NTU CSIE Open Source System Software

2016.03.29

An Introduction to the Android Framework

-- a core architecture view from apps to the kernel --

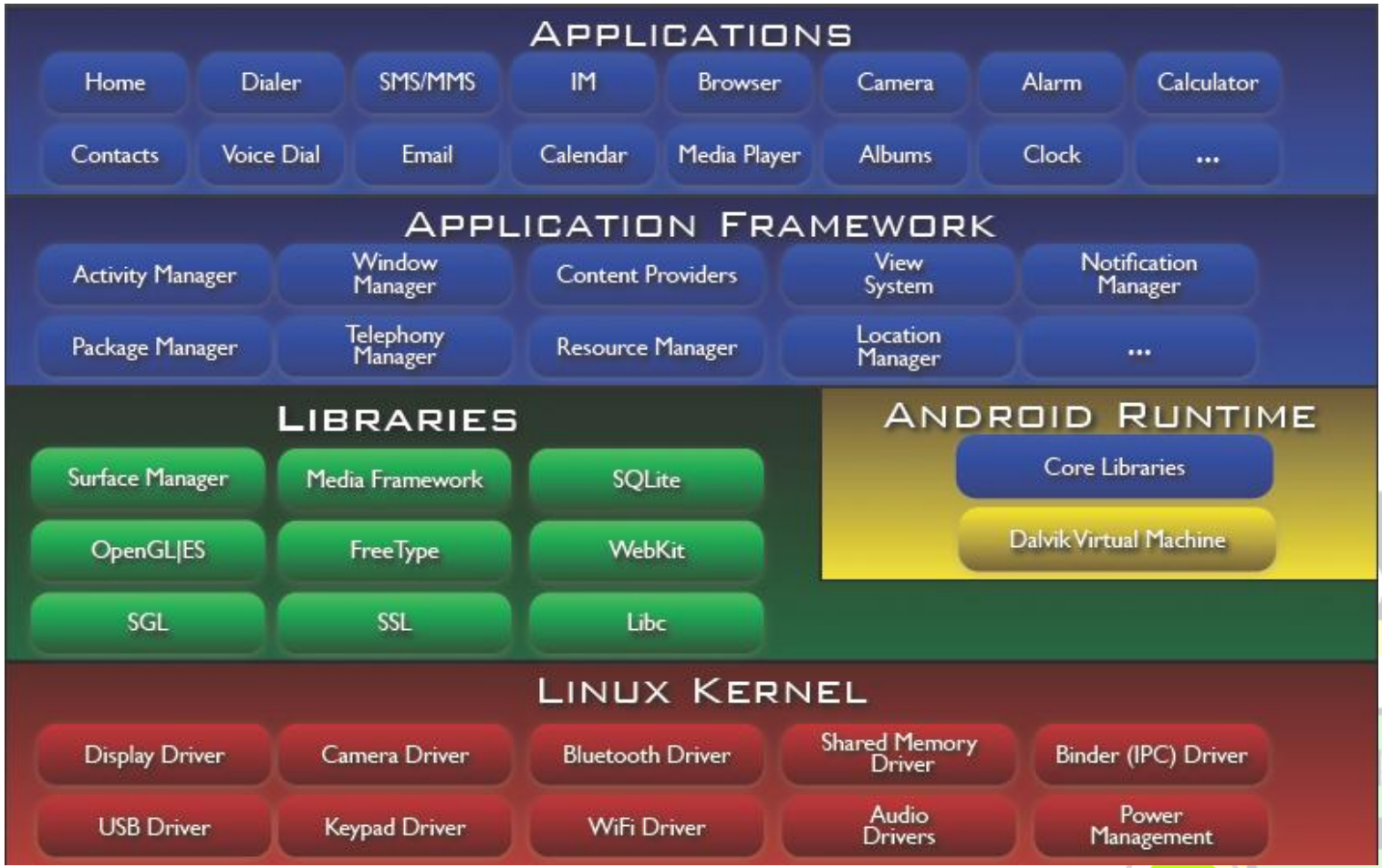
William W.-Y. Liang (梁文耀), Ph. D.

http://www.ntut.edu.tw/~wyliang

for 台大資工系開源系統軟體課程

hosted by Prof. Shih-Hao Hung

Android Architecture



Source from http://developer.android.com/

Think: user-space Android?

Advantage of the Android Operating System

- High application portability
- Consistent user experience
- Component-based design
- Suitable for resource-limited handheld devices
- Consider the performance issue
- Acceptable effort to port to different hardware

Think: Architecture design and considerations

- Multi-layer design
- Programming languages
- License issues

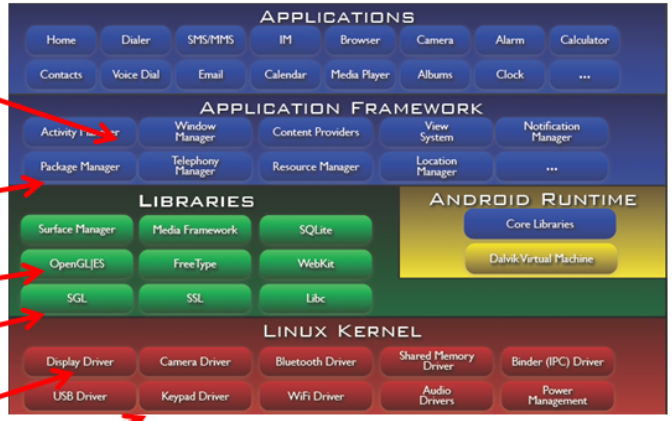
Android/Linux

- Android differs from the typical GNU/Linux in that it adopts only the Linux kernel, not everything.
- The first process ‘init’ transfers to Android’s own middleware and application environment.
- BSD libc is used instead of glibc or uClibc.
- As a result, it is called **Android/Linux**, not GNU/Linux.

Think: **how different they are?**

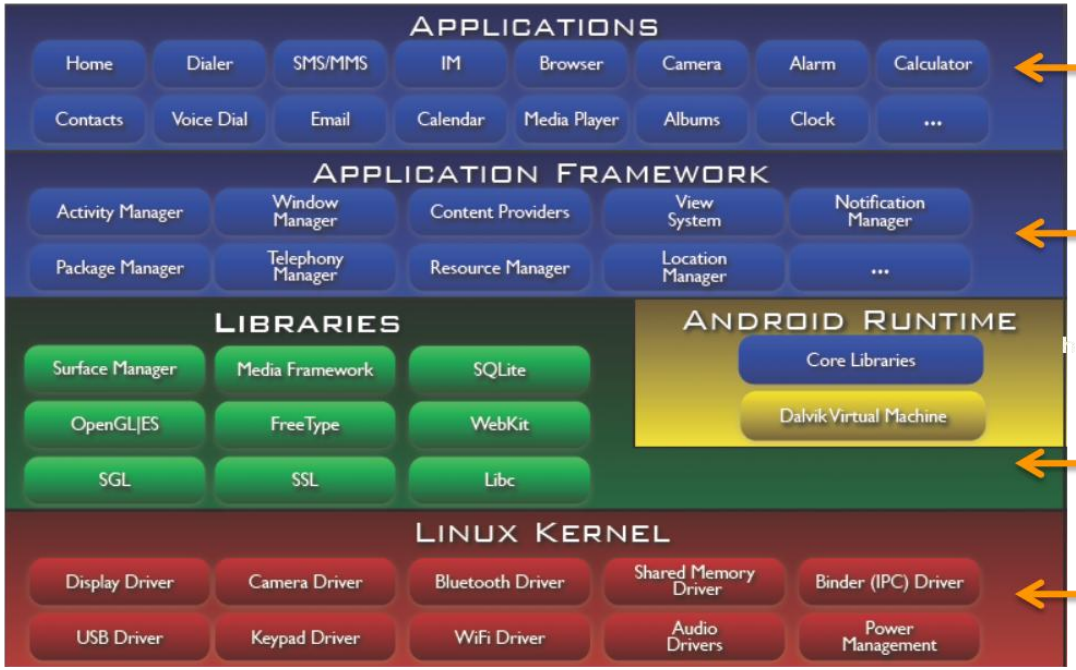
Android/Linux System Integration

- Android Application
- Android Application Framework
- Android Java/C/C++ Interface
- Android/Linux Native Code
- Linux System Calls
- Linux Device Driver
- Hardware and Control



Think: typical efforts for Android system integration

Interface between Layers



Develop Apps

Using Framework Components

Add Framework

Over the JNI interface

Add Native Control Code

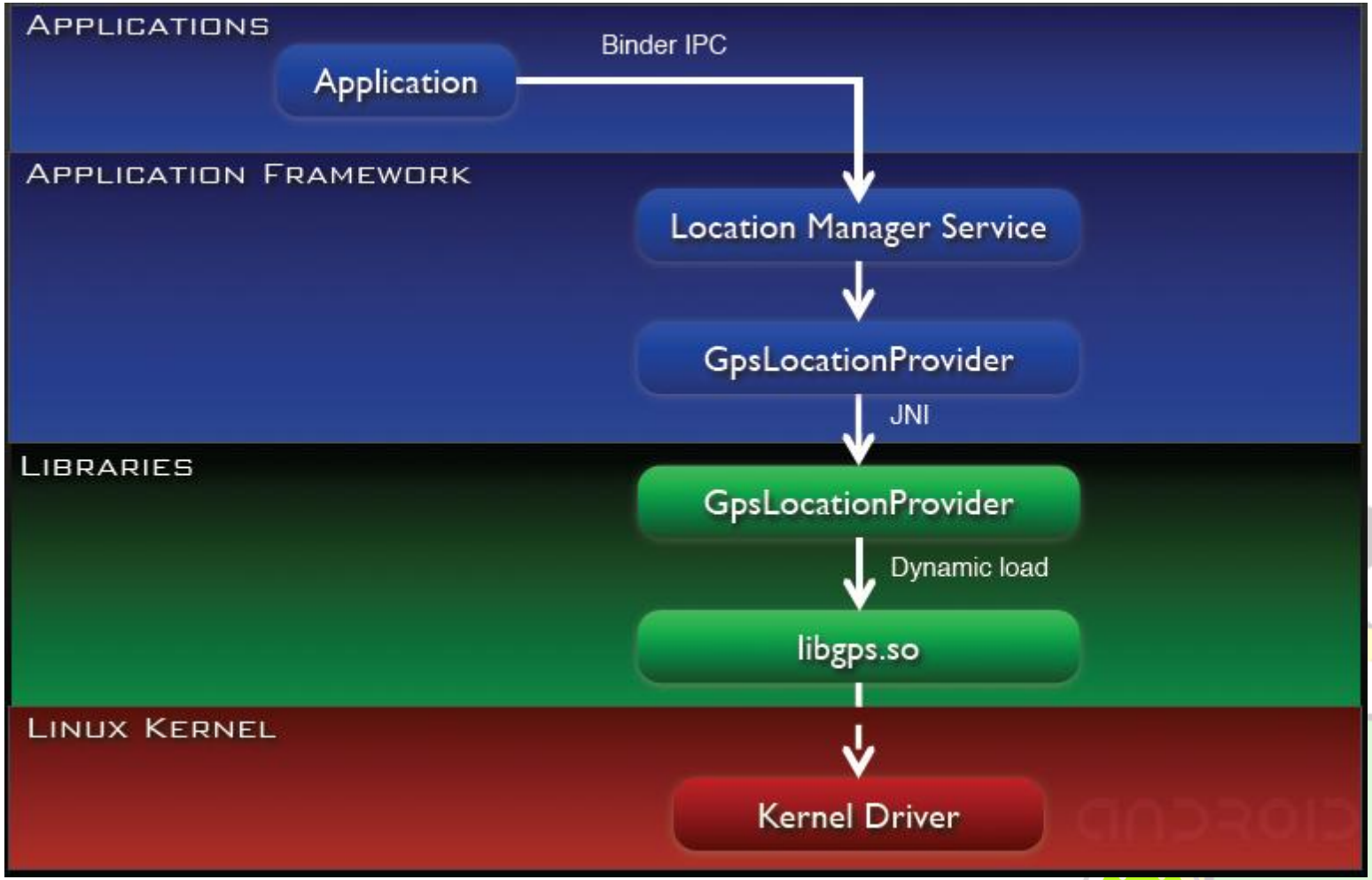
Over the system call

Add Driver

Android System Integration Procedure

1. Add driver
2. Add user-space JNI C/C++ native library
3. Add the Java Middleware
   - Service
   - Framework component
4. Develop the application

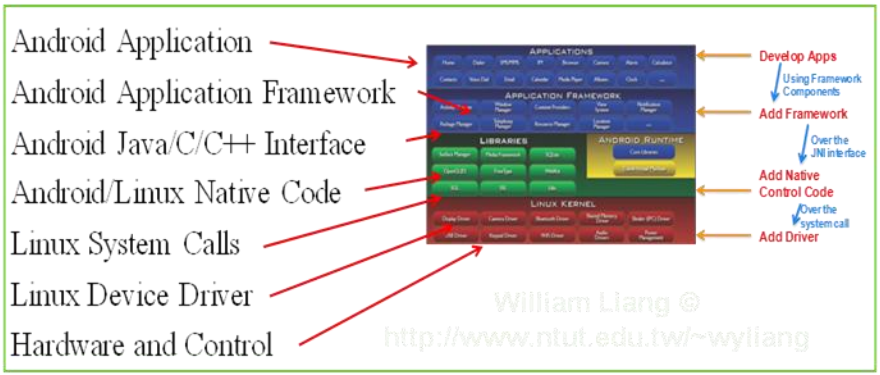
Example

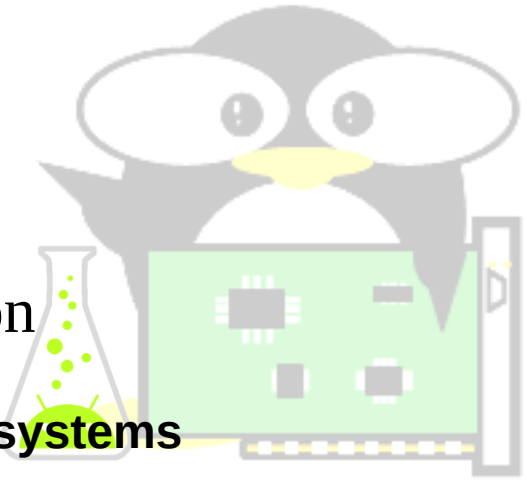


Slides from "Android Anatomy and Physiology," Patrick Brady ©

Important Subjects for the Core Architecture

- Android task and process
- Activity and service
- The binder IPC
- Thread operations
- Concurrency control and thread safety
- Android framework components
- User-space device control
- Android Java native interface for C/C++
- Hardware abstraction layer
- Linux device driver and device manipulation





Think: Android boot flow, system services, and subsystems

Activity -- the Primary Framework Component

- Displays a user interface component and responds to system/user.
- When an application has a user interface, it contains one or more “Activities”.
- Each “Activity” can be invoked by the same or other apps.
- The new Java class must extend the framework “Activity” class.
- Created “Activity” must be defined into the application’s Manifest.xml.
- An existing “Activity” can be replaced with a new one that fulfill the same contract (intent).

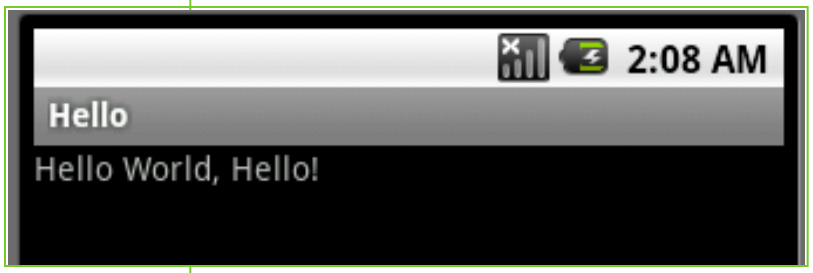
Source from “Deep inside Android,” Gilles Printemps, Esmertec ©

Think: activity stack and process

UI in XML

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />

</LinearLayout>
```



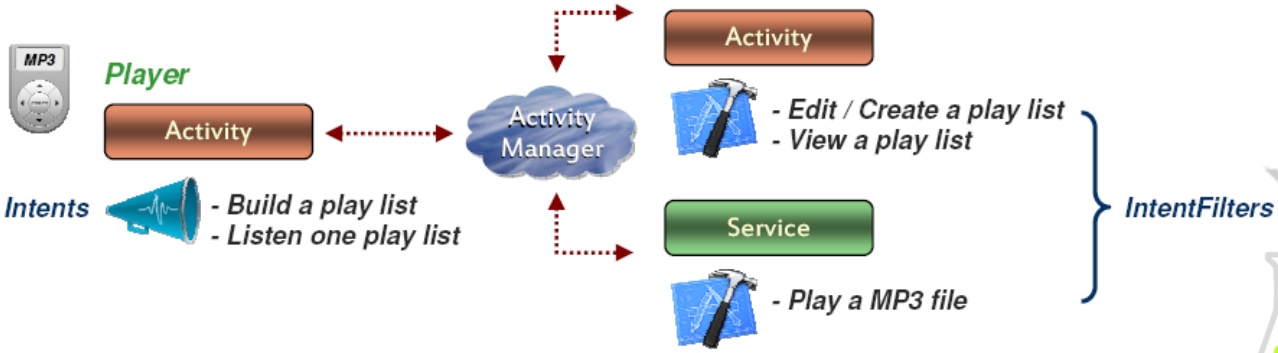

```
package hello.test;

import android.app.Activity;

public class Hello extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        /* pass resource id to attach to screen */
        setContentView(R.layout.main);
    }
}
```

Think: how UI XML gets deployed

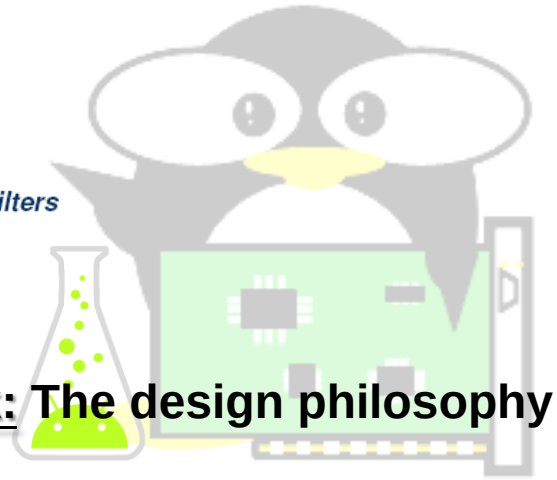
Source from http://developer.android.com/

Intents and Intent Filters

- Provide a **late runtime (asynchronous) binding** between the code in different applications (for activities, services, and broadcast receivers)
- **Intents**: Simple message objects that represent an intention to do something
- **Intent Filters**: A declaration of capacity and interest in offering assistance to those in need

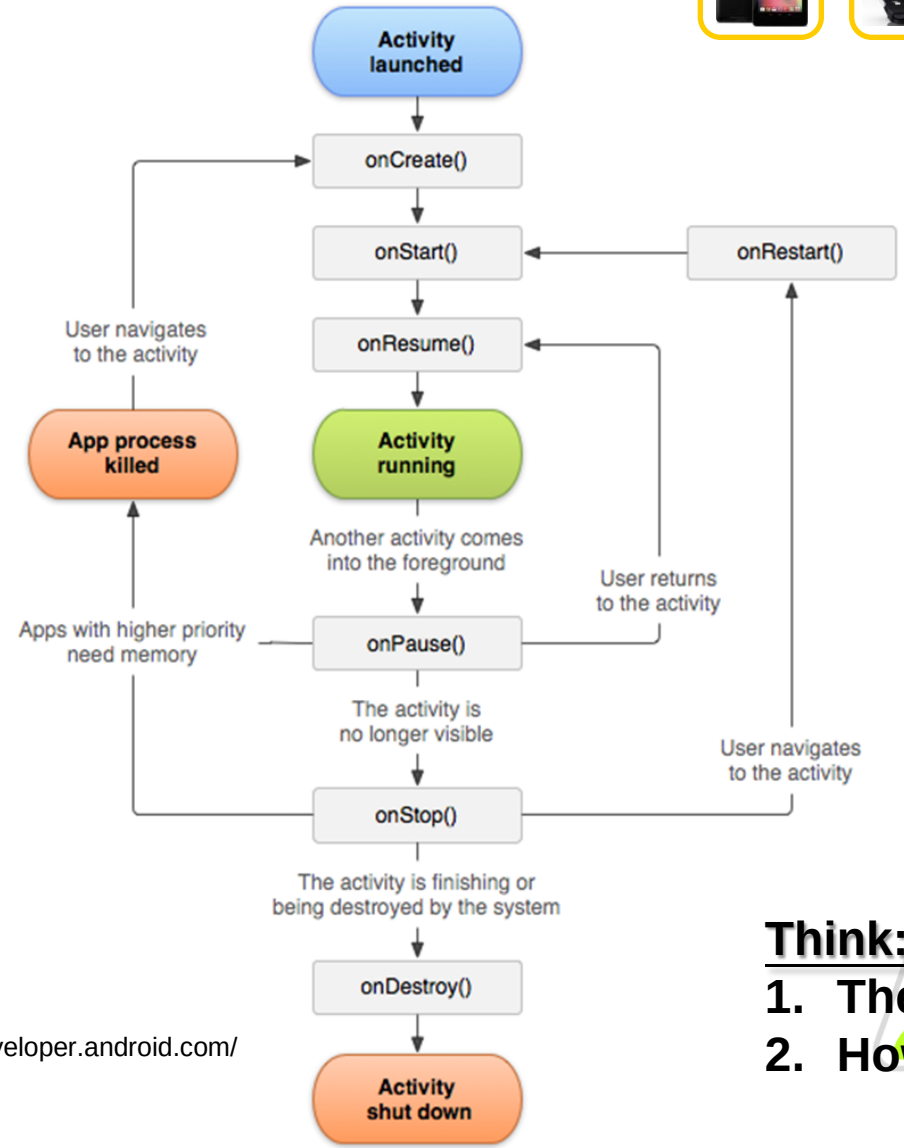


Slide from “Deep inside Android,” Gilles Printemps, Esmertec ©



Think: The design philosophy

Life Cycle of an Activity



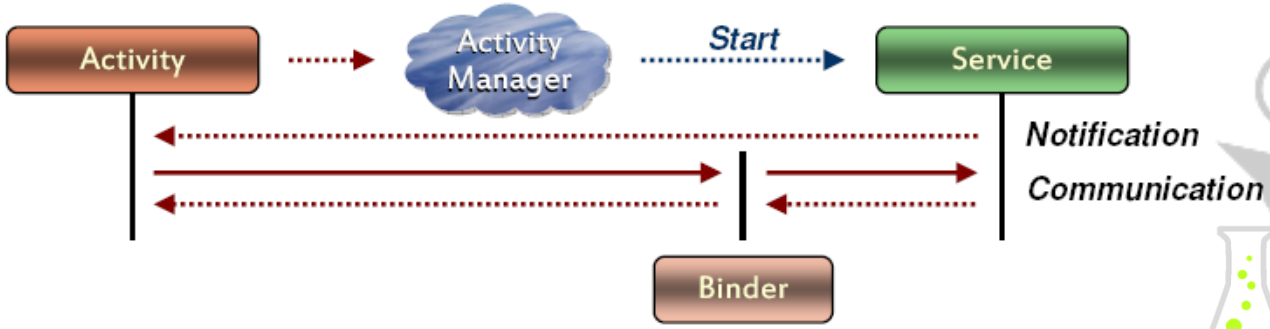
Source from http://developer.android.com/

Think:

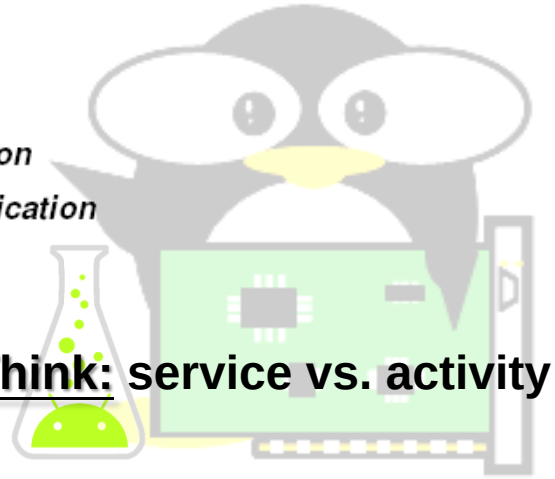
1. **The design philosophy**
2. **How to deal with data loss?**

The Service Framework Component

- Similar to activities, but without UI
- For long-running background tasks
- Extended from the Service class
- It's possible to connect to (bind to) an ongoing service.
- The connected service can be communicated through the Binder interface.

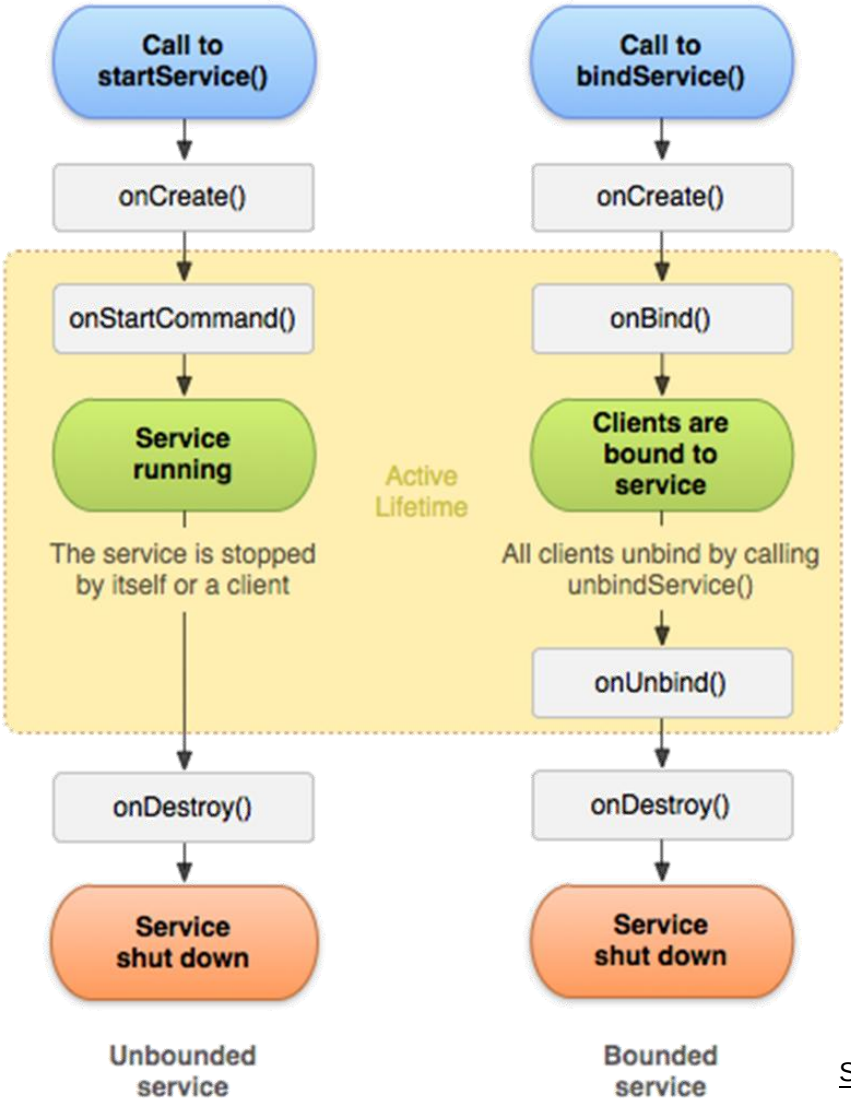


Source from “Deep inside Android,” Gilles Printemps, Esmertec ©



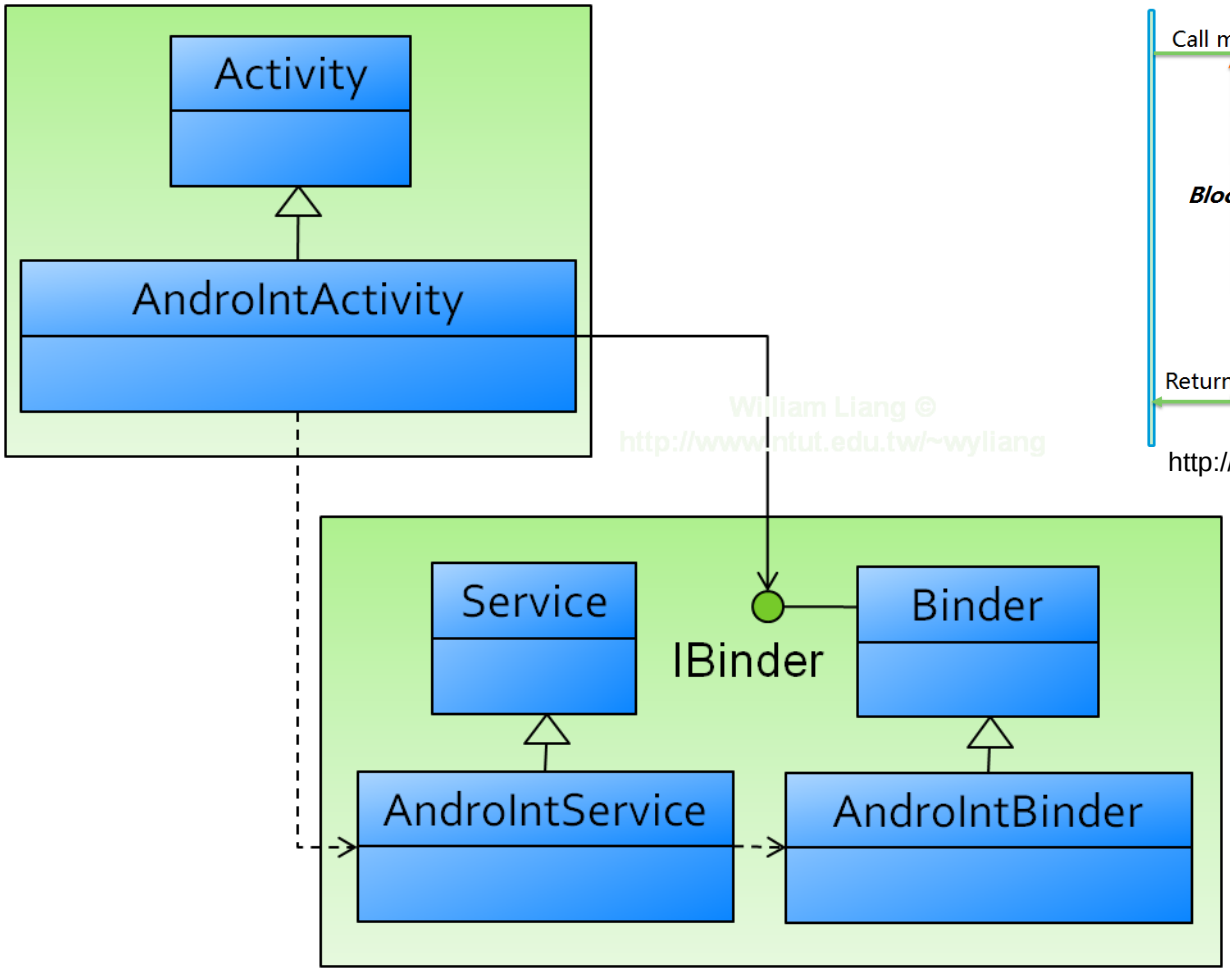
Think: service vs. activity

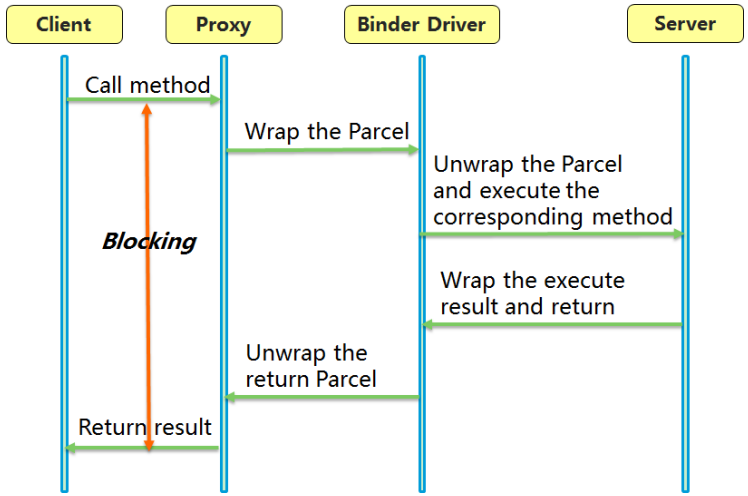
Service Lifecycle



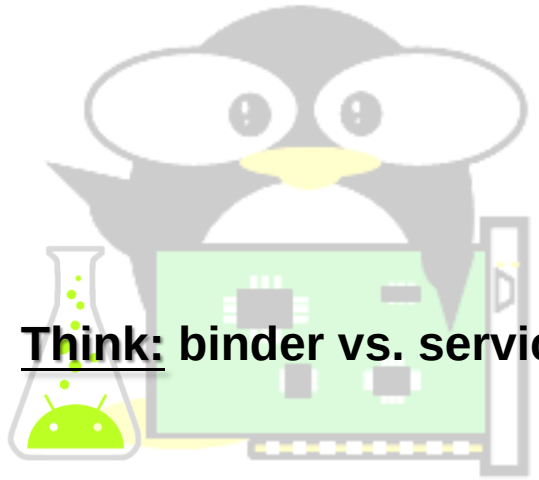
Source from http://developer.android.com/

Example





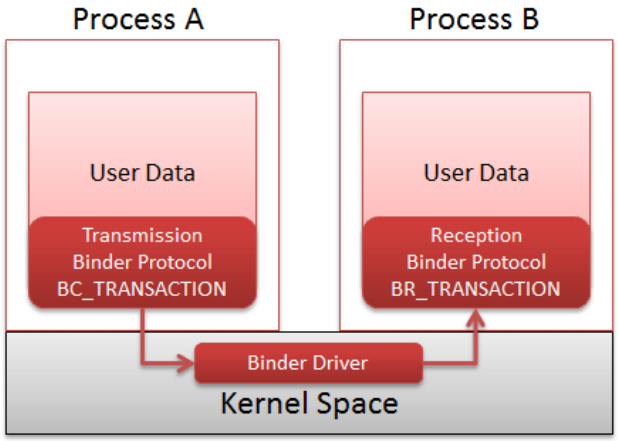
http://www.cnblogs.com/yingcare/p/3746751.html



Think: binder vs. service

Binder IPC

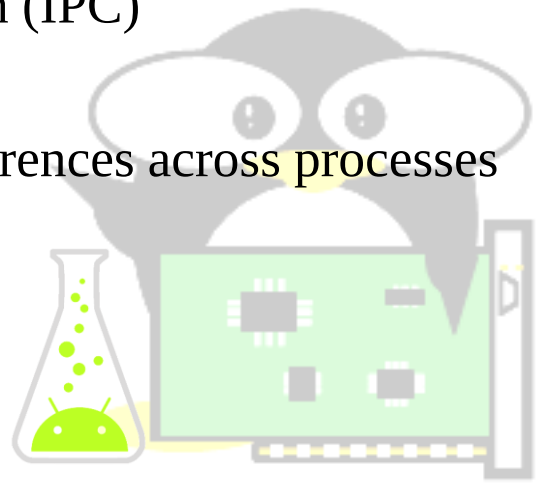
- Problems to solve
  - Applications and Services may run in separate processes but must communicate and share data.
  - IPC can introduce significant processing overhead and security holes.



https://kushinbunte.wordpress.com/2015/01/29/binder-communication-mechanism-of-android-processes/

- Solution
  - Driver to facilitate inter-process communication (IPC)
  - High performance through shared memory
  - Reference counting, and mapping of object references across processes
  - Synchronous calls between processes

Think: IPC overheads



Source from "Android Anatomy and Physiology," Patrick Brady ©

AIDL – a Wrapper of the Binder IPC

- AIDL: Android Interface Definition Language
  - Used to generate code in a remote procedure call (RPC) form that marshals the Binder IPC communication details.

```
private ServiceConnection mConnection = new ServiceConnection() {
    public void onServiceConnected(ComponentName className, IBinder ibinder) {
        mAddService = IAddService.Stub.asInterface((IBinder) ibinder);
    }

    public void onServiceDiscon
        mAddService = null;
    }
};

private OnClickListener mAddListener = new OnClickListener() {
    public void onClick(View view) {
        int a = Integer.parseInt(mEditText1.getText().toString());
        int b = Integer.parseInt(mEditText2.getText().toString());
        int result = 0;

        try {
            result = mAddService.add(a, b);
        } catch (RemoteException e) {
            e.printStackTrace();
        }

        mButton.setText(String.valueOf("A+B=" + result));
    }
};
```

```
interface IAddService {
    int add(in int a, in int b);
}
```

```
lic class AndroIntService extends Service {
 private IBinder mBinder = null;

 private static final String LOG_TAG = "AdroIntService";

 @Override
 public void onCreate() {
     mBinder = new AndroIntBinder();
     Log.i(LOG_TAG, "Service created, binder interface init
 }

 public int onStartCommand(Intent intent, int flags, int st

 @Override
 public IBinder onBind(Intent arg0) {
     Log.i(LOG_TAG, "Service bind requestd");
     return mBinder;
 }

 public void onDestroy() {

 private class AndroIntBinder extends IAddService.Stub {
     public int add(int a, int b) throws RemoteExceptin {
         return a+b;
     }
```

Think: how it works?

Thread Issues for Android

- The Andoid UI toolkit is not thread-safe.
- Do not manipulate your UI from a worker thread
- Two rules to Android's single UI thread model:
  - Do not access the UI toolkit from outside the UI thread
  - Do not block the UI thread

Problematic Example

```
public void onClick(View v) {
    new Thread(new Runnable() {
        public void run() {
            Bitmap b = loadImageFromNetwork("http://example.com/image.png");
            mImageView.setImageBitmap(b);
        }
    }).start();
}
```

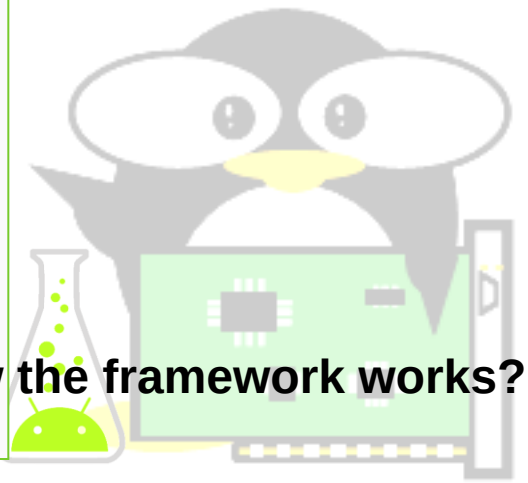
Think:

1. **How to solve the problem**
2. **The ANR issue**

The Message Queue

- Message Queue
- Looper
- Handler

```
class LooperThread extends Thread {
    public Handler mHandler;

    public void run() {
        Looper.prepare();

        mHandler = new Handler() {
            public void handleMessage(Message msg) {
                // process incoming messages here
            }
        };

        Looper.loop();
    }
}
```



Think: how the framework works?

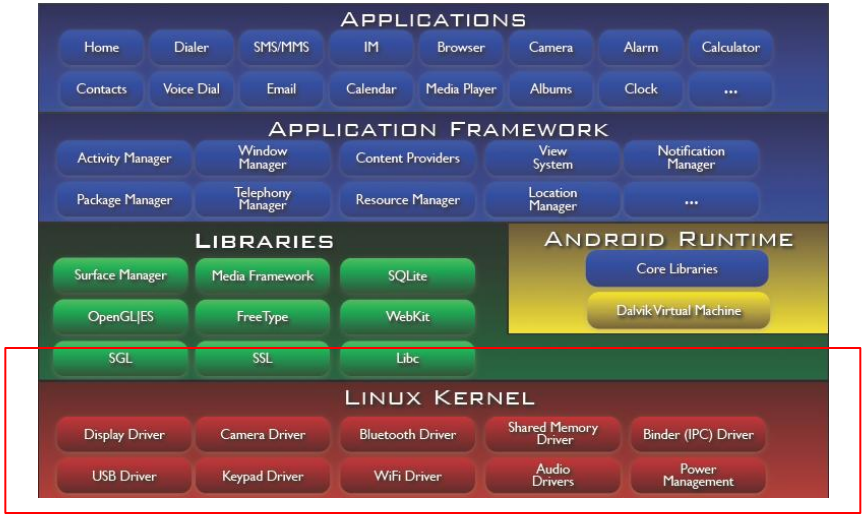
Process Lifecycle

- In Android, the system needs to remove old processes when memory runs low.
- To determine which processes to keep and which to kill
- There are five levels, listed in order of importance:
  - Foreground process
  - Visible process
  - Service process
  - Background process
  - Empty process

Think: when it changes and how it affects?

The Linux Kernel

- Android relies on Linux kernel 2.6 and above for core system services, e.g. memory management, process management, network stack, and driver model, security, etc.
- The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.



Features that Android added in Kernel

- **Binder**: Binder driver provides high performance inter-process communication (IPC) through shared memory.
- **Ashmem**: The Android shared memory driver, which create a memory region to be shared between processes.
- **Pmem** and **ION**: Used to allocate and manage a physically contiguous memory for user space driver.
- **Power**: The android power management driver
- **Alarm**: It's a driver which provides a real time clock.

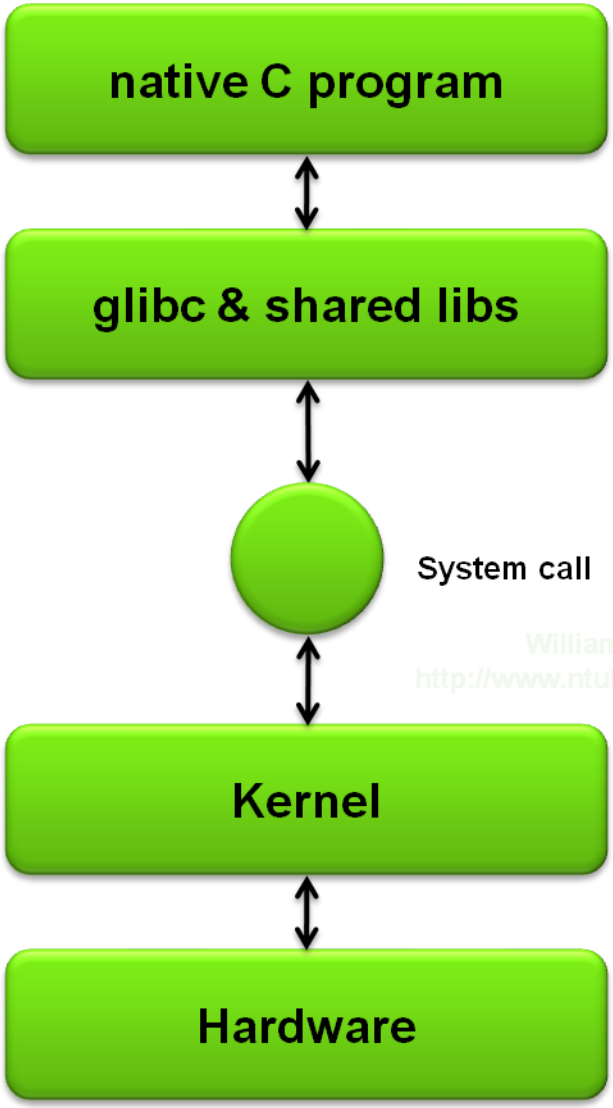
Think: **Linux mainline kernel vs. Android kernel**

User-space Device Control

- Typical device drivers exist in the Kernel
- However, some of the device control logic can be performed from the user space.
- Two methods to do so
  - Communicate with the kernel-space drivers through the device files or other system interfaces such as sysfs
  - Direct hardware access through memory mapped I/O, by *mmap*

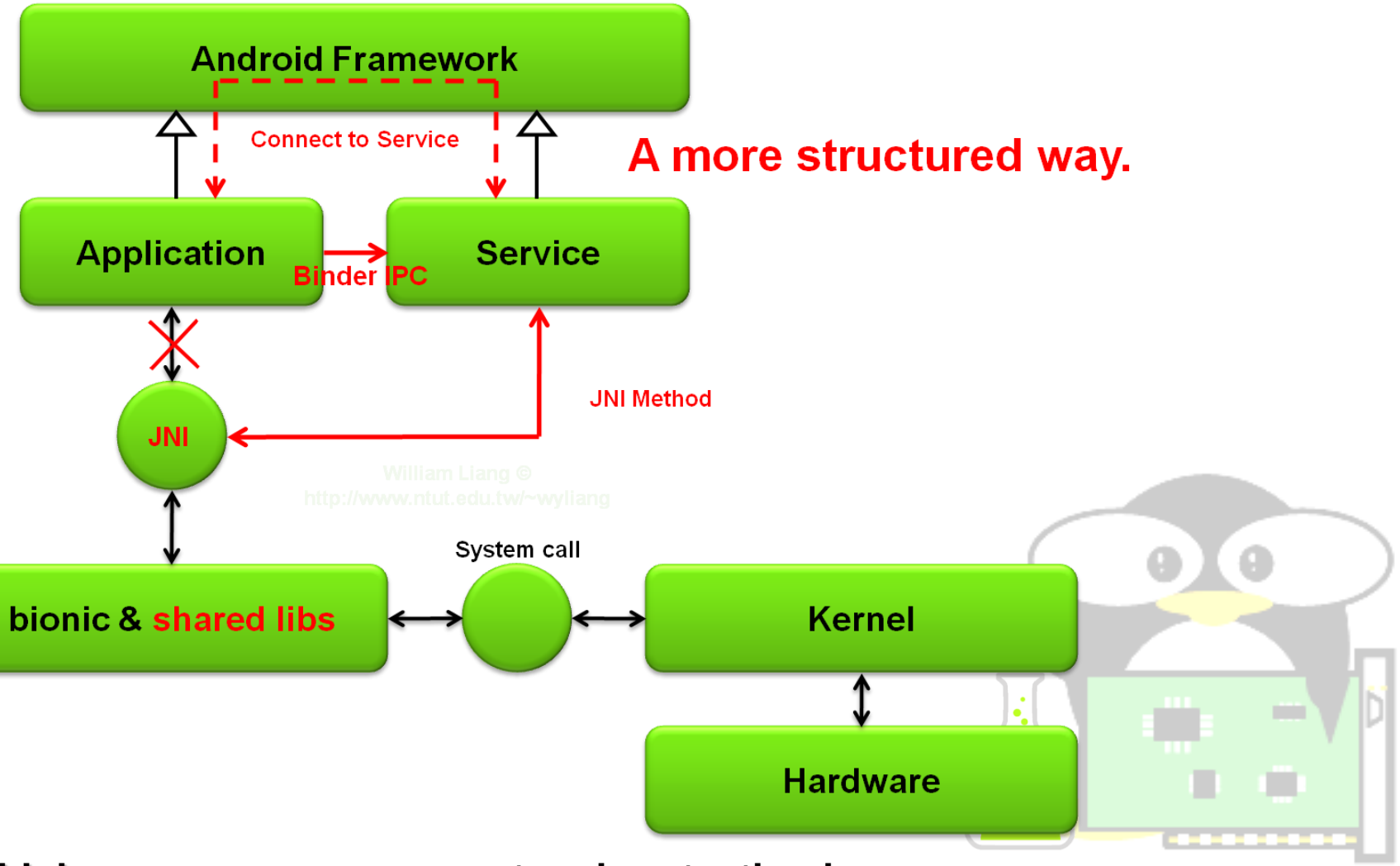
Think: performance issues for the above methods

The Traditional Linux Device Control Method




```c
#include <stdio.h>
#include <fcntl.h>

#define DEVFILE "/dev/androint"

int main() {
        int fd;
        int in[2] = {10, 20};
        int out;

        printf("AndroInt virtual adder driver test:\n");
        printf("Input A: ");
        scanf("%d", in);
        printf("Input B: ");
        scanf("%d", in+1);
        printf("Input: %d %d\n", in[0], in[1]);

        fd = open(DEVFILE, O_RDWR);
        write(fd, in, sizeof(in));
        read(fd, &out, sizeof(out));
        close(fd);

        printf("Output: %d\n", out);

        return 0;
}
```

The Formal Android Device Control Model



Think: resource management and protection issues

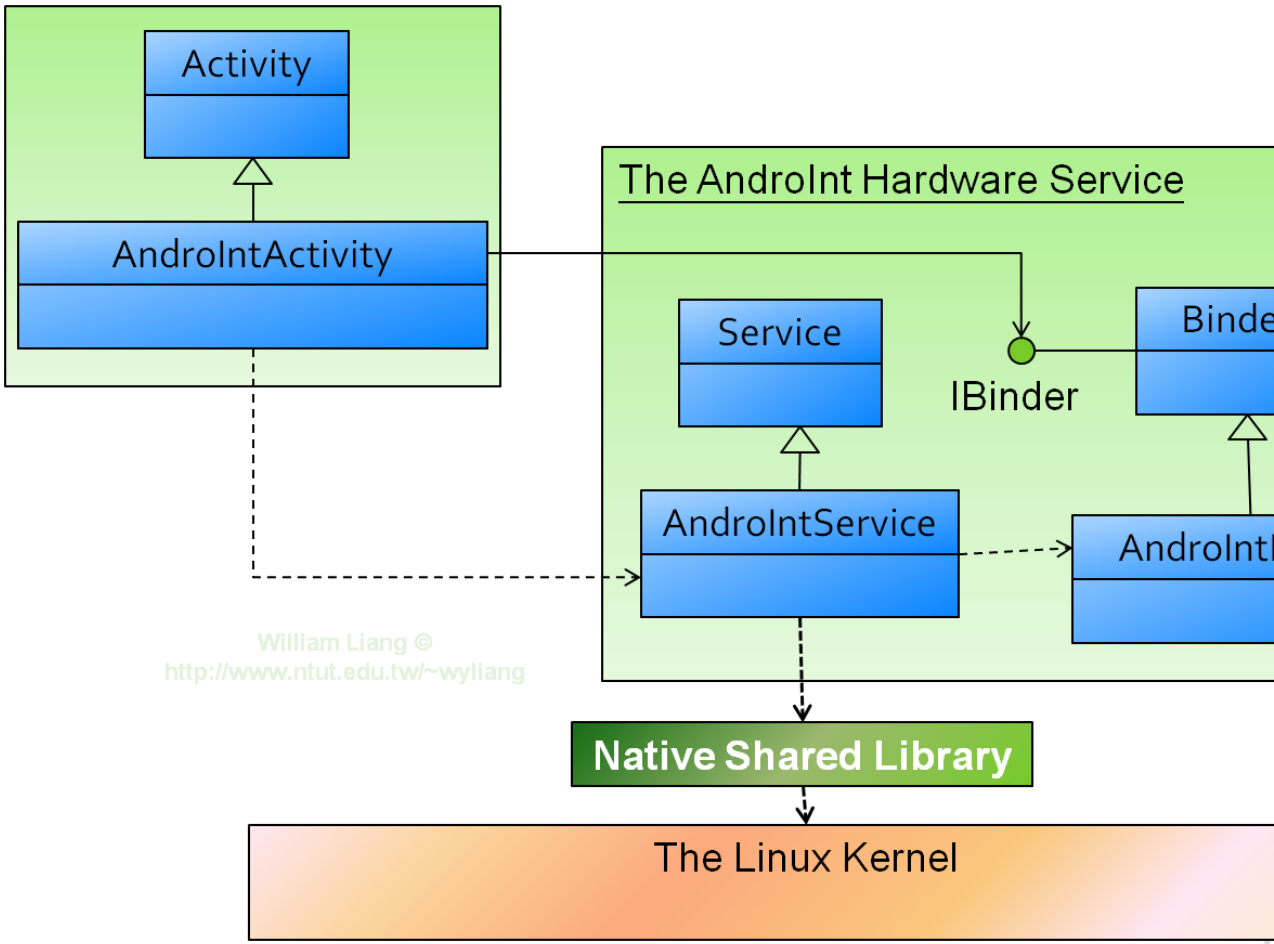
Android System Services

- Core Services
  - Activity Manager
  - Package Manager
  - Window Manager
  - Resource Manager
  - Content Providers
  - View System
- Hardware Services
  - Telephony Service
  - Location Service
  - Bluetooth Service
  - WiFi Service
  - USB Service
  - Sensor Service

Source from “Deep inside Android,” Gilles Printemps, Esmertec ©

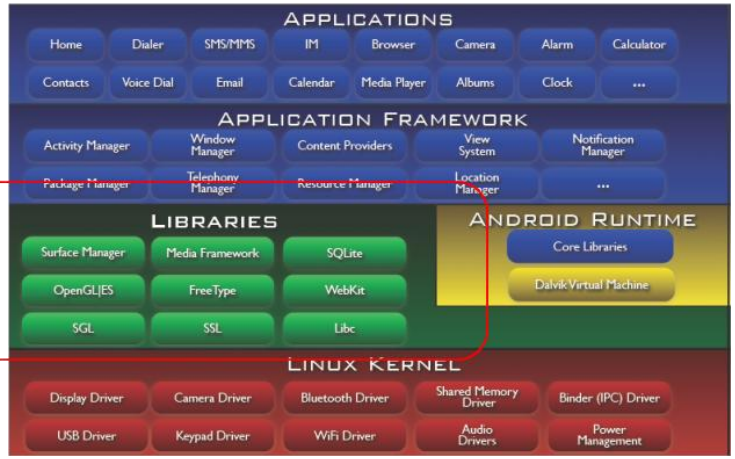
Think: compare with the concept used in micro-kernel

Hardware Service Example



Android Native Libraries

- C Library
  - Android includes a set of C/C++ libraries used by various components of the Android system.
  - Bionic C Library: A BSD-derived implementation of the standard C library, tuned for embedded Linux-based devices
- Example Media Libraries
  - SQLite
  - Webview
  - SGL
  - FreeType
  - 3D libraries
  - Many more …



Think:

1. **The Dynamic Shared Library**
2. **Performance Issue**

Cross the Language: Java, C/C+, and JNI

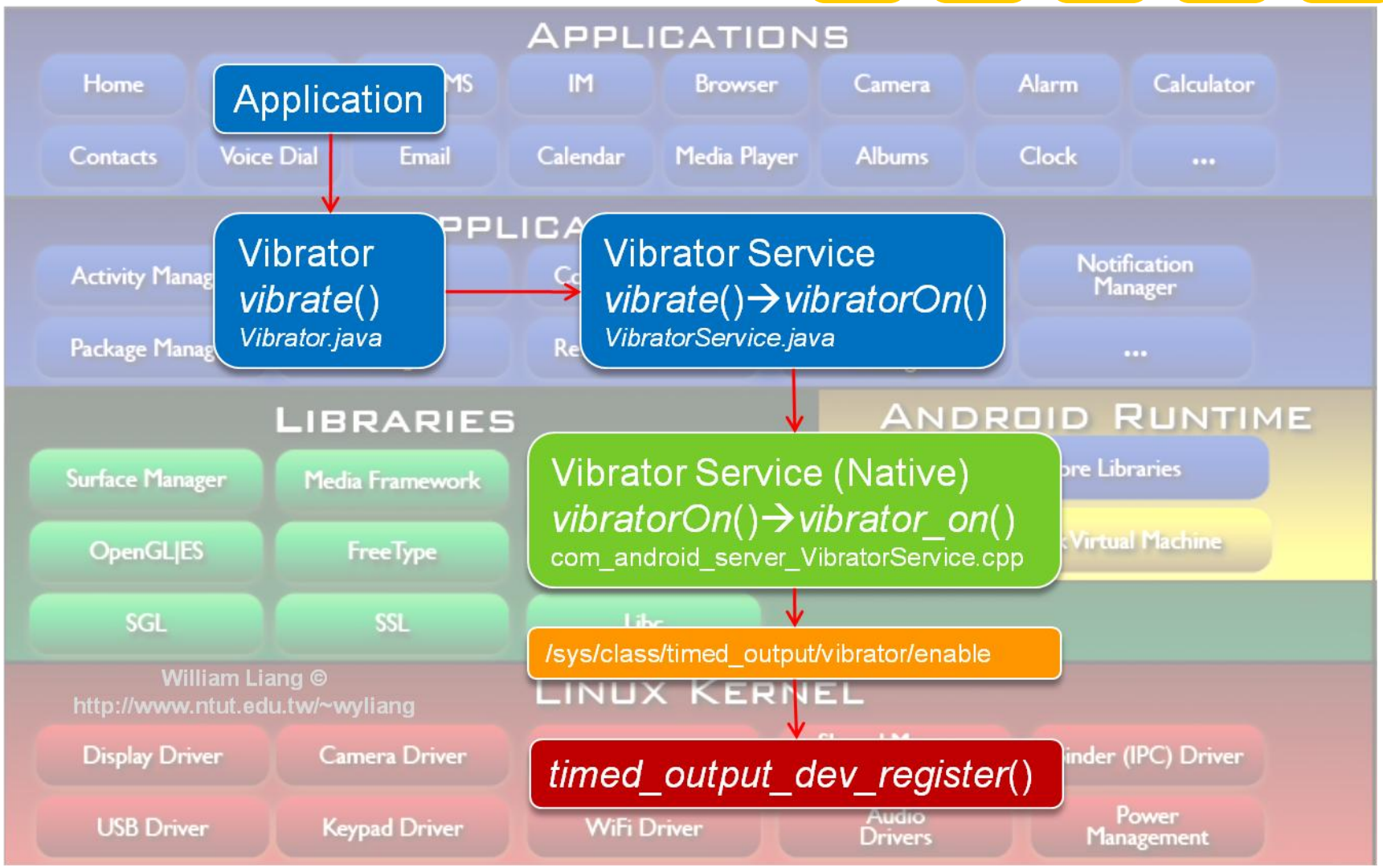
- JNI: Java Native Interface
- Standard interface between Java and C/C++

```
class HelloWorld {
    public native void helloworld();

    static {
        System.loadLibrary("hello");
    }

    public static void main(String[] args) {
        new HelloWorld().helloworld();
    }
}
```

Think:

1. **Bi-directional invocation**
2. **Dalvik VM, JIT, and ART**

A Simple Legacy System Service: Vibrator Service

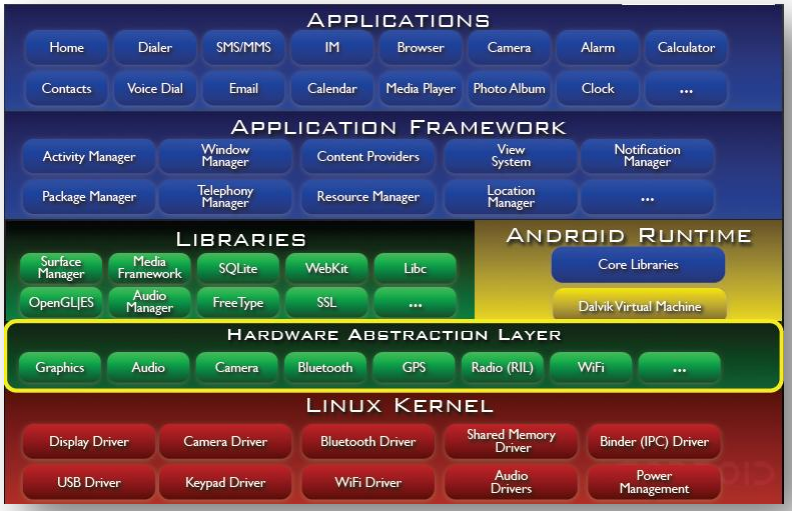


The Android Hardware Abstraction Layer

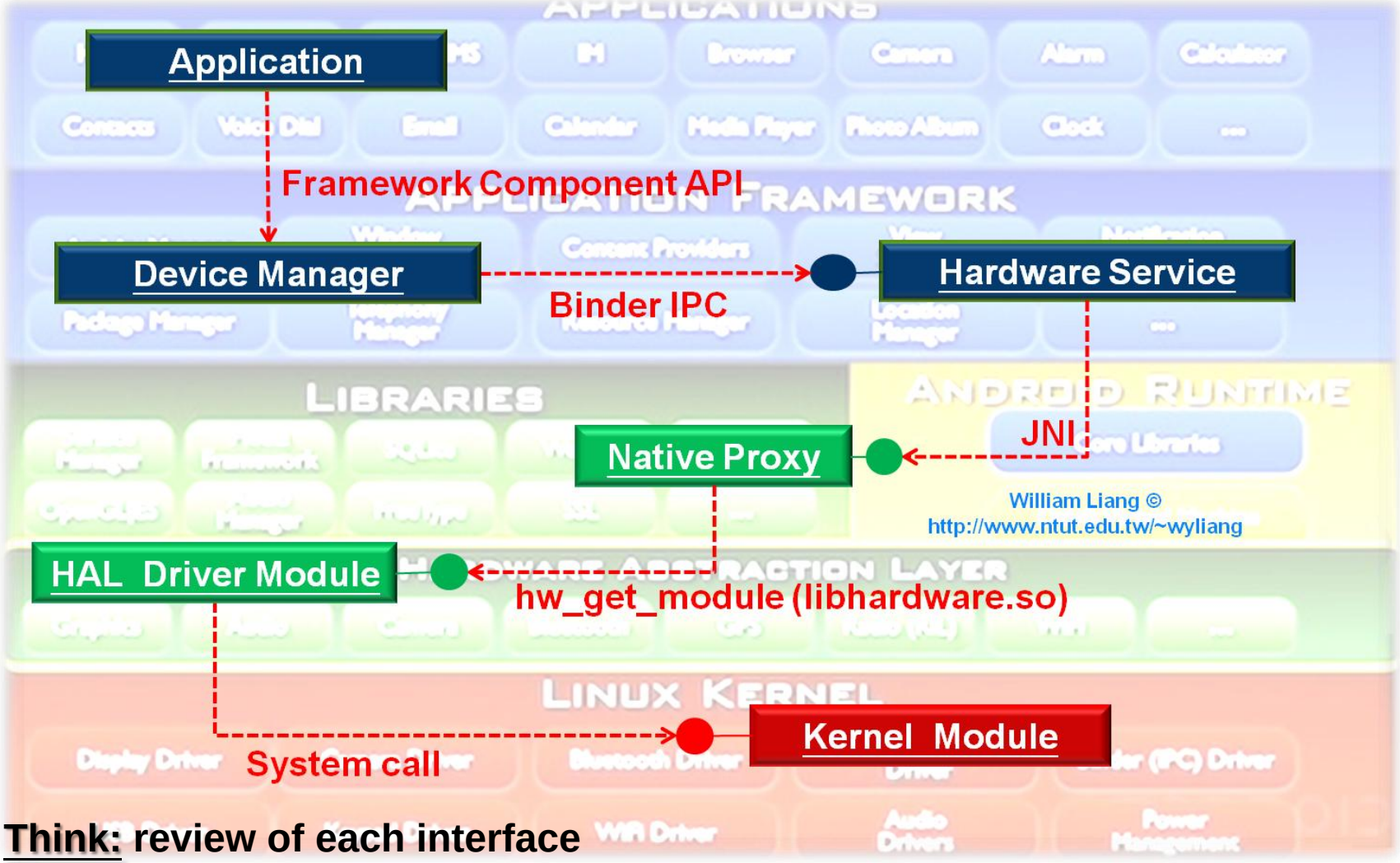
- User space C/C++ library layer
- Defines the interface that Android requires hardware “drivers” to implement
- Separates the Android platform logic from the hardware interface

Source from "Android Anatomy and Physiology," Patrick Brady ©

- Kernel drivers are GPL which may expose the proprietary IP.

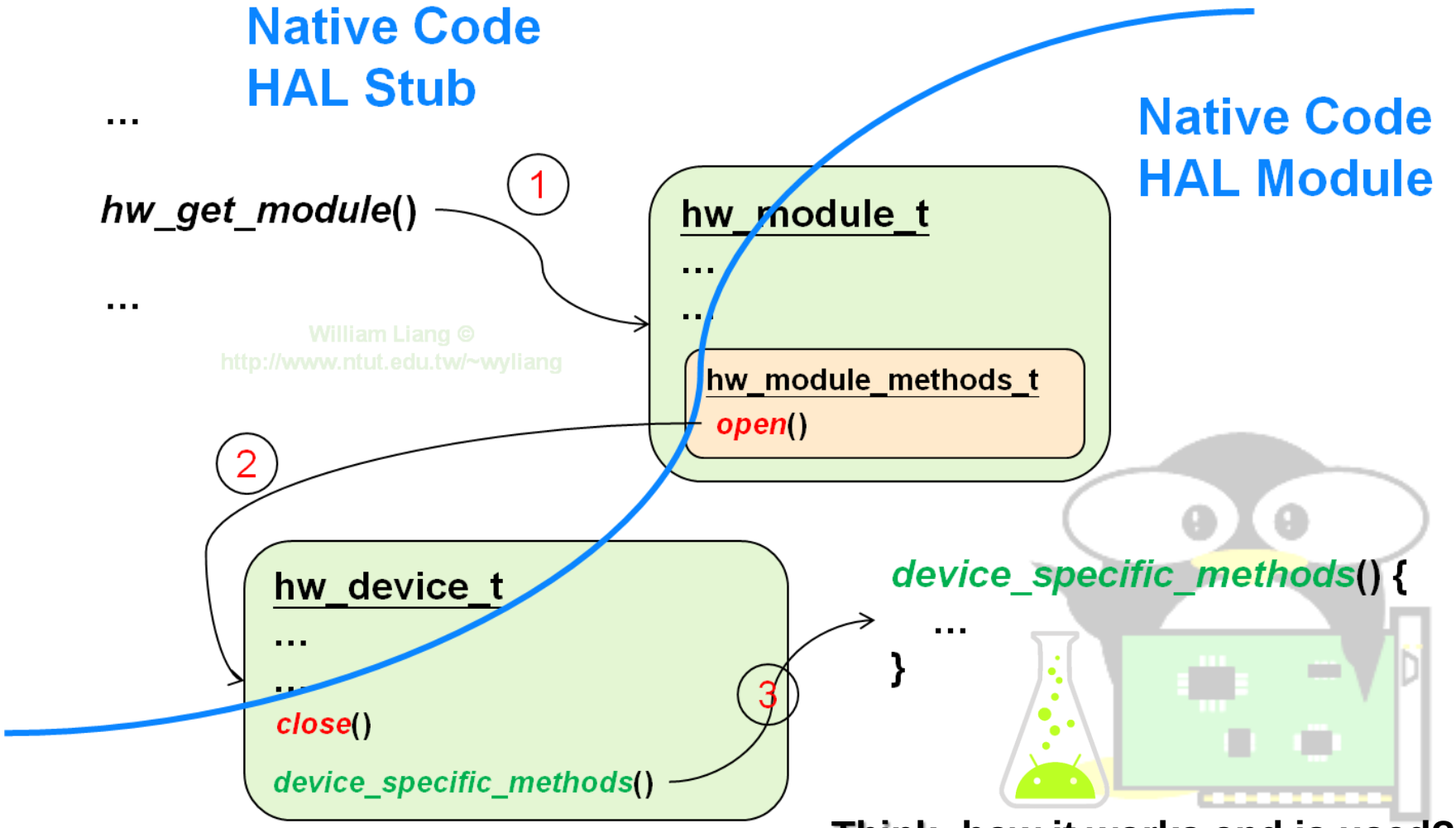


Android Hardware Control Flow Diagram



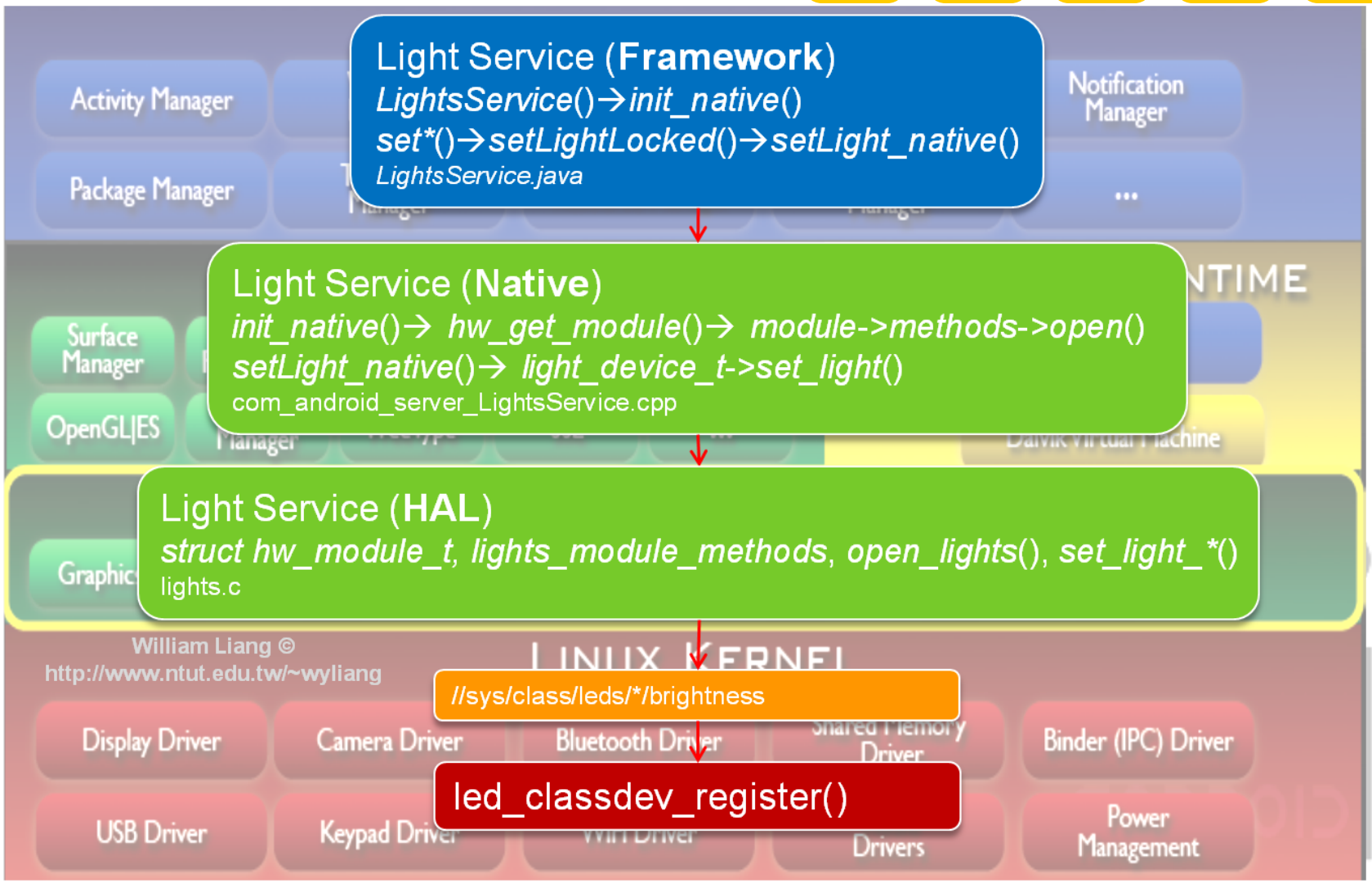
Think: review of each interface

Flow for Retrieving HAL Module and Methods



Think: how it works and is used?

HAL Example: Lights Service



Q & A

william.wyliang@gmail.com
http://www.ntut.edu.tw/~wyliang
http://www.facebook.com/william.wyliang