CS 318 Principles of Operating Systems

Fall 2021

Lecture 21: Mobile Operating System

Prof. Ryan Huang

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Administrivia

Next week Thanksgiving break

- No class
- Assignments
  - food, lots of it
  - sleep, lots of it
  - warm clothes, winter is coming
  - **Stay safe**

Preview

Next two lectures again preview advanced systems topics

- Each topic has enough depth to be covered in an entire course by itself
- We will only cover basic concepts

Today: mobile operating system

- History of mobile device and OS
- Mobile OS vs. traditional OS
- How does Android OS work?

Quizzes

When did the 1st-gen iPhone come out?

A: 2000-2005

B: 2005-2010

C: 2010-2015

D: 2015-2020

Which mobile OS has the most market share?

A: Android

B: iOS

C: Windows

D: Blackberry

History of Mobile OS (1)

Early “smart” devices are PDAs (touchscreen, Internet)

Symbian, first modern mobile OS

- released in 2000
- run in Ericsson R380, the first ‘smartphone’ (mobile phone + PDA)
- only support proprietary programs

History of Mobile OS (2)

Many smartphone and mobile OSes followed up

- Kyocera 6035 running Palm OS (2001)
- Windows CE (2002)
- Blackberry (2002)
  - was a prominent vendor
  - known for secure communications
- Moto Q (2005)
- Nokia N70 (2005)
  - 2-megapixel camera, bluetooth
  - 32 MB memory
  - Symbian OS
  - Java games

One More Thing…

Introduction of iPhone (2007)

- revolutionize the smartphone industry
- 4GB flash memory, 128 MB DRAM, multi-touch interface
- runs iOS, initially only proprietary apps
- App Store opened in 2008, allow third party apps

Android – An Unexpected Rival of iPhone

Android Inc. founded by Andy Rubin et al. in 2003

- original goal is to develop an OS for digital camera
- shift focus on Android as a mobile OS

The startup had a rough time [story]

- run out of cash, landlord threatens to kick them out
- later bought by Google
- no carrier wants to support it except for T-Mobile
- while preparing public launch of Android, iPhone was released

Android 1.0 released in 2008 (HTC G1)

Today: ~88% of mobile OS market

- iOS ~11%

Why Are Mobile OSes Interesting?

- **They are running in every mobile device as an essential part of people's daily life, even for non-technical users**
  - In many developing countries, the only computing device one has is a phone
- **Mobile OSes and traditional OSes share the same core abstractions but also have many unique designs**
  - Comparing and contrasting helps you understand the whole OS design space
- **It will make you a more efficient mobile user and developer**

Design Considerations for Mobile OS

Resources are very constrained

- Limited memory
- Limited storage
- Limited battery life
- Limited processing power
- Limited network bandwidth
- Limited size

User perception are important

- Latency ≫ throughput
    - Users will be frustrated if an app takes several seconds to launch

Environment are frequently changing

- The whole point about being mobile
- Cellular signals from strong to weak and then back to strong

Process Management in Mobile OS (1)

In desktop/server: an application = a process

Not true in mobile OSes

- When you see an app present to you, doesn't mean an actual process is running
- Multiple apps might share processes
- An app might make use of multiple processes
- When you "close" an app, the process might be still running
  - **Why?**
  - "*all applications are running all of the time*"

Different user-application interaction patterns

- Check Facebook for 1 min, switch to Reminder for 10s, Check Facebook again
- Server: launch a job, waits for result

Process Management in Mobile OS (2)

Multitasking is a luxury in mobile OS

- Early versions of iOS don't allow multi-tasking
  - Not because the CPU doesn't support it, but because of battery life and limited memory
- Only one app runs in the foreground, all other user apps are suspended
- OS's tasks are multi-tasked because they are assumed to be well-behaving
- Starting with iOS 4, the OS APIs allow multi-tasking in apps
  - But only available for a limited number of app types

Different philosophies among mobile OSes

- Android more liberal: apps are allowed to run in background
  - Define Service class, e.g., to periodically fetch tweets
  - When system runs low in memory, kill an app

Memory Management in Mobile OS

Most desktop and server OSes today support swap space

- Allows virtual memory to grow beyond physical memory size
- When physical memory is full utilized, evict some pages to disk

Smartphones use flash memory rather than hard disk

- Capacity is very constrained: 16 GB vs. 512 GB
- Limited number of writes in its lifetime
- Poor throughput between main memory and flash memory

Mobile OSes typically don't support swapping!

- iOS ***asks*** applications to voluntarily relinquish allocated memory
- Android will terminate an app when free memory is running low

App developers must be very careful about memory usage

Storage in Mobile OS

App privacy and security is hugely important in mobile device

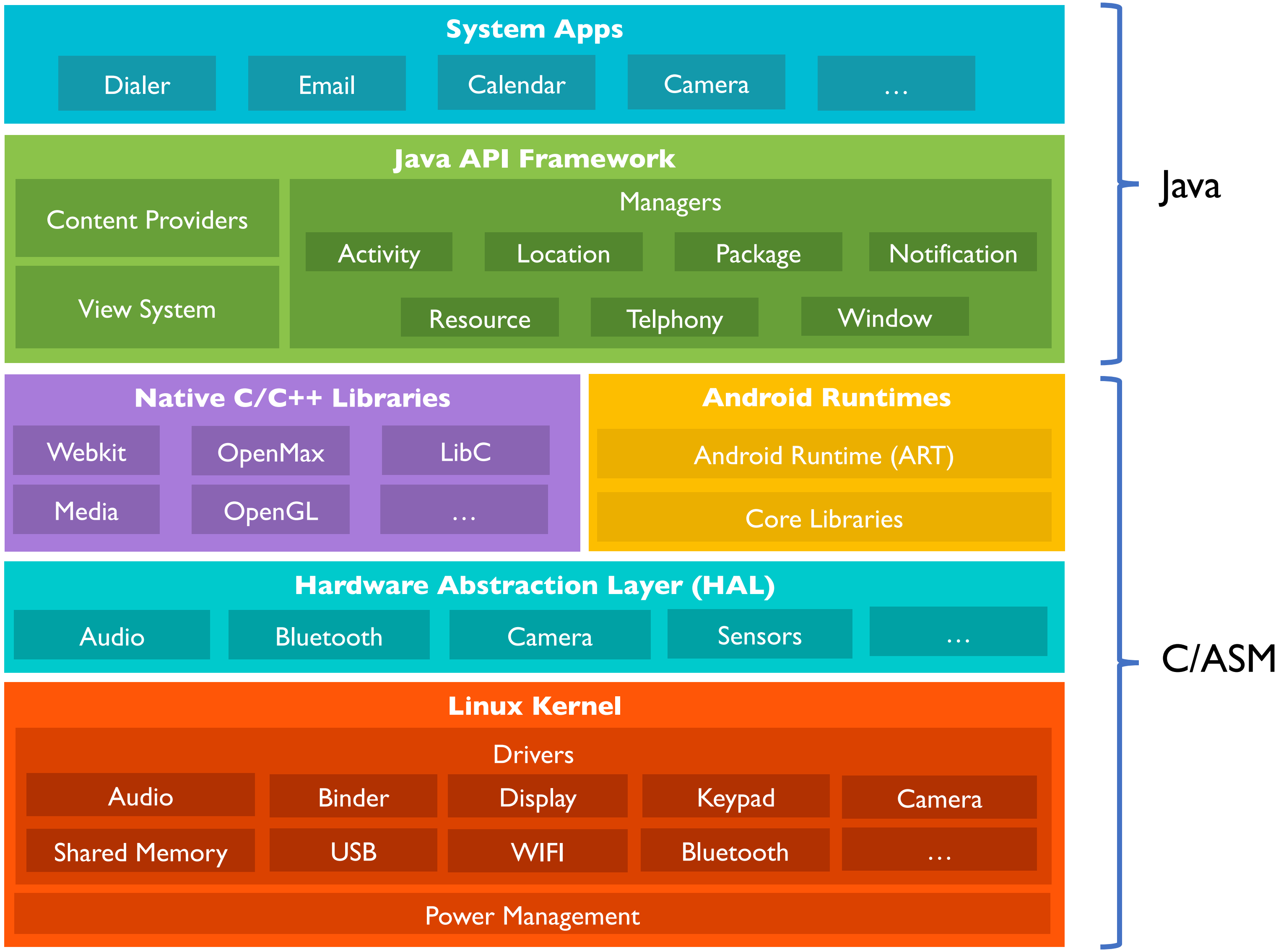
- Each app has its own private directory that other app can't access
- Only shared storage is external storage
  - /sdcard/

High-level abstractions

- Files
- Database (SQLite)
- Preferences (key-value pairs)

```
ryan@orderlab:~$adb shell
shell@shamu:/ $ cd /data/app
shell@shamu:/data/app $ ls
opendir failed, Permission denied
255|shell@shamu:/data/app $ su
root@shamu:/data/app # ls
com.android.chrome-2
com.android.vending-2
com.facebook.katana-1
com.google.android.apps.docs.editors.docs-1
com.google.android.apps.maps-1
com.google.android.apps.messaging-1
com.google.android.gms-2
com.google.android.googlequicksearchbox-1
com.google.android.instantapps.supervisor-2
com.google.android.play.games-1
com.google.android.youtube-1
com.jumobile.manager.systemapp-1
com.ketchapp.stack-1
com.progames01.tanks.playtank-1
com.rovio.angrybirds-1
com.snapchat.android-1
edu.jhu.order.appstatstracker-1
```

Android OS Stack



Linux Kernel vs. Android Kernel

Linux kernel is the foundation of Android platform

New core code

- binder - interprocess communication mechanism
- ashmem - shared memory mechanism
- logger

Performance/power

- wakelock
- low-memory killer
- CPU frequency governor

and much more . . . [361 Android patches for the kernel]

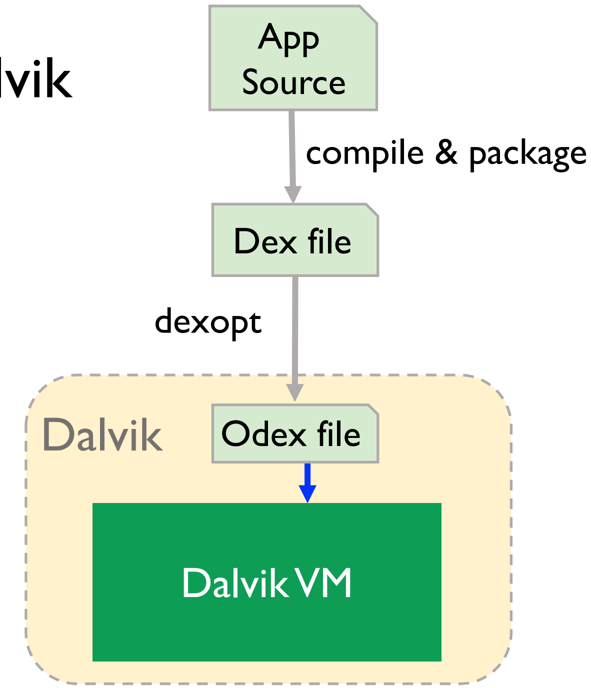
Android Runtime

What is a runtime?

- A component provides functionality necessary for the execution of a program
  - E.g., scheduling, resource management, stack behavior

Prior to Android 5.0, Dalvik is the runtime

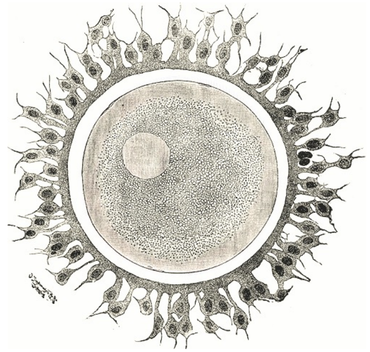
- Each Android app has its own process, runs its own instance of the Dalvik virtual machine (*process virtual machine*)
- The VM executes the Dalvik executable (.dex) format
- Register-based compared to stack-based of JVM



ART introduced in Android 5.0

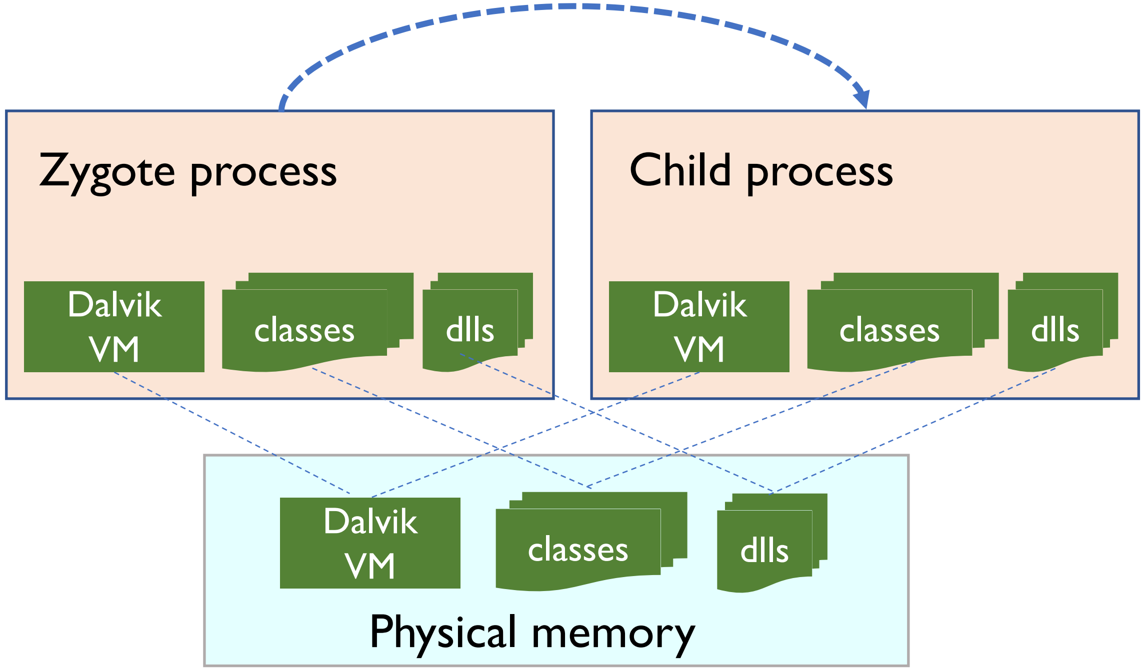
- Backward compatible for running Dex bytecode
- New feature: Ahead-of-time (AOT) compilation
- Improved garbage collection

Android Runtime - Zygote



All Android apps derive from a process called Zygote

- Zygote is started as part of the init process
- Preloads Java classes, resources, starts Dalvik VM
- Registers a Unix domain socket
- Waits for commands on the socket
- Forks off child processes that inherit the initial state of VMs



Uses Copy-on-Write

- Only when a process writes to a page will a page be allocated

Java API Framework

The main Android "OS" from app point of view

- Provide high-level services and environment to apps
- Interact with low-level libraries and Linux kernel

Example

- Activity Manager
  - Manages the lifecycle of apps
- Package Manager
  - Keeps track of apps installed
- Power Manager
  - Wakelock APIs to apps

Native C/C++ Libraries

Many core Android services are built from native code

- Require native libraries written in C/C++
- Performance benefit
- Some of them are exposed through the Java API framework as native APIs
  - E.g., Java OpenGL API

Technique: JNI – Java Native Interface

App developer can use Android NDK to include C/C++ code

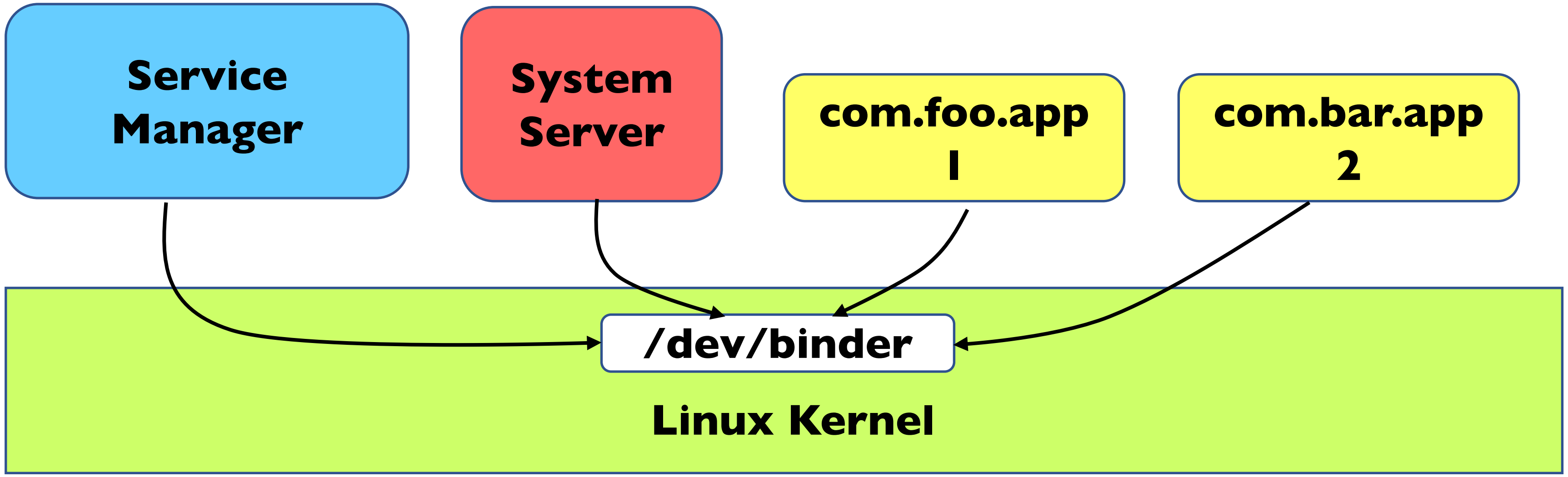
- Common in gaming apps

Android Binder IPC

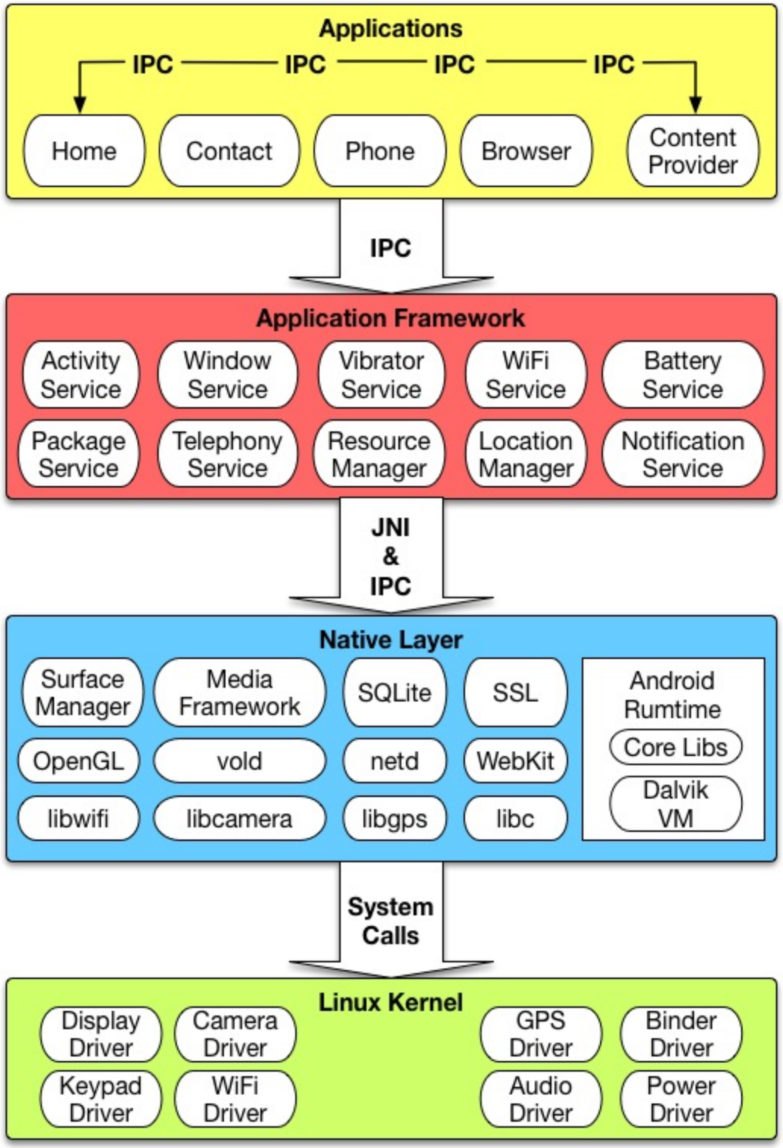
An essential component in Android for Inter-Process Communication (IPC)

- Allows communication among apps, between system services, and between app and system service

Data sent through "parcels" in "transactions"



IPC Is Pervasive in Android



How Is Binder Implemented: As RPC!

Developer defines methods and object interface in an .aidl file

```
package com.example.android; // IRemoteService.aidl

/** Example service interface */
interface IRemoteService {
    /** Request the process ID of this service, to do evil things with it. */
    int getPid();
    /** Pause the service for a while */
    void pause(long time);
}
```

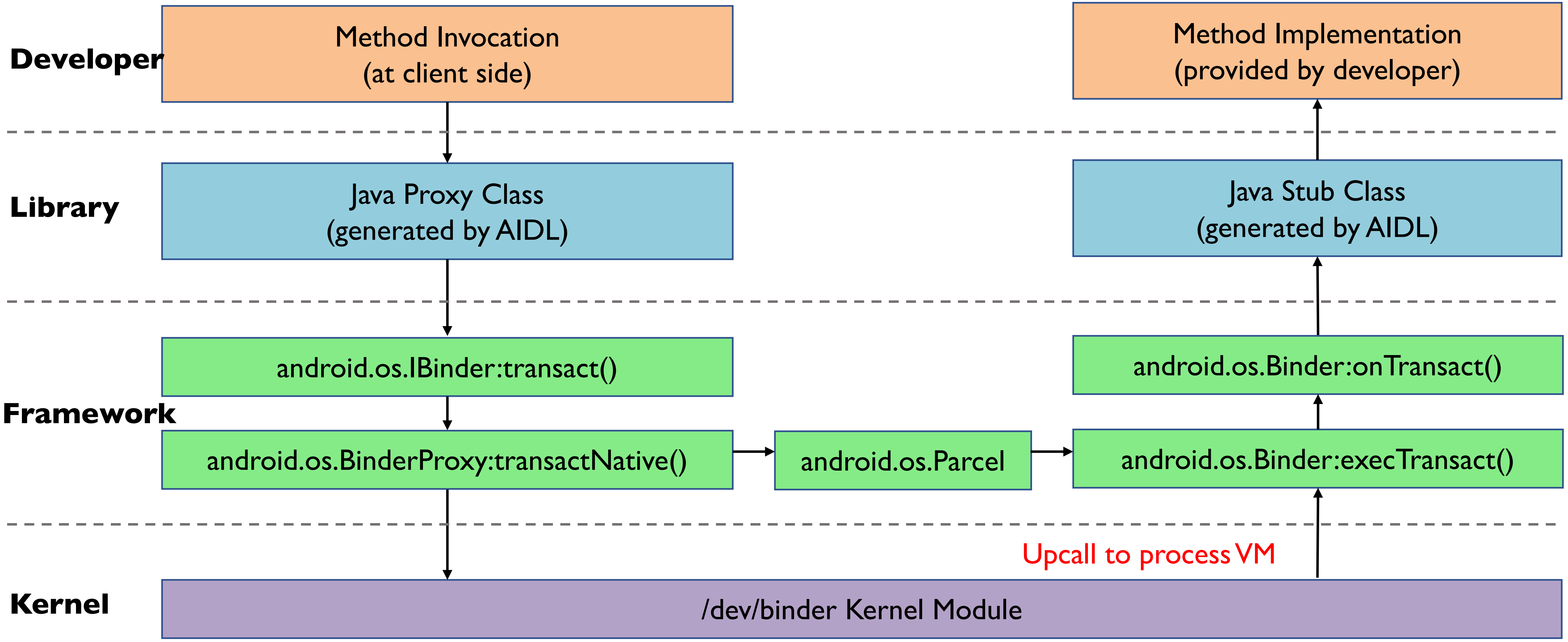
Android SDK generate a stub Java file for the .aidl file

- Developer implements the stub methods
- Expose the stub in a Service

Client copies the .aidl file to its source, Android SDK generates a stub (a.k.a proxy) for it as well

- Client invoke the RPC through the stub

Binder Information Flow



Some Other Interesting Topics in Mobile OS

Energy management

- ECOSystem: Managing Energy as a First Class Operating System Resource
- Drowsy Power Management
- A Case for Lease-Based, Utilitarian Resource Management on Mobile Devices

Dealing with misbehaving apps

- DefDroid: Towards a More Defensive Mobile OS Against Disruptive App Behavior
- eDoctor: Automatically Diagnosing Abnormal Battery Drain Issues on Smartphones

Security

- CLKSCREW: Exposing the Perils of Security-Oblivious Energy Management

Summary

Smartphone has become a ubiquitous computing device

- Long history but past decade is disruptive

Mobile OS is an interesting and challenging subject

- Constrained resources
- Different user interaction patterns
- Frequently changing environment
- Untrusted, immature third-party apps

Some unique design choices

- Application $\neq$ process
- Multitasking
- No swap space
- Private storage