Disclaimer: The contents in this document are only my personal opinions, do not reflect the opinions of my employer or anyone else.

Android overview

—— From a system design perspective

Xiao-Feng Li
xiaofeng.li@gmail.com

2012-09-21

Android overview

—— From a system design perspective

PART I

Design principles

Question 1

- Q: What do you expect from a mobile phone?

Answer To Q1

- Q: What do you expect from a mobile phone?
- A: (no correct answer, but reasonable answer)
  - Easy communications
  - Long battery life
  - Fast boot-up/shutdown
  - Smooth operations
  - Delicate industry design
  - Applications
  - … …

Could be a close environment

Question 2

- Q: What do you expect from a smart-phone?

Answer To Q2

- Q: What do you expect from a smart-phone?
- A: (no correct answer, but reasonable answer)
  - All of those features with a mobile phone, plus
  - PC features
    - Programming environment
    - Portal to Internet
    - And security risks
  - Gaming station features
    - Sensors, 3D
  - … …

Must be an open environment

Question 3

- Q: What build an open pleasant environment?

Answer To Q3

- Q: What build an open pleasant environment?
- A: (no correct answer, but reasonable answer)
  - Favorable APIs to developers
  - Choices of applications to users
  - Seamless access to Internet/Cloud services
  - Pleasant user experience

Maximize information flow

Question 4

- Q: What prevent an open environment?

Answer To Q4

- Q: What prevent an open environment?
- A: (no correct answer, but reasonable answer)
  - Openness vs. Security
  - User experience vs. Battery life
  - Cloud readiness vs. Local computation

Law of conservation of energy

ITJ paper: Mobile OS Architecture Trends, Xiao-Feng Li, et. al. 16(4), 2012

Question 5

- Q: What does Android do?
  - APIs
  - Applications
  - Internet/Cloud
  - Security
  - User experience
  - Battery life

Answer To Q5

- Q: What does Android do?
- A: (no correct answer, but reasonable answer)
  - APIs: SDK, NDK, RS, HTML/JS
  - Applications: Google Play, web apps
  - Internet/Cloud: Chrome/webview/HTML5
  - Security: Java, Permission, Linux security, sig
  - User experience: touch, mem, perf
  - Battery life: Linux PM, wake lock

Feature list does not tell the truth

Question 6

- Q: How does Android put them together?

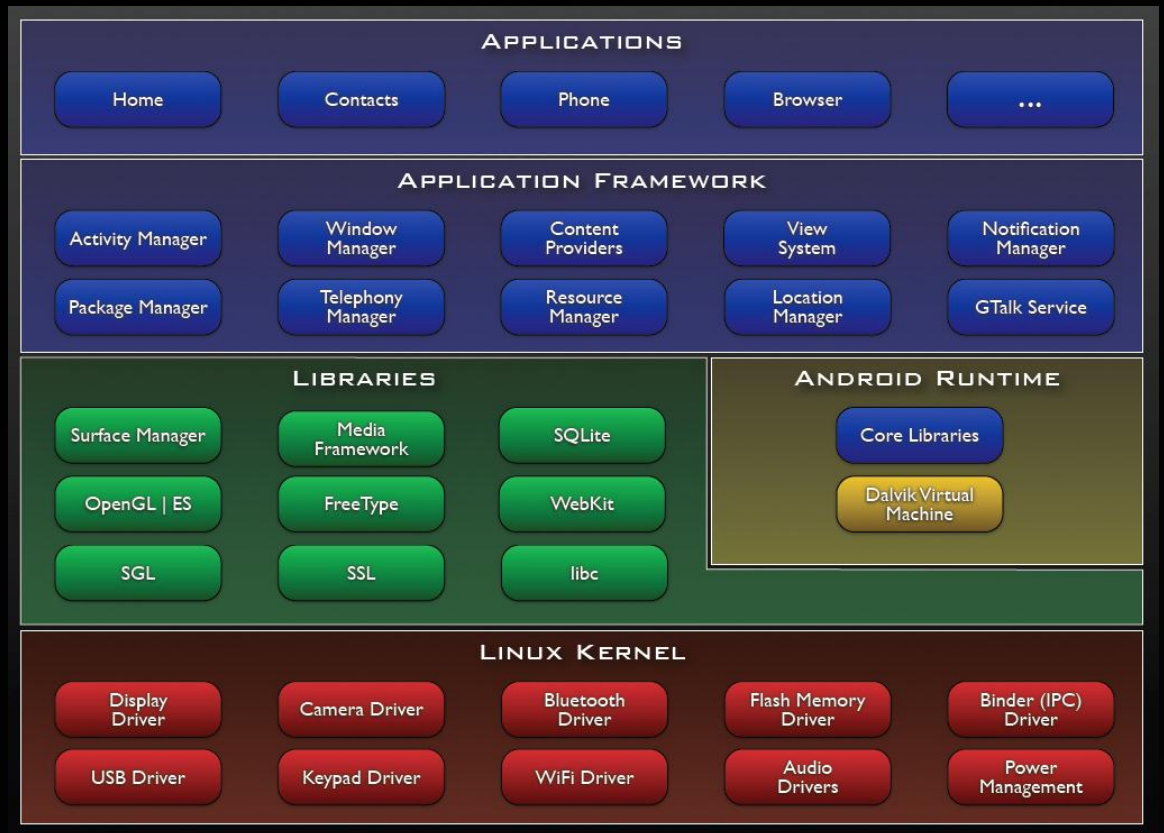
Answer To Q6

- Q: How does Android put them together?
- A: (no correct answer, but reasonable answer)
  - 1. Java as the primary API language
    - Top popularity is not without rationality
  - 2. Application framework
    - Service-oriented, component-based
  - 3. Applications are wired into framework
  - 4. Hardware abstraction layer (HAL)
  - 5. Linux kernel

Holistic design makes the cathedral

Android Stack

- Layers make the holistic design possible



Question 7

- Q: What does a common OS distribution do?
  - APIs
  - Applications
  - Internet/Cloud
  - Security
  - User experience
  - Battery life

Answer To Q7

- Q: What does a common Linux distro do?
- A: (no correct answer, but reasonable answer)
  - APIs: open choices
  - Applications: abundant
  - Internet/Cloud: Mozilla Firefox, etc.
  - Security: Linux security
  - User experience: Gnome, KDE, etc.
  - Battery life: Linux PM

More features may not always excel

Question 8

- Q: How does a common Linux distro put them together?

Answer to Q8

- Q: How does a common Linux distro put them together?
- A: (no correct answer, but reasonable answer)
  - APIs: open choices
  - Applications: abundant
  - Internet/Cloud: Mozilla Firefox, etc.
  - Security: Linux security
  - User experience: Gnome, KDE, etc.
  - Battery life: Linux PM

Bazar is still bazar

Question 9

- Q: What does J2ME do?

Answer to Q9

- Q: What does J2ME do?
- A: (no correct answer, but reasonable answer)
  - APIs: roughly ok
  - Applications: roughly ok
  - Internet/Cloud: roughly ok
  - Security: roughly ok
  - User experience: roughly ok
  - Battery life: roughly ok

Roughly ok is not ok

Question 10

- Q: How does J2ME put them together?

Answer to Q10

- Q: How does J2ME put them together?
- A: (no correct answer, but reasonable answer)
  - APIs: ok + OS
  - Applications: ok + OS
  - Internet/Cloud: ok + OS
  - Security: ok + OS
  - User experience: ok + OS
  - Battery life: ok + OS

Who are you, J2ME or OS?

Question 11

- Q: What does Windows desktop do?
- A: (no correct answer, but reasonable answer)
  - APIs
  - Applications
  - Internet/Cloud
  - Security
  - User experience
  - Battery life

Answer to Q11

- Q: What does Windows desktop do?
- A: (no correct answer, but reasonable answer)
  - APIs: Win API in C++
  - Applications: abundant
  - Internet/Cloud: IE
  - Security: oops!
  - User experience: “(Not responding)”
  - Battery life: …

Time does not solve all problems

Question 12

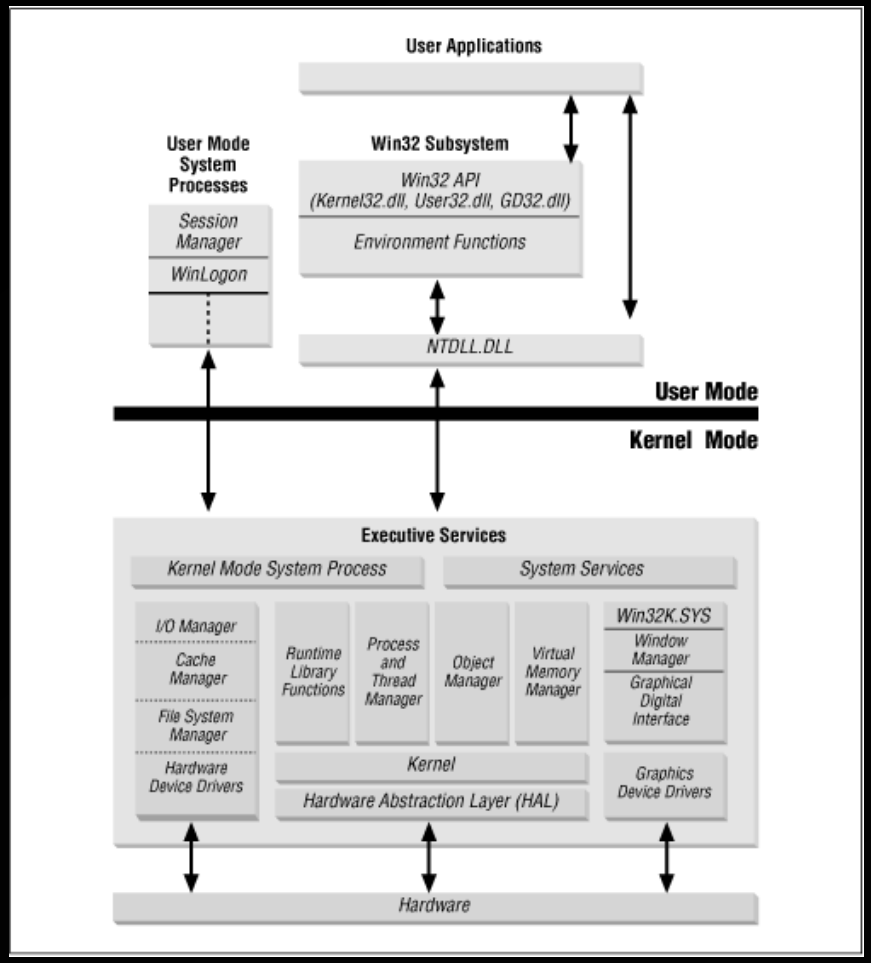
- Q: How does Windows put them together?

Answer to Q12

- Q: How does Windows put them together?
- A: (no correct answer, but reasonable answer)
  - Win API
  - Kernel

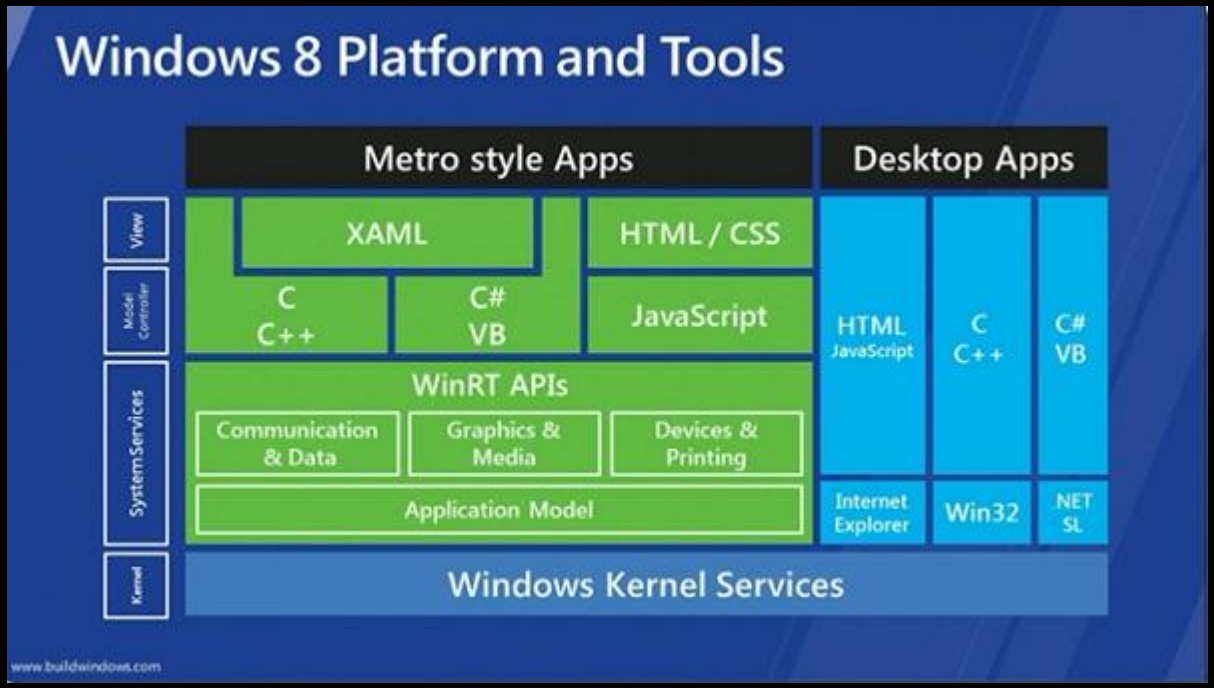
Time freezes at 80s

Windows Stack



Services built on top of API

Bonus Slide: Windows 8 Stack



- Much better as a smartphone OS as I expected

API built on top of services

Android Design Explained

- Complete, consistent, modular
  - API: Full sets of high level APIs for all the tasks
  - Apps: components to be wired into the system
  - App framework: rich set of services/abstractions
  - Security: every application is a Linux user
  - User experience: mem/power/perf, smoothness/responsiveness

Android overview

—— From a system design perspective

PART II

Runtime model

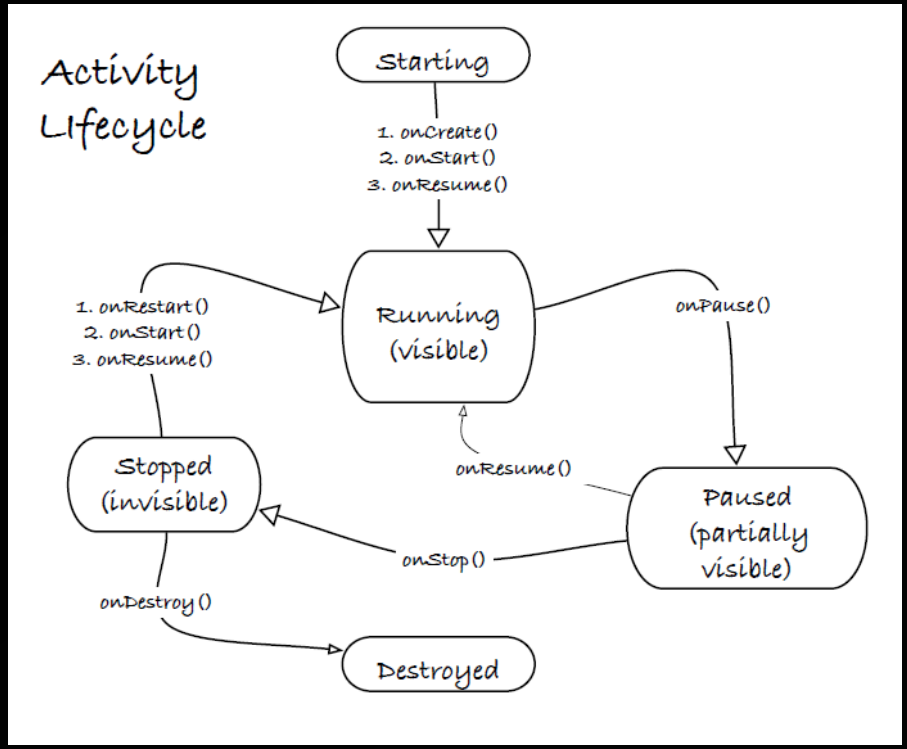
Android Application

- Not a standalone application, but components of the system

Activity	UI component typically corresponding to one screen.
IntentReceiver	Set and respond to notifications or status changes. Can wake up your app.
Service	Faceless task that runs in the background.
ContentProvider	Enable applications to share data.

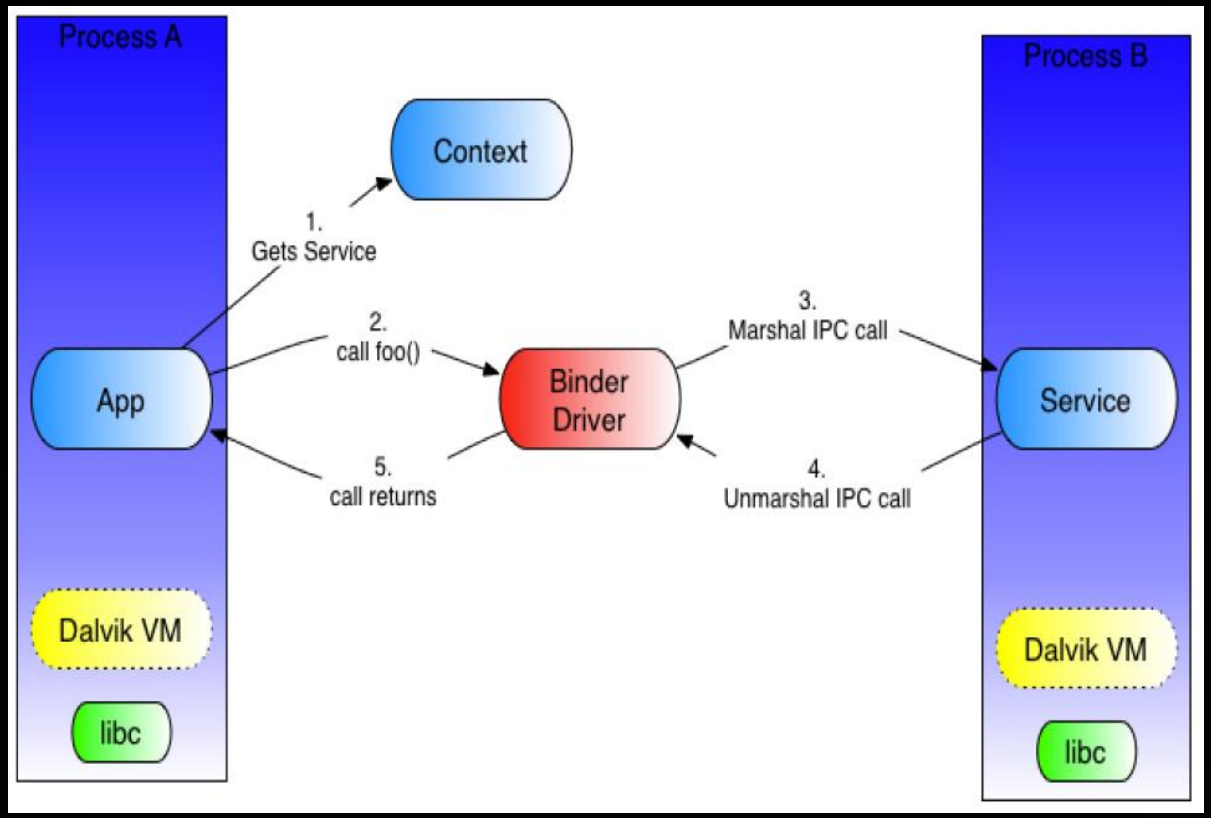
Activity Lifecycle

- Activity is integrated with Activity Manager Service through binder



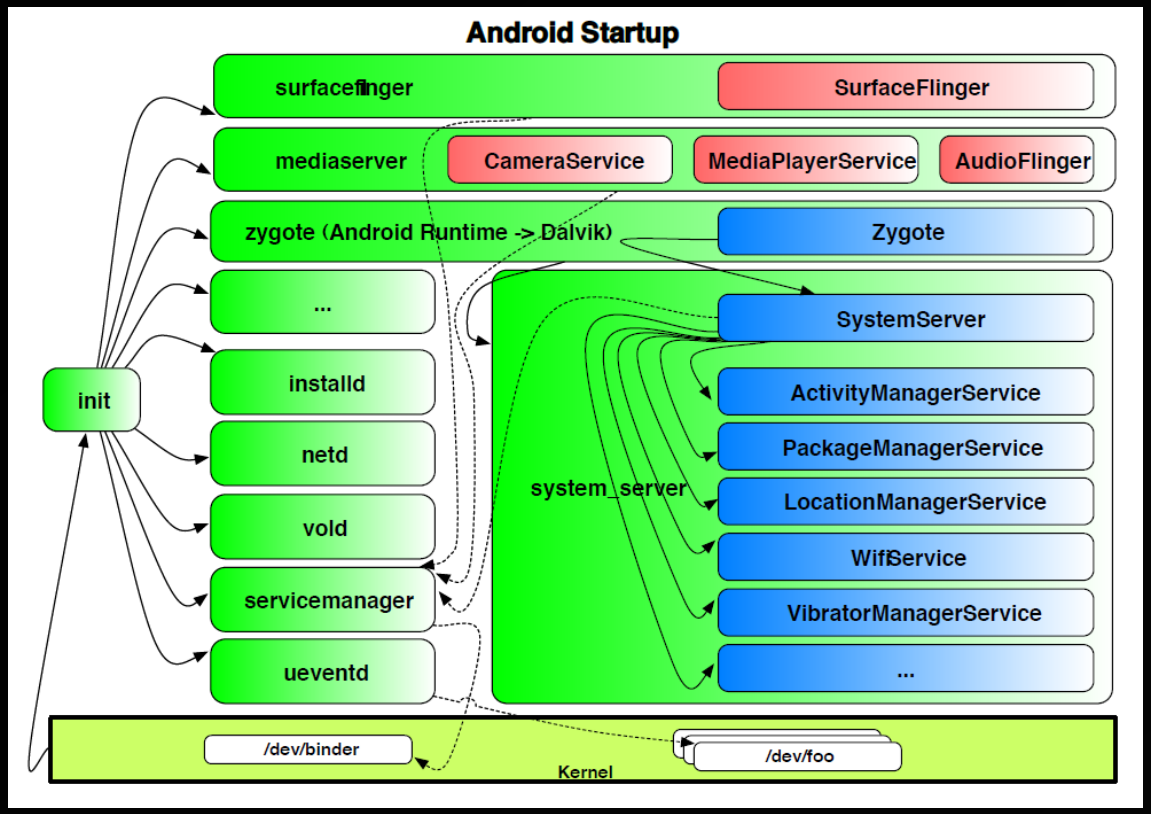
Android Service

- Android is service-oriented design



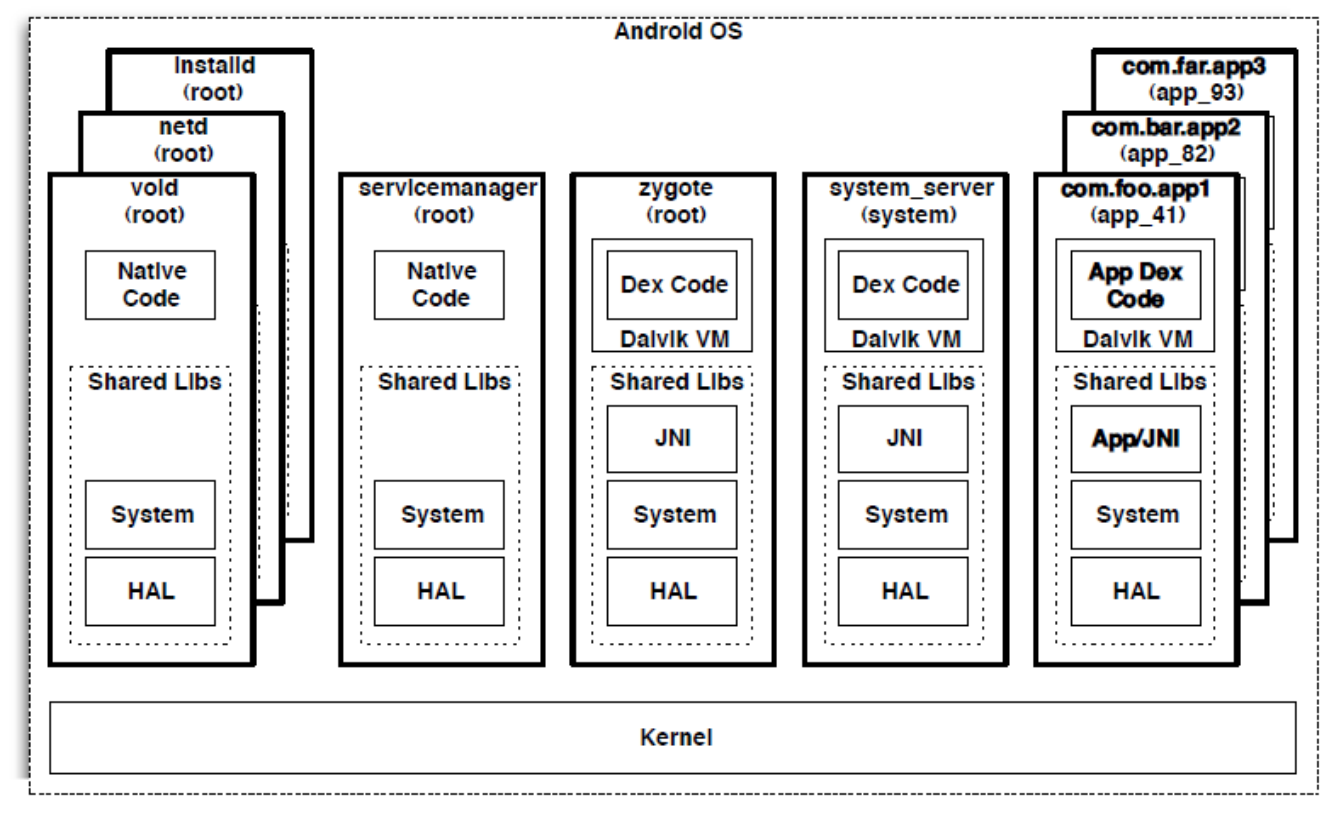
Android Startup

- Launch all the system services



Android Running

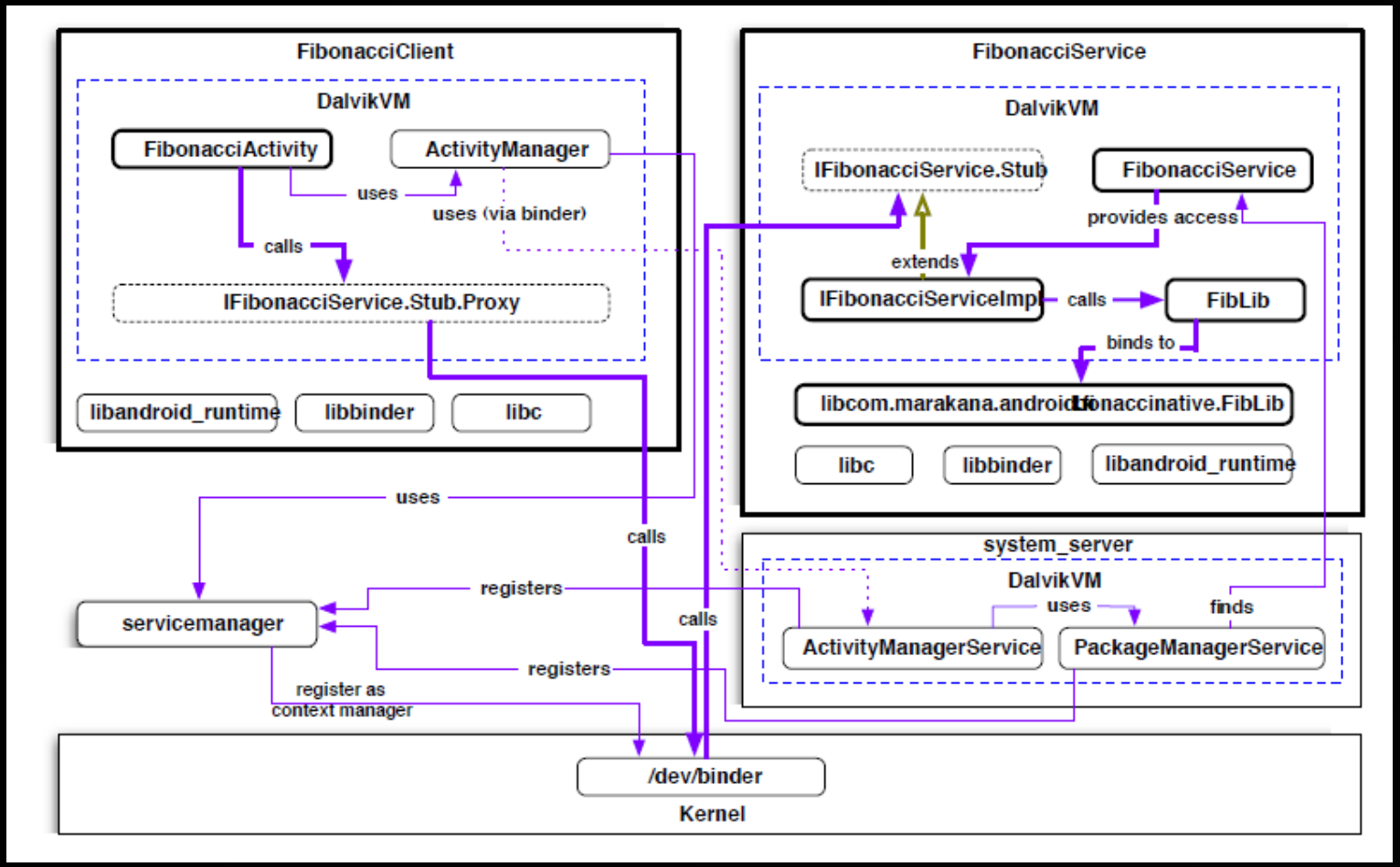
- Sanboxed applications (and services)



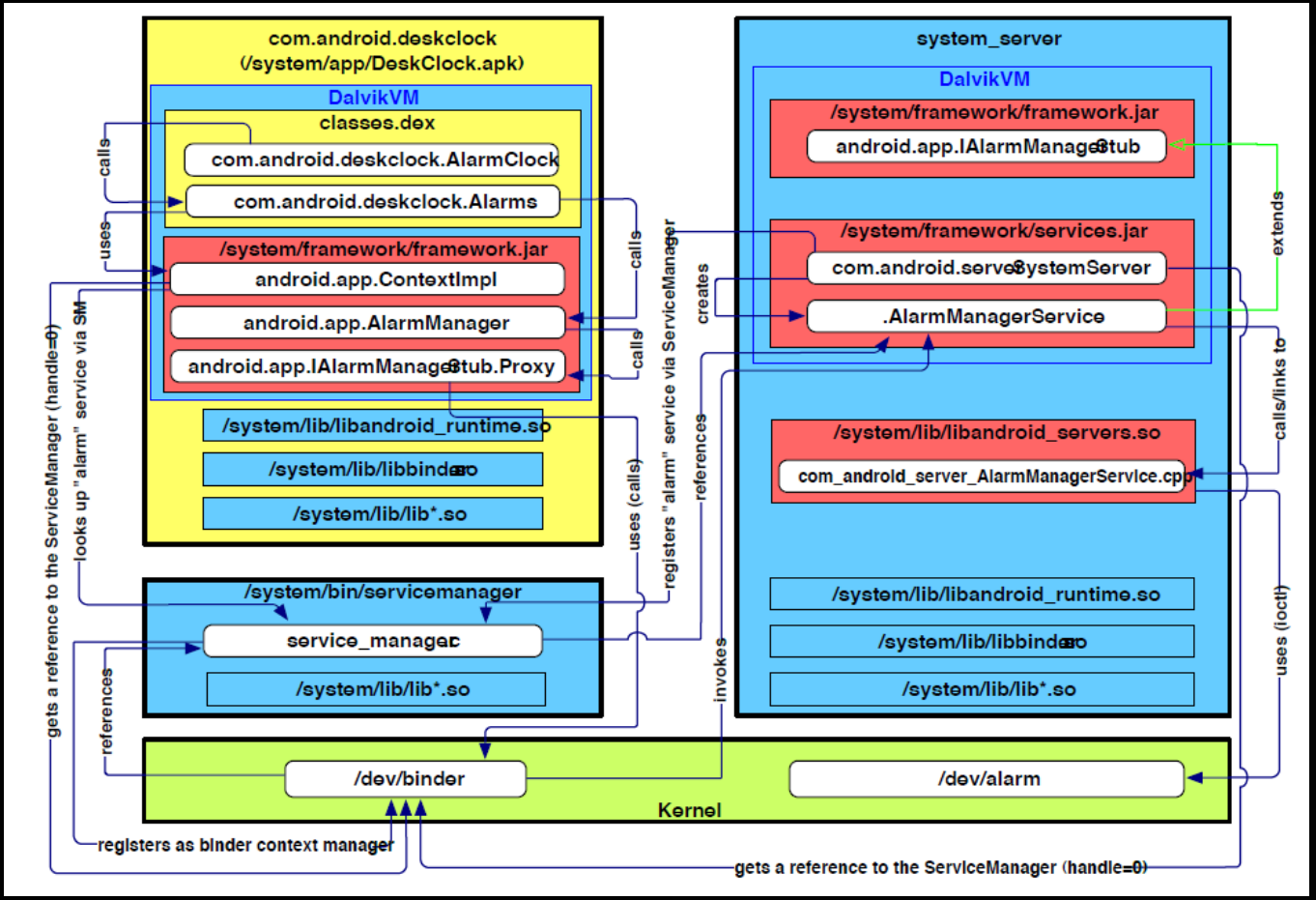
Android Binder

- Core IPC mechanism that enables Android client/server modular design
  - Like CORBA, but much simpler and more efficient
    - Only for intra-OS IPC
  - AIDL to define the service interface (stub, proxy)
    - Support both C++ and Java binding
  - Linux kernel modified to support binder
    - Also for IPC performance and security

Binder Running

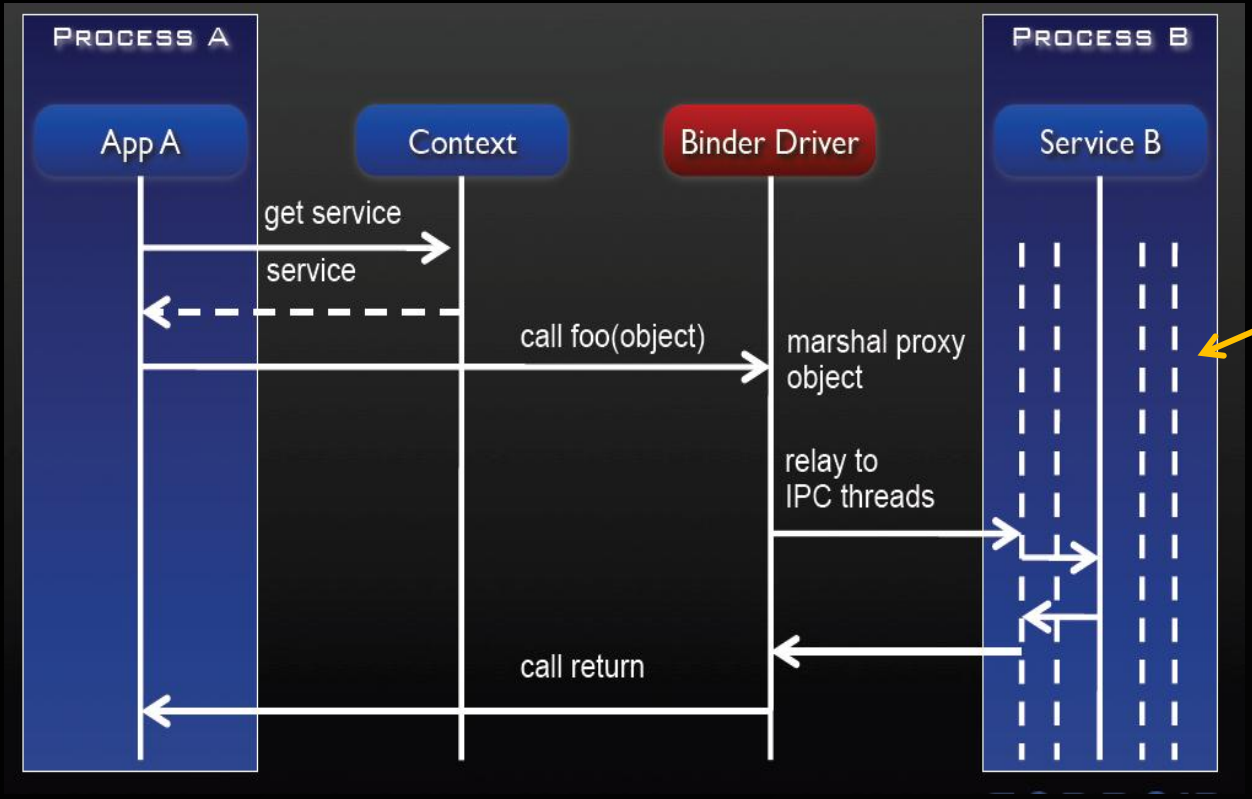


Example: Alarm Service



Binder Threads

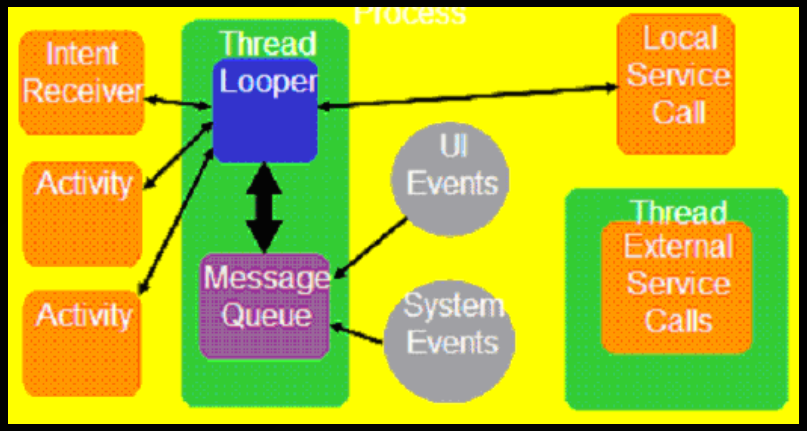
- Binder threads exist invisibly for IPC

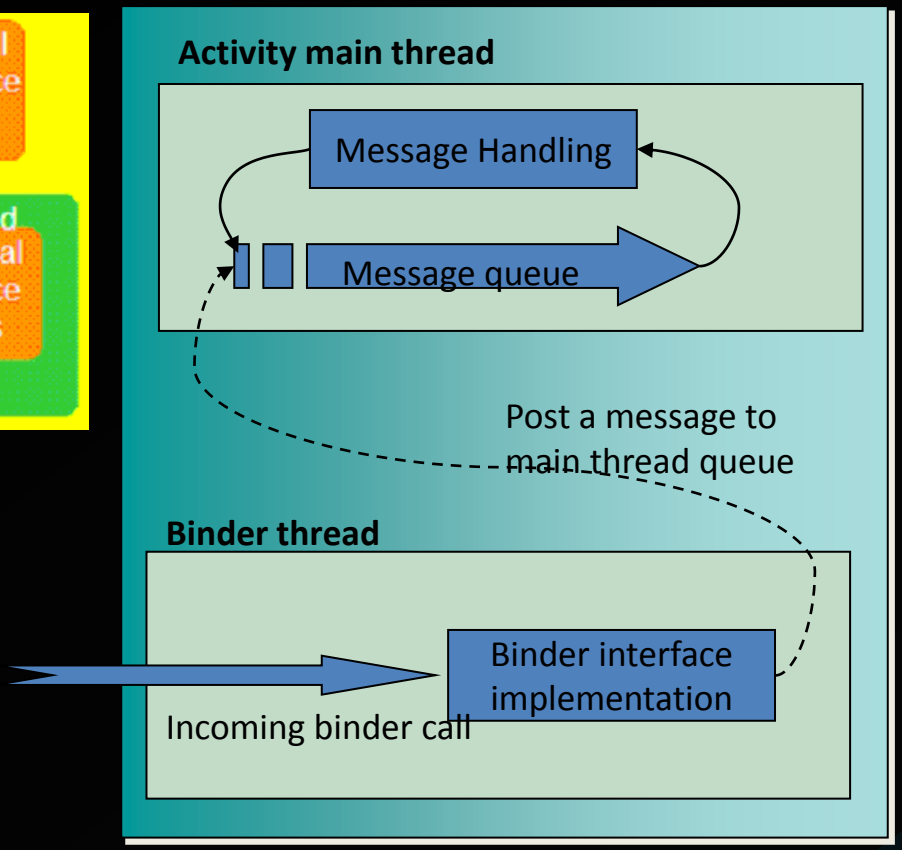


Binder threads created by system automatically

Android Application Structure

- Role of binder thread

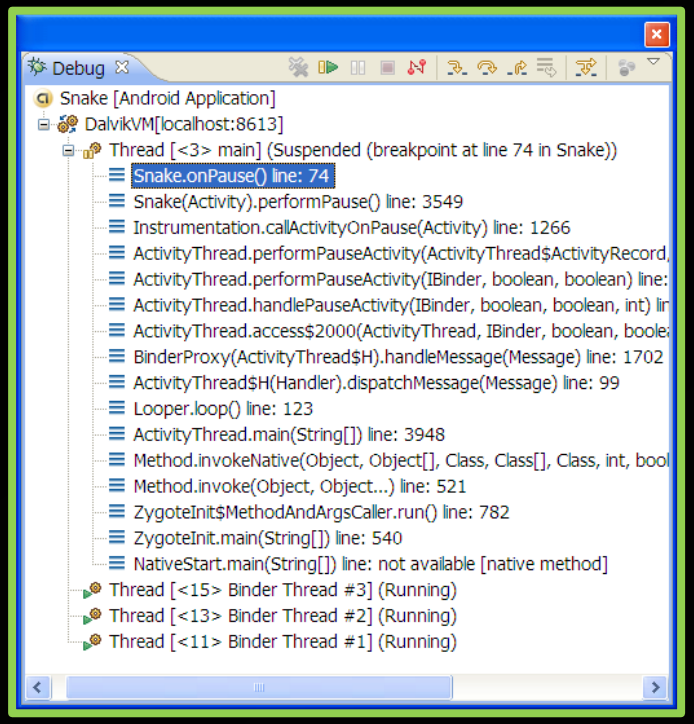


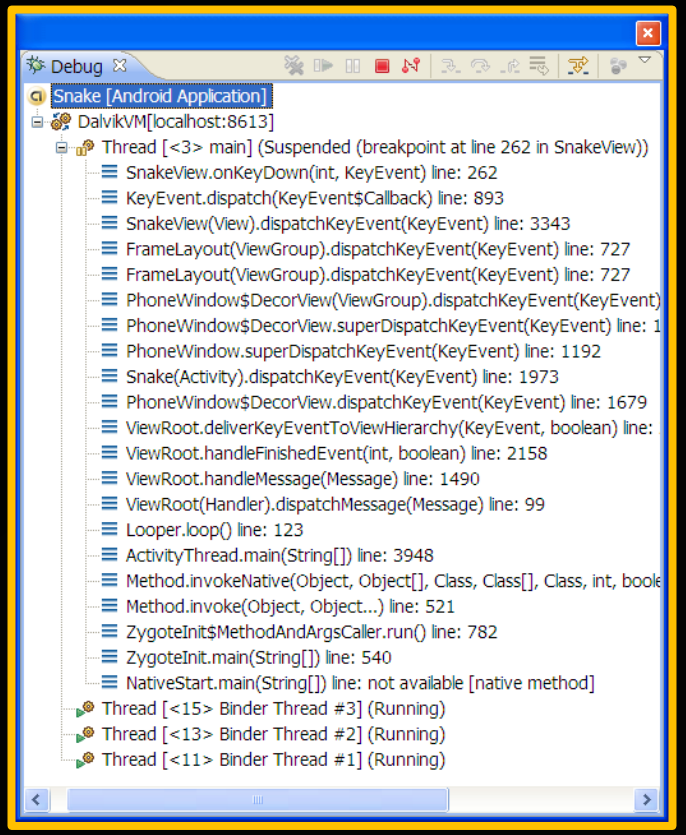


Binder Threads Running

Binder #1: IApplicationThread

Activity lifecycle transition callback is triggered by the incoming call on binder IApplicationThread, that was initiated by AM service.

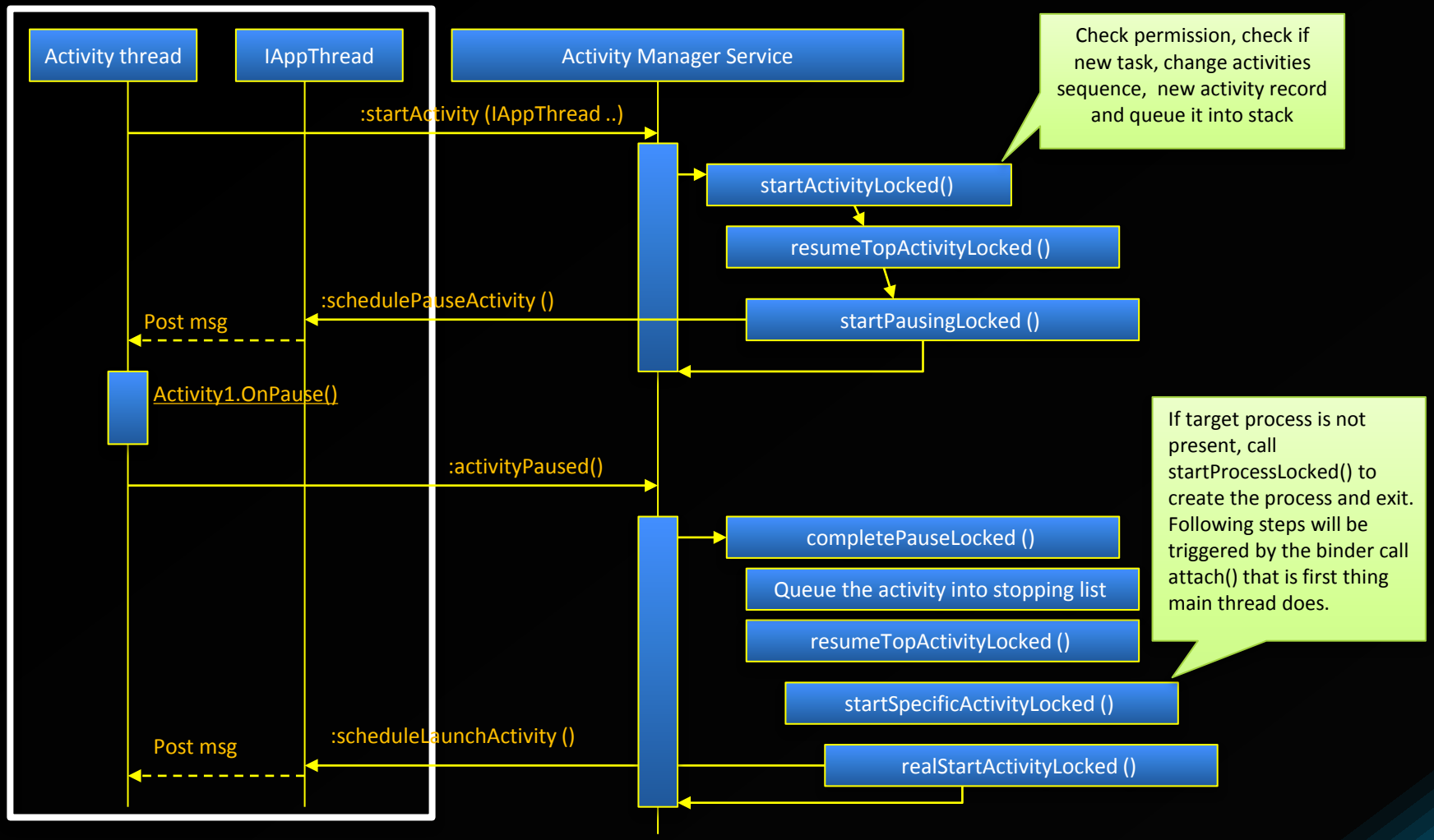




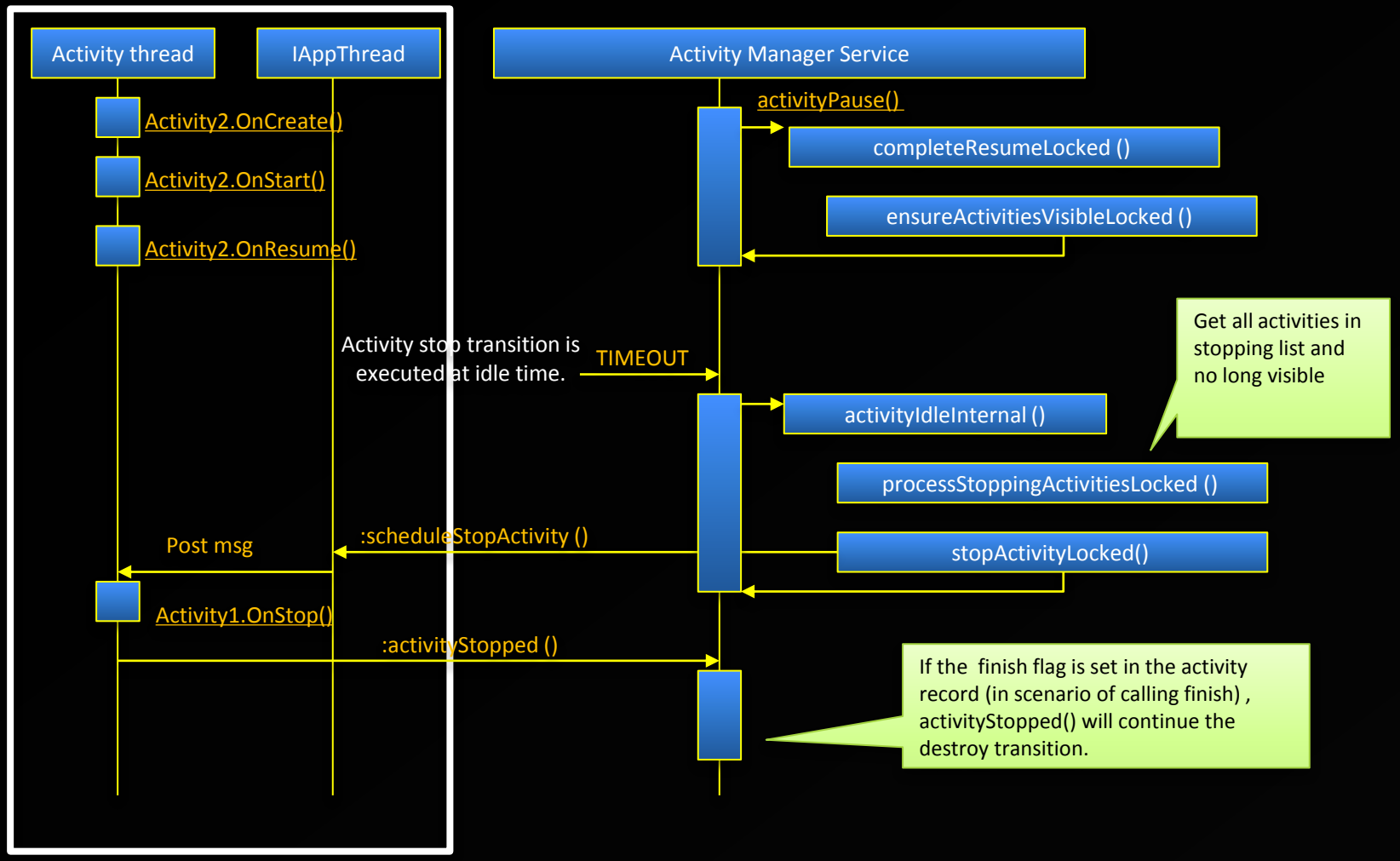
Binder #2: IWindow.stub

UI events handling triggered by **IWindow.stub** binder calls

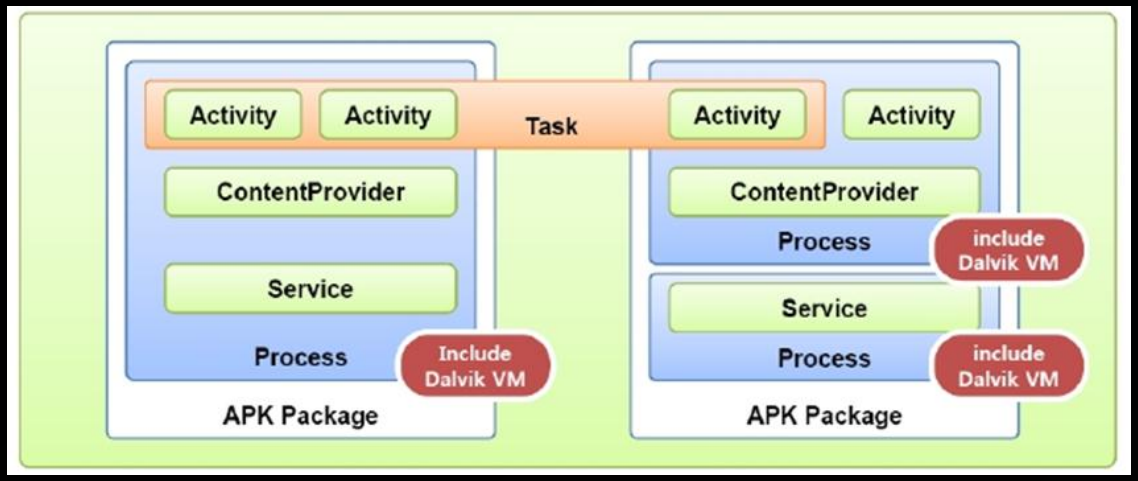
Start Activity (1/2)

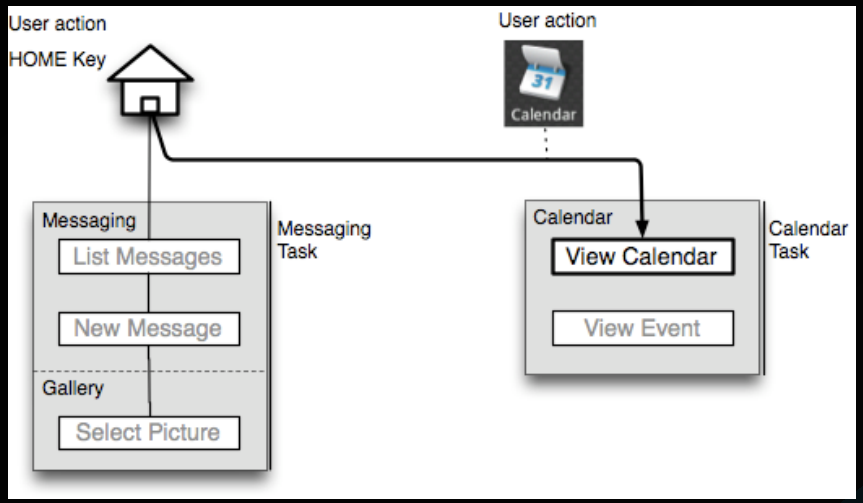


Start Activity (2/2)

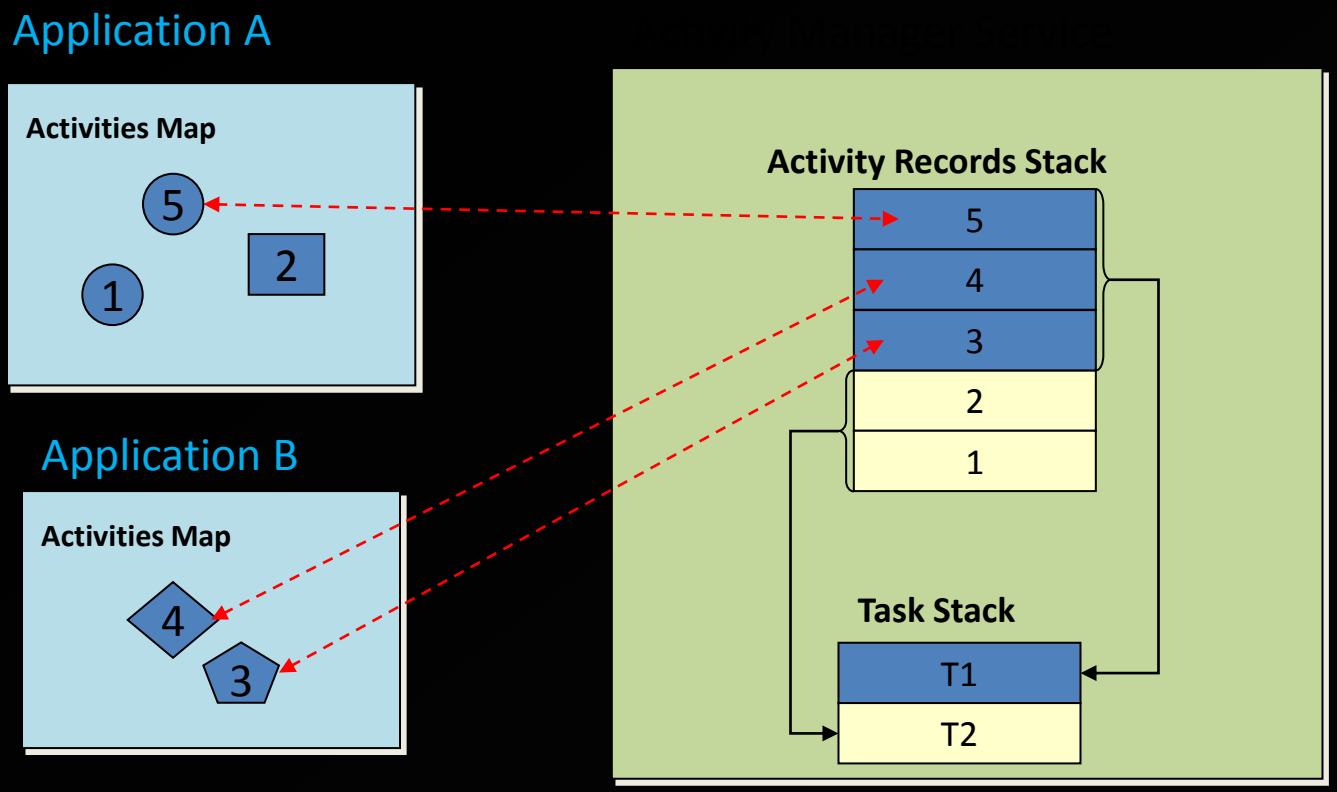


Activities And Tasks (1/2)





Activities And Tasks (1/2)



Android overview

—— From a system design perspective

(Not yet put together here …)

PART III

Framework design

Summary

- Smartphone OS has to be open and elegant
  - Comprehensive and consistent API
  - Service oriented framework to back the API
  - App components are wired into the framework
  - HAL to hide hardware differences
  - UID/process based security support to apps
  - Traditional OS kernel for platform management

Acknowledgements

- Some of the figures are taken from materials of their respective owners