Android Internals

Lecture 3

Security of Mobile Devices

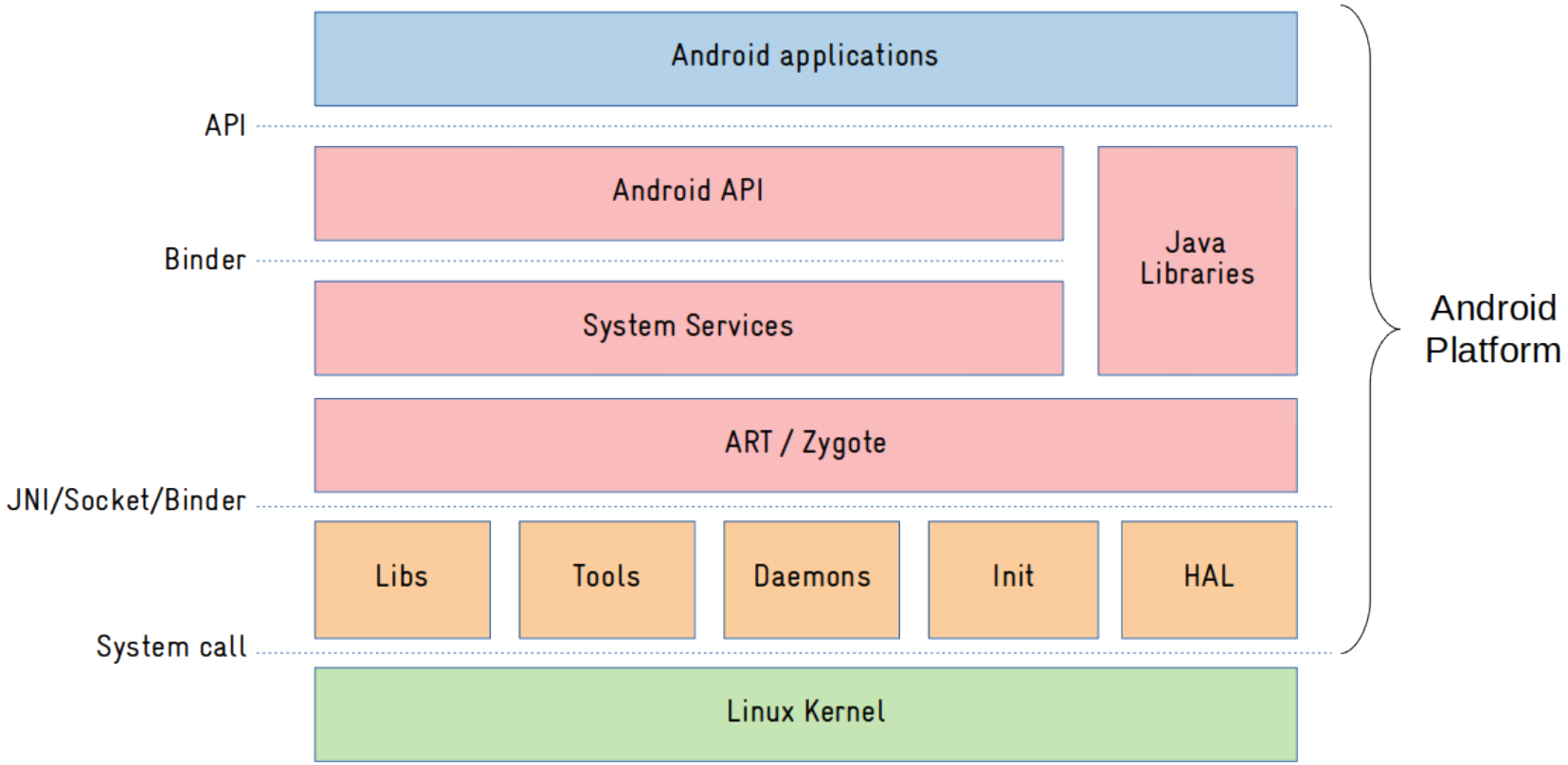
2022

Android Arhitecture

- Linux Kernel
- Binder
- Native Userspace
- ART
- Zygote
- Logd
- System Services

Outline

Android Architecture



Source: https://embeddedbits.org/what-differs-android-from-other-linux-based-systems/

Outline

Kernel

- Linux Kernel
- With some additions
  - Low Memory Killer
  - Wake Locks
  - Binder IPC
- device drivers

Wakelocks

- On desktops and laptops
  - The user decides when the system goes to sleep
- The Android kernel goes to sleep as often as possible
- Sometimes you want to keep the system from going to sleep
  - Input from the user, critical operations
- Wakelocks keep the system awake
- A wakelock must be obtained by the application when it needs to stay awake

Wakelocks

- Apps use abstractions that handle locking
- Apps can request wakelocks directly from PowerManager Service
- Device drivers call in-kernel wakelock primitives
- Permission `android.permission.WAKE_LOCK`
- `acquire()` and `release()` methods

Low Memory Killer

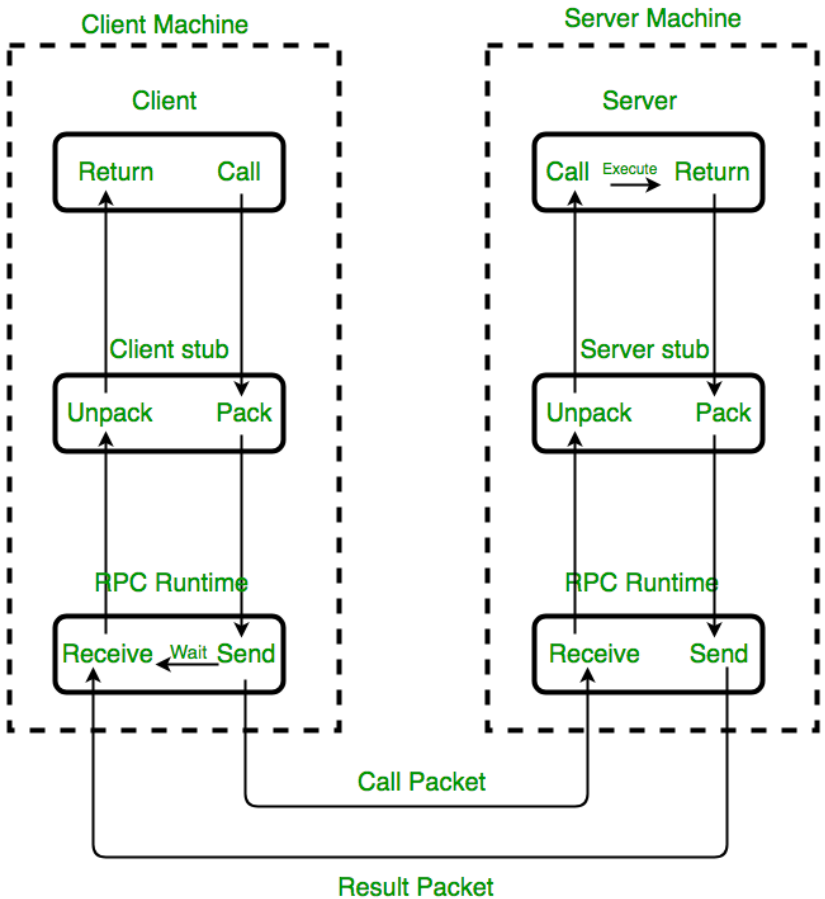
- Many processes => low memory, delays
- Memory pressure
- Low Memory Killer driver
  - Based on hardcoded values
  - Rigid
  - Removed from kernel 4.12
- `lmkd` daemon - memory monitoring, killing processes

Outline

Binder History

- RPC mechanism
- Initially in BeOS (then bought by Palm)
- OpenBinder project
- OpenBinder developers working in Android team
- Android Binder does not derive from OpenBinder
  - Clean re-write of the same functionality
- OpenBinder documentation for understanding the mechanism
- Binder driver in the mainline from kernel 3.19

RPC principles



Implementation of RPC mechanism

Source: `https://www.geeksforgeeks.org/remote-procedure-call-rpc-in-operating-system/`

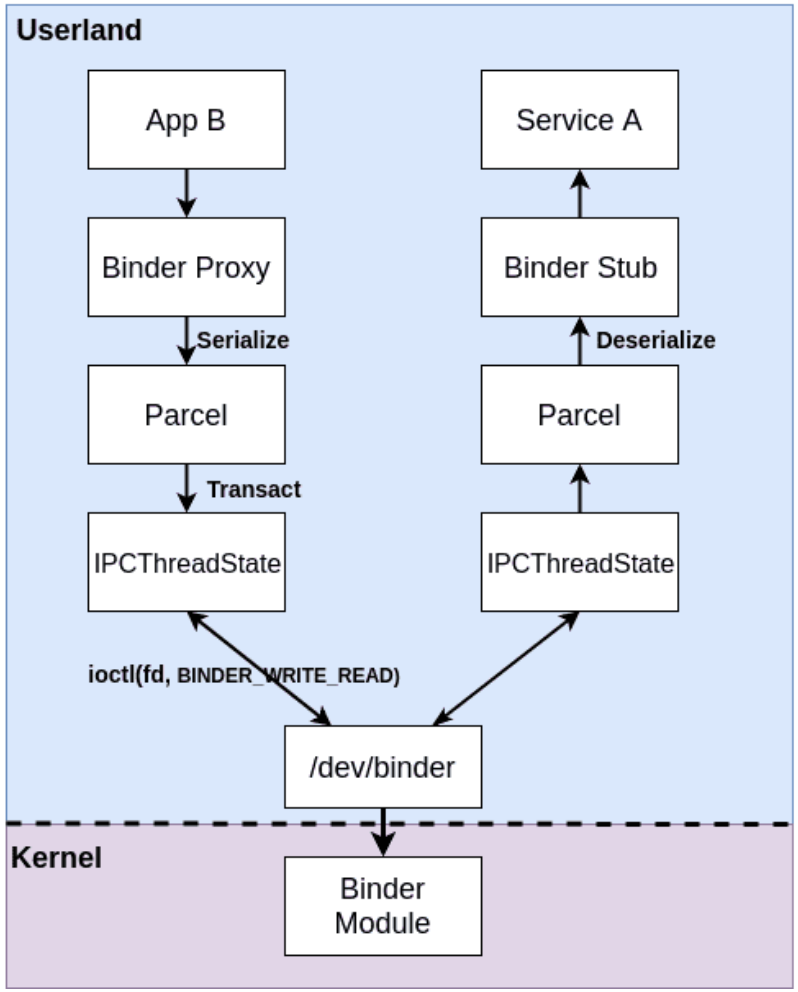
Binder

- Remote object invocation
  - Remote services as objects
  - Interface definition and reference to it
- Cornerstone of Android architecture
  - Apps talk to systems services
  - Apps talk to application services
- Developers don't use the Binder directly
- Use interfaces and stubs generated with the `aidl` tool
- Public API uses stubs to communicate with system services

Binder Driver

- Part of the Binder implemented in a kernel driver
- Character device
- `/dev/binder`
- `ioctl()` calls
- Transmit parcels of data (serialized) between entities

Binder Communication



Source: `https://www.synacktiv.com/en/publications/binder-transactions-in-the-bowels-of-the-linux-kernel.html`

Binder Domains

IPC Domain	Description
`/dev/binder`	IPC between framework/app processes with AIDL interfaces
`/dev/hwbinder`	IPC between framework/vendor processes with HIDL interfaces IPC between vendor processes with HIDL interfaces
`/dev/vndbinder`	IPC between vendor/vendor processes with AIDL Interfaces

Source: `https://www.synacktiv.com/en/publications/binder-transactions-in-the-bowels-of-the-linux-kernel.html`

Outline

Init Process

- Native userspace:
  - Init process
  - Native daemons
  - Native libraries
  - Tools
  - HAL
- Init - started by the Linux Kernel after booting

Init Process

- Configures the execution environment of the OS
  - Export environment variables
  - Handling permissions
  - Setting SeLinux
  - Mounting file systems
  - Handling links
- Starts and monitors daemons
- Manages system properties

Android Daemons

- Run in background, indefinitely
- Resposible with some system functionality
- Started by the init process
- Interface between the Android Framework and system resources

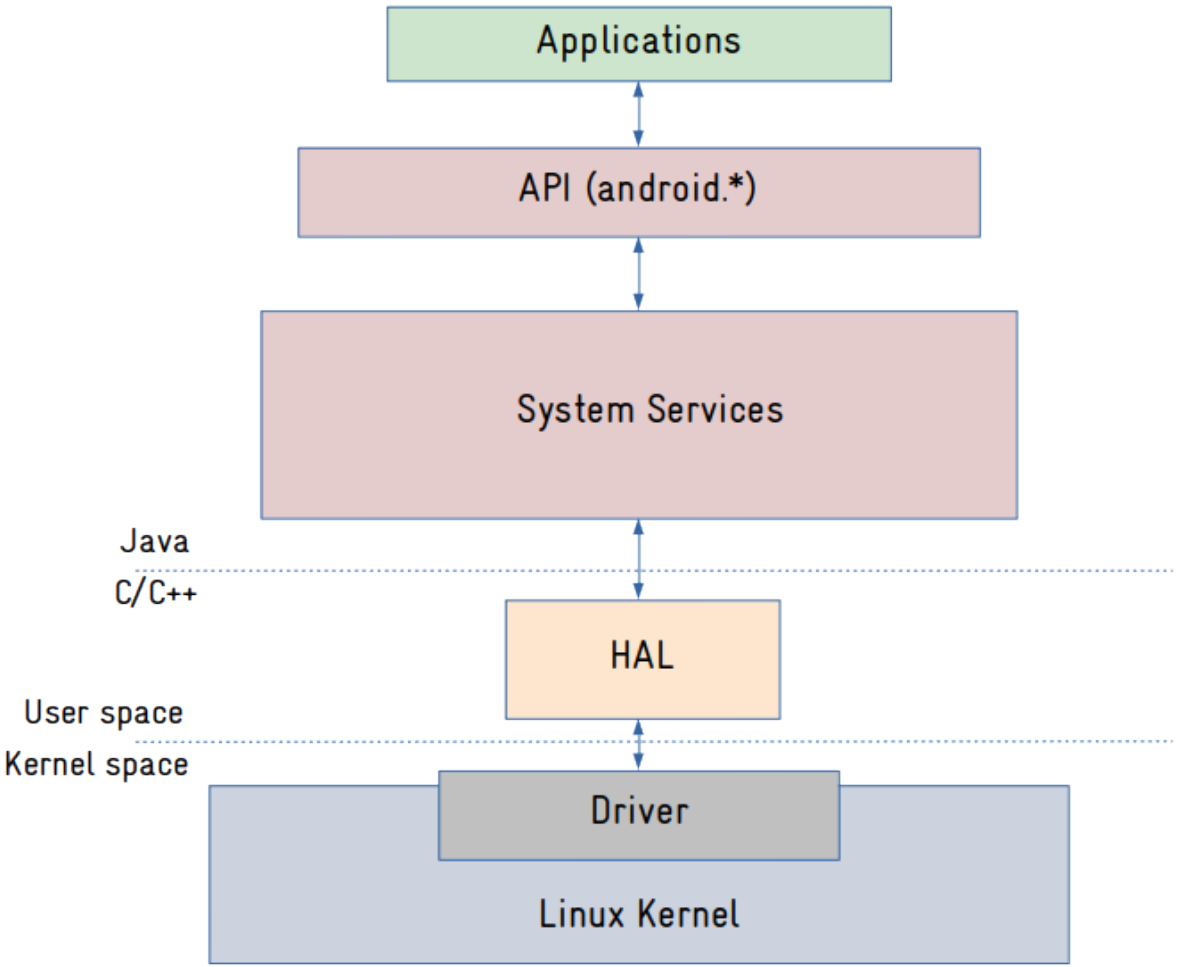
Android Daemons

- `logd` - handles logging
- `lmkd` - low memory killer
- `rild` - communication with the radio chip
- `vold` - handling storage devices
- `installd` - installs Android apps
- `netd` - manages network connections
- `ueventd` - manages connections to hardware devices

Android HAL

- Abstracts access to hardware devices
- Decouple system services from the Linux Kernel
- Kernel interface chages => just modify the HAL
- HAL accessed through the Binder
- Interfaces written in HIDL
- Sensors, Audio, Camera, Display, etc.

Android HAL



Source: https://embeddedbits.org/what-differs-android-from-other-linux-based-systems/

Outline

Android Framework

- On top of the native userspace
- `android.*` packages, System Services, Android Runtime
- Code in `frameworks/` directory in AOSP
- Key building blocks: Service Manager, ART, Zygote

Android Runtime (ART)

- Available from Android 4.4
- Default from Android 5.0
- Dalvik Executable format (dex)
- Ahead-of-Time compilation (AoT)
  - Translate the dex file into an executable for the target device
  - At installation time
  - Replaces JIT compilation and Dalvik interpretation
  - Installation takes longer
  - Executables occupy storage space
  - Additional verifications

ART

- Improved garbage collection
  - More efficient
- Support for sampling profiler
  - Does not affect app performance
- More debugging features
  - Especially for monitoring and GC
- More details in case of exceptions and crash reports

Outline

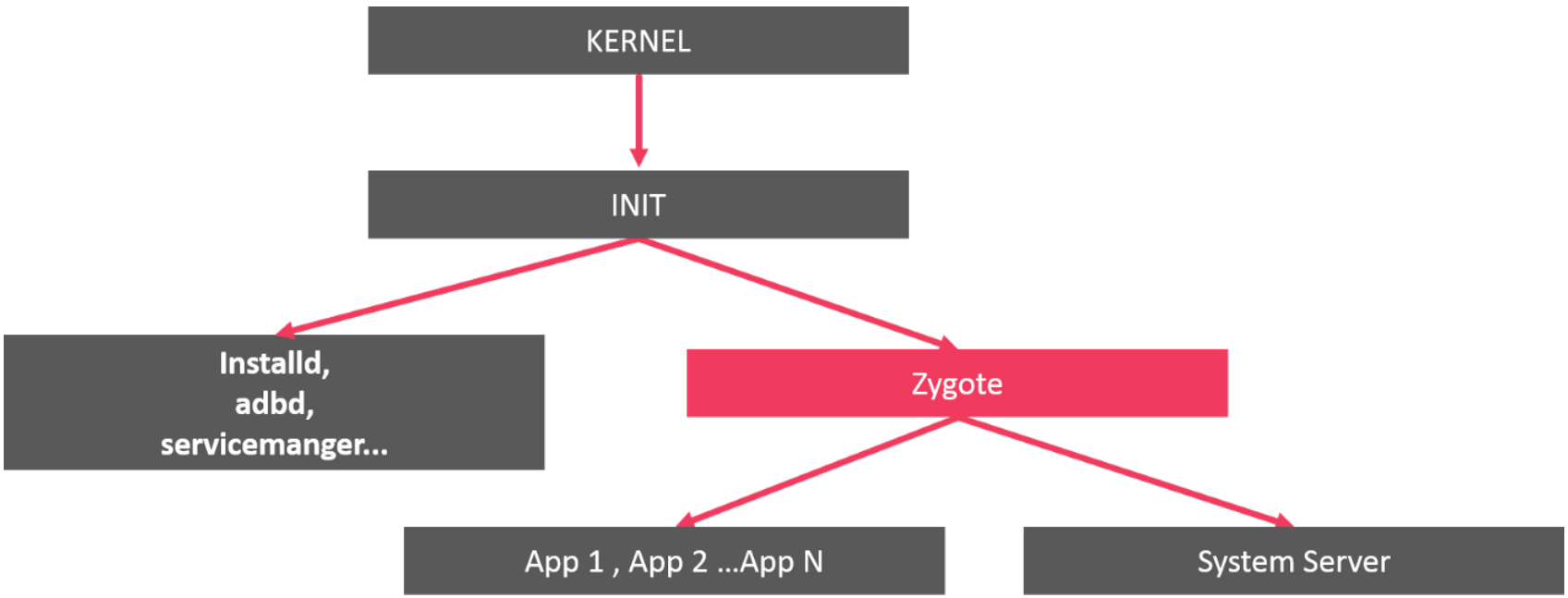
Zygote

- Daemon used to launch apps
- Parent of all applications
- Preloads in RAM all Java classes and resources needed by apps
- Listens to connections on its socket for requests to start apps
  - `/dev/socket/zygote`
- When it gets a request, it forks itself and launches the app

Zygote

- Copy-on-write (COW)
- Classes and resources are not modified, so all apps use them from Zygote
  - A single version of classes and resources in RAM
- The System Server is started explicitly by Zygote
- The PPID of all apps is the PID of Zygote

Zygote



Source: https://medium.com/@khetanrajesh/android-boot-up-process-zygote-507e184a15e1

Zygote

```
hero2lte:/ # ps
USER          PID   PPID  NAME
root          1     0     /init
root          3279  1     zygote
system        3689  3279  system_server
system        5063  3279  com.samsung.android.radiobasedlocation
u0_a10        5090  3279  com.samsung.android.providers.context
advmodem      5117  3279  com.samsung.android.networkdiagnostic
u0_a99        5271  3279  com.samsung.android.widgetapp.briefing
u0_a45        5287  3279  com.samsung.android.service.peoplestripe
u0_a4         5313  3279  com.samsung.android.app.aodservice
u0_a128       5922  3279  com.samsung.android.sdk.handwriting
u0_a6         6178  3279  com.samsung.android.contacts
system        6927  3279  com.samsung.ucs.agent.boot
u0_a108       6939  3279  com.samsung.ucs.agent.ese
u0_a37        12229 3279  com.samsung.klmsagent
system        24833 3279  com.samsung.android.lool
system        25118 3279  com.samsung.android.securitylogagent
system        25354 3279  com.samsung.android.sm.provider
```

Outline

Logd

- From Android 5.0
- Logd daemon
- Centralized user-mode logger
- Addresses the disadvantages of circular buffers
- Integration with SELinux
  - Registers as auditd
  - Receive messages via netlink

Logd

- Uses 4 sockets
- `/dev/socket/logd` - control
- `/dev/socket/logdw` - write-only
- `/dev/socket/logdr` - read-only
- Unnamed netlink socket - SELinux

Logd

- Write log messages:
  1. `Log` class
  2. Liblog library
  3. `/dev/socket/logdw` socket
- Read log messages:
  1. `logcat`
  2. Liblog library
  3. `/dev/socket/logdr` socket

Outline

System Services

- Form an object-oriented OS on top of Linux
- System Server
  - All components run in the `system_server` process
  - Many Java-based services/managers, 2 C-based services
  - Power Manager, Activity Manager, Location Manager, etc.
  - Surface Flinger, Sensor Service (C/C++)
- Media Server
  - `mediaserver` process
  - C/C++ code
  - Audio Flinger, Media Player Service, Camera Service

Service Manager

- Performs system service handle lookups
- The Yellow pages book of all system services
- A service must be registered to the Service Manager to be available
- Started by `init` before any other service
- Opens `/dev/binder` and becomes the Context Manager of the Binder
- Binder ID 0 = "magic object" = Service Manager

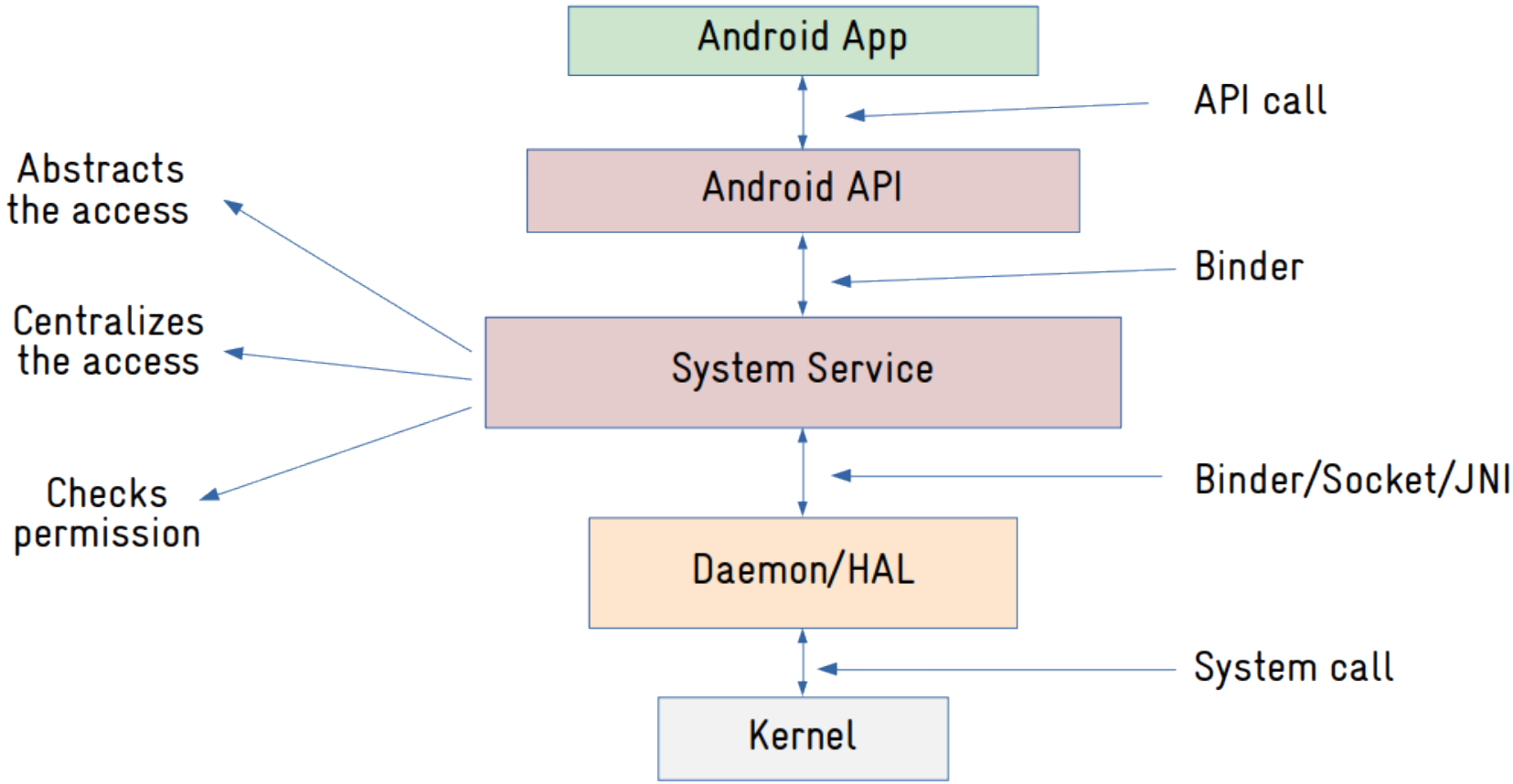
Service Manager

- System Server registers every service with the Service Manager
- Any component that wants to talk to a system service:
  - Asks the Service Manager for a handle
  - `getSystemService()`
  - Invokes the methods of the service using the handle
- Only to access system services
- Used by the `dumpsys` utility to obtain the status of the system services

Access a System Service

- System services accessed through the Binder
- Example - reading data from sensors:
  - App calls methods from SensorManager (API)
  - SensorManager calls SensorService through the Binder
  - System sevice manages access to data
  - Verifies permissions of the calling app
  - Calls Sensors HAL through the Binder
  - HAL calls kernel driver through system calls to get data

Access a System Service



Source: `https://embeddedbits.org/what-differs-android-from-other-linux-based-systems/`

Activity Manager

- One of the most important services in the System Server
- Handles activity lifecycle
- Sends intents
- Starts new components (activities, services)
- Obtains content providers

Activity Manager

- Responsible with the Application Not Responding (ANR) messages
- Involved in
  - Permission checks
  - Task management

Activity Manager

- Starts the Launcher (with `Intent.CATEGORY_HOME`)
- When an app is started from Launcher
  - Launcher's `onClick()` callback is called
  - Launcher calls the `startActivity()` from ActivityManager (through Binder)
  - ActivityManager calls `startViaZygote()` method
  - Opens socket to Zygote and asks to start the activity
- `am` command for invoking the functionality of the ActivityManager

Package Manager

- Manages the `.apk` files in the systems
- API for installing, uninstalling, upgrading `.apk` files
- Works with files located in `/data/system/`
  - `packages.xml` - permissions and packages
  - `packages.list` - details about packages

Package Manager

- Runs in `system_server` (system user)
- Uses `installd` daemon for operations (root user)
- Resolves intents
  - Searches in Manifest files
- `pm` command for invoking the functionality of the PackageManager
  - List packages, list permissions, install/uninstall/disable packages, etc.

Power Manager

- Control the power state of the device
- Handles WakeLocks
- Includes the WakeLock class
  - `acquire()`, `release()`
- Apps request WakeLocks from PowerManager

Power Manager

- All calls to the Power Management (kernel) go through PowerManager
- Can force device to go to sleep
- Set the brightness of the backlights

Bibliography

- https://source.android.com/devices/architecture
- https://developer.android.com/reference/android/os/PowerManager#newWakeLock(int,%20java.lang.String)
- https://source.android.com/devices/tech/perf/lmkd
- https://source.android.com/devices/architecture/hidl/binder-ipc
- https://source.android.com/devices/tech/dalvik
- https://embeddedbits.org/what-differs-android-from-other-linux-based-systems/
- https://www.synacktiv.com/en/publications/binder-transactions-in-the-bowels-of-the-linux-kernel.html

Keywords

- Linux Kernel
- Wake Locks
- Low Memory Killer
- Binder
- Init process
- Daemons
- HAL
- ART
- Zygote
- Logd
- System services
- Service Manager
- Activity Manager
- Package Manager
- Power Manager