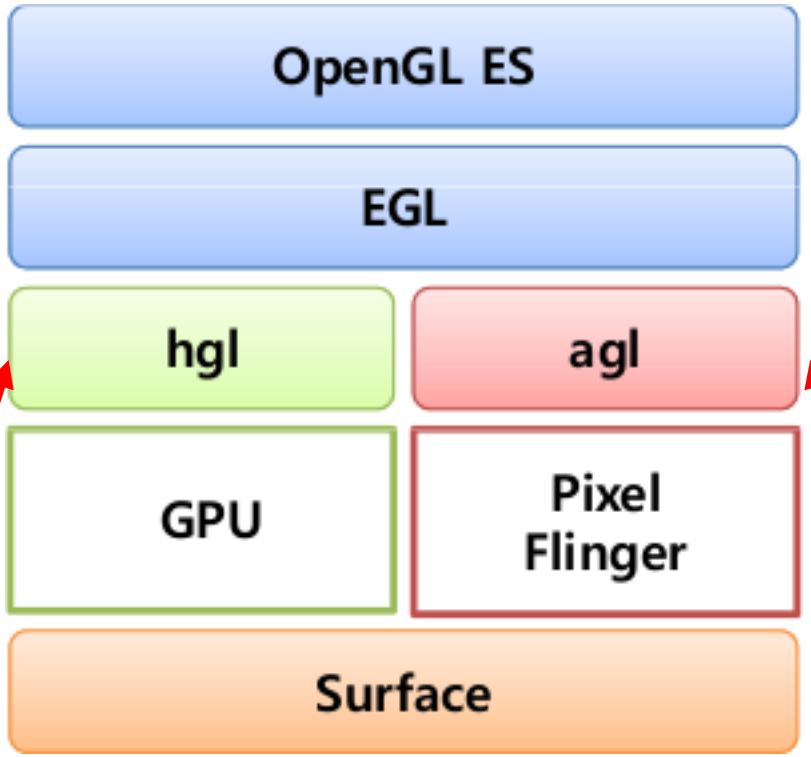
- EGL

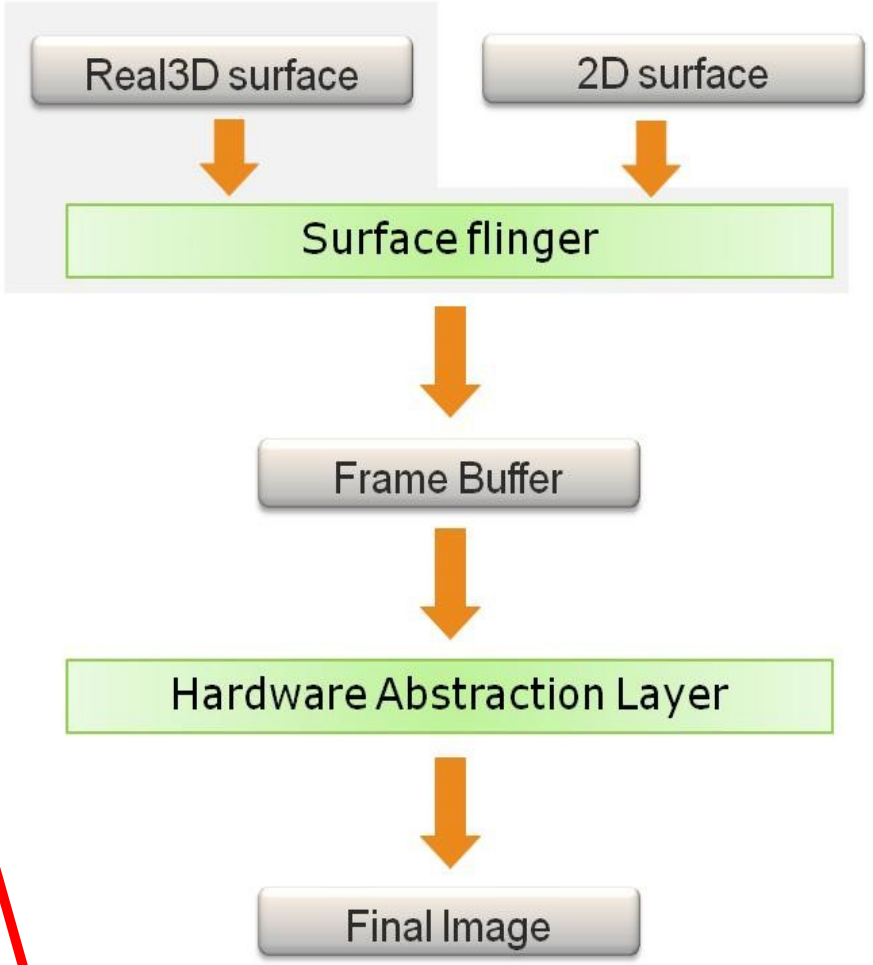
  The EGL API defines a portable mechanism for creating GL contexts and windows for rendering into, which may be used in conjunction with different native platform window systems using the WSEGL layer

EGL

- EGL (Embedded-System Graphics Library) is an interface between Khronos rendering APIs (such as OpenGL ES or OpenVG) and the underlying native platform window system.
- EGL handles graphics context management, surface/buffer binding, and rendering synchronization and enables high-performance, accelerated, mixed-mode 2D and 3D rendering using other Khronos APIs.
- EGL Surfaces
  - **windows - on-screen rendering**
  - **pbuffers - off-screen rendering**
  - **pixmaps - off-screen rendering**

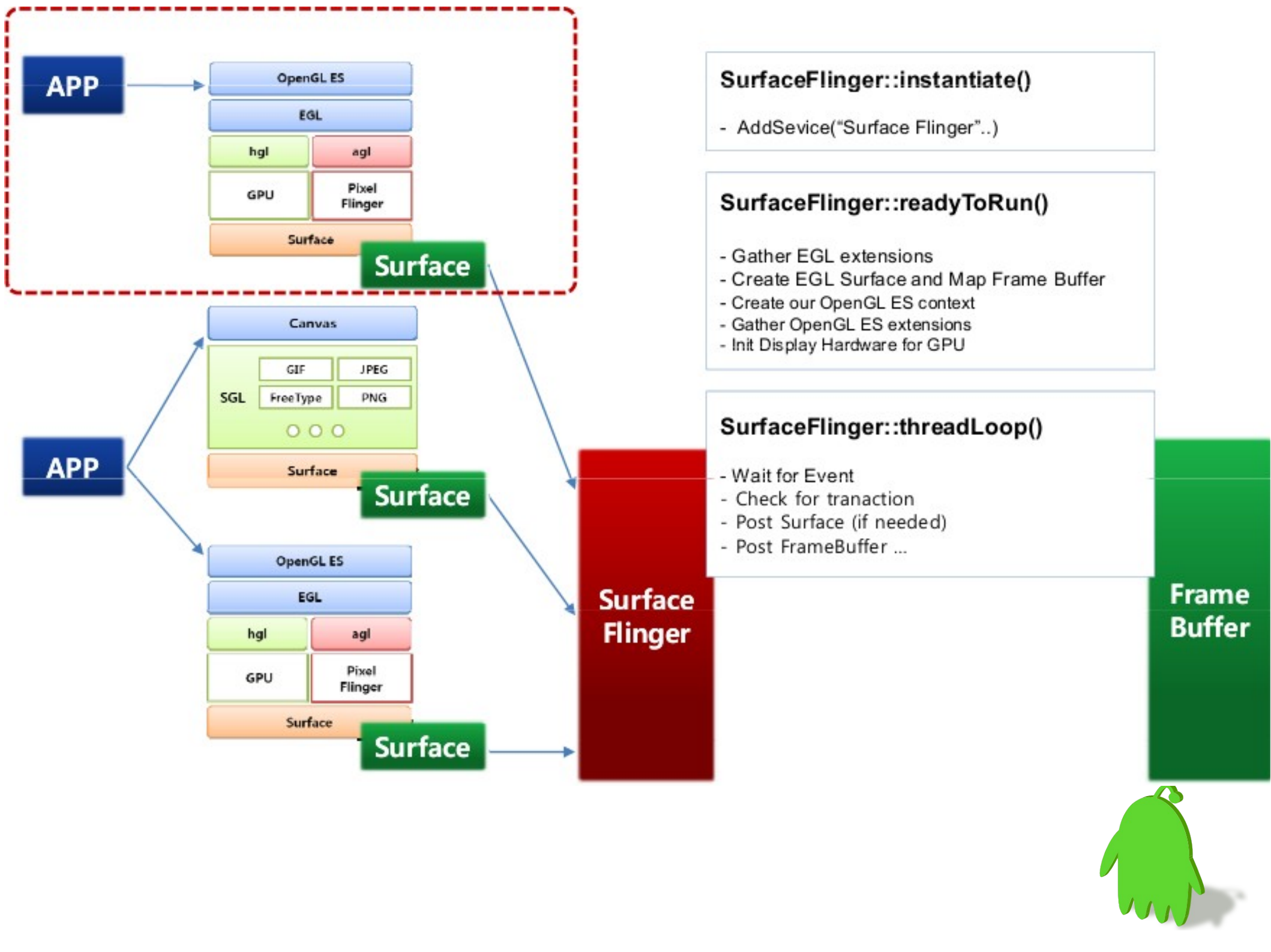
from EGL to SurfaceFlinger





hgl = hardware OpenGL|ES

agl = android software OpenGL|ES renderer


APP
OpenGL ES
EGL
hgl
agl
GPU
Pixel Flinger
Surface
Surface
APP
Canvas
SGL
GIF
JPEG
FreeType
PNG
Surface
Surface
OpenGL ES
EGL
hgl
agl
GPU
Pixel Flinger
Surface
Surface
Surface Flinger
SurfaceFlinger::instantiate()
- AddSevice("Surface Flinger"..)
SurfaceFlinger::readyToRun()
- Gather EGL extensions
- Create EGL Surface and Map Frame Buffer
- Create our OpenGL ES context
- Gather OpenGL ES extensions
- Init Display Hardware for GPU
SurfaceFlinger::threadLoop()
- Wait for Event
- Check for tranaction
- Post Surface (if needed)
- Post FrameBuffer ...
Frame Buffer

http://0xlab.org