Android Anatomy and Physiology

Agenda

- Android Anatomy
  - Linux Kernel
  - Native Libraries
  - Android Runtime
  - Application Framework
- Android Physiology
  - Start-up Walkthrough
  - Layer Interaction

Android Anatomy

Applications

Home | Dialer | SMS/MMS | IM | Browser | Camera | Alarm | Calculator

Contacts | Voice Dial | Email | Calendar | Media Player | Albums | Clock | ...

Application Framework

Activity Manager | Window Manager | Content Providers | View System | Notification Manager

Package Manager | Telephony Manager | Resource Manager | Location Manager | ...

Libraries

Surface Manager | Media Framework | SQLite

OpenGL|ES | FreeType | WebKit

SGL | SSL | Libc

Android Runtime

Core Libraries

Dalvik Virtual Machine

Linux Kernel

Display Driver | Camera Driver | Bluetooth Driver | Shared Memory Driver | Binder (IPC) Driver

USB Driver | Keypad Driver | WiFi Driver | Audio Drivers | Power Management

Agenda

- Android Anatomy
  - Linux Kernel
  - Native Libraries
  - Android Runtime
  - Application Framework
- Android Physiology
  - Start-up Walkthrough
  - Layer Interaction

Linux Kernel

- Android is built on the Linux kernel, but Android is not Linux
- No native windowing system
- No glibc support
- Does not include the full set of standard Linux utilities

android

Linux Kernel

- Standard Linux 2.6.24 Kernel
- Patch of “kernel enhancements” to support Android

android

Why Linux Kernel?

- Great memory and process management
- Permissions-based security model
- Proven driver model
- Support for shared libraries
- It's already open source!

android

Kernel Enhancements

- Alarm
- Ashmem
- Binder
- Power Management
- Low Memory Killer
- Kernel Debugger
- Logger

android

Binder: Problem

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

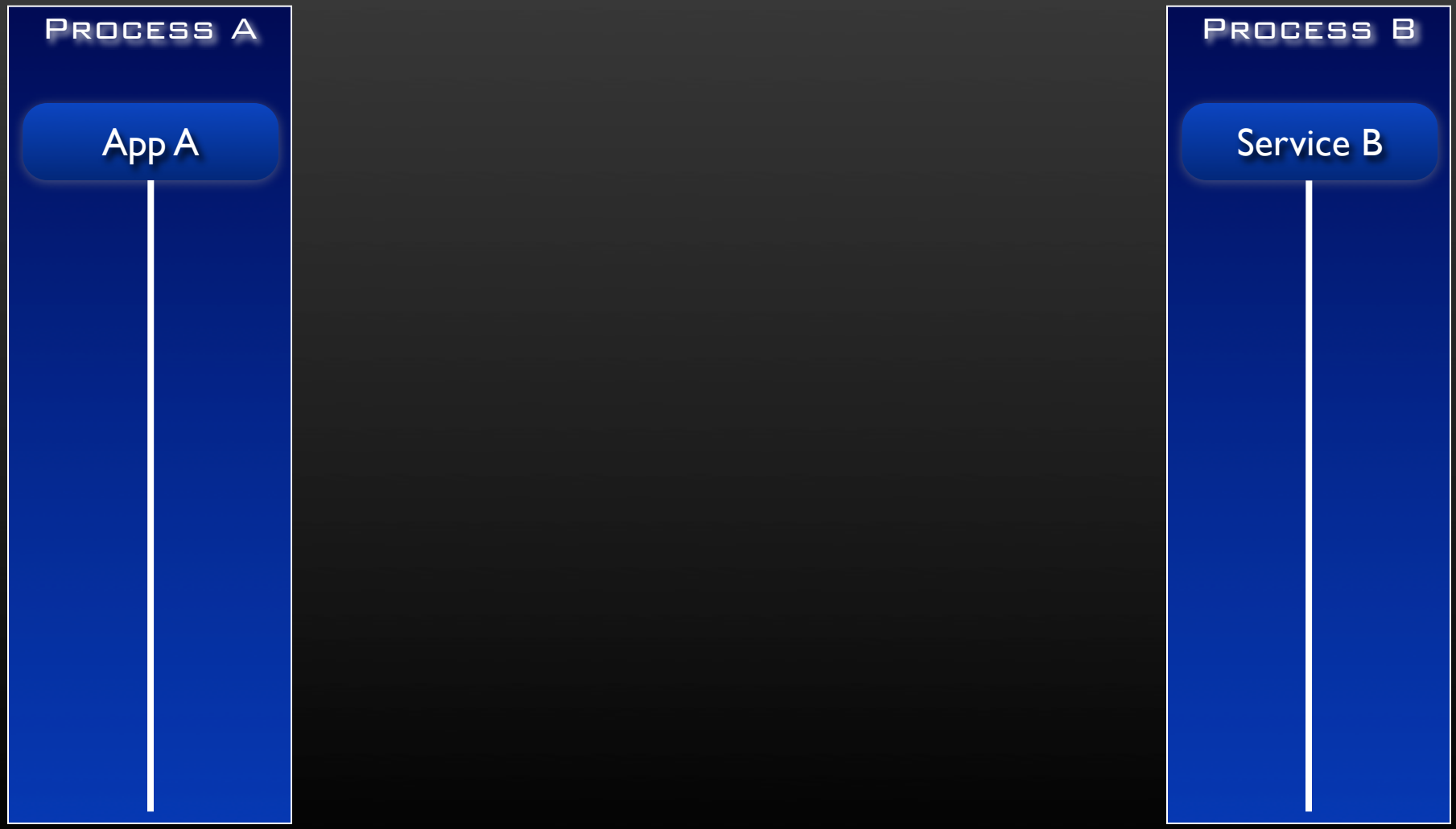
android

Binder: Solution

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

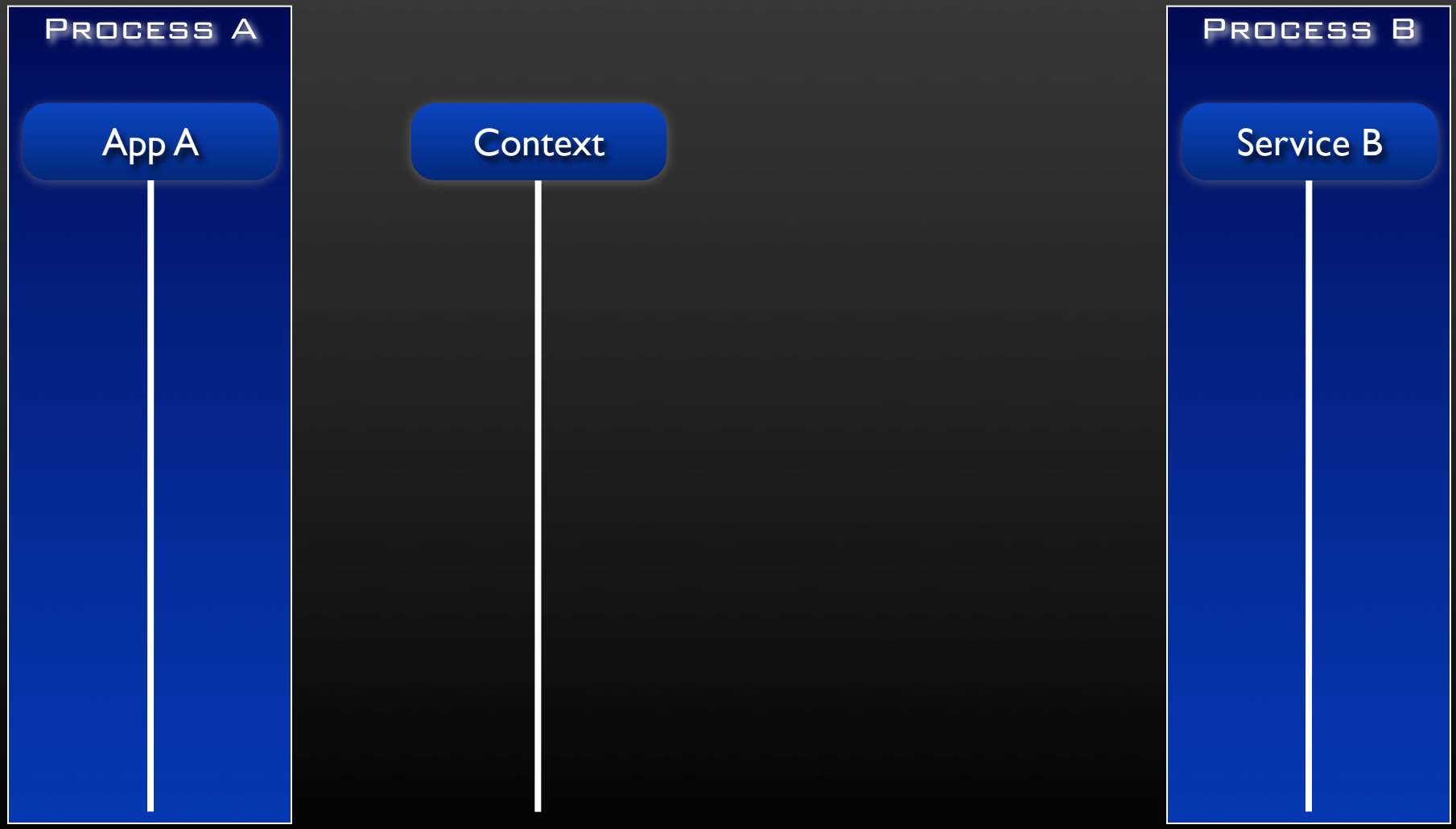
android

Binder in Action



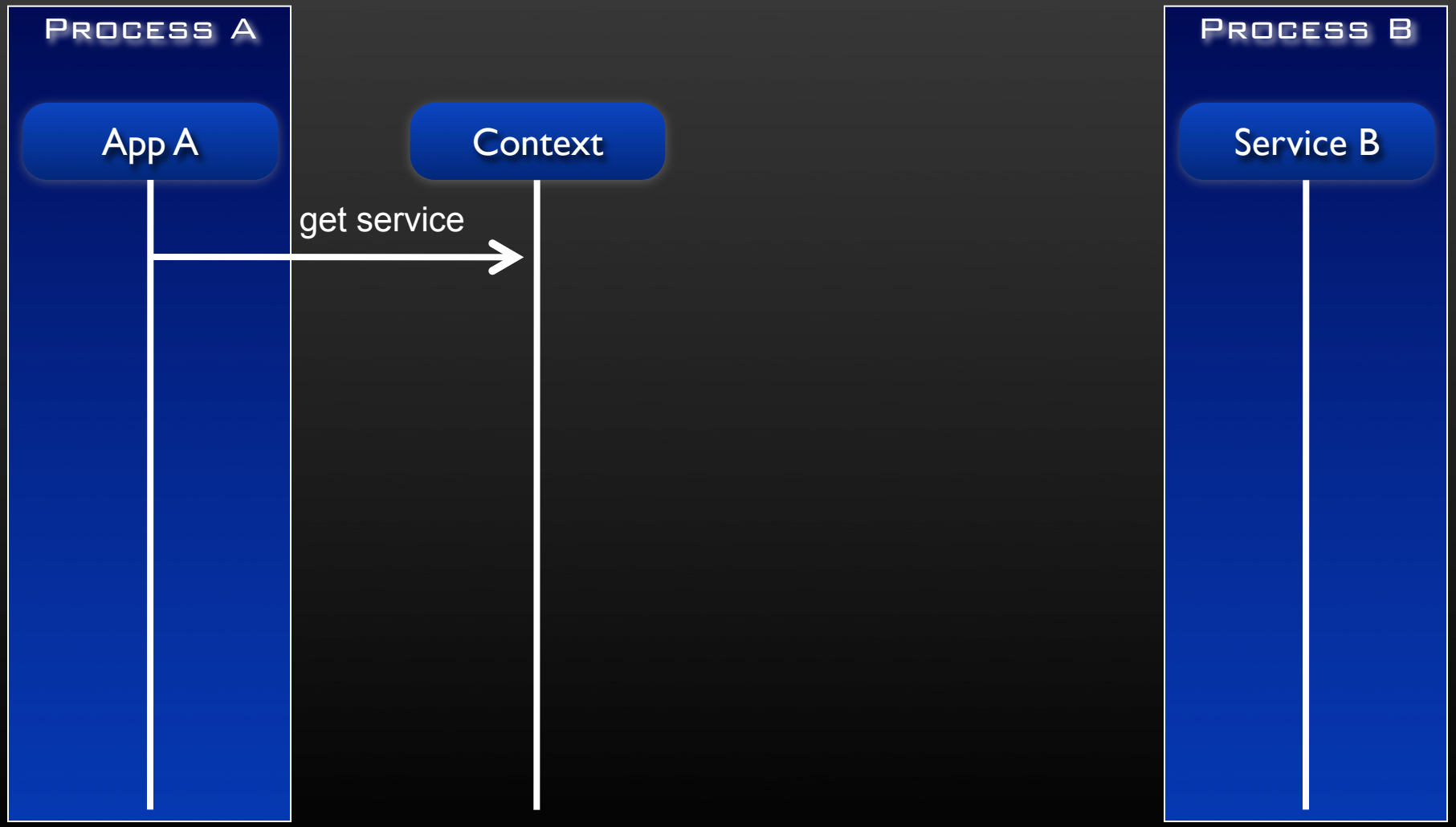
android

Binder in Action



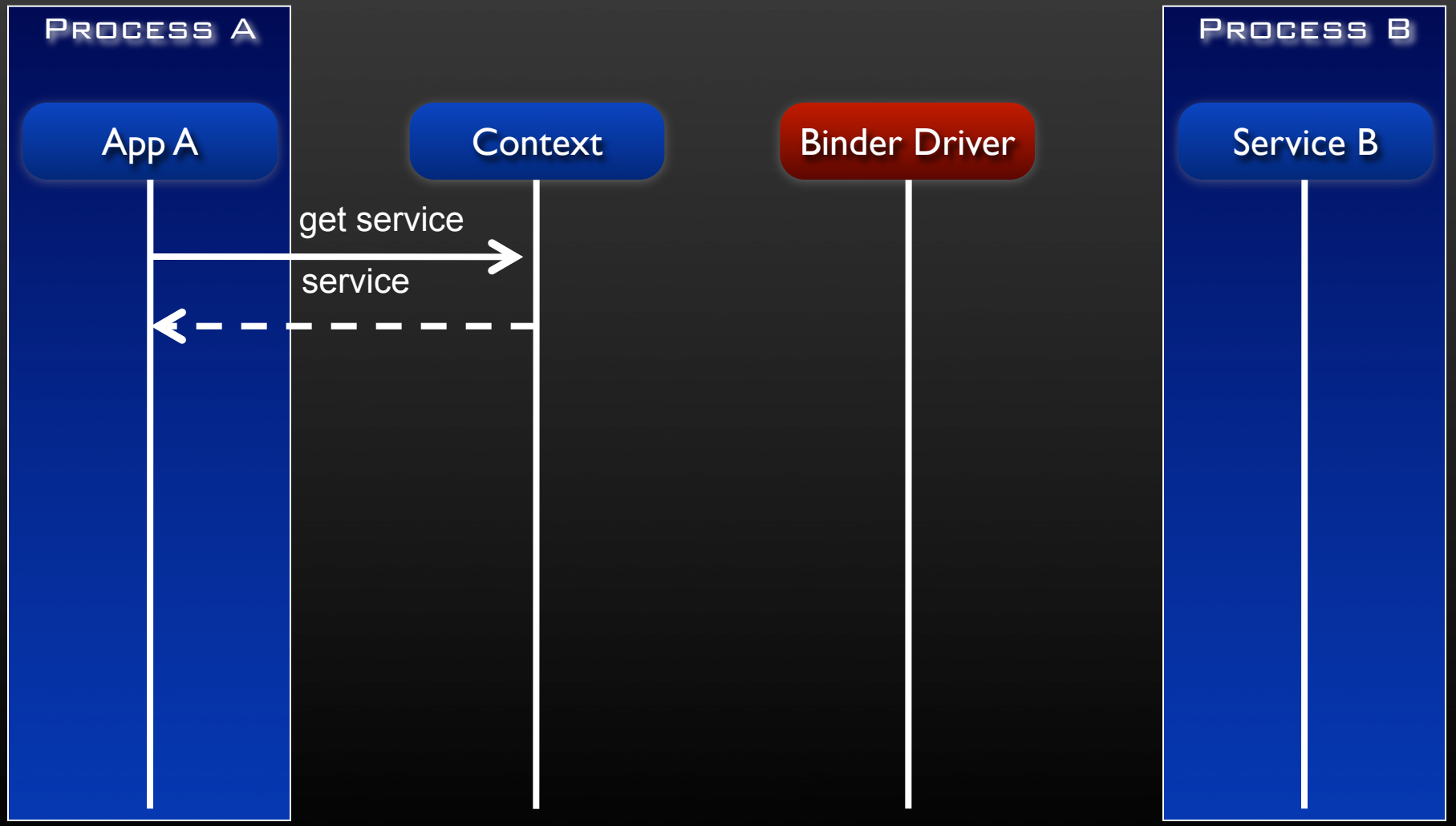
android

Binder in Action



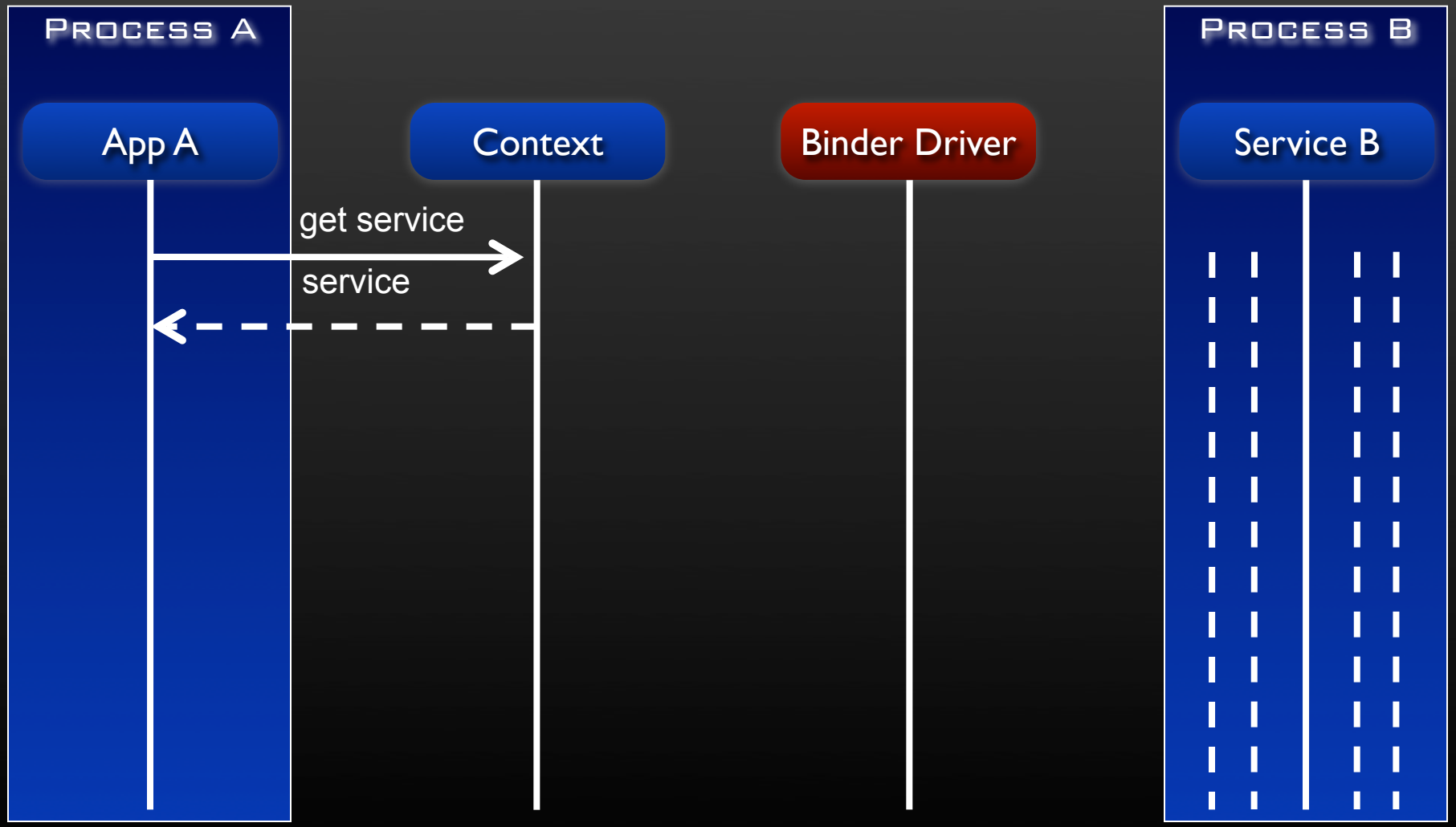
android

Binder in Action



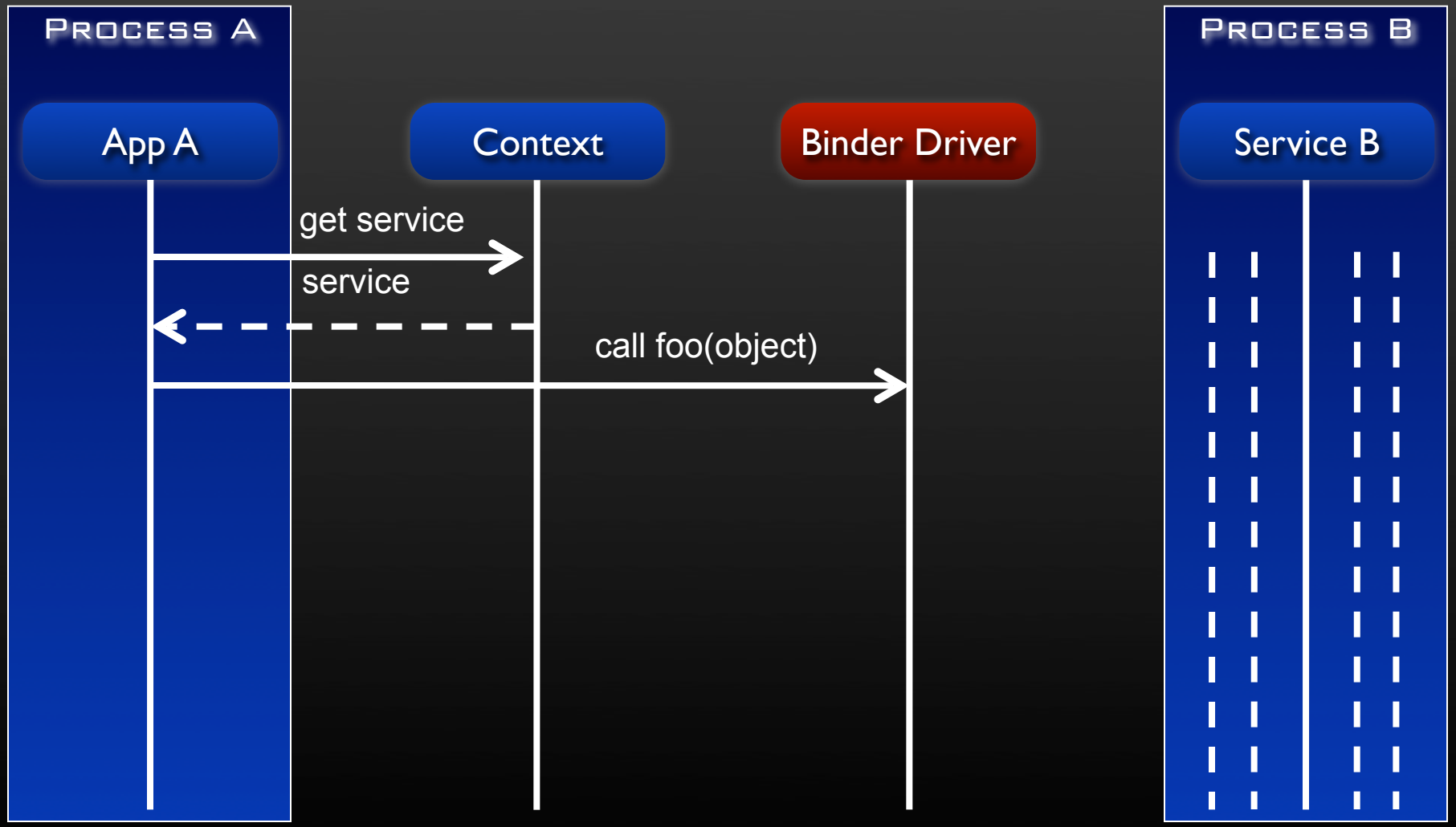
android

Binder in Action



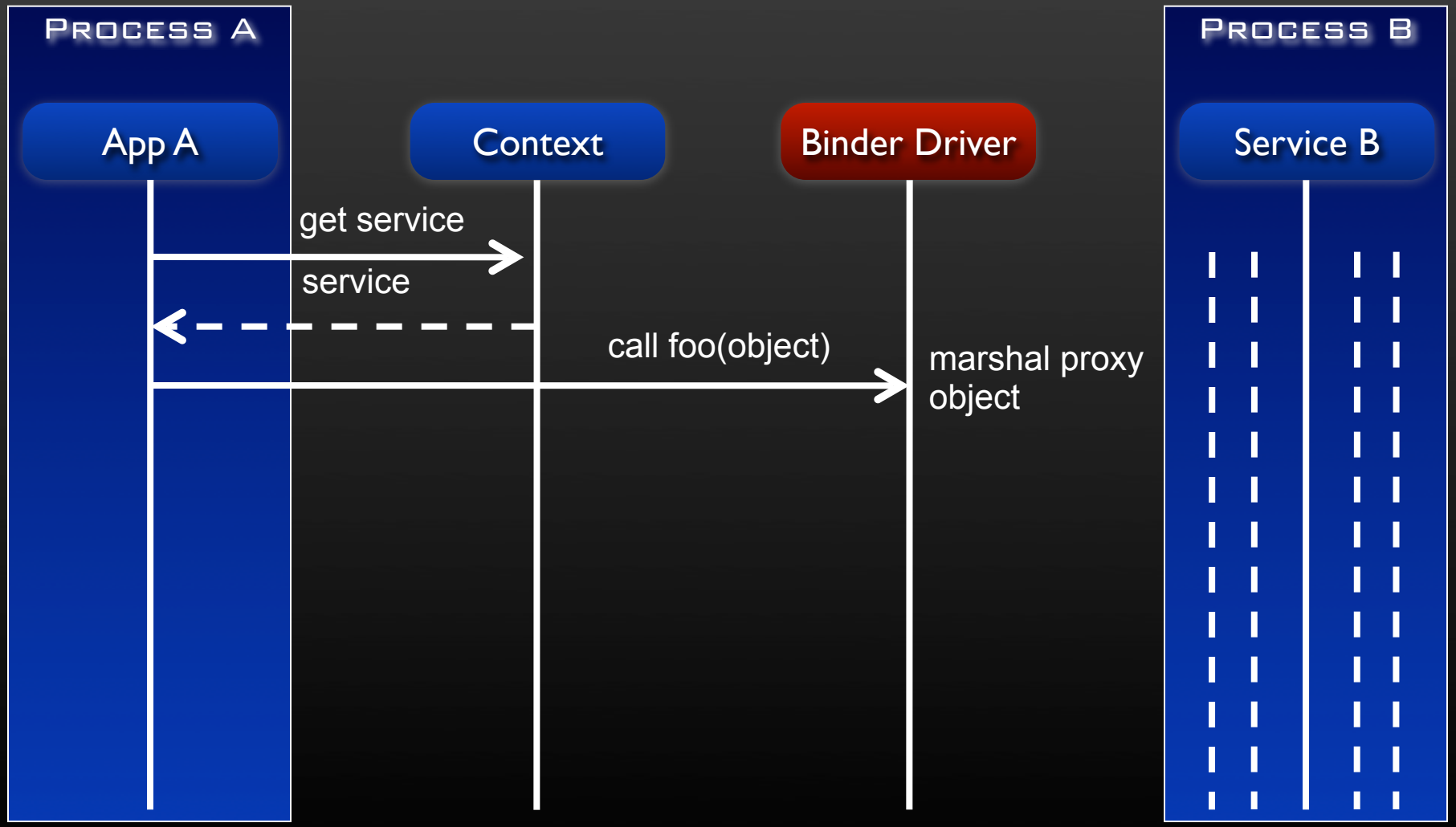
android

Binder in Action



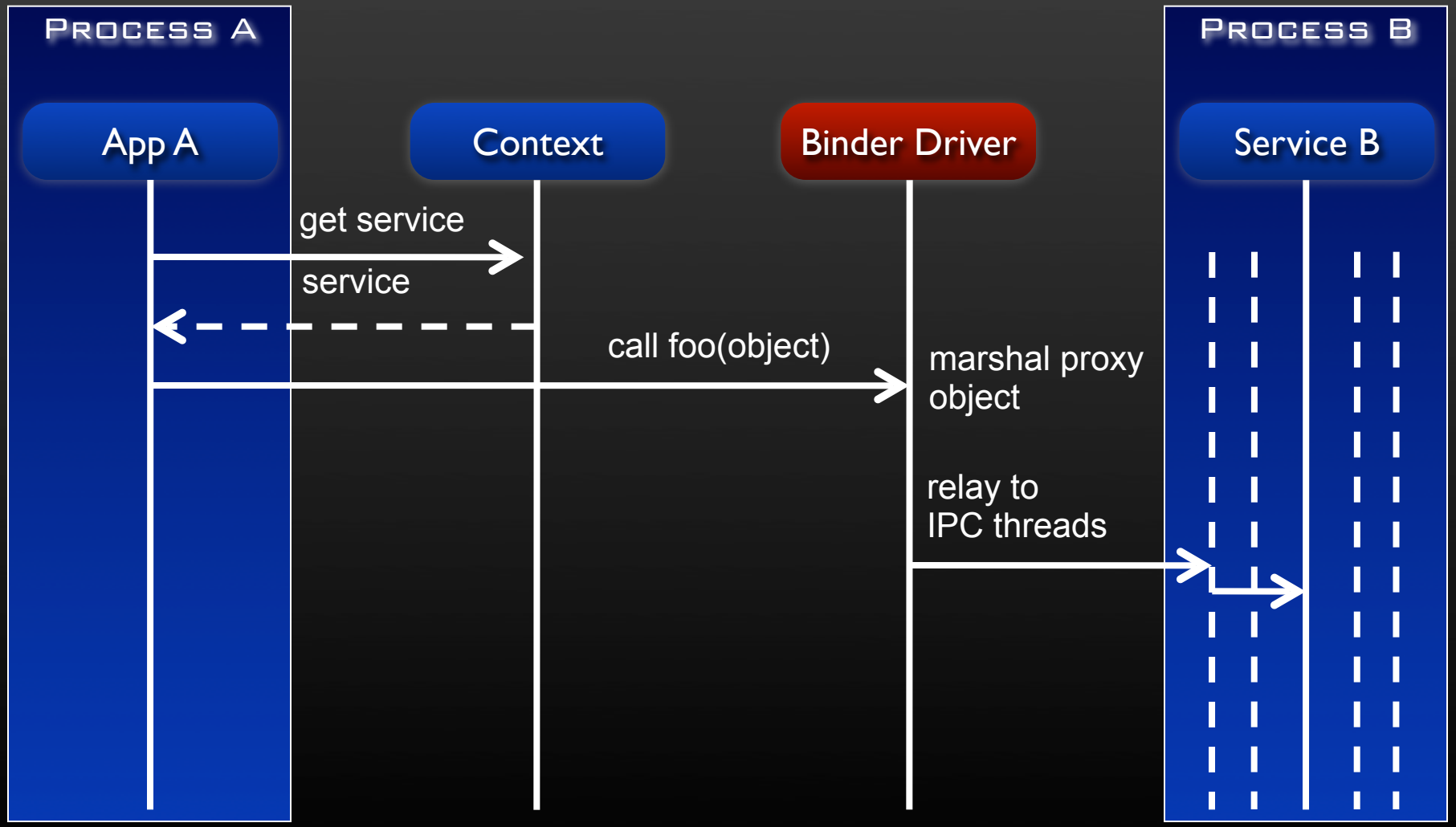
android

Binder in Action



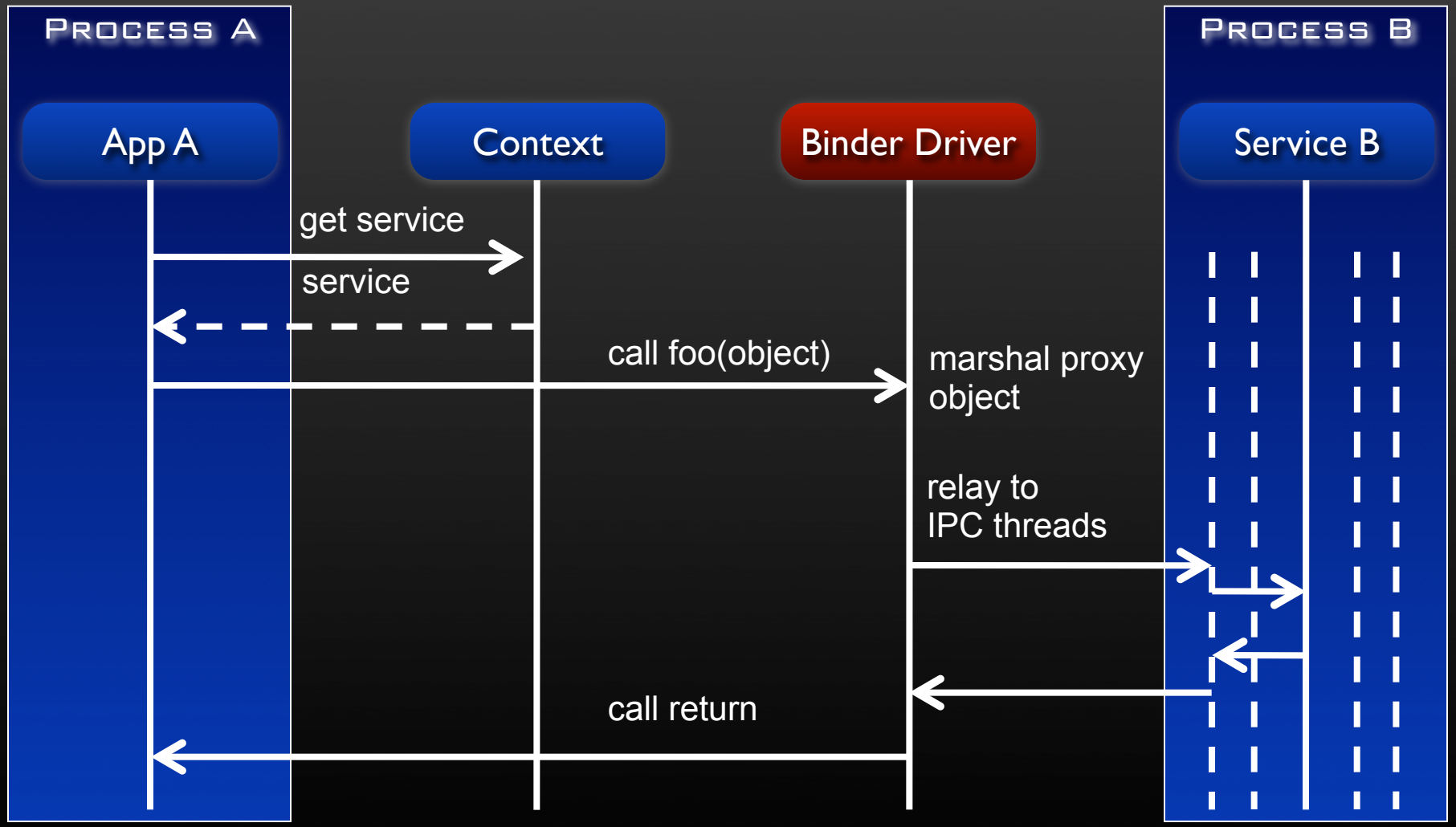
android

Binder in Action



android

Binder in Action



android

Binder

Android Interface Definition Language (AIDL)

- http://code.google.com/android/reference/aidl.html

android

PM Problem

- Mobile devices run on battery power
- Batteries have limited capacity

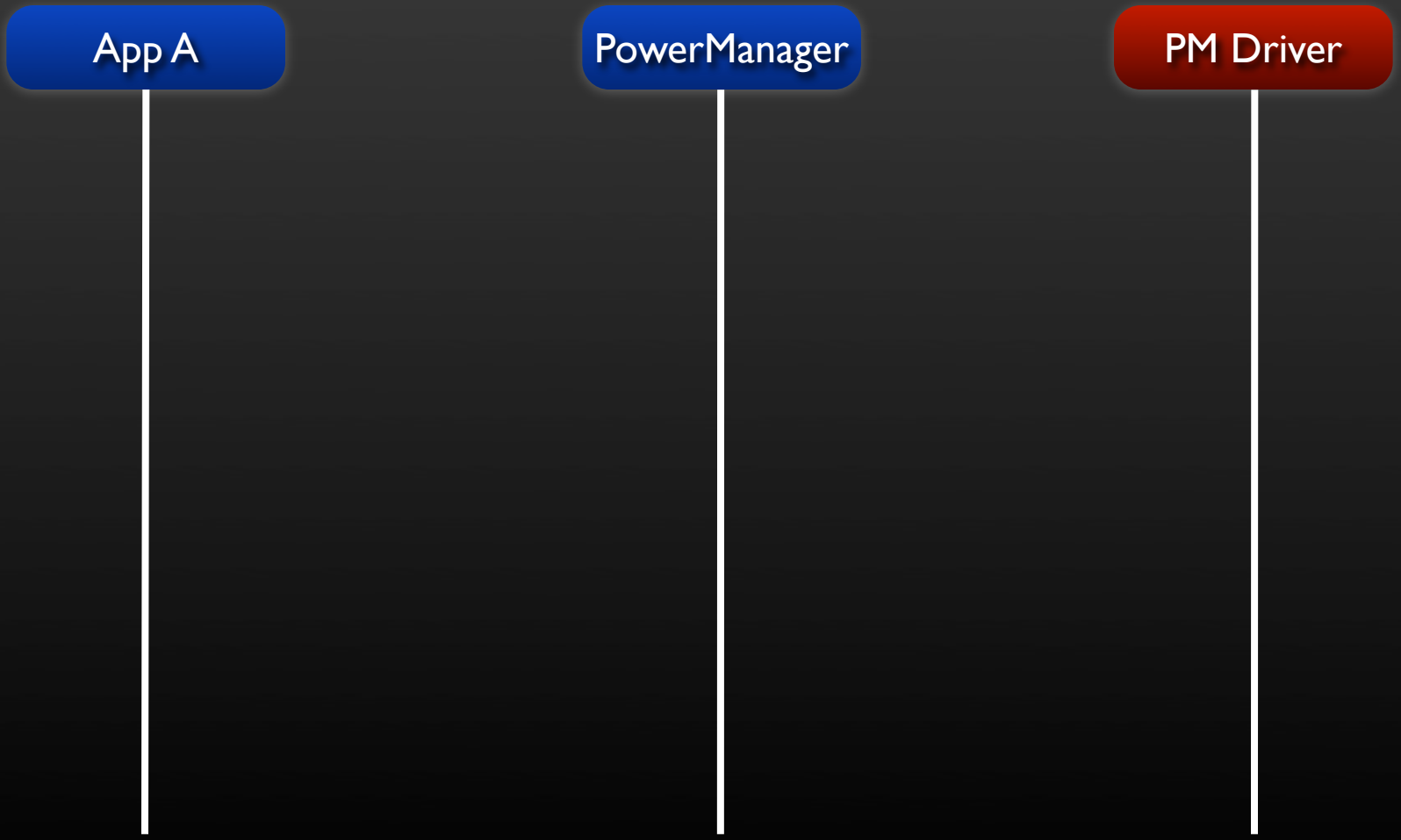
android

PM Solution

- Built on top of standard Linux Power Management (PM)
- More aggressive power management policy
- Components make requests to keep the power on through *“wake locks”*
- Supports different types of wake locks

android

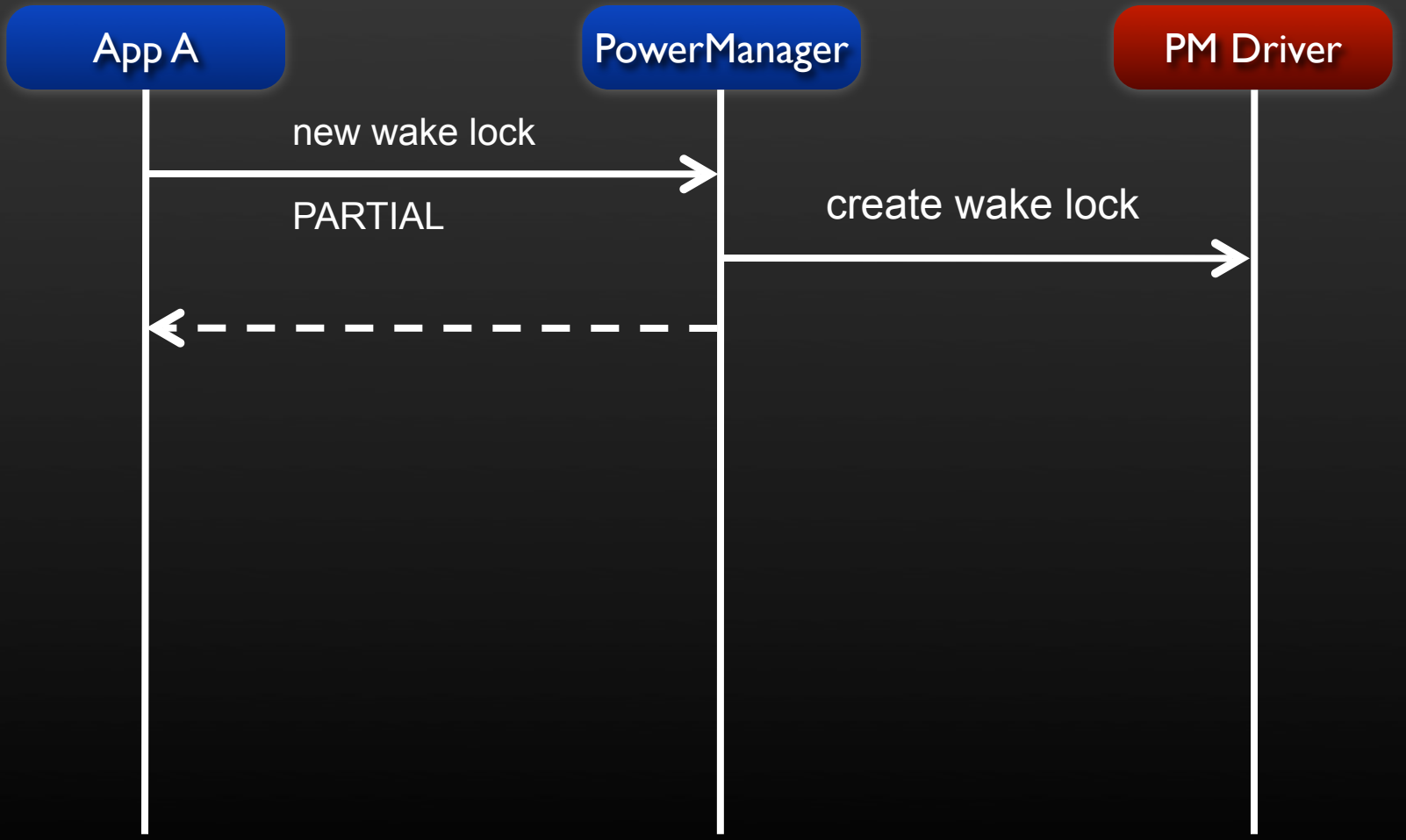
Android PM in Action



android

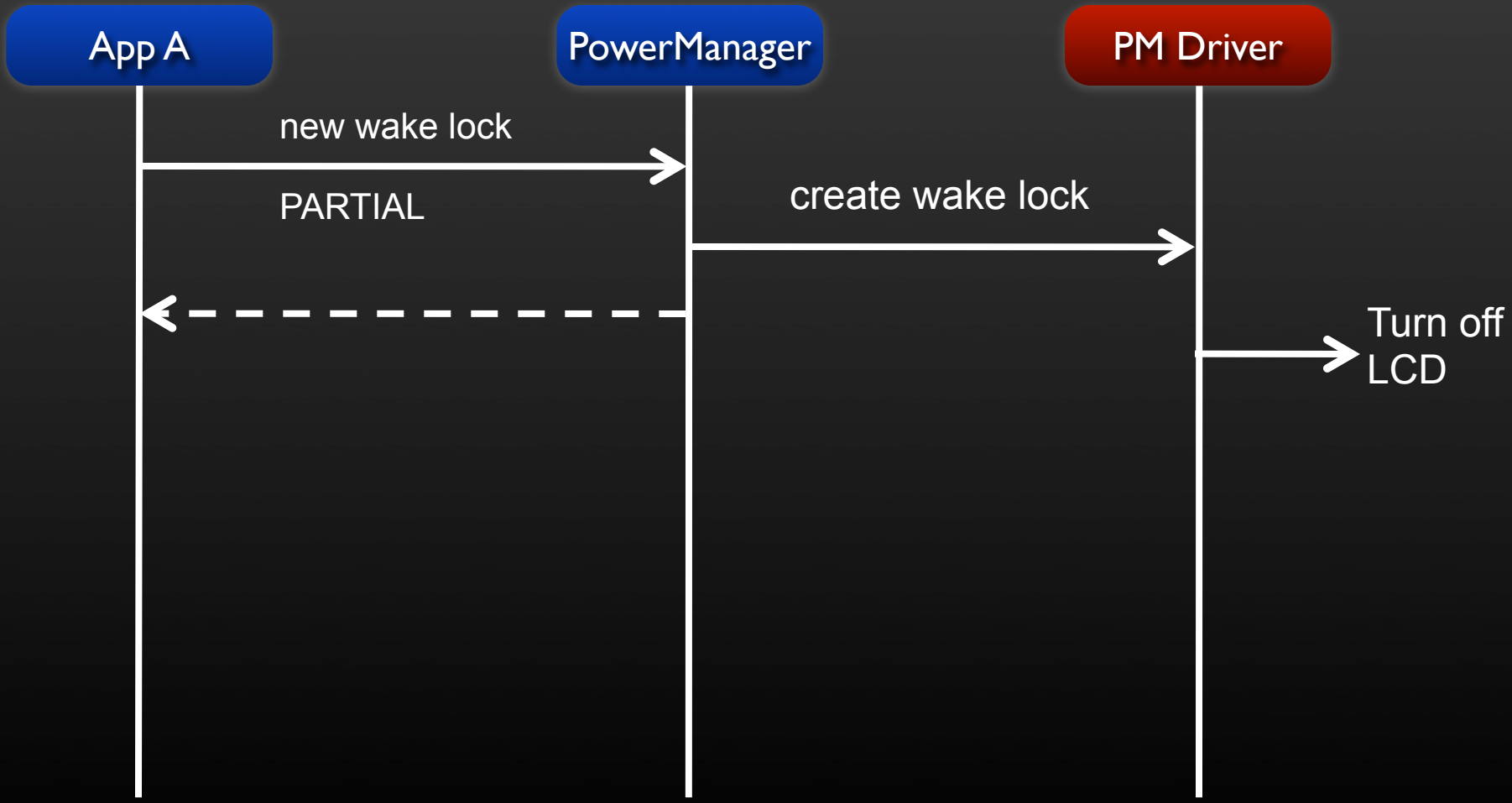
Android PM in Action

Android PM in Action



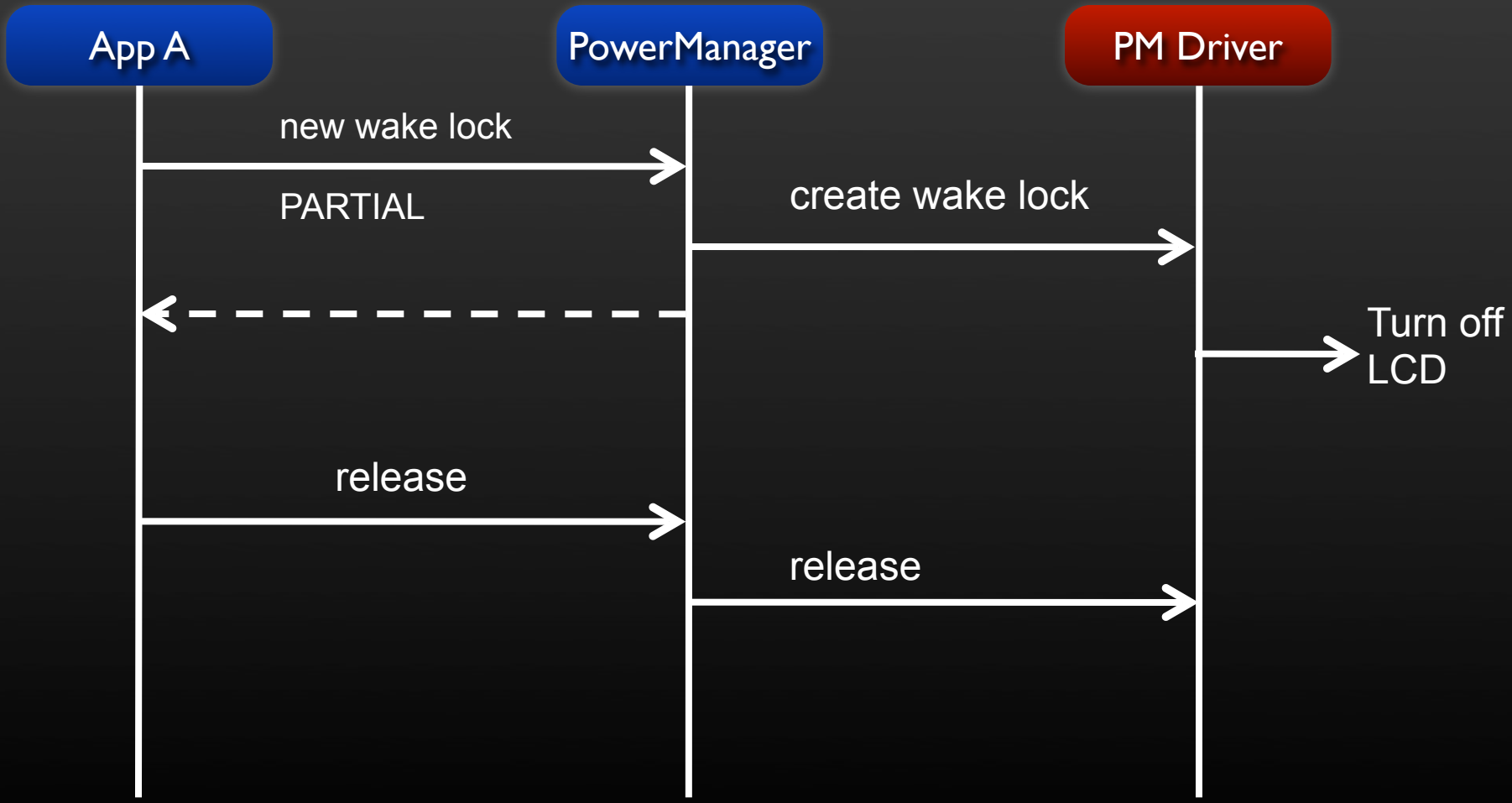
android

Android PM in Action

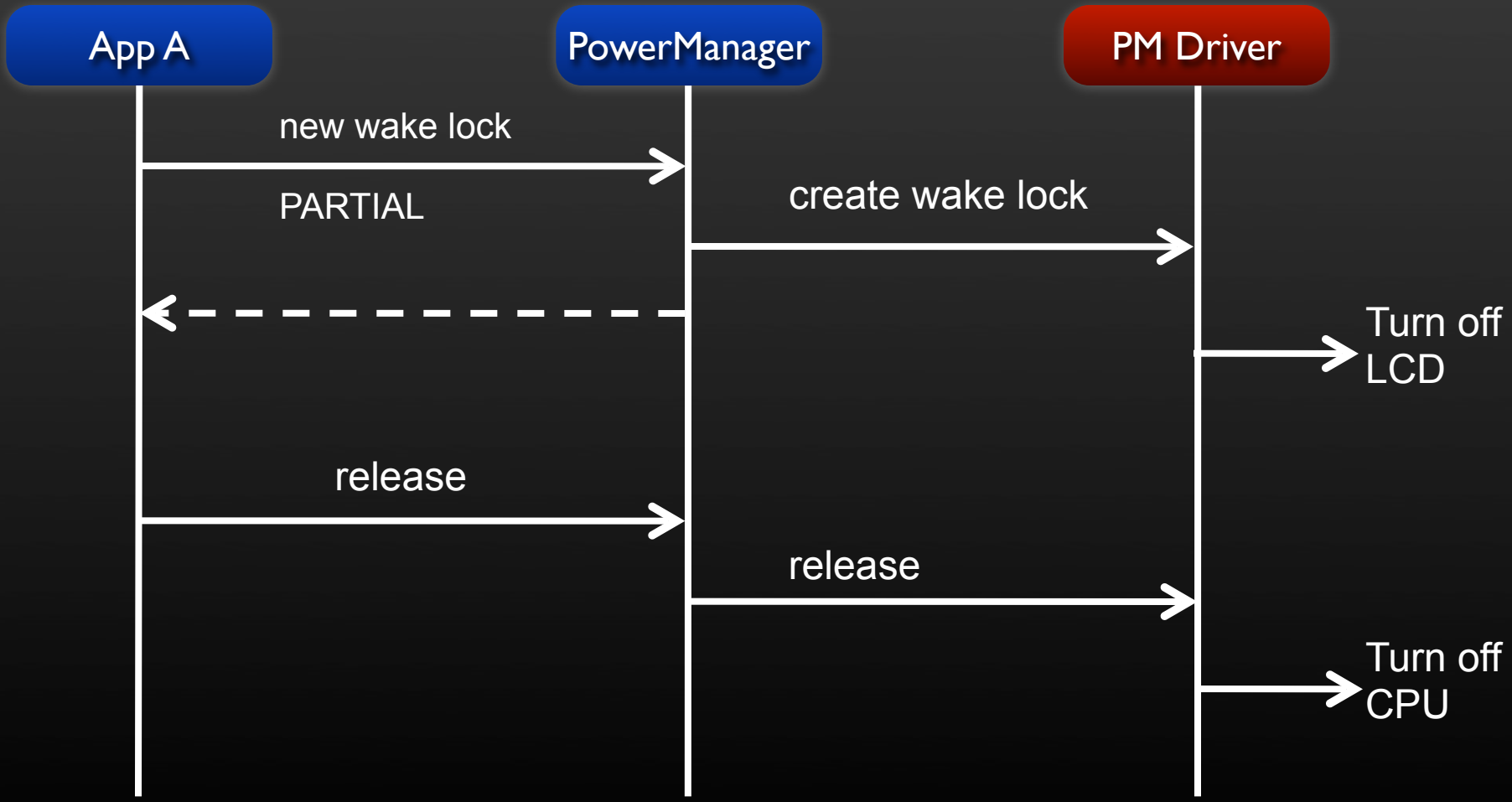


android

Android PM in Action



Android PM in Action



android

Android PM

android.os.PowerManager

- Use wake locks carefully!

- `userActivity(long when, …);`

android

Kernel

The Android kernel source is available today at:

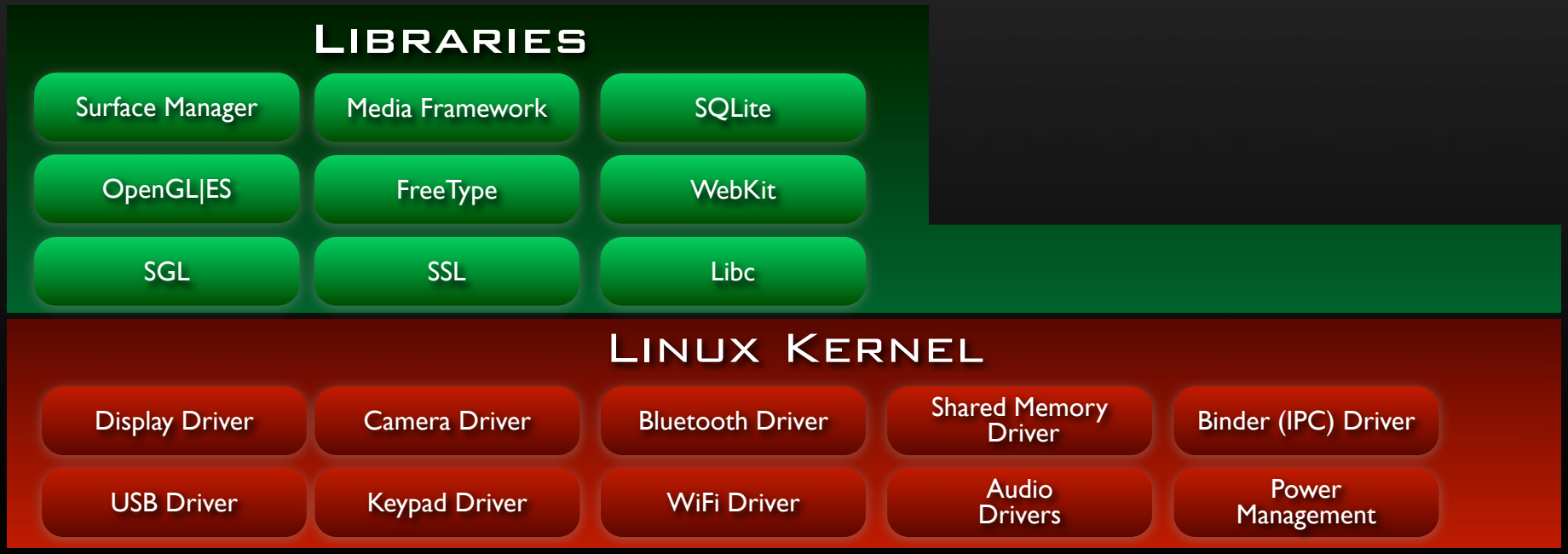
http://git.android.com

android

Agenda

- Android Anatomy
  - Linux Kernel
  - Native Libraries
  - Android Runtime
  - Application Framework
- Android Physiology
  - Start-up Walkthrough
  - Layer Interaction

Android Anatomy



Native Libraries

- Bionic Libc
- Function Libraries
- Native Servers
- Hardware Abstraction Libraries

android

Native Libraries

- Bionic Libc
- Function Libraries
- Native Servers
- Hardware Abstraction Libraries

android

What is Bionic?

- What is bionic?
  - Custom libc implementation, optimized for embedded use.

android

Why Bionic?

Why build a custom libc library?

- License: we want to keep GPL out of user-space
- Size: will load in each process, so it needs to be small
- Fast: limited CPU power means we need to be fast

android

Bionic libc

- BSD License
- Small size and fast code paths
- Very fast and small custom pthread implementation

android

Bionic libc

- Built-in support for important Android-specific services
  - system properties

```
getprop("my.system.property", buff, default);
```

  - log capabilities

```
LOGI("Logging a message with priority 'Info'");
```

android

Bionic libc

- Doesn't support certain POSIX features
- Not compatible with Gnu Libc (glibc)
- All native code must be compiled against bionic

android

Native Libraries

- Bionic Libc
- Function Libraries
- Native Servers
- Hardware Abstraction Libraries

android

WebKit

- Based on open source WebKit browser: http://webkit.org
- Renders pages in full (desktop) view
- Full CSS, Javascript, DOM, AJAX support
- Support for single-column and adaptive view rendering

android

Media Framework

- Based on PacketVideo OpenCORE platform
- Supports standard video, audio, still-frame formats
- Support for hardware / software codec plug-ins

android

SQLite

- Light-weight transactional data store
- Back end for most platform data storage

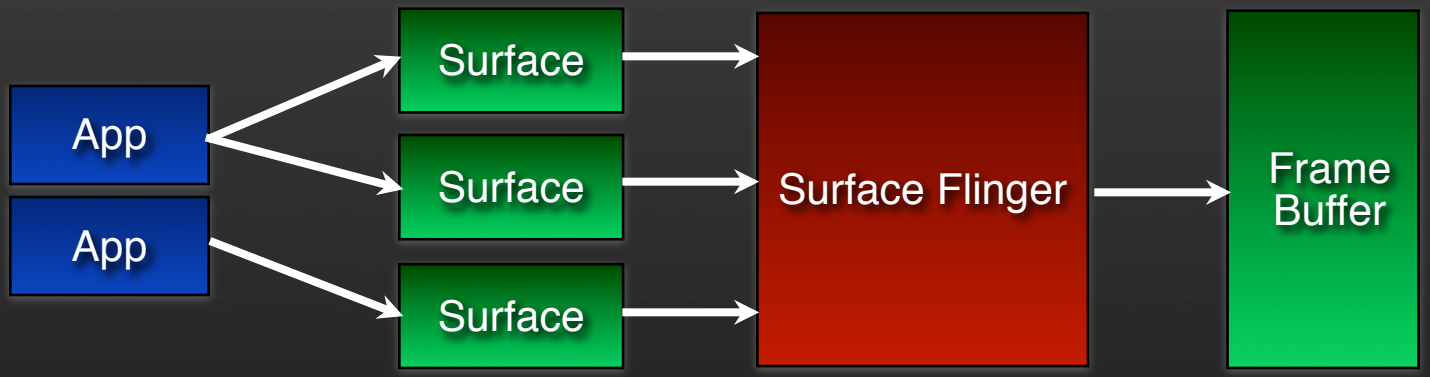
android

Native Libraries

- Bionic Libc
- Function Libraries
- Native Servers
- Hardware Abstraction Libraries

android

Surface Flinger



- Provides system-wide surface “composer”, handling all surface rendering to frame buffer device
- Can combine 2D and 3D surfaces and surfaces from multiple applications

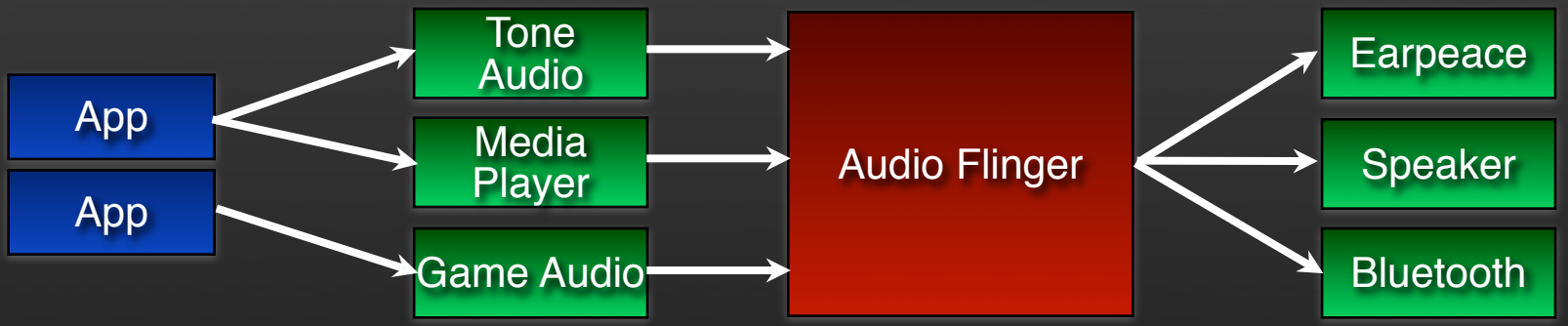
android

Surface Flinger

- Surfaces passed as buffers via Binder IPC calls
- Can use OpenGL ES and 2D hardware accelerator for its compositions
- Double-buffering using page-flip

android

Audio Flinger



- Manages all audio output devices
- Processes multiple audio streams into PCM audio out paths
- Handles audio routing to various outputs

android

Native Libraries

- Bionic Libc
- Function Libraries
- Native Servers
- Hardware Abstraction Libraries

ANDROID

Hardware Abstraction Layer

Applications

Home | Dialer | SMS/MMS | IM | Browser | Camera | Alarm | Calculator

Contacts | Voice Dial | Email | Calendar | Media Player | Photo Album | Clock | ...

Application Framework

Activity Manager | Window Manager | Content Providers | View System | Notification Manager

Package Manager | Telephony Manager | Resource Manager | Location Manager | ...

Libraries

Surface Manager | Media Framework | SQLite | WebKit | Libc

OpenGL|ES | Audio Manager | FreeType | SSL | ...

Android Runtime

Core Libraries

Dalvik Virtual Machine

Hardware Abstraction Layer

Graphics | Audio | Camera | Bluetooth | GPS | Radio (RIL) | WiFi | ...

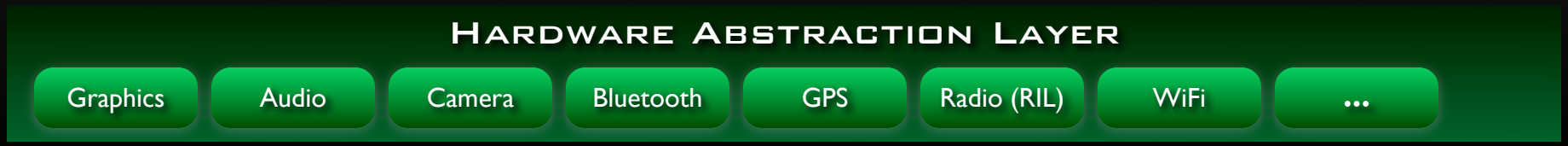
Linux Kernel

Display Driver | Camera Driver | Bluetooth Driver | Shared Memory Driver | Binder (IPC) Driver

USB Driver | Keypad Driver | WiFi Driver | Audio Drivers | Power Management

Hardware Abstraction Libraries

- 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

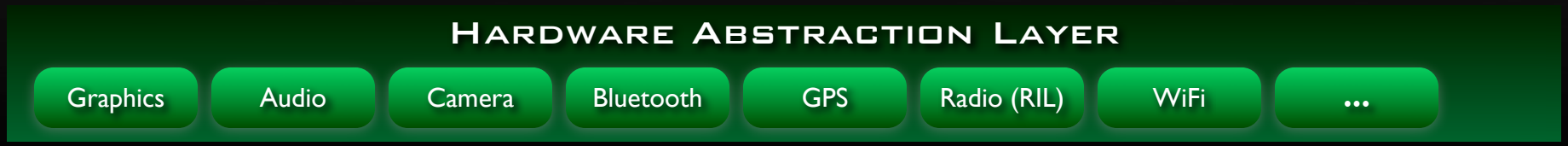


android

Hardware Abstraction Libraries

Why do we need a user-space HAL?

- Not all components have standardized kernel driver interfaces
- Kernel drivers are GPL which exposes any proprietary IP
- Android has specific requirements for hardware drivers



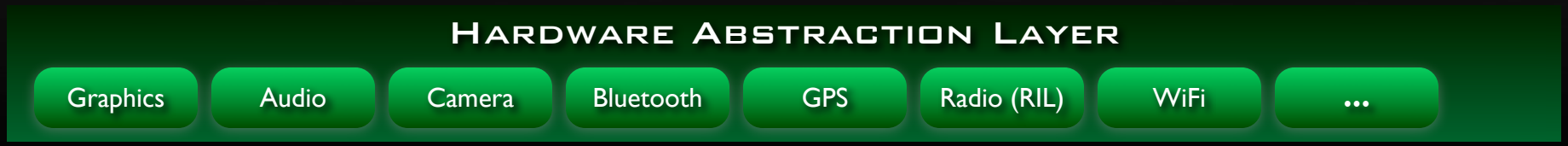
android

HAL Header Example

```
// must be provided by each Acme hardware implementation
typedef struct {
    int   (*foo)( void );
   char  (*bar)( void );
   …

} AcmeFunctions;


const AcmeFunctions *Acme_Init(const struct Env *env, int argc, char
   **argv);
```



android

Hardware Abstraction Libraries

- Libraries are loaded dynamically at runtime as needed

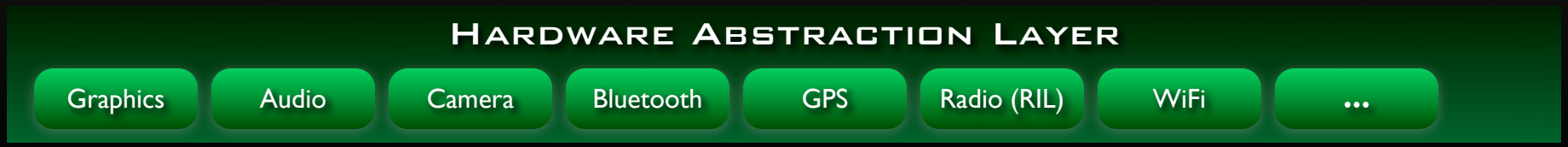
```
dlHandle = dlopen("/system/lib/libacme.so", RTLD_NOW);

...

acmeInit = (const AcmeFunctions *(*)(const struct Env *,
   int, char **))dlsym(dlHandle, "Acme_Init");

...

acmeFuncs = acmeInit(&env, argc, argv);
```

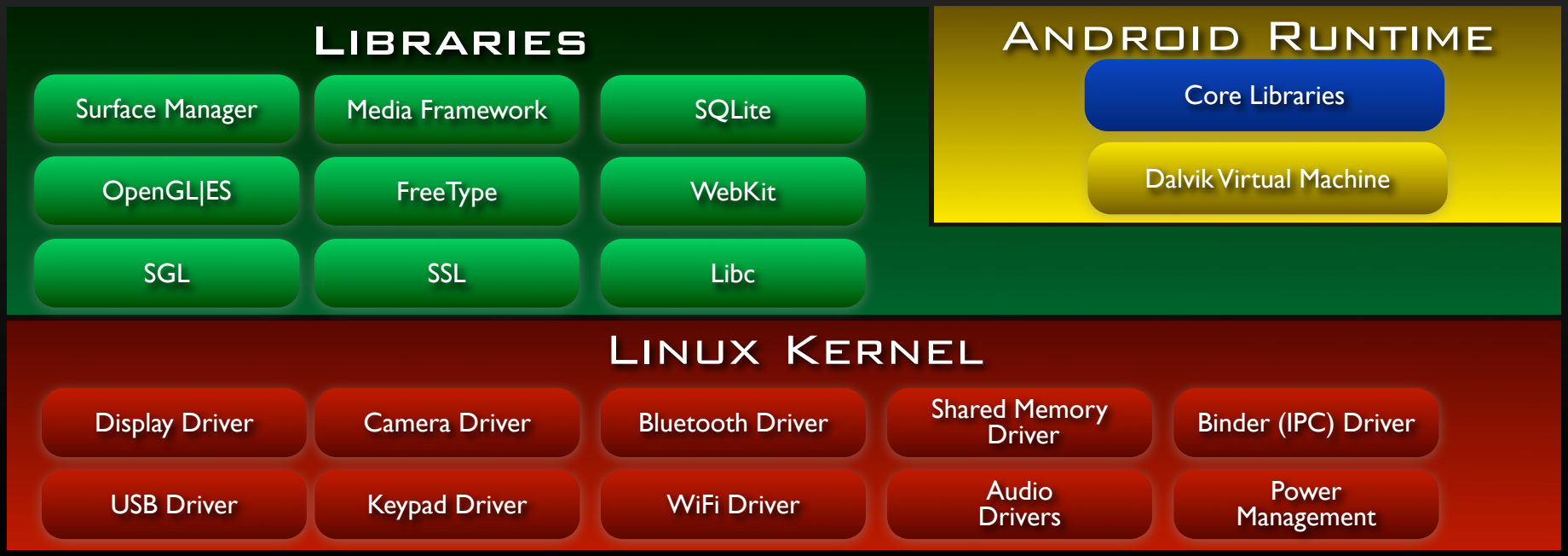


android

Agenda

- Android Anatomy
  - Linux Kernel
  - Native Libraries
  - Android Runtime
  - Application Framework
- Android Physiology
  - Start-up Walkthrough
  - Layer Interaction

Android Anatomy



Dalvik Virtual Machine

- Android's custom clean-room implementation virtual machine
  - Provides application portability and runtime consistency
  - Runs optimized file format (.dex) and Dalvik bytecode
  - Java .class / .jar files converted to .dex at build time

android

Dalvik Virtual Machine

- Designed for embedded environment
  - Supports multiple virtual machine processes per device
  - Highly CPU-optimized bytecode interpreter
  - Uses runtime memory very efficiently

android

Core Libraries

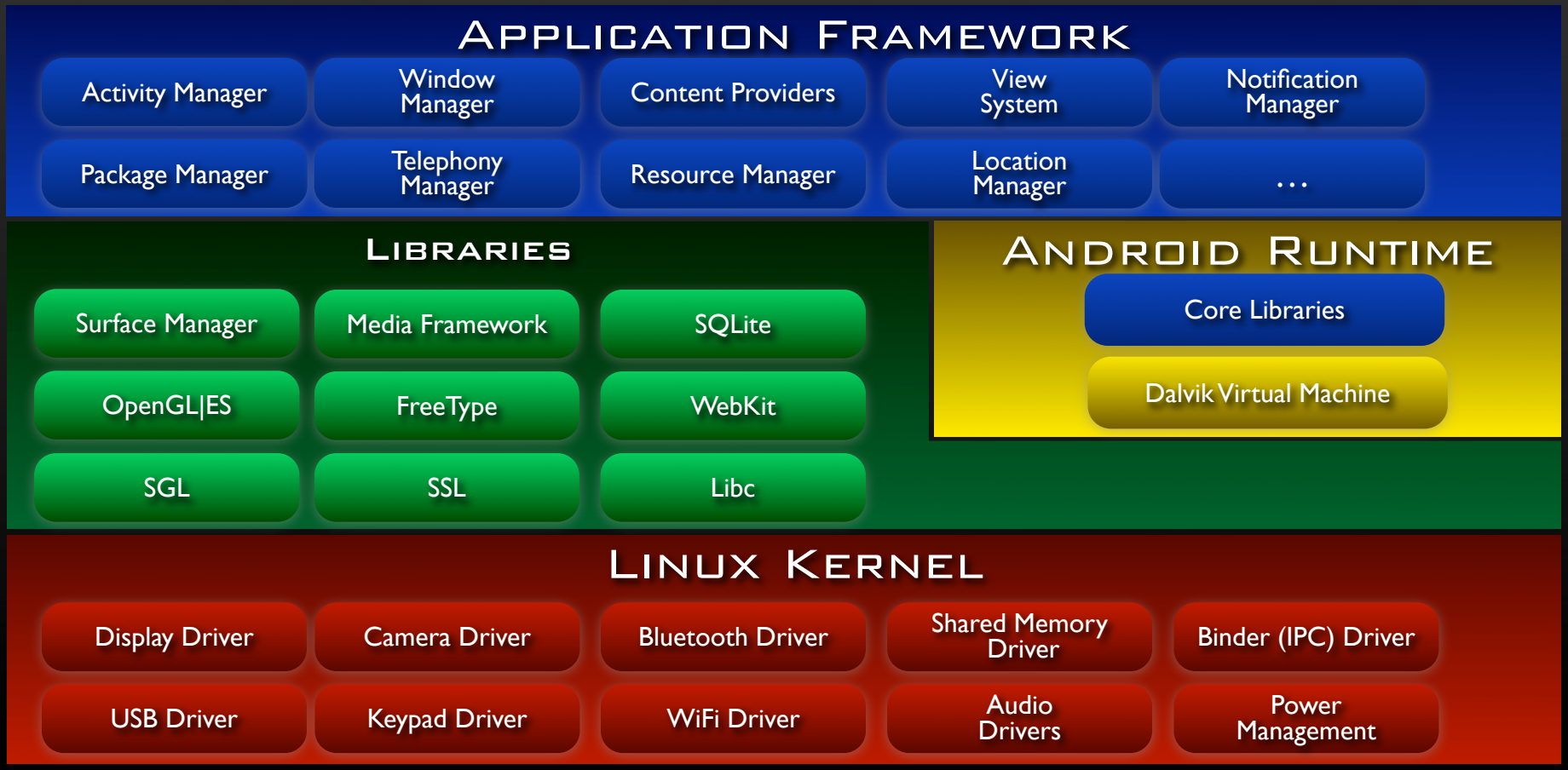
- Core APIs for Java language provide a powerful, yet simple and familiar development platform
  - Data structures
  - Utilities
  - File access
  - Network Access
  - Graphics
  - …

android

Agenda

- Android Anatomy
  - Linux Kernel
  - Native Libraries
  - Android Runtime
  - Application Framework
- Android Physiology
  - Start-up Walkthrough
  - Layer Interaction

Android Anatomy



Core Platform Services

- Services that are essential to the Android platform
- Behind the scenes - applications typically don't access them directly

android

Core Platform Services

- Activity Manager

android

Core Platform Services

- Activity Manager
- Package Manager

android

Core Platform Services

- Activity Manager
- Package Manager
- Window Manager

android

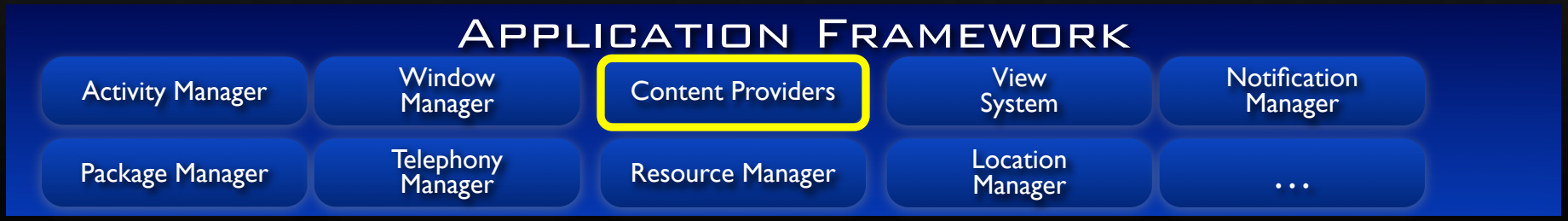
Core Platform Services

- Activity Manager
- Package Manager
- Window Manager
- Resource Manager

android

Core Platform Services

- Activity Manager
- Package Manager
- Window Manager
- Resource Manager
- Content Providers



android

Core Platform Services

- Activity Manager
- Package Manager
- Window Manager
- Resource Manager
- Content Providers
- View System

android

Hardware Services

- Provide access to lower-level hardware APIs

android

Hardware Services

- Provide access to lower-level hardware APIs
- Typically accessed through local *Manager* object

```
LocationManager lm = (LocationManager)
   Context.getSystemService(Context.LOCATION_SERVICE);
```

android

Hardware Services

- Telephony Service

android

Hardware Services

- Telephony Service
- Location Service

android

Hardware Services

- Telephony Service
- Location Service
- Bluetooth Service

android

Hardware Services

- Telephony Service
- Location Service
- Bluetooth Service
- WiFi Service

android

Hardware Services

- Telephony Service
- Location Service
- Bluetooth Service
- WiFi Service
- USB Service

android

Hardware Services

- Telephony Service
- Location Service
- Bluetooth Service
- WiFi Service
- USB Service
- Sensor Service

android

Application Framework

More Information

- At Google I/O
  - “Inside the Android Application Framework”
- Online
  - http://code.google.com/android

android

Android Anatomy

Applications

Home | Dialer | SMS/MMS | IM | Browser | Camera | Alarm | Calculator

Contacts | Voice Dial | Email | Calendar | Media Player | Albums | Clock | ...

Application Framework

Activity Manager | Window Manager | Content Providers | View System | Notification Manager

Package Manager | Telephony Manager | Resource Manager | Location Manager | ...

Libraries

Surface Manager | Media Framework | SQLite

OpenGL|ES | FreeType | WebKit

SGL | SSL | Libc

Android Runtime

Core Libraries

Dalvik Virtual Machine

Linux Kernel

Display Driver | Camera Driver | Bluetooth Driver | Shared Memory Driver | Binder (IPC) Driver

USB Driver | Keypad Driver | WiFi Driver | Audio Drivers | Power Management

Agenda

- Android Anatomy
  - Linux Kernel
  - Native Libraries
  - Application Framework
- Android Physiology
  - Start-up Walkthrough
  - Layer Interaction

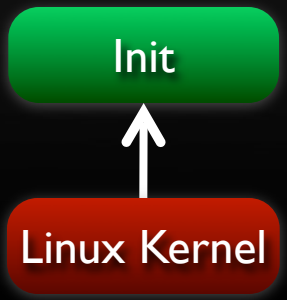
Agenda

- Android Anatomy
  - Linux Kernel
  - Native Libraries
  - Application Framework
- Android Physiology
  - Start-up Walkthrough
  - Layer Interaction

Runtime Walkthrough

It all starts with init…

Similar to most Linux-based systems at startup, the bootloader loads the Linux kernel and starts the init process.

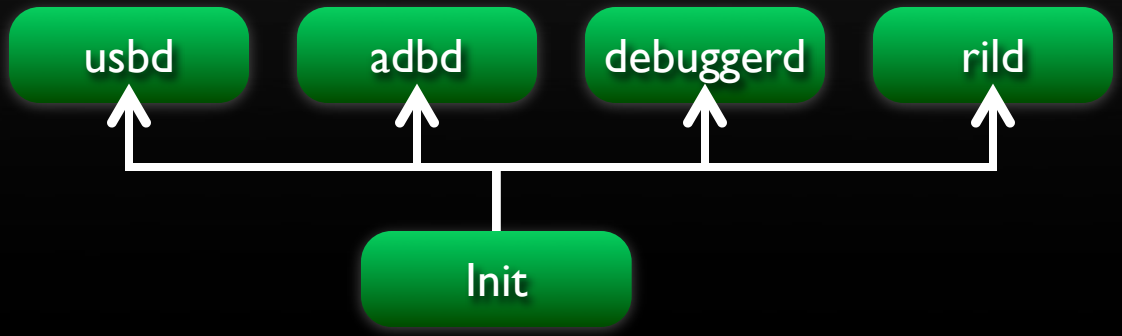


ANDROID

Runtime Walkthrough

Init starts Linux daemons, including:

- USB Daemon (usbd) to manage USB connections
- Android Debug Bridge (adbd) to manage ADB connections
- Debugger Daemon (debuggerd) to manage debug processes requests (dump memory, etc.)
- Radio Interface Layer Daemon (rild) to manage communication with the radio

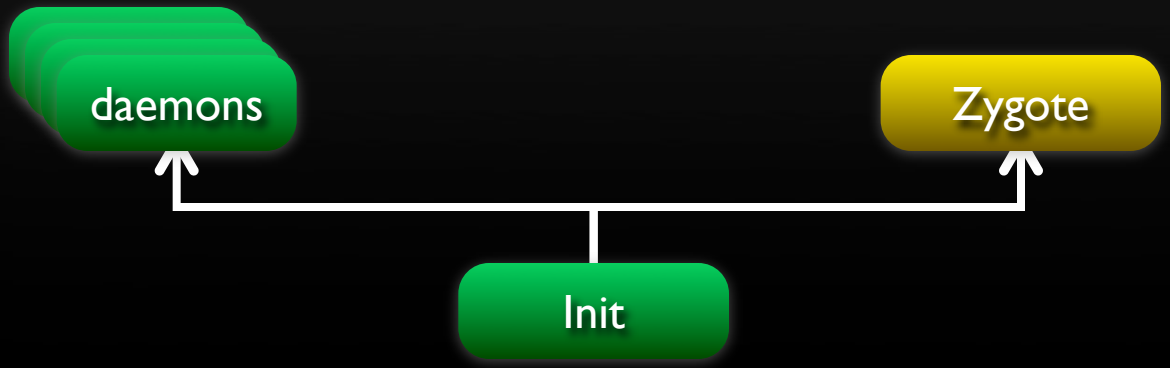


ANDROID

Runtime Walkthrough

Init process starts the zygote process:

- A nascent process which initializes a Dalvik VM instance
- Loads classes and listens on socket for requests to spawn VMs
- Forks on request to create VM instances for managed processes
- Copy-on-write to maximize re-use and minimize footprint

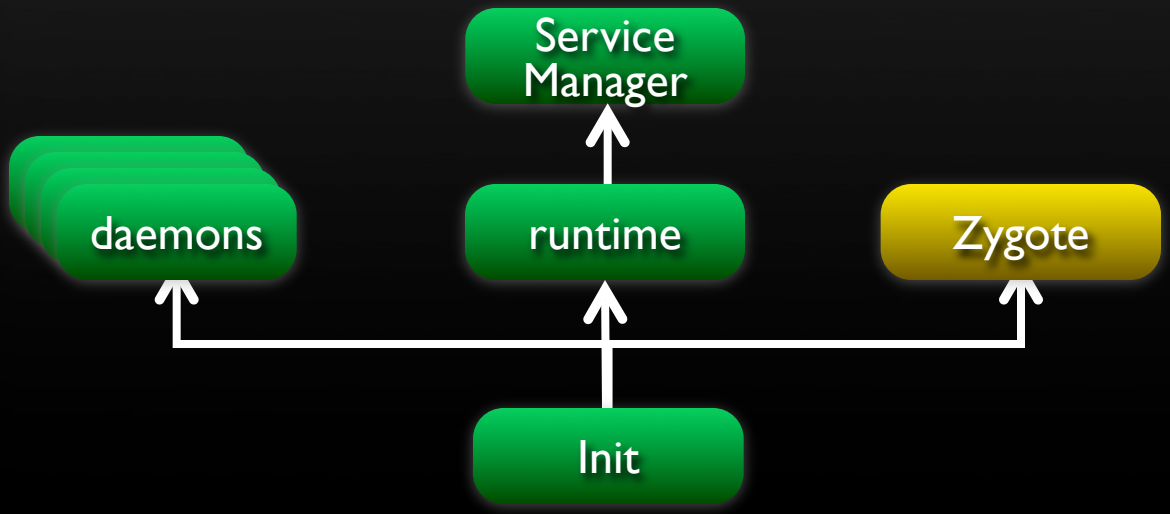


ANDROID

Runtime Walkthrough

Init starts runtime process:

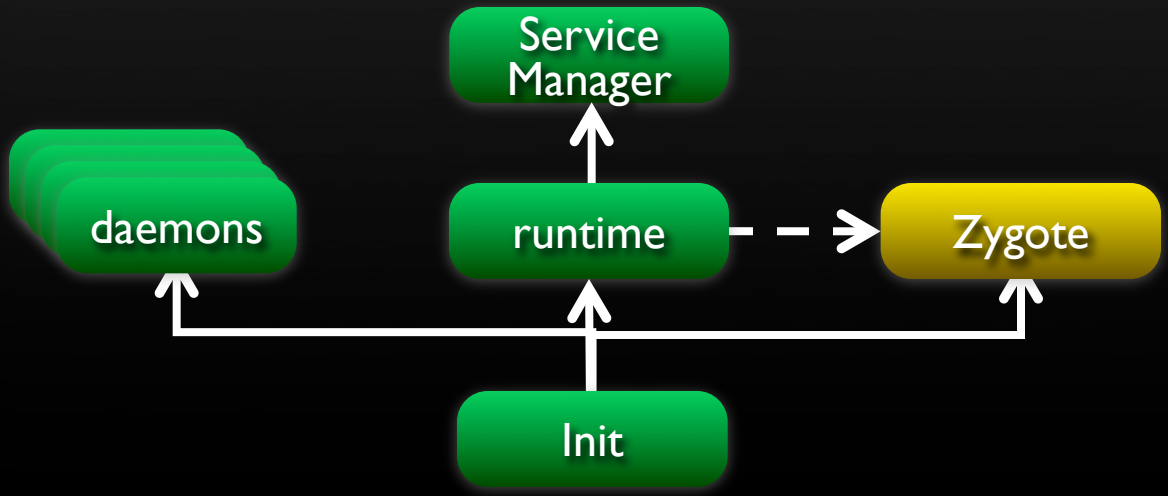
- Initializes Service Manager – the context manager for Binder that handles service registration and lookup
- Registers Service Manager as default context manager for Binder services



ANDROID

Runtime Walkthrough

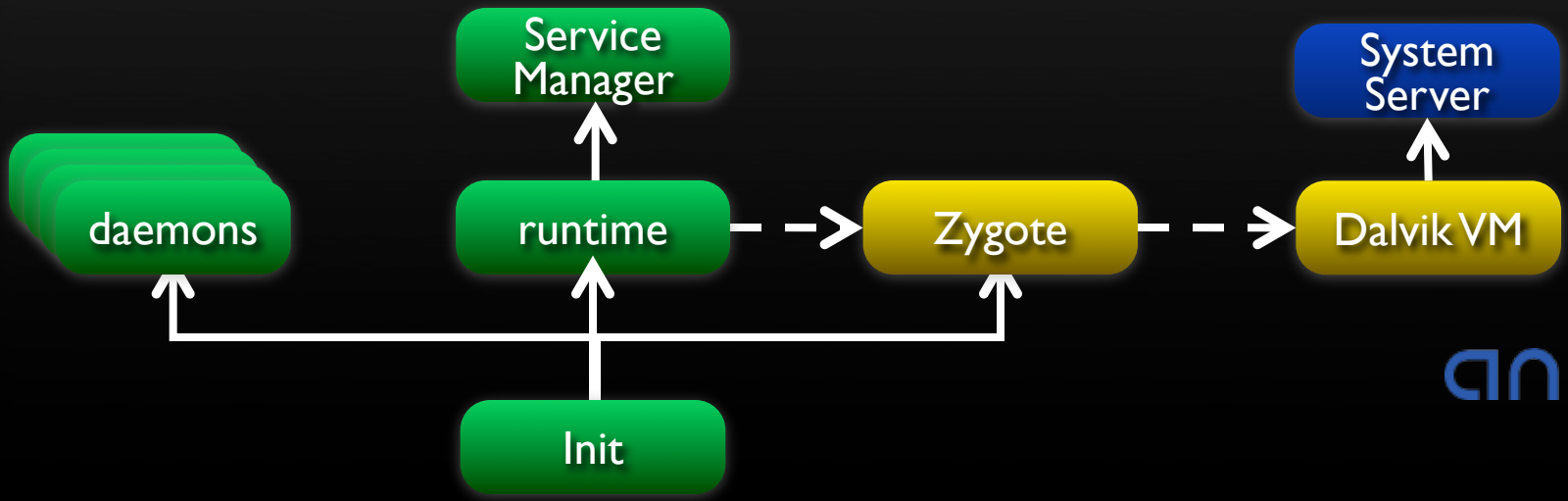
Runtime process sends request for Zygote to start System Service



Runtime Walkthrough

Runtime process sends request for Zygote to start System Server

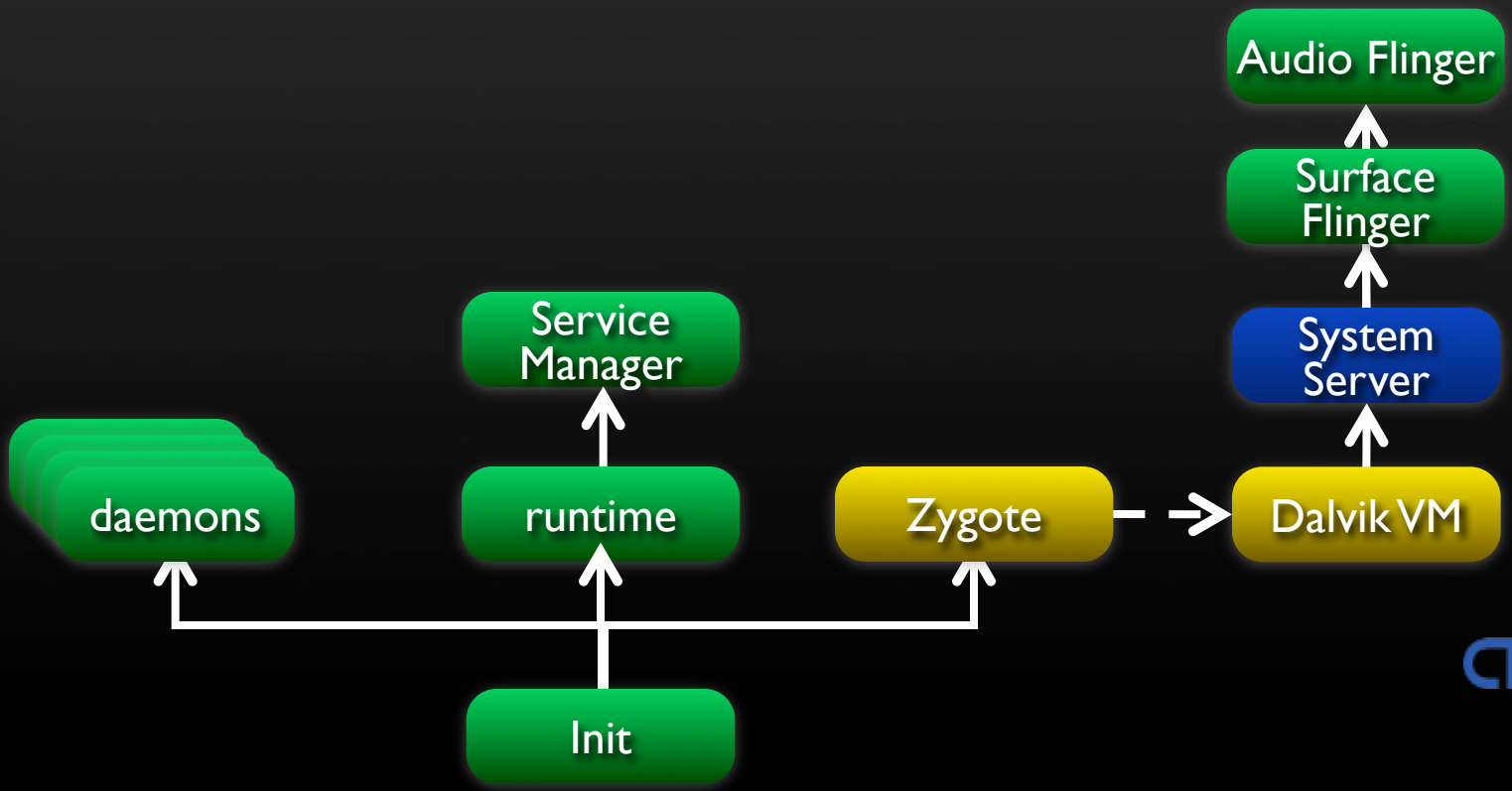
- Zygote forks a new VM instance for the System Service process and starts the service



Runtime Walkthrough

System Service starts the native system servers, including:

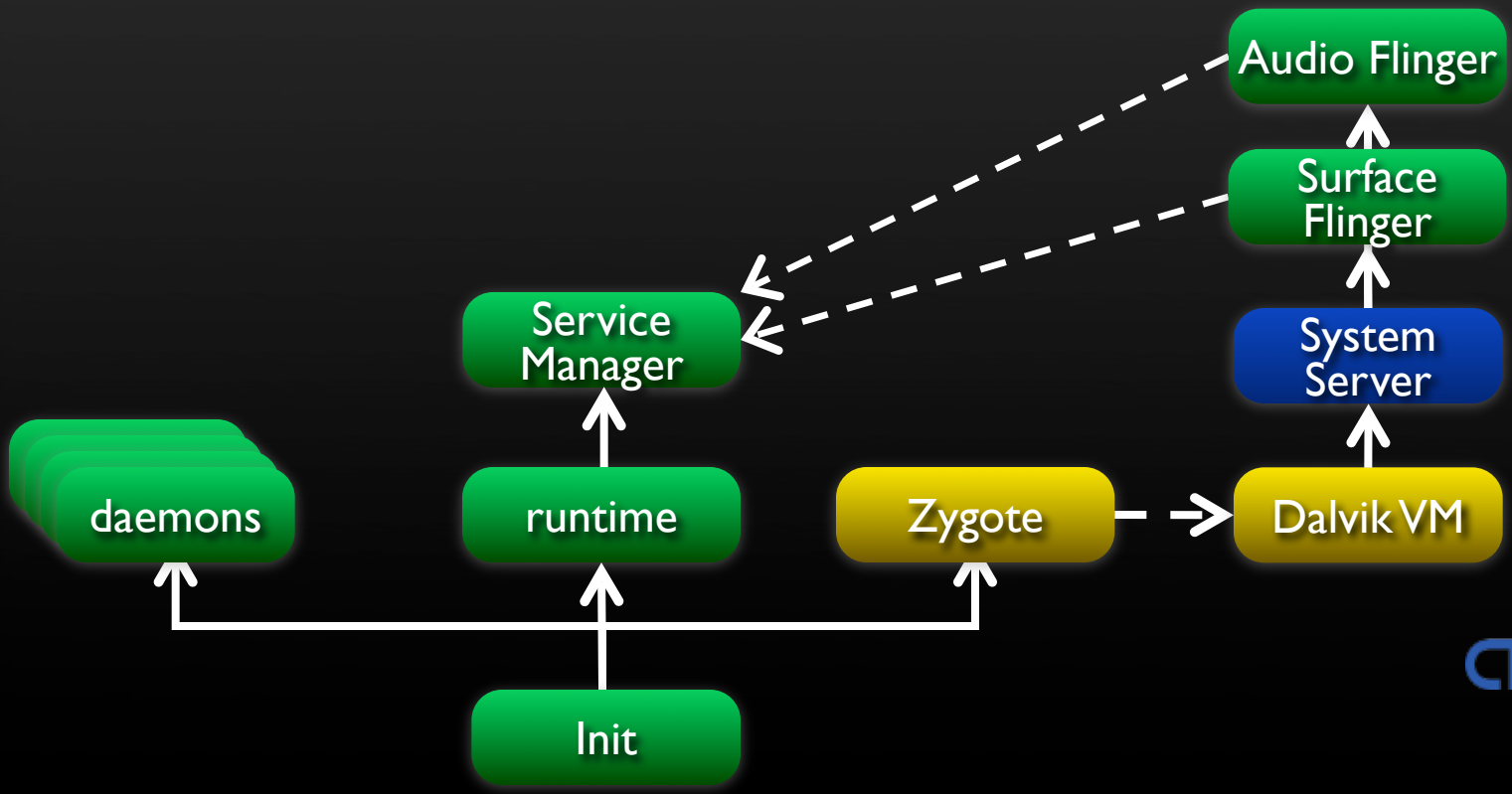
- Surface Flinger
- Audio Flinger



android

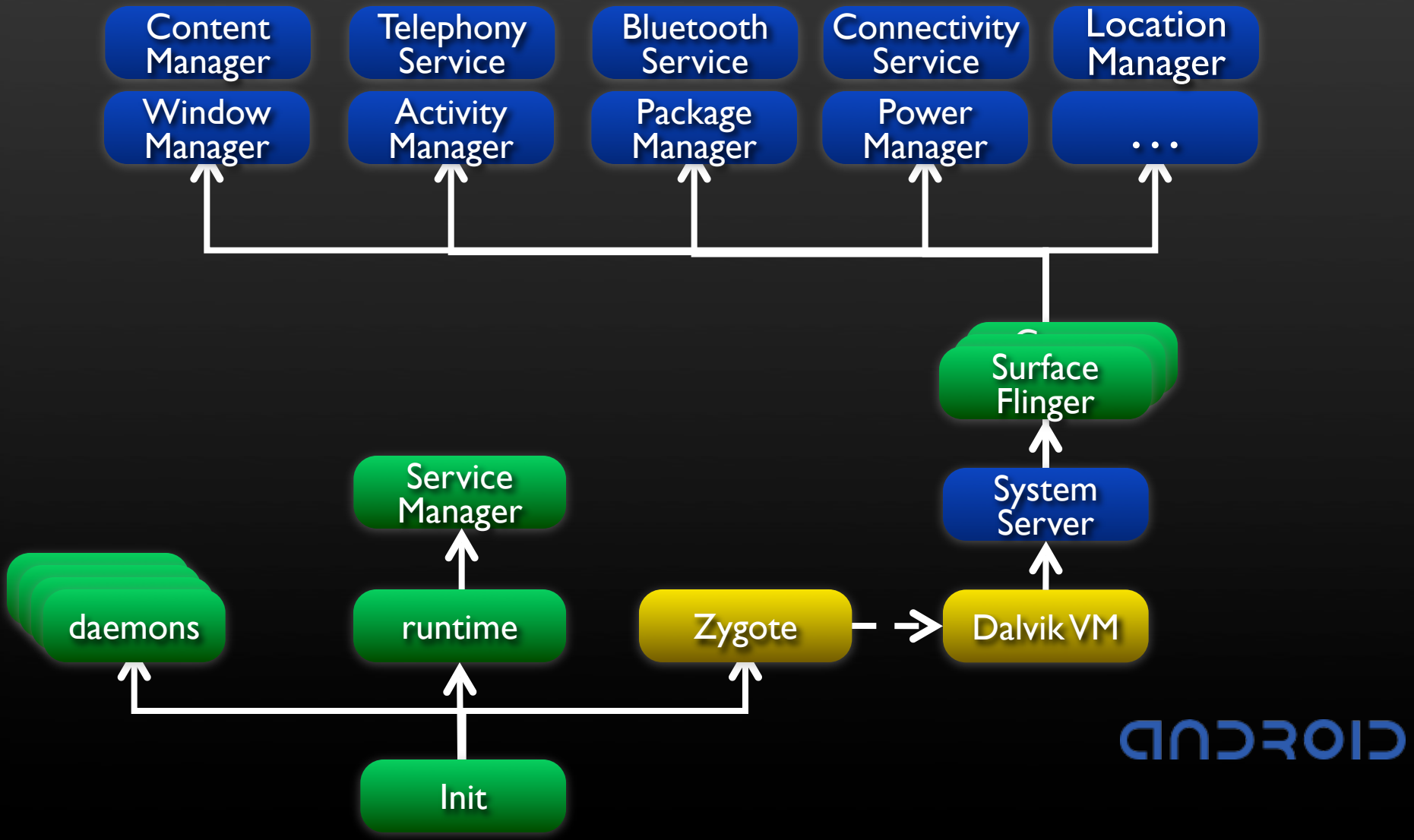
Runtime Walkthrough

Native system servers register with Service Manager as IPC service targets:



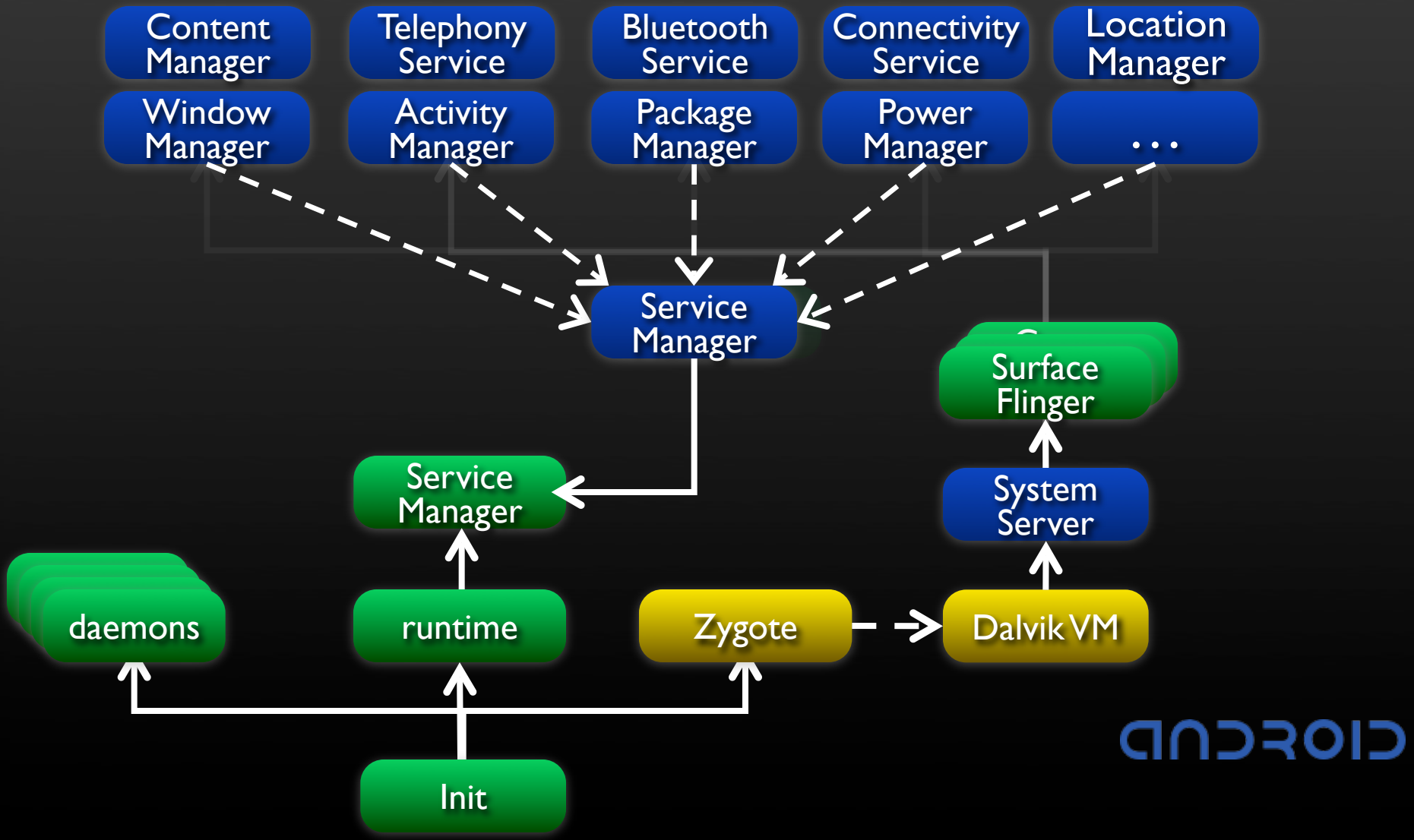
Runtime Walkthrough

System Service starts the Android managed services:

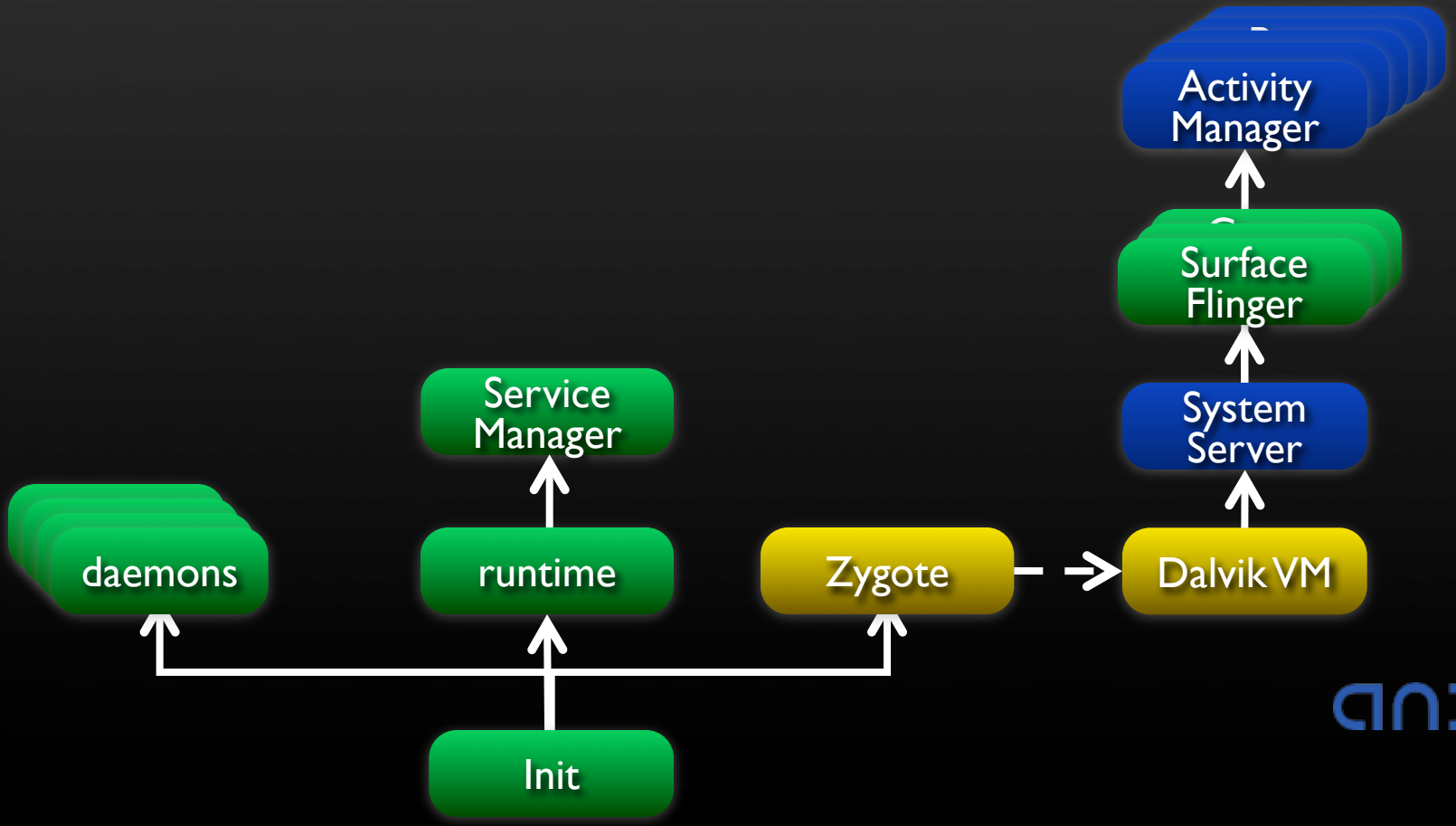


Runtime Walkthrough

Android managed Services register with Service Manager:



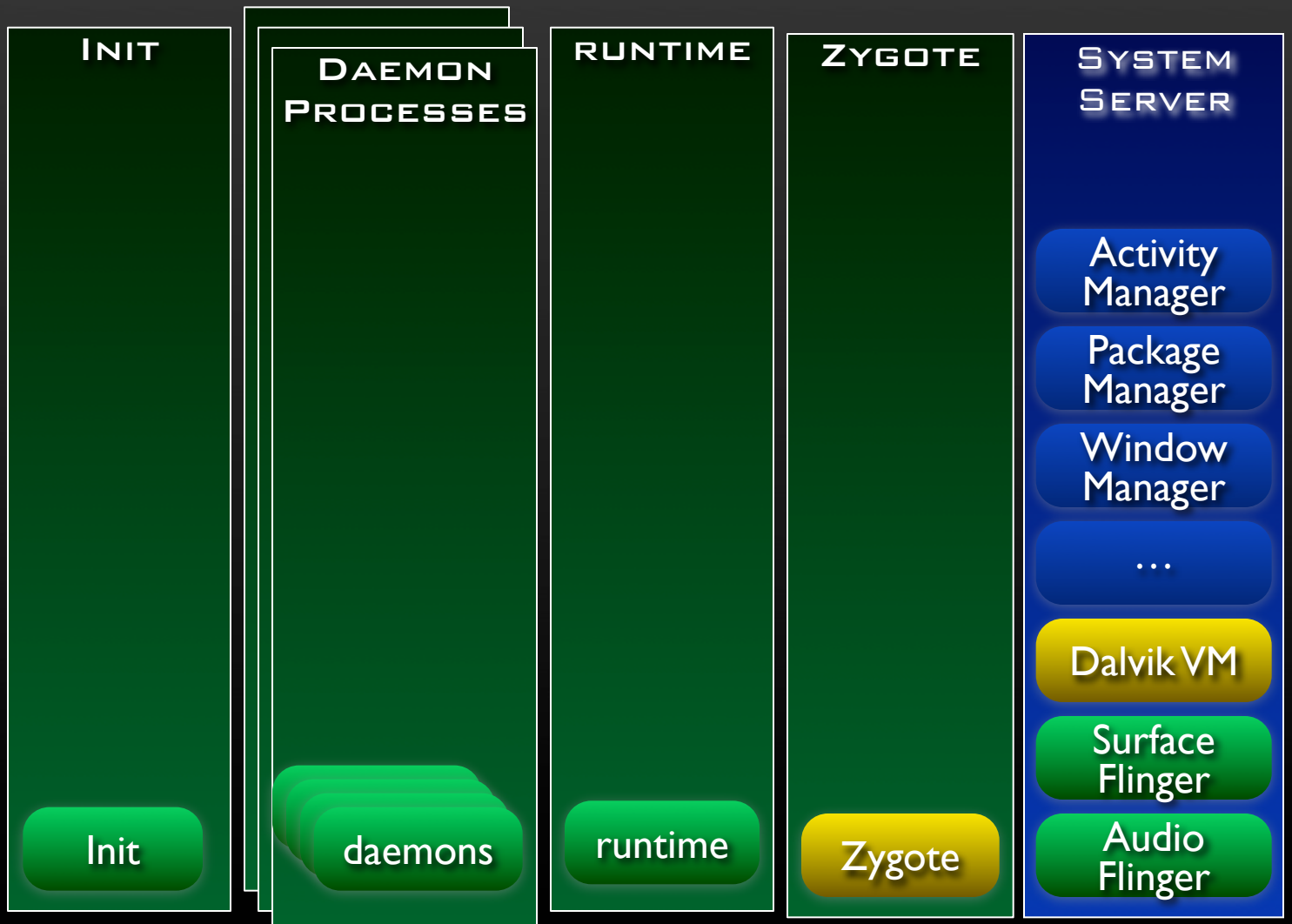
Runtime Walkthrough



android

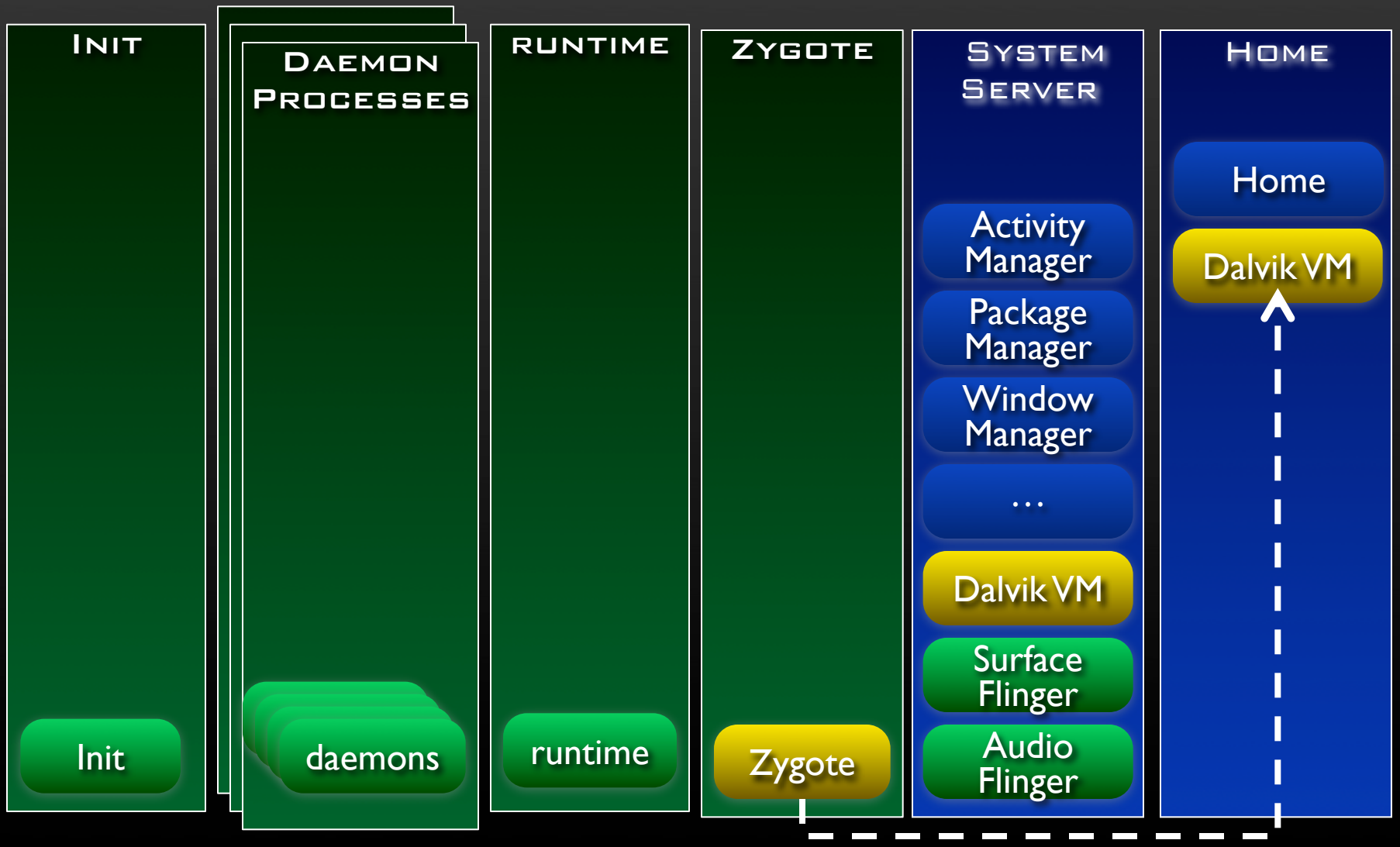
Runtime Walkthrough

After system server loads all services, the system is ready…



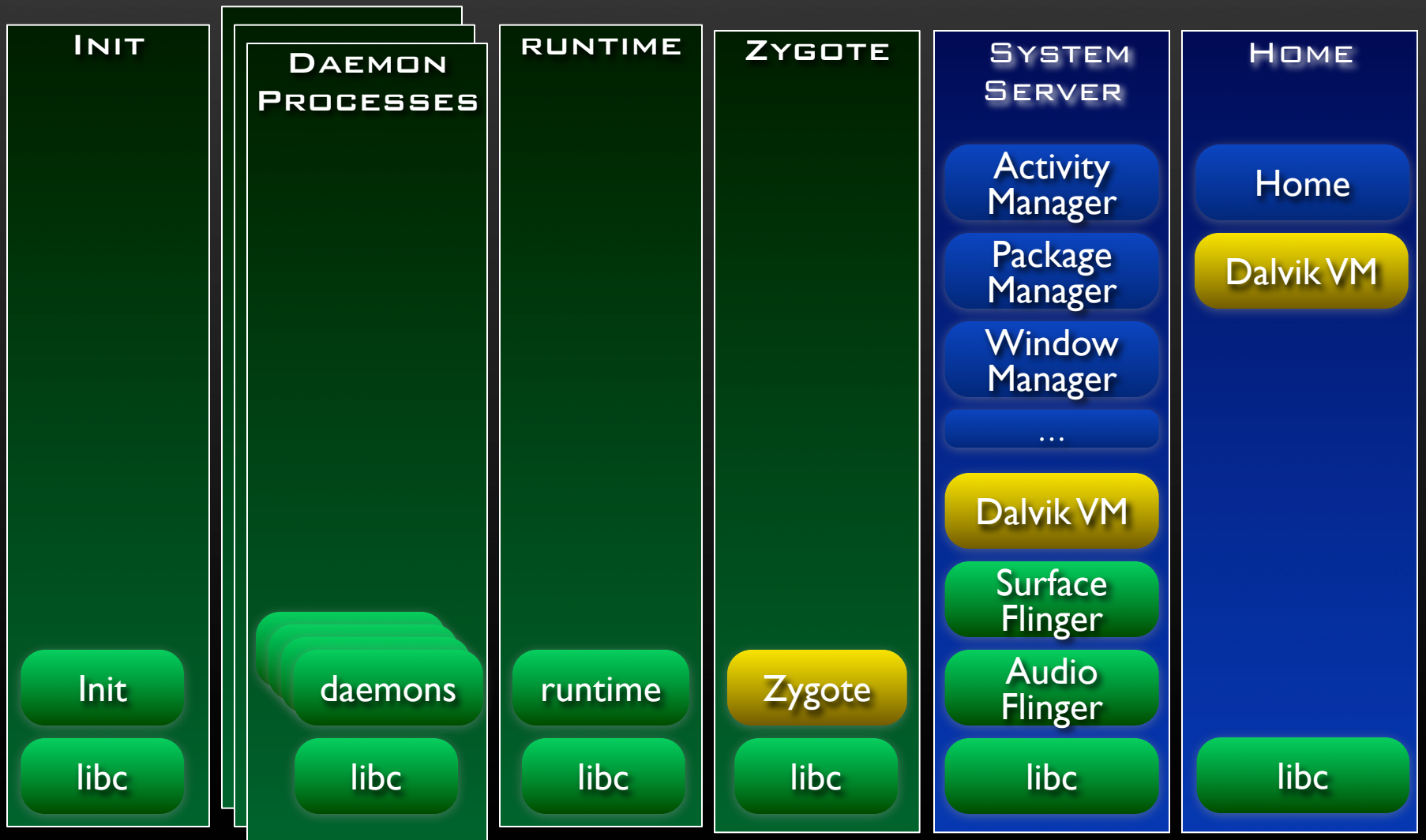
Runtime Walkthrough

After system server loads all services, the system is ready…



Runtime Walkthrough

After system server loads all services, the system is ready…



Runtime Walkthrough

Each subsequent application is launched in it's own process

Agenda

- Android Anatomy
  - Linux Kernel
  - Native Libraries
  - Framework Services
- Android Physiology
  - Start-up Walkthrough
  - Layer Interaction

Layer Interaction

There are 3 main flavors of Android layer cake:

- App → Runtime Service → lib
- App → Runtime Service → Native Service → lib
- App → Runtime Service → Native Daemon → lib

ANDROID

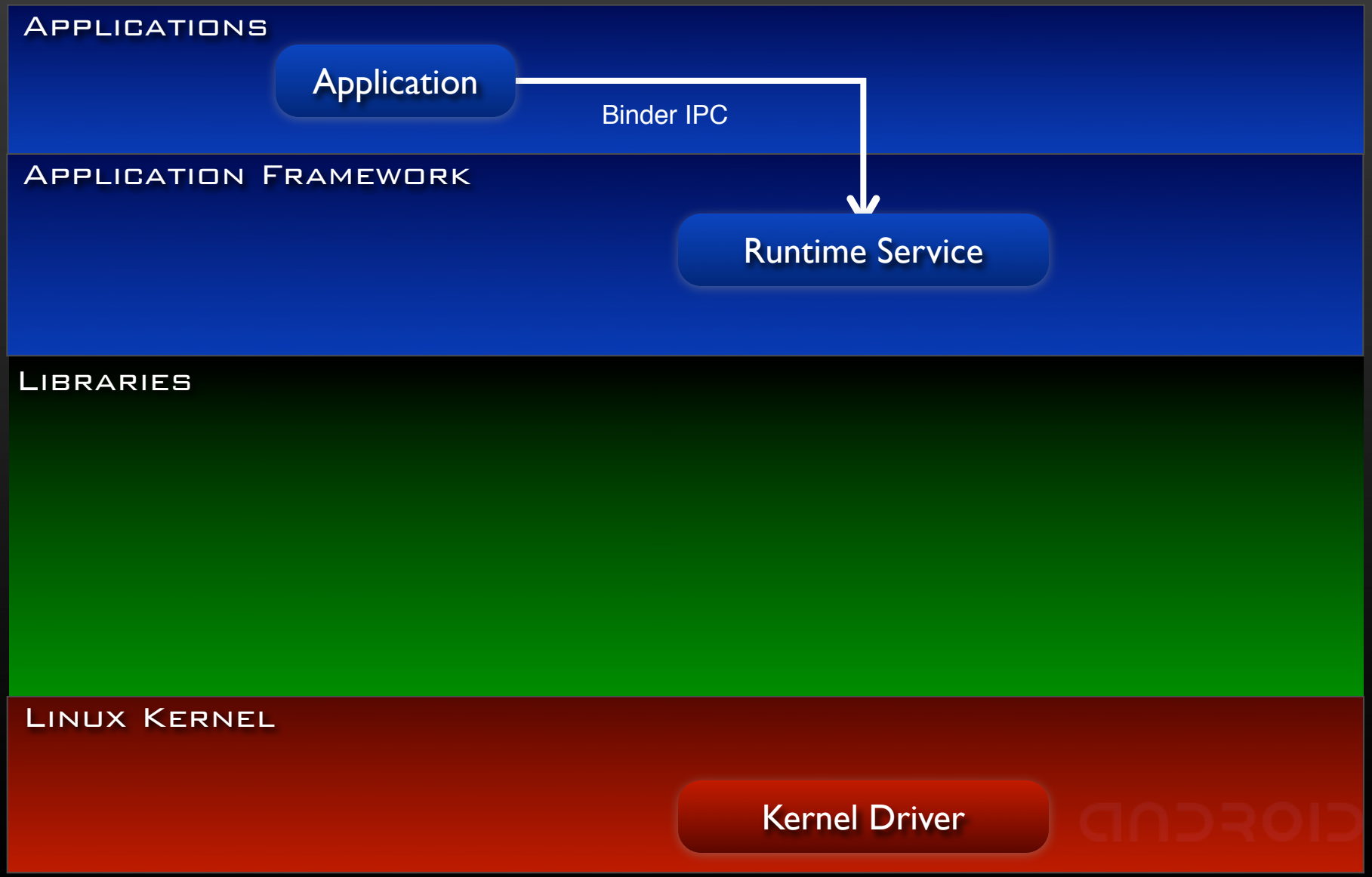
Layer Interaction

There are 3 main flavors of Android layer cake:

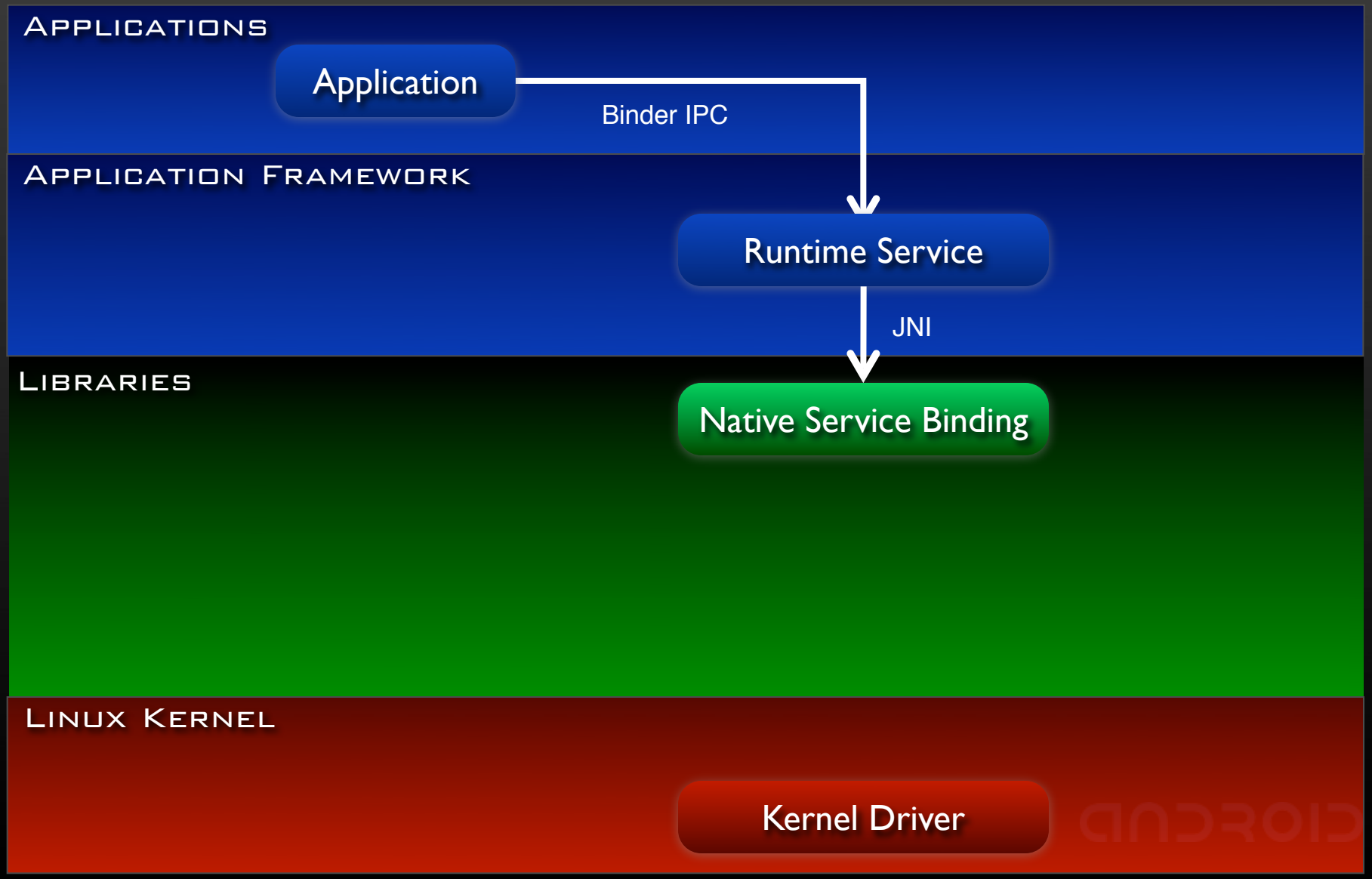
- App → Runtime Service → lib
- App → Runtime Service → Native Service → lib
- App → Runtime Service → Native Daemon → lib

android

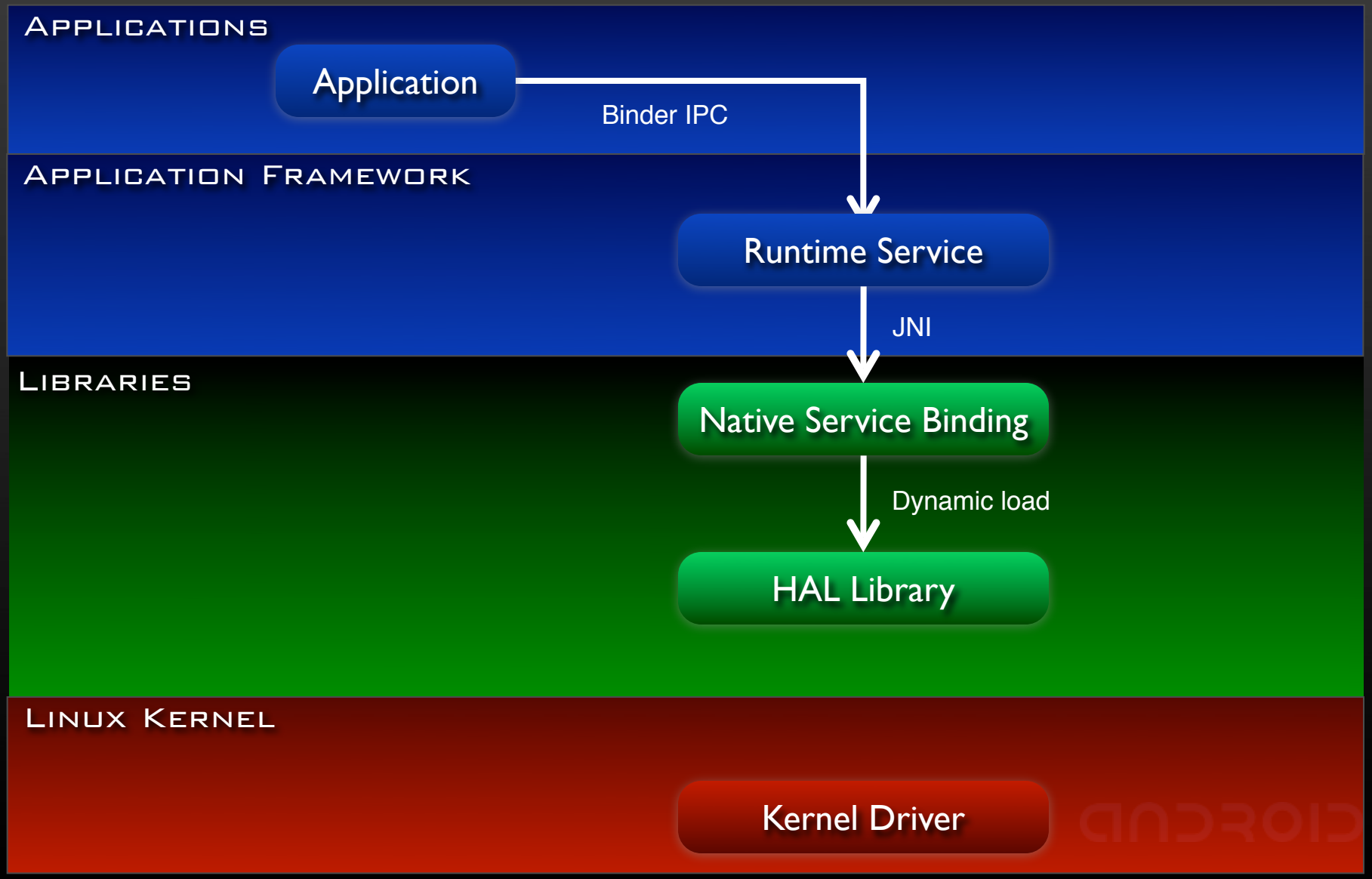
Android Runtime Services



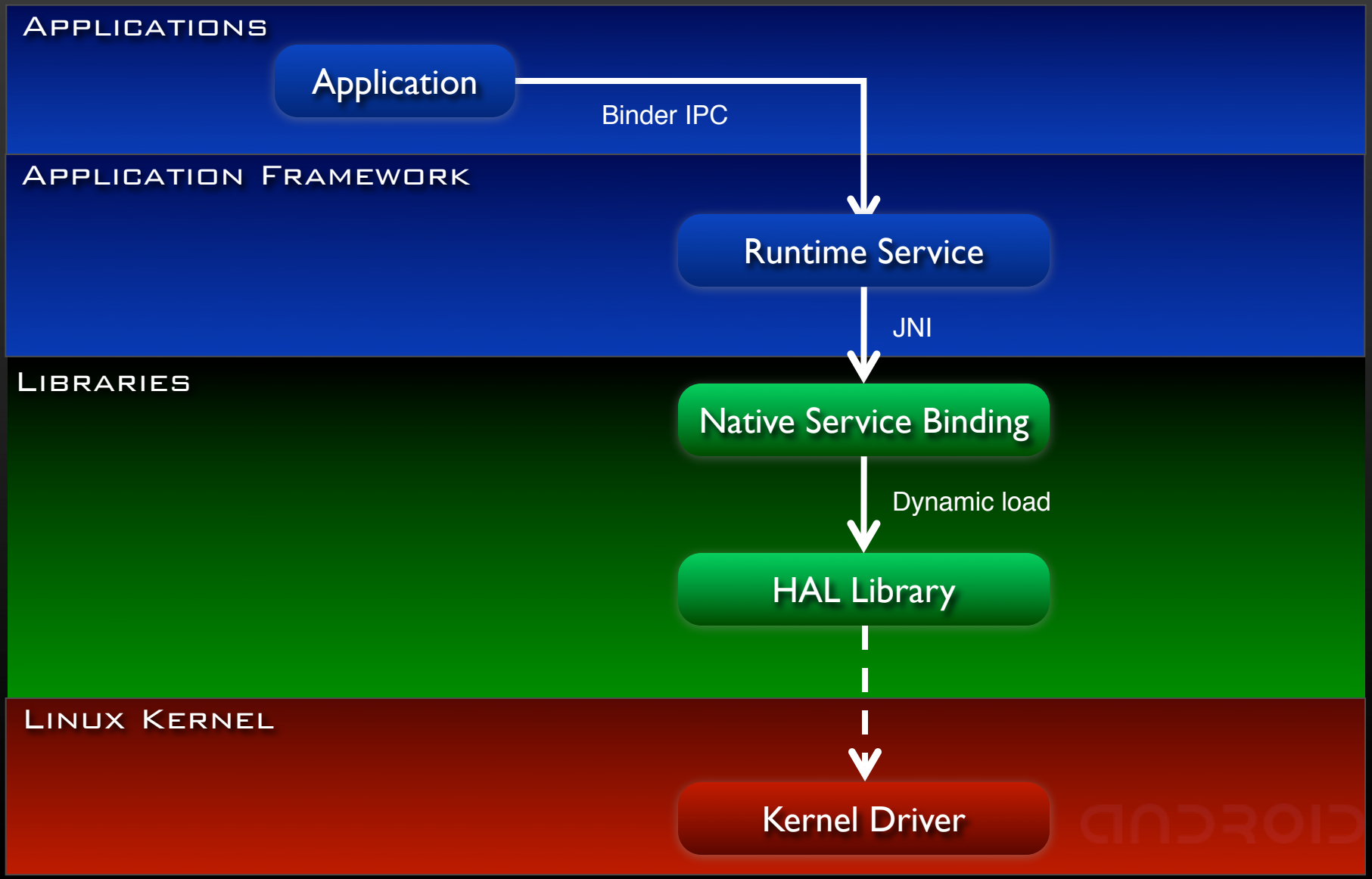
Android Runtime Services



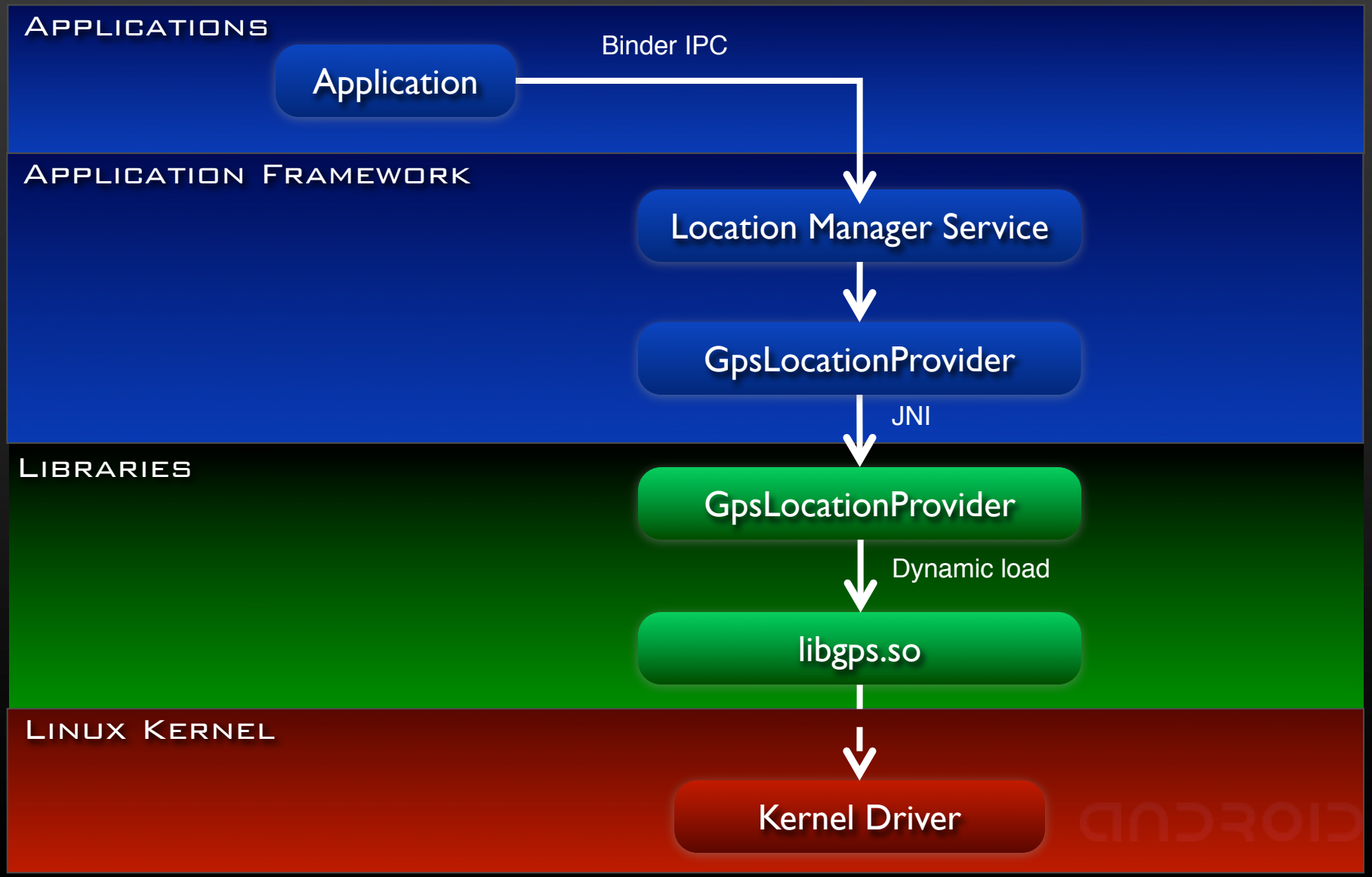
Android Runtime Services



Android Runtime Services



Example: Location Manager



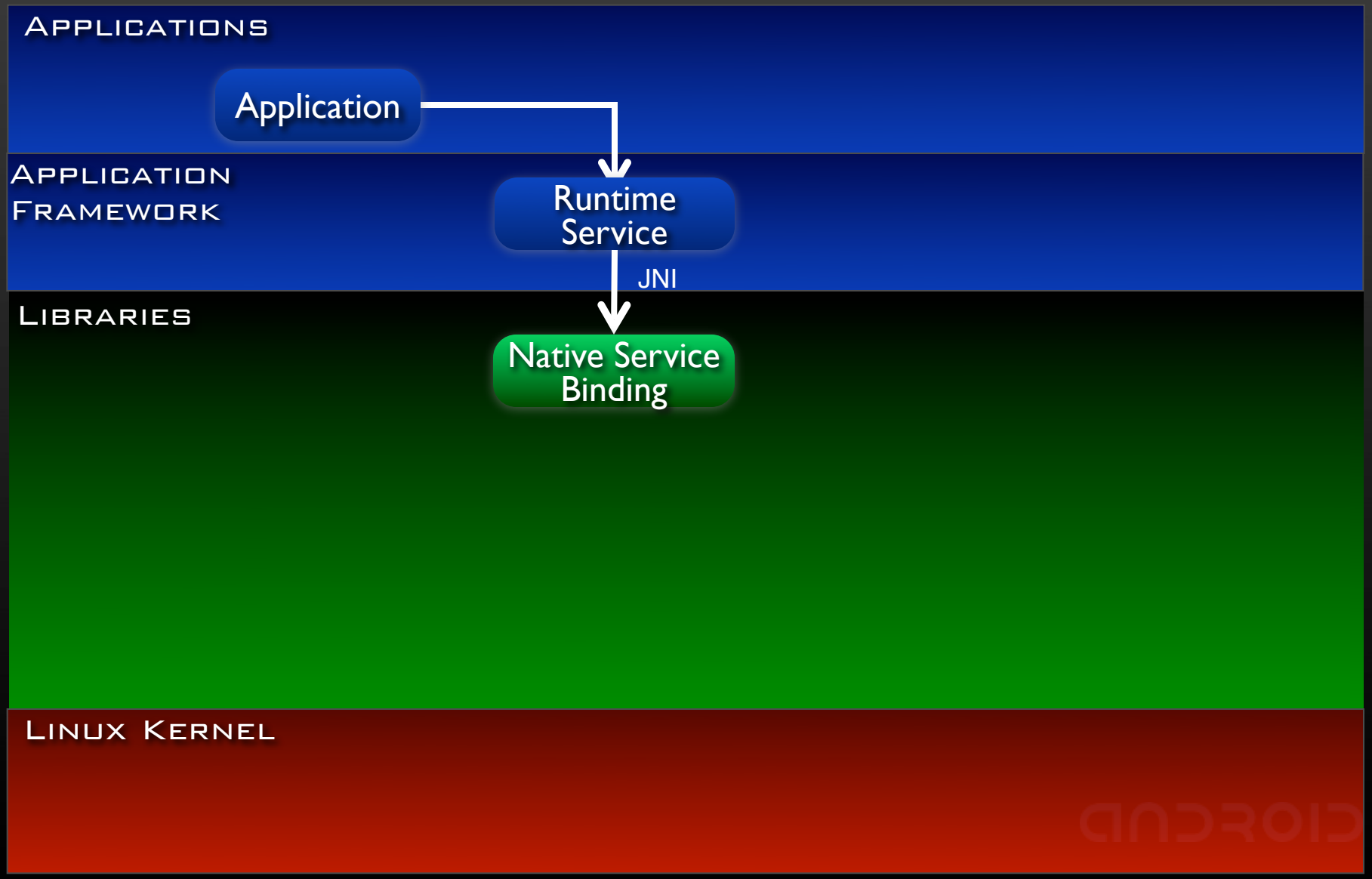
Layer Interaction

There are 3 main flavors of Android layer cake:

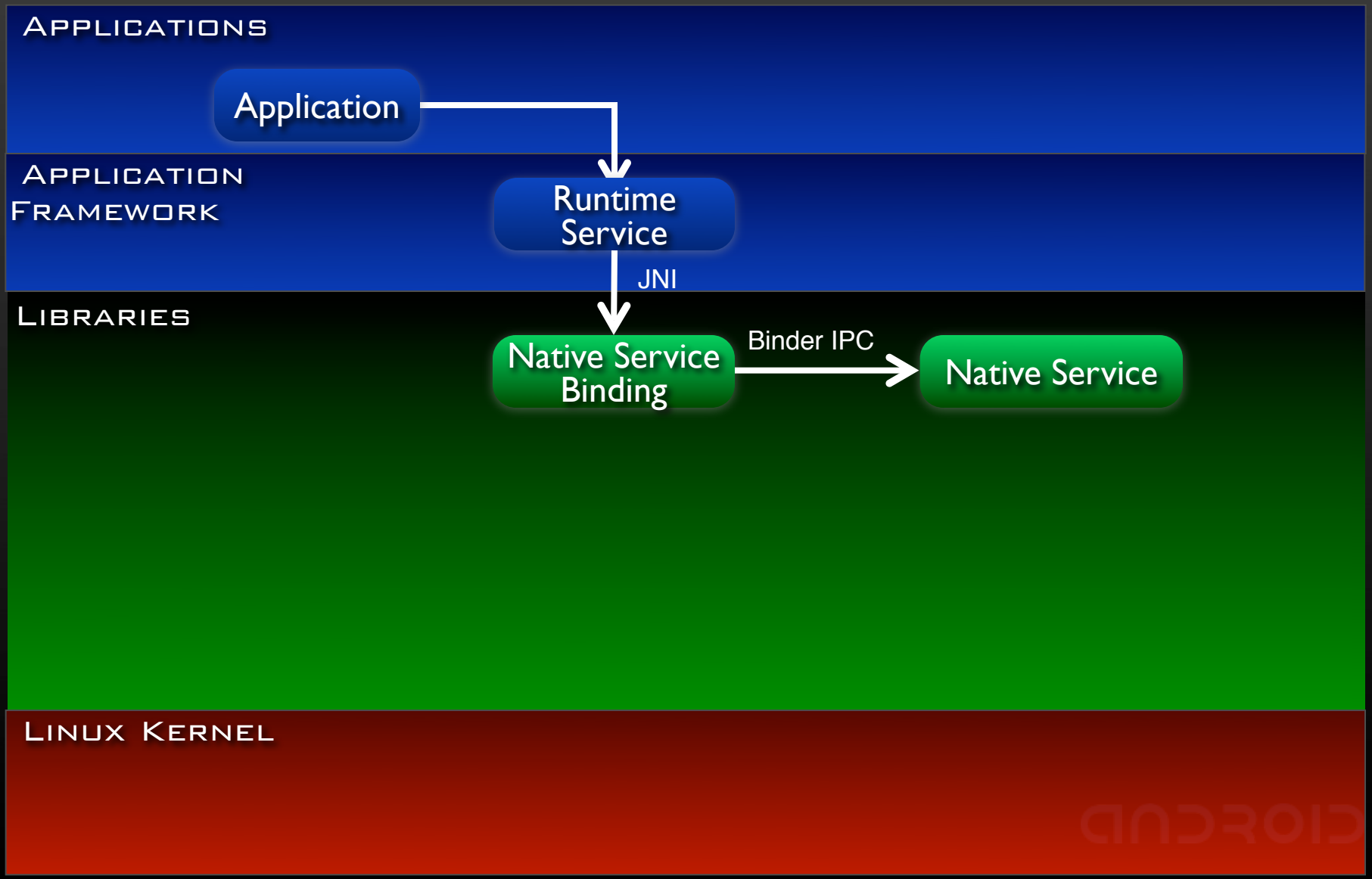
- App → Runtime Service → lib
- App → Runtime Service → Native Service → lib
- App → Runtime Service → Native Daemon → lib

android

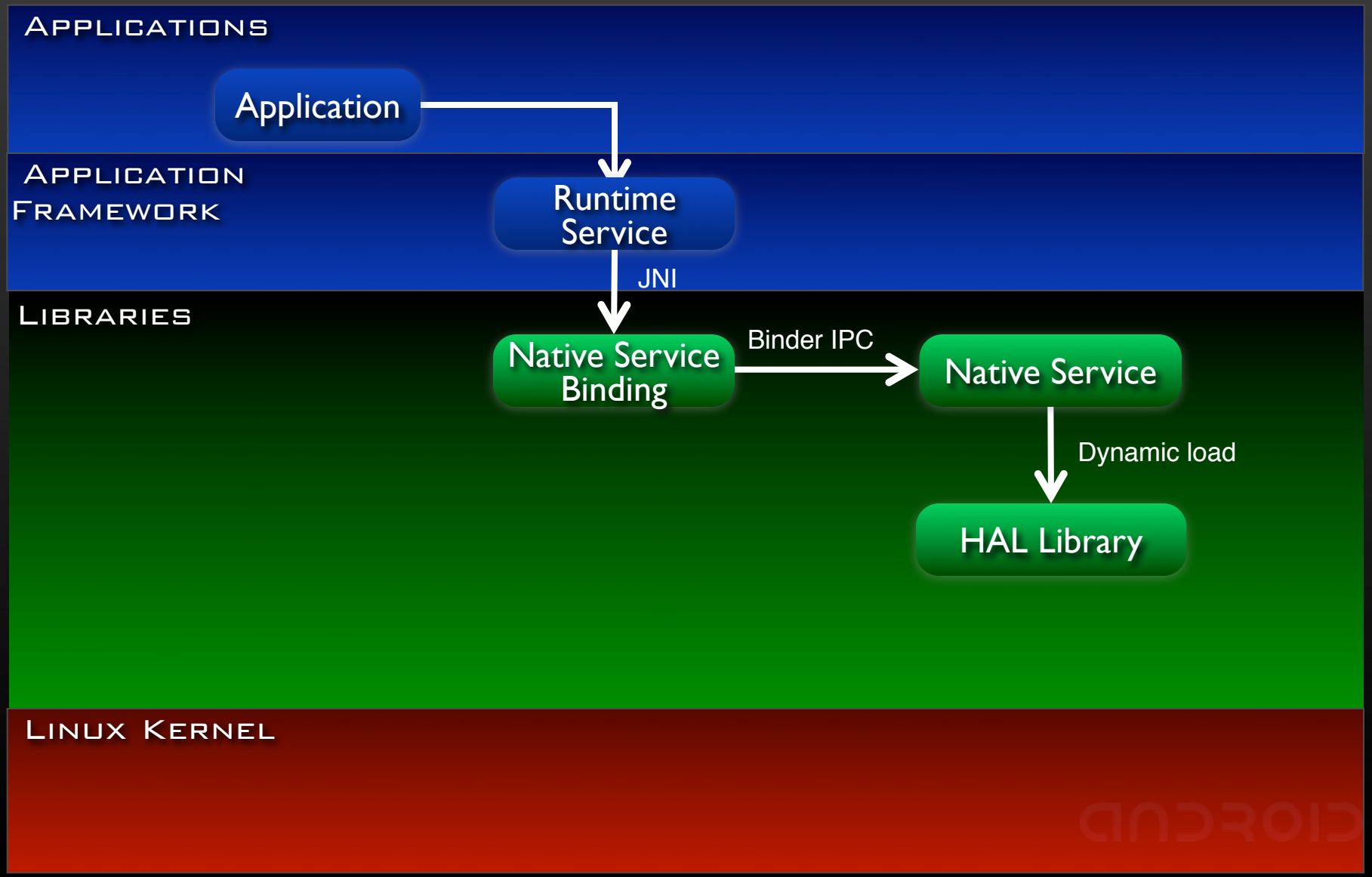
Android Native Services



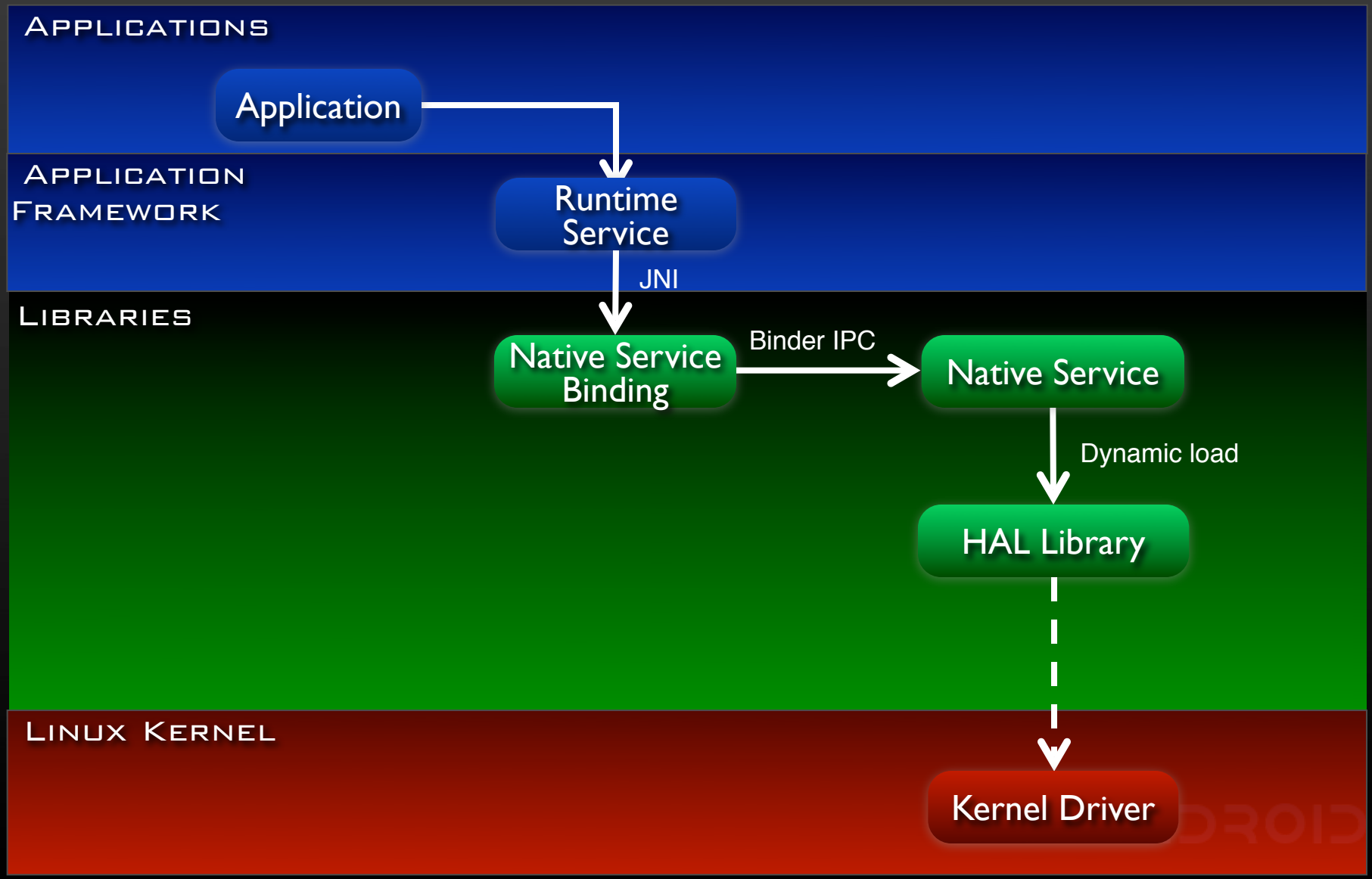
Android Native Services



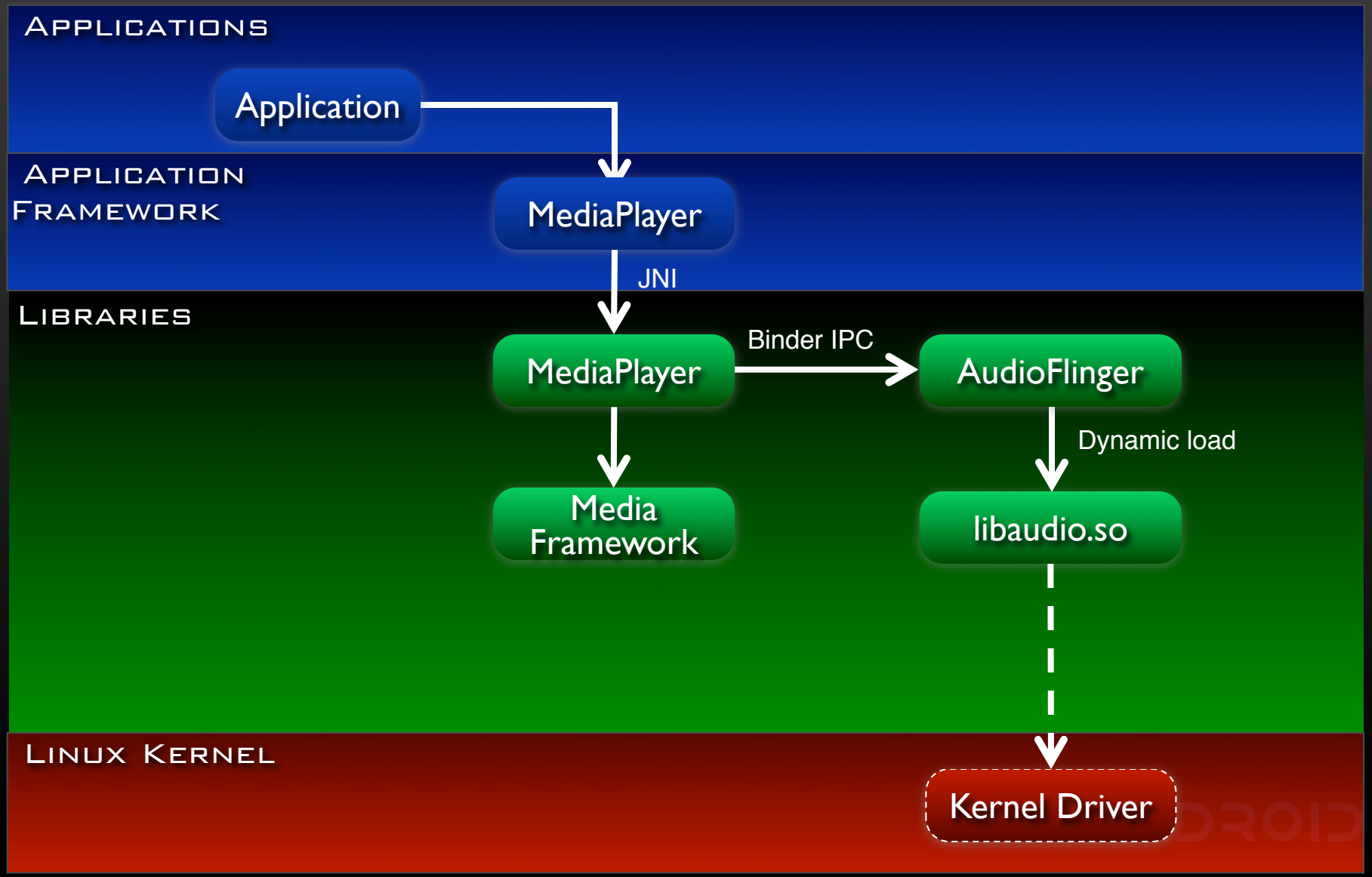
Android Native Services



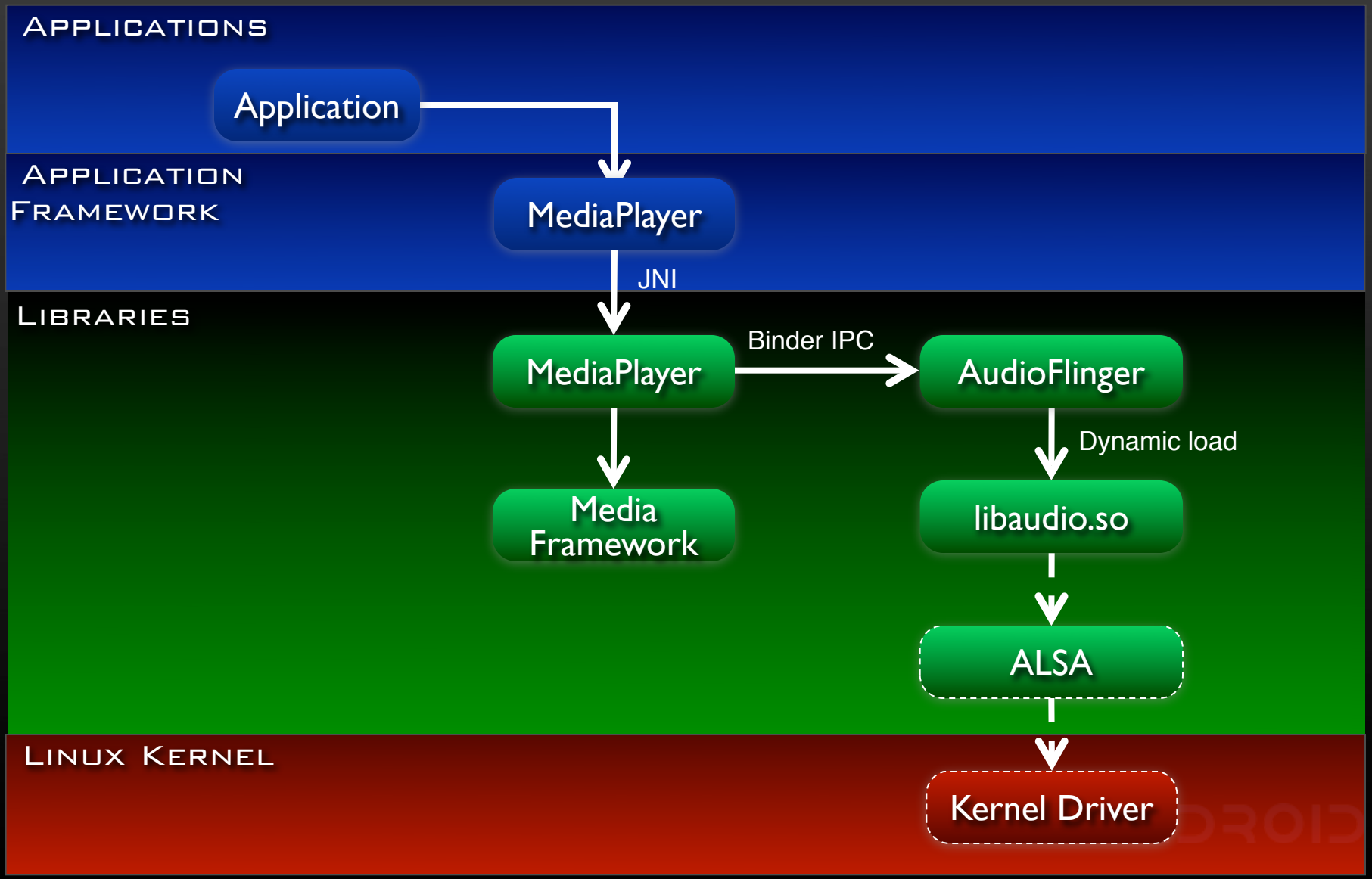
Android Native Services



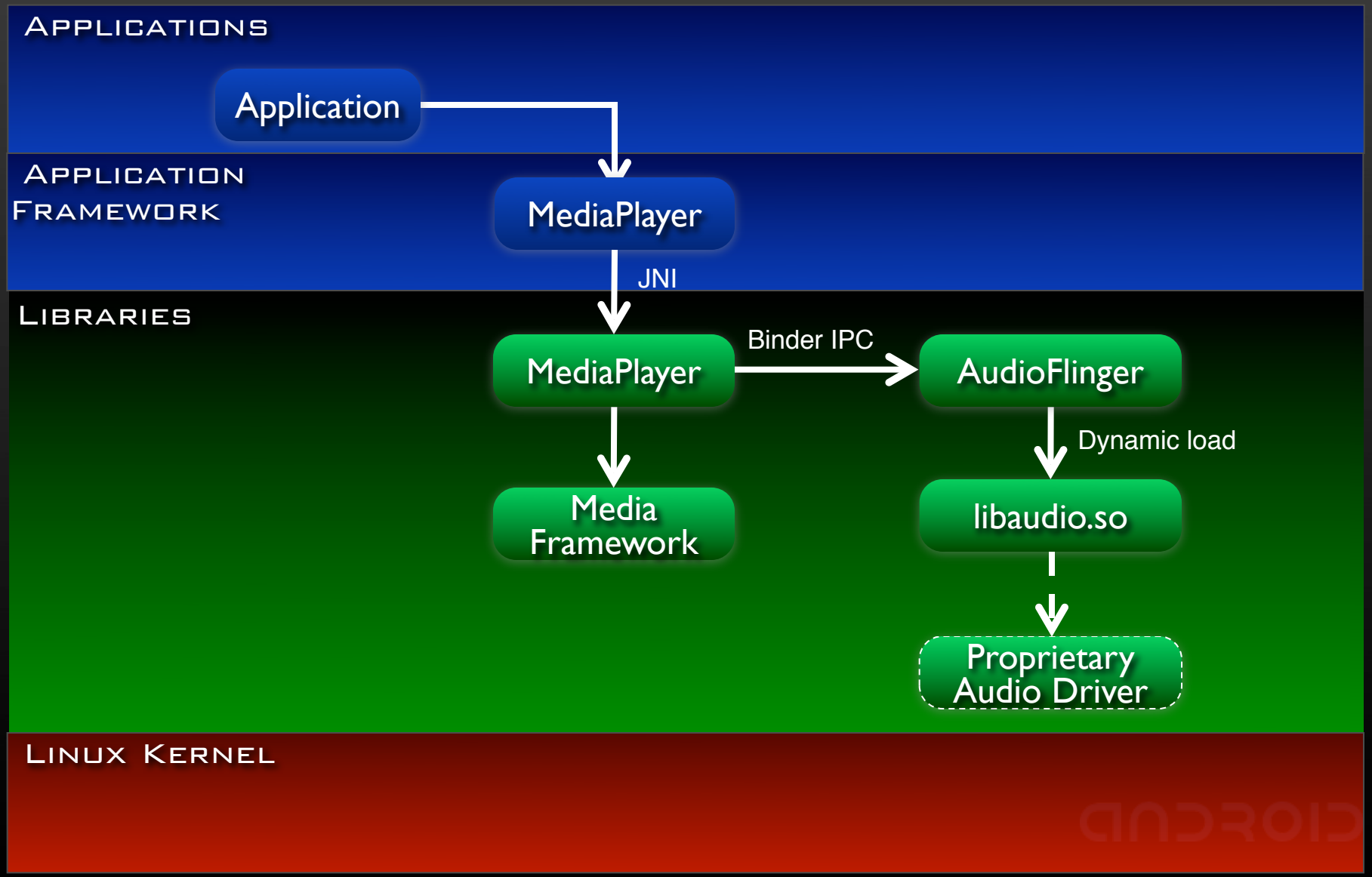
Android Native Services



Android Native Services



Android Native Services



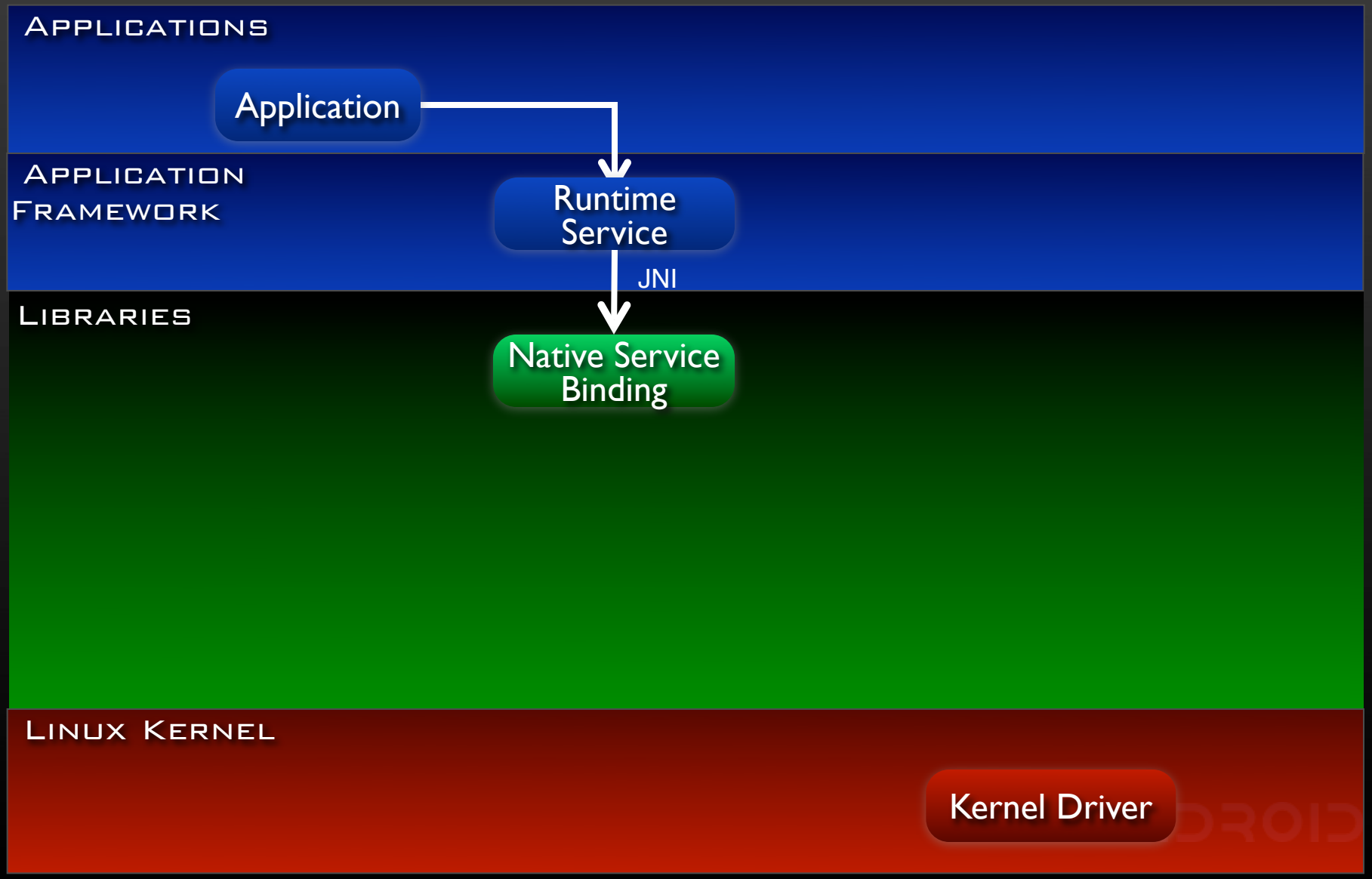
Layer Interaction

There are 3 main flavors of Android layer cake:

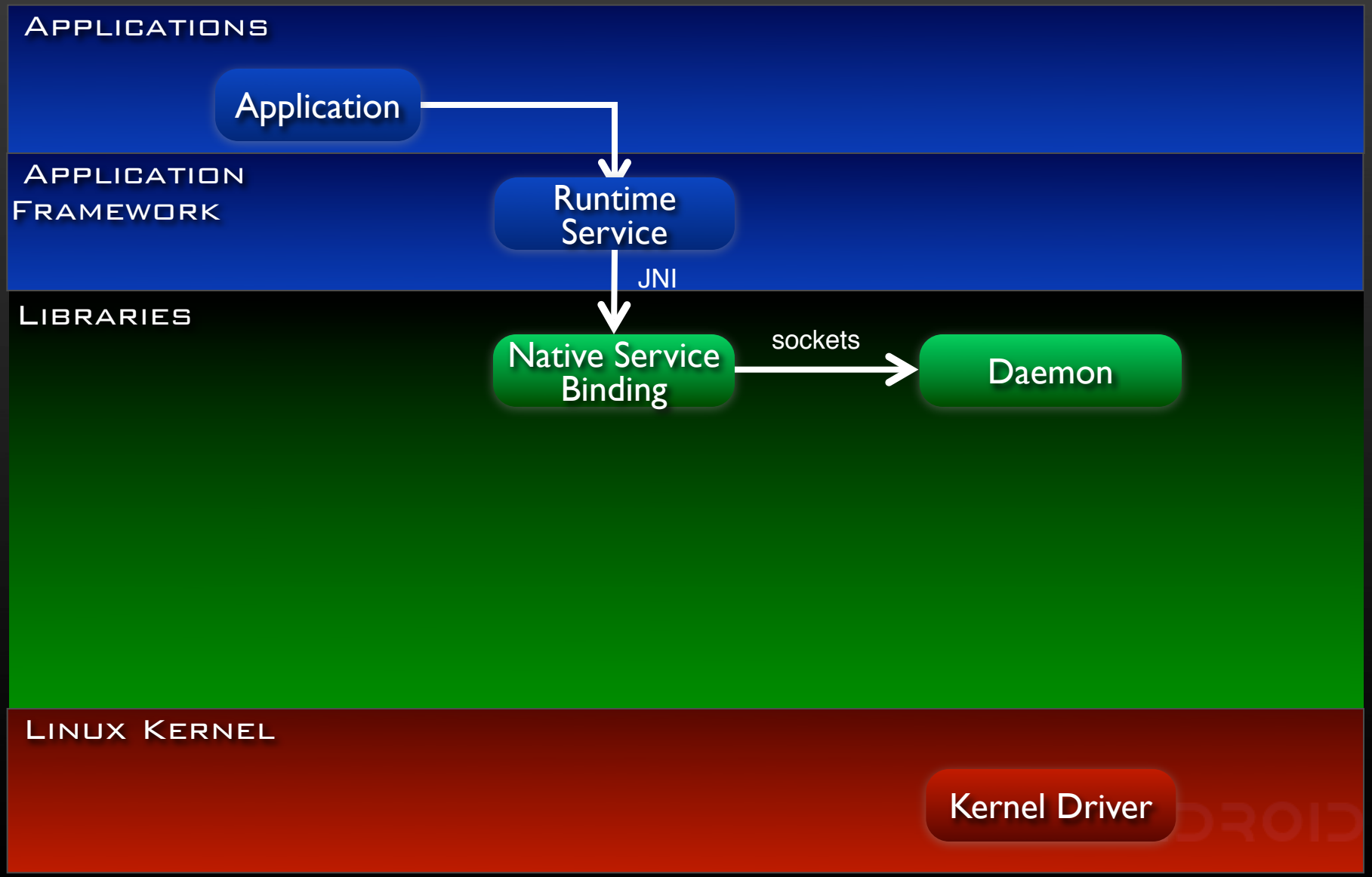
- App → Runtime Service → lib
- App → Runtime Service → Native Service → lib
- App → Runtime Service → Native Daemon → lib

android

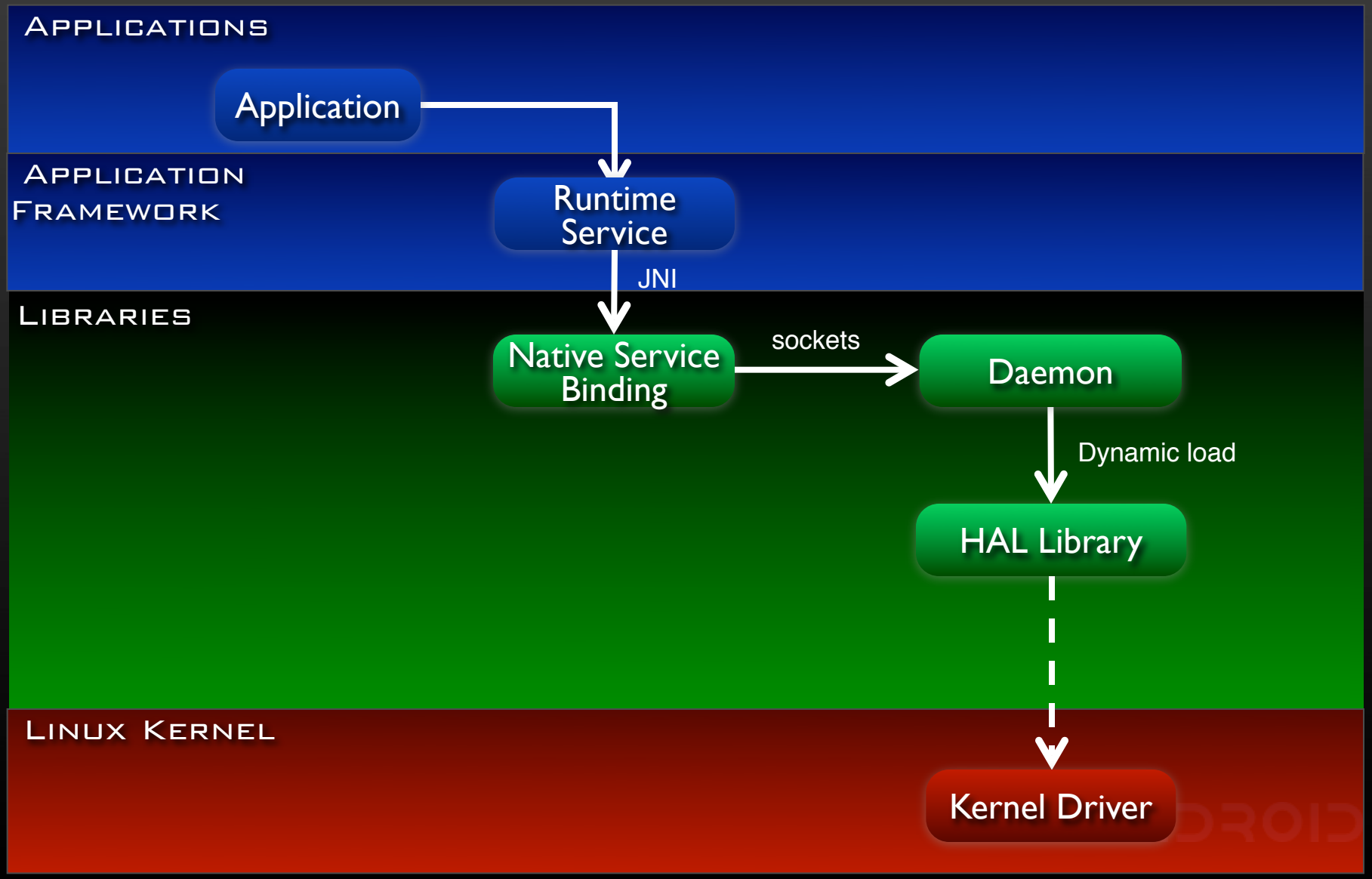
Daemon Connection



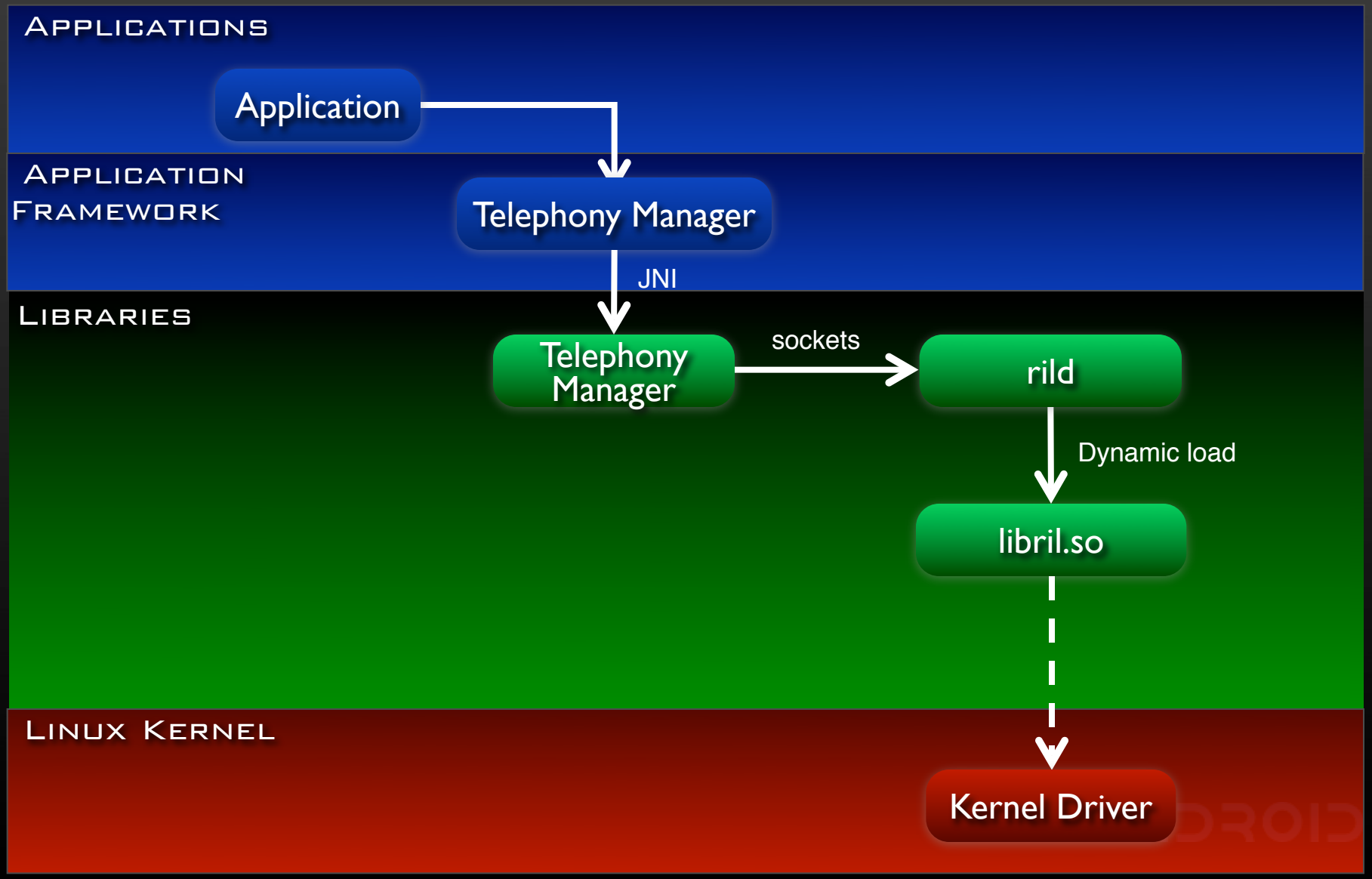
Daemon Connection



Daemon Connection



Daemon Connection



Layer Interaction

There are 3 main flavors of Android layer cake:

- App → Runtime Service → lib
- App → Runtime Service → Native Service → lib
- App → Runtime Service → Native Daemon → lib

ANDROID

Android Anatomy

Applications

Home | Dialer | SMS/MMS | IM | Browser | Camera | Alarm | Calculator

Contacts | Voice Dial | Email | Calendar | Media Player | Albums | Clock | ...

Application Framework

Activity Manager | Window Manager | Content Providers | View System | Notification Manager

Package Manager | Telephony Manager | Resource Manager | Location Manager | ...

Libraries

Surface Manager | Media Framework | SQLite

OpenGL|ES | FreeType | WebKit

SGL | SSL | Libc

Android Runtime

Core Libraries

Dalvik Virtual Machine

Linux Kernel

Display Driver | Camera Driver | Bluetooth Driver | Shared Memory Driver | Binder (IPC) Driver

USB Driver | Keypad Driver | WiFi Driver | Audio Drivers | Power Management

The End

code.google.com

Questions

Q&A