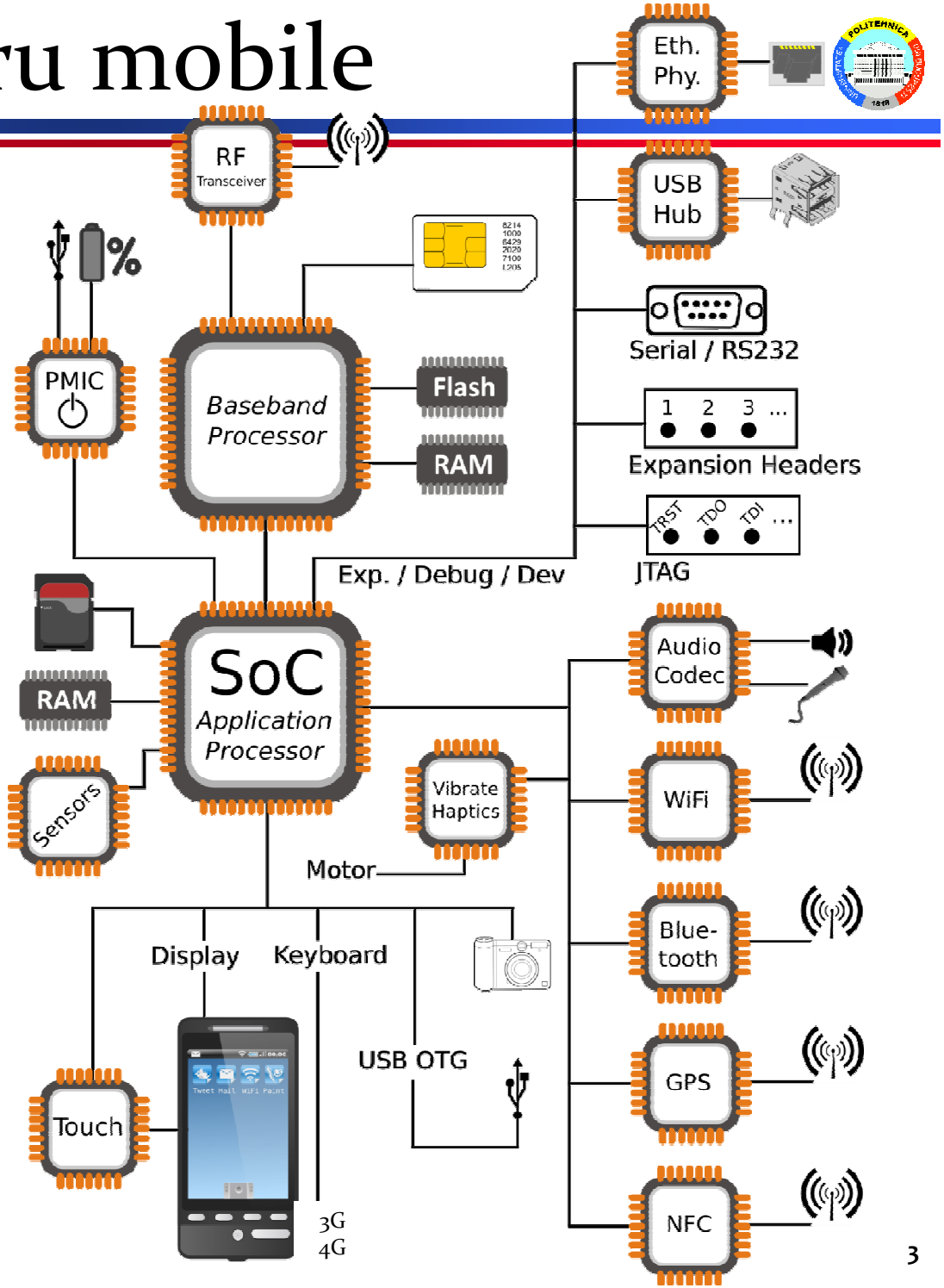
Sumar

- Android internals
- Noțiuni generale despre radio
- Accesul la mediu
  - SDMA, FDMA, TDMA, CDMA
  - CSMA/CA
- Sisteme de comunicații mobile
  - 2G: GSM
  - 3G: UMTS
  - 4G: LTE
- WiFi
  - 802.11a/b/g/n/ac/ad
  - Infrastructuri
- Mobile IP
  - Locator/Identifier split
  - Routing
- Mobilitate nivel transport
  - I-TCP, middlebox-uri
- VoIP
  - QoS, SIP
- Descoperire servicii
  - zeroconf, mDNS, DNS-SD
- Servicii de locație
  - Exterior: GPS, CellID
  - Interior: WiFi

Android Internals

Hardware pentru mobile

- ARM, nu x86
- Memorie limitată
- Baterie mică
- Flash, nu disc
- GUI
- Radio – BT, WiFi, 4G
- GPS



Android

- De ce bazat pe Linux?
  - Drivere
  - Rețea, servicii
  - Gestiune procese, memorie
- Imens
- Se schimbă repede
- Schimbări imprevizibile
  - Schimbări interne, dar API relativ stabile

Android

Au fosta dăugate

- Biblioteci C proprii (bionic)
- Mediu de rulare JAVA (dalvik, ART)
- Framework aplicații
- Binder, ashmem
- Gestiune consum
  - wakelocks

Is Android really Open Source?

~90% of development is behind closed doors

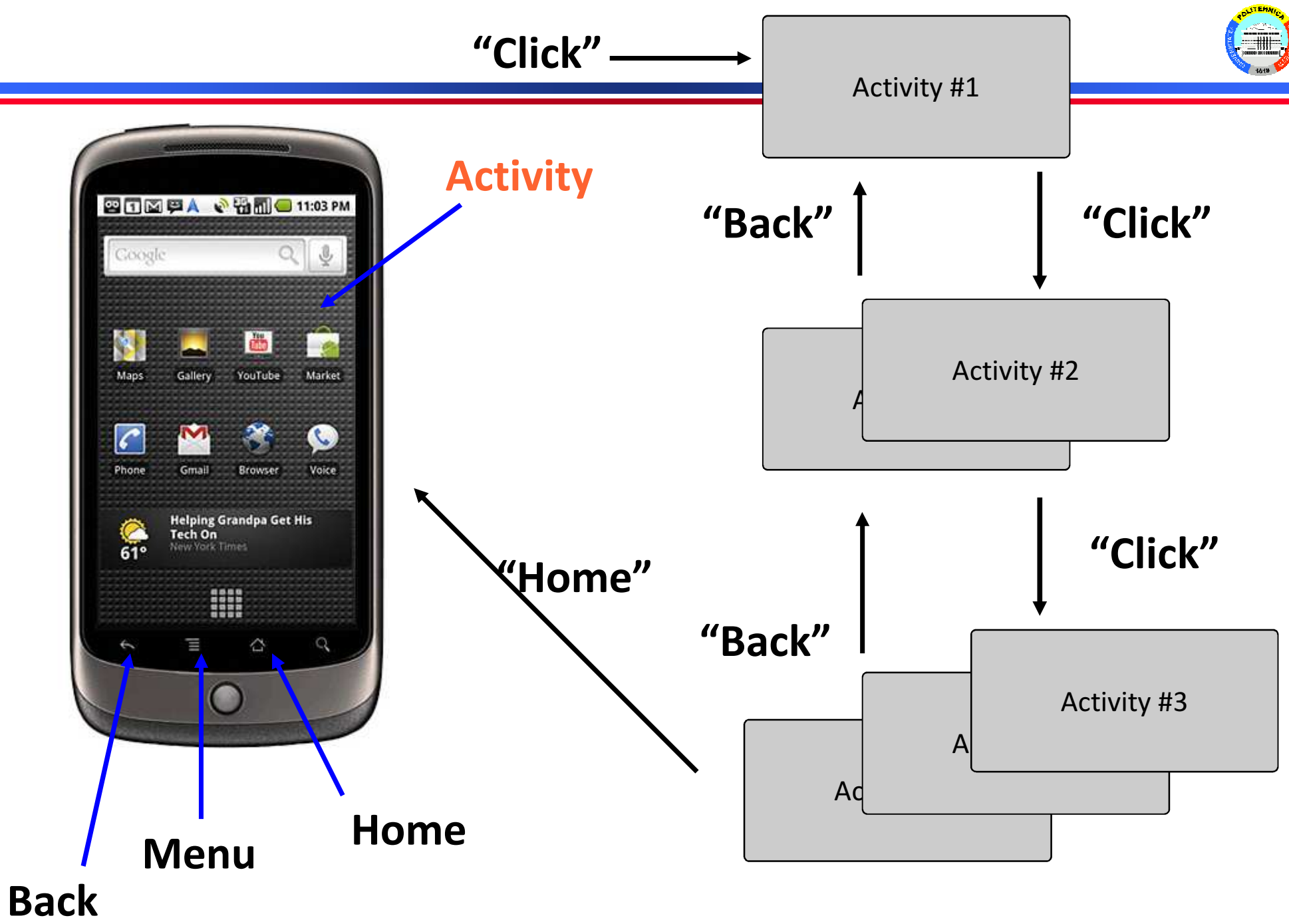
- Only Google&partners see source code
- Only Google&partners accept (or ignore/deny) patches
  - Google is both the KeyMaster and the Architect
- Very few public discussions occur on AOSP design principles
- No public roadmap of features
- However….
  - Google accepts huge responsibility for standardization & long term stability of platform
  - Major contributions to upstream Linux kernel
  - Huge money influx for PR releases, OEM standardization efforts, etc

"Open source is different than a community-driven project. Android is light on the community-driven side and heavy on the open source. Everything we do ends up in the open source repository."

- Andy Rubin, Cofounder of Android Inc, SVP@Google until 2013

See https://source.android.com/source/code-lines.html#about-private-code-lines

Slide from Hamilton Turner 2015


"Click"
Activity #1
Activity
"Back"
"Click"
Activity #2
"Home"
"Click"
"Back"
Activity #3
Back
Menu
Home

Comportarea aplicațiilor

- Filozofie diferită de aplicații desktop (Unix, Windows)
- Fără punct de intrare unic… fărămain() !?
- Procesele și aplicațiile terminate aleator
  - Developerul trebuie să țină cont
- UI deconectat de “creierul” aplicației
- Aplicațiile sunt foarte izolate
- **Comportarea controlată de**
  - **Memorie redusă**
  - **Baterie redusă**

Concepte Linux

- Processes (fork() și prietenii)
- Signals (kill() ... or be killed)
- Sockets / Pipes / Fifos / SysV IPC
- Hardware devices ca fisiere(/dev)
- Daemons
- Shell / scripts
- Useri(root vs. ceilalți-- # vs. $)
- ELF files
- GNU toolchain
- ... 40 de ani de Unix

Concepte Android

- Components
- Intents
- Manifest file
- Component lifecycle
- Processes and threads
- Remote procedure calls
- Permissions
- Storage
- Native development

Componente

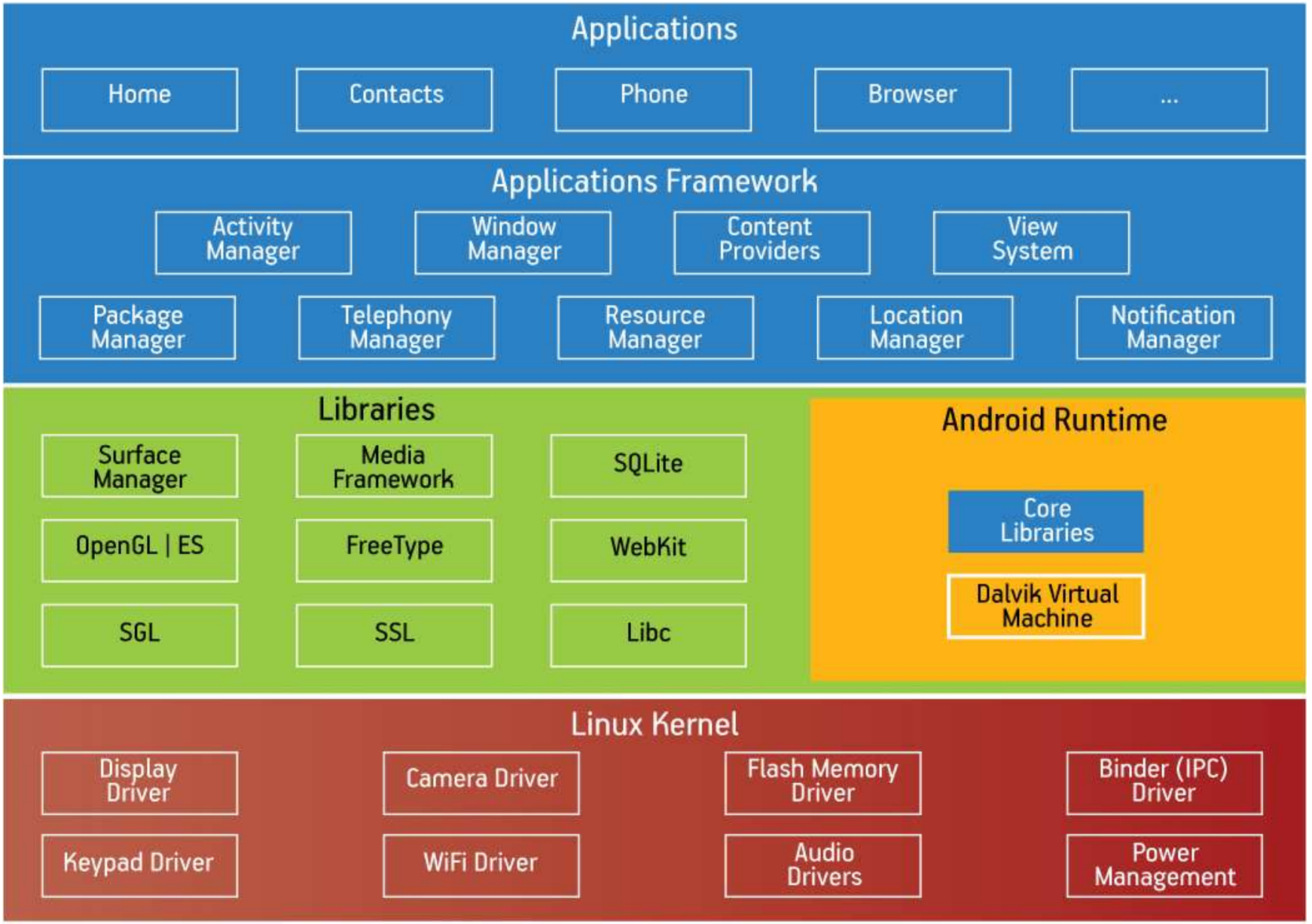
- 1 Aplicație = N Componente
- aplicațiile pot partaja componente
- Procesele aplicațiilor sunt pornite automat când o componentă e apelată
- **N puncte de intrare, !1, !main()**
- Componente:
  - Activities
  - Services
  - Broadcast Receivers
  - Content Providers

Intents

- Intent = mesaj asincron cu/fără target
- Ca un signal polimorfic de Unix signal, fără target
- Intent Filters specificate în Manifest file

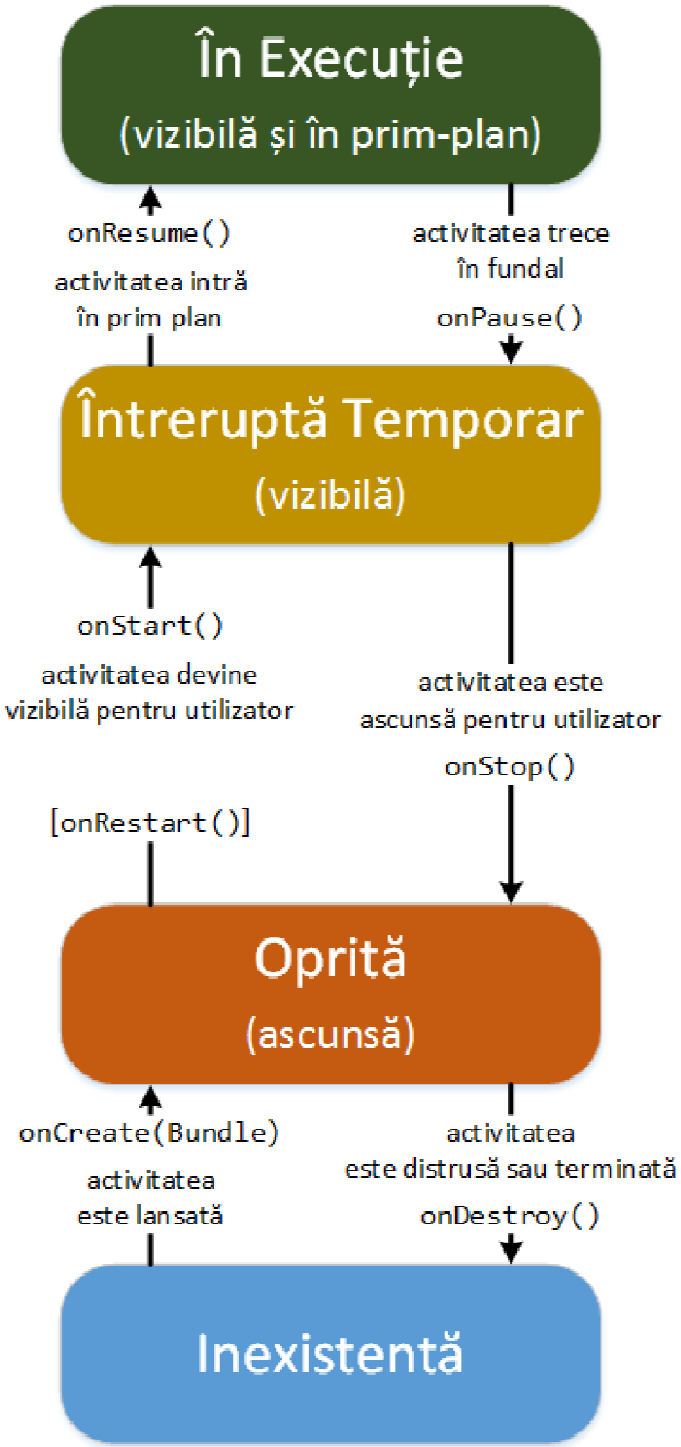
Manifest file

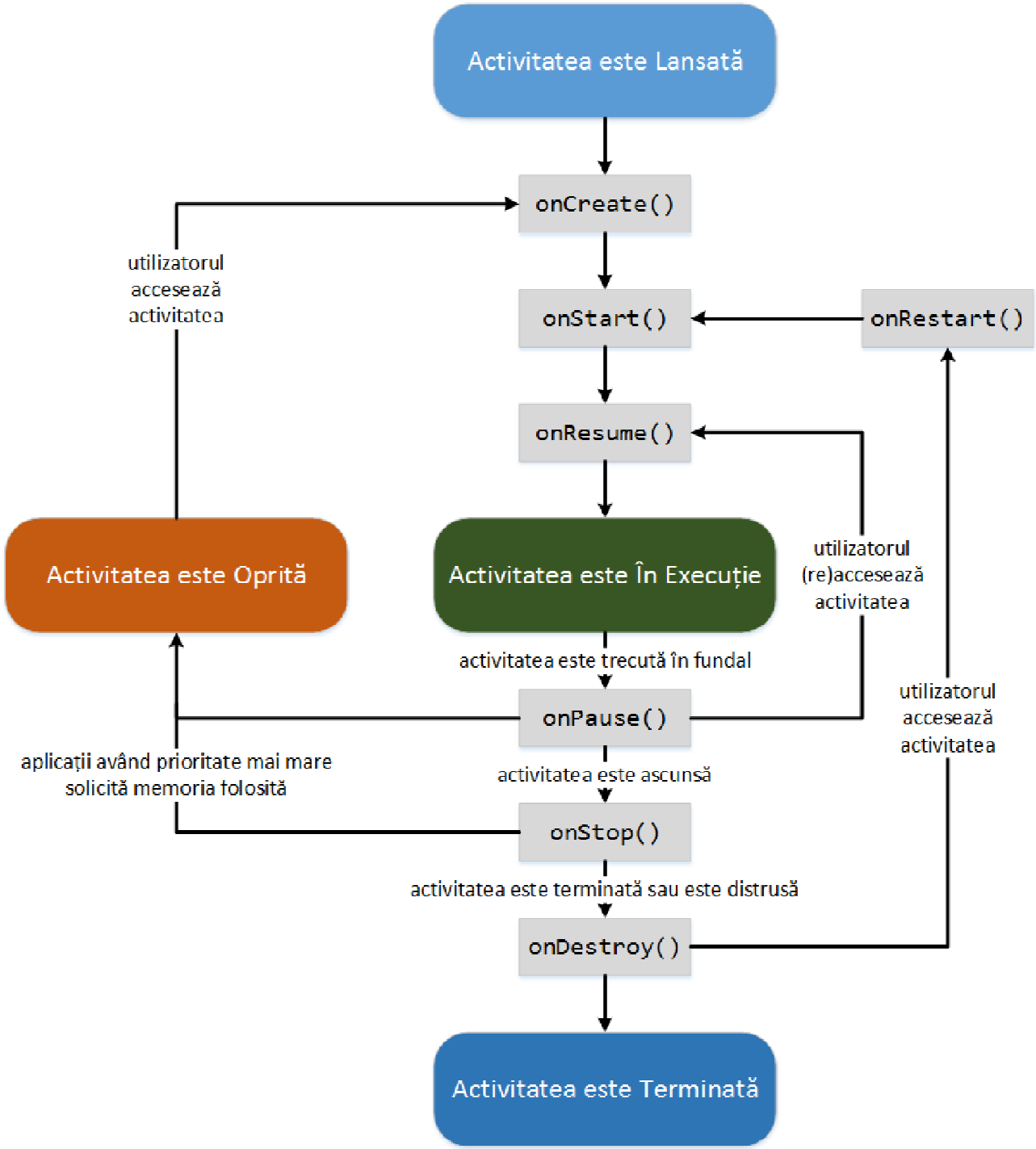
- Informează sistemul despre componentele aplicației
- Numit mereu AndroidManifest.xml
- Activity = <activity> ... static
- Service = <service> ... static
- Broadcast Receiver:
  - Static = <receiver>
  - Dynamic = Context.registerReceiver()
- Content Provider = <provider> ... static


Applications
Home
Contacts
Phone
Browser
...
Applications Framework
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification 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
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

Ciclul de viață al componentelor

- Sistem pornește/oprește/omoară procesele automat
- **Toată comportarea sistemului ține cont de**
  - **Memorie redusă**
  - **Baterie redusă**
- Sistemul
  - Apelează callback-uri când e necesar
  - Gestionează ciclul de viață
- Unele comopnente sunt mai complex de gestionat


În Execuție
(vizibilă și în prim-plan)
onResume()
activitatea intră
în prim plan
activitatea trece
în fundal
onPause()
Întreruptă Temporar
(vizibilă)
onStart()
activitatea devine
vizibilă pentru utilizator
activitatea este
ascunsă pentru utilizator
onStop()
[onRestart()]
Oprită
(ascunsă)
onCreate(Bundle)
activitatea
este lansată
activitatea
este distrusă sau terminată
onDestroy()
Inexistentă


Activitatea este Lansată
onCreate()
onStart()
onRestart()
onResume()
Activitatea este În Execuție
Activitatea este Oprită
Activitatea este Terminată
onPause()
onStop()
onDestroy()
utilizatorul accesează activitatea
utilizatorul (re)accesează activitatea
activitatea este trecută în fundal
activitatea este ascunsă
aplicații având prioritate mai mare solicită memoria folosită
activitatea este terminată sau este distrusă
utilizatorul accesează activitatea

Procese și threaduri

- **Procese**
  - default: ∀callback către∀ componentă ->main thread
  - nu se fac operații blocante în thread-ul principal:
    - Threaduri suplimentare
  - terminare/restartare proces – la discreția sistemului!
  - **Sistemul gestionează ciclul de viață**

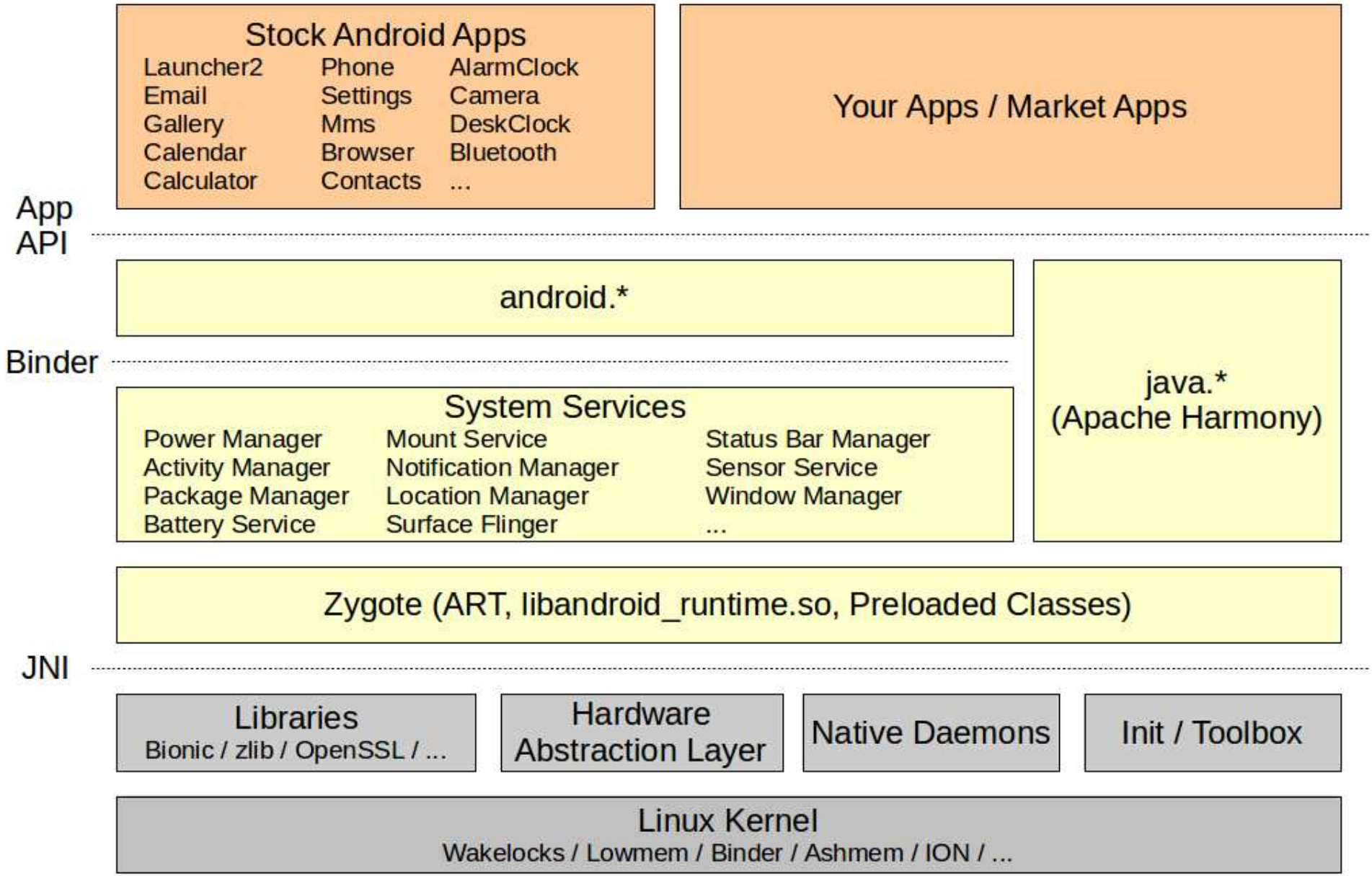
- **Threaduri**
  - create cu obiectul Java Thread
  - nu GUI în thread-ul de networking
  - nu networking în thread-ul de GUI

Remote procedure calls

- Nu există IPC de UNIX (semafoare , cozi de mesaje, pipe)
  - Dificilă portarea programelor UNIX
- Android RPC = Binder
- Binder in kernelul de Linux 3.19 (2015)

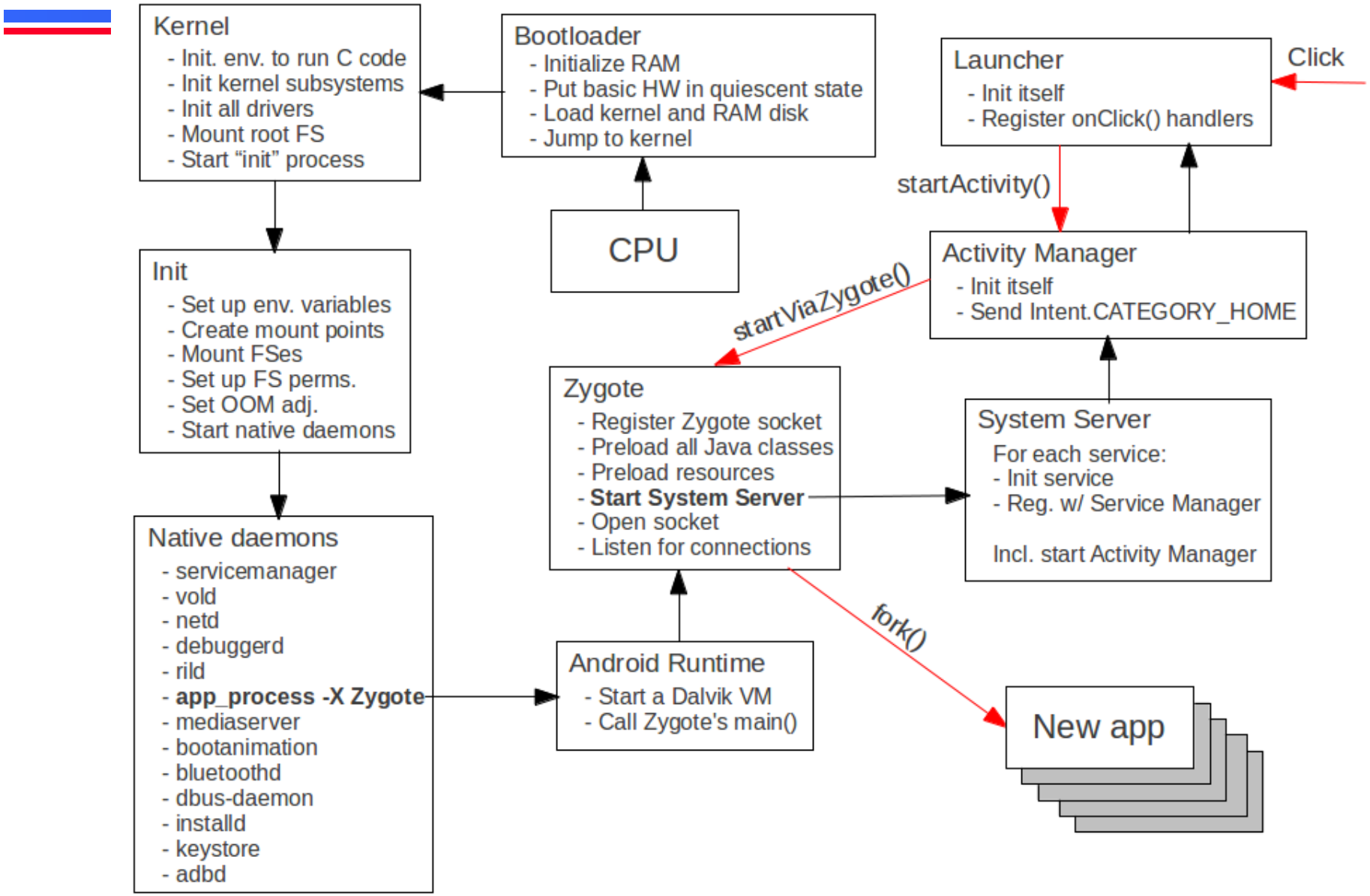
Securitate/Permisiuni

- Securitate la nivel de proces: UID, GID
- Permisiunile implementează restricții la:
  - Operații per-process
  - Acces per-URI
- Aplicațiile rulate în sandbox
- Permisiuni speciale pentru a ieși din sandbox
- Se dă accesul bazat pe:
  - Certificate
  - User prompt
- Toate permisiunile declarate static


Stock Android Apps
Launcher2
Email
Gallery
Calendar
Calculator
Phone
Settings
Mms
Browser
Contacts
AlarmClock
Camera
DeskClock
Bluetooth
...
Your Apps / Market Apps
App API
android.*
java.* (Apache Harmony)
Binder
System Services
Power Manager
Activity Manager
Package Manager
Battery Service
Mount Service
Notification Manager
Location Manager
Surface Flinger
Status Bar Manager
Sensor Service
Window Manager
...
Zygote (ART, libandroid_runtime.so, Preloaded Classes)
JNI
Libraries
Bionic / zlib / OpenSSL / ...
Hardware Abstraction Layer
Native Daemons
Init / Toolbox
Linux Kernel
Wakelocks / Lowmem / Binder / Ashmem / ION / ...

Secvența de startup

- Bootloader
- Kernel
- Init
- Zygote
- System Server
- Activity Manager
- Launcher (Home)


Kernel
- Init. env. to run C code
- Init kernel subsystems
- Init all drivers
- Mount root FS
- Start "init" process
Bootloader
- Initialize RAM
- Put basic HW in quiescent state
- Load kernel and RAM disk
- Jump to kernel
Launcher
- Init itself
- Register onClick() handlers
Click
CPU
startActivity()
Activity Manager
- Init itself
- Send Intent.CATEGORY_HOME
Init
- Set up env. variables
- Create mount points
- Mount FSes
- Set up FS perms.
- Set OOM adj.
- Start native daemons
startViaZygote()
Zygote
- Register Zygote socket
- Preload all Java classes
- Preload resources
- Start System Server
- Open socket
- Listen for connections
System Server
For each service:
- Init service
- Reg. w/ Service Manager
Incl. start Activity Manager
Native daemons
- servicemanager
- vold
- netd
- debuggerd
- rild
- app_process -X Zygote
- mediaserver
- bootanimation
- bluetoothd
- dbus-daemon
- installd
- keystore
- adbd
Android Runtime
- Start a Dalvik VM
- Call Zygote's main()
fork()
New app

Zygote

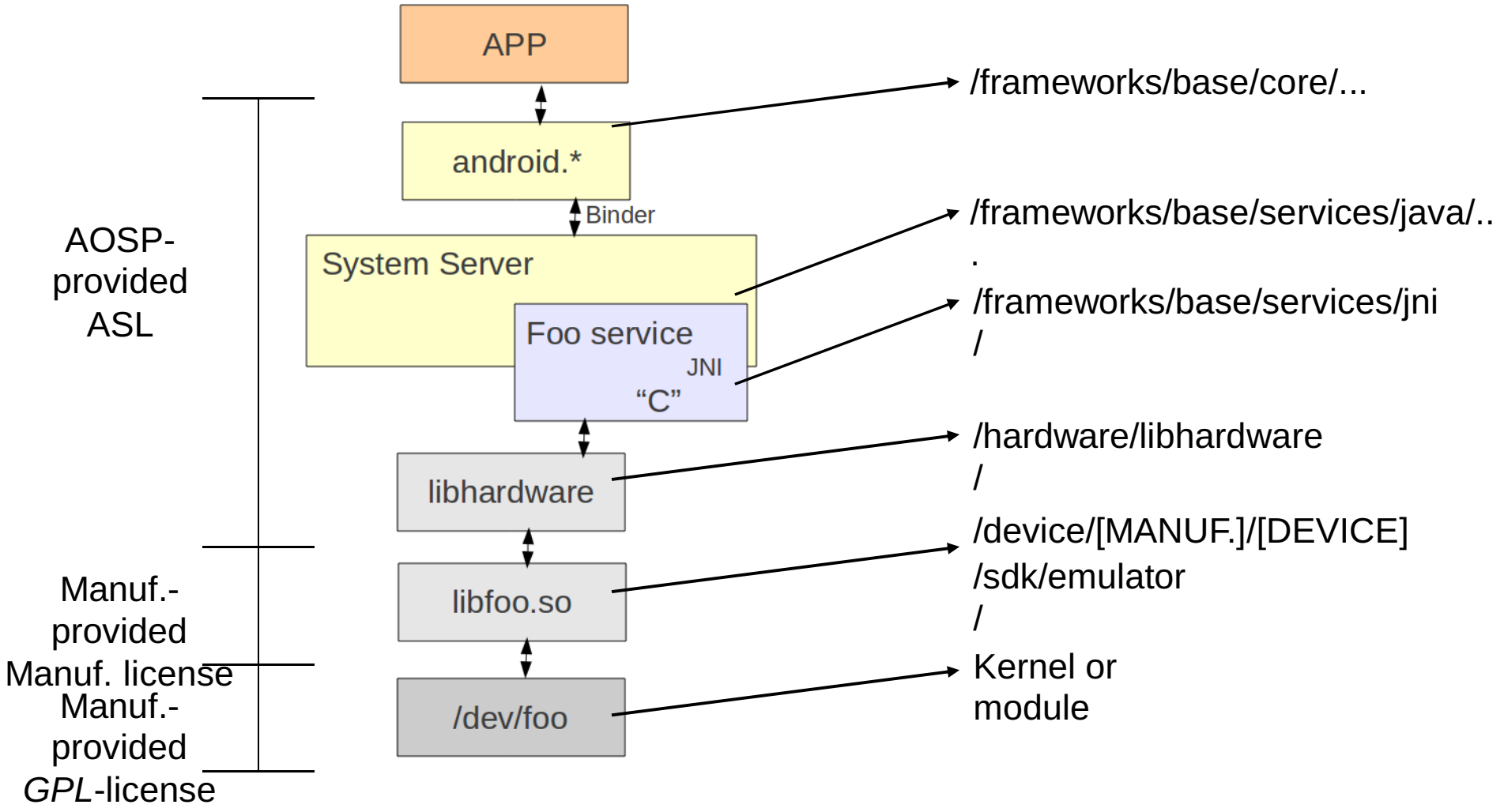
- Pornește după startup-ultipic de Linux
- Are toate bibliotecile încărcate
- Conține masina virtuala Java
  - Dalvik< 4
  - ART (Android RunTime) >= 4
- Fiecare proces nou este fork de zygote

Hardware Support

- hardware-ul nu este accesat direct
- HAL = hardware abstraction layer
- HAL “modules” are .so files

- Activity Recognition
- Audio
- Bluetooth
- Camera
- ConsumerIr
- Framebuffer
- Fingerprint
- Fused Location
- GPS
- Gralloc
- HWcomposer
- Keymaster
- Lights
- NFS
- Power
- Sensors

Hardware Abstraction Layer



User-Space Nativ

- Principale:
  - /data =>User data
  - /system => System components
  - /cache => Cache (& OTA update)
- Sistem Linux
  - /dev
  - /proc
  - /sys
  - /sbin
  - /mnt
- Multe directoare Linux lipsesc...

Java din Android

- Oracle (Sun)
  - Java = Java language + JVM + JDK libs
- Android < 5 (Lollipop)
  - Java = Java language + Dalvik
- Android < 7 (Nougat)
  - Java = Java language + ART + Apache Harmony
- Android >= 7
  - Java = Java language + ART + OpenJDK

Mașina virtuală java

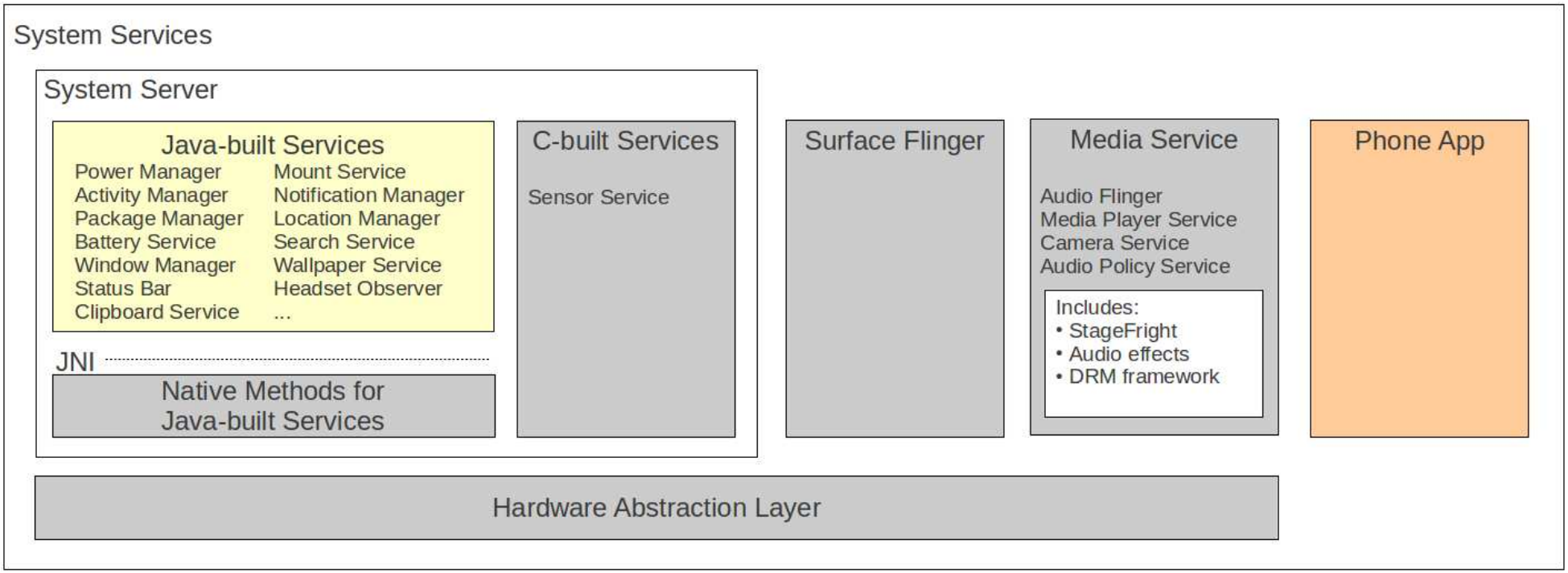
- Aplicațiile Android rulează propriul proces
  - Mașina virtuală java

- ART = Android RunTime
  - 64 bit
  - Multi-core
  - AOT instead of JIT

- Dalvik

Servicii de sistem

- Sistemul Android
- ~100 servicii (Lollipop)
- 5 - 6 servicii noi la fiecare release
- adb shell “service list” pentru listare
  - WifiService
  - ActivityManagerService
  - PowerService
  - PackageManagerService
  - LocationManagerService


System Services
System Server
Java-built Services
Power Manager
Activity Manager
Package Manager
Battery Service
Window Manager
Status Bar
Clipboard Service
Mount Service
Notification Manager
Location Manager
Search Service
Wallpaper Service
Headset Observer
...
JNI
Native Methods for Java-built Services
C-built Services
Sensor Service
Surface Flinger
Media Service
Audio Flinger
Media Player Service
Camera Service
Audio Policy Service
Includes:
• StageFright
• Audio effects
• DRM framework
Phone App
Hardware Abstraction Layer

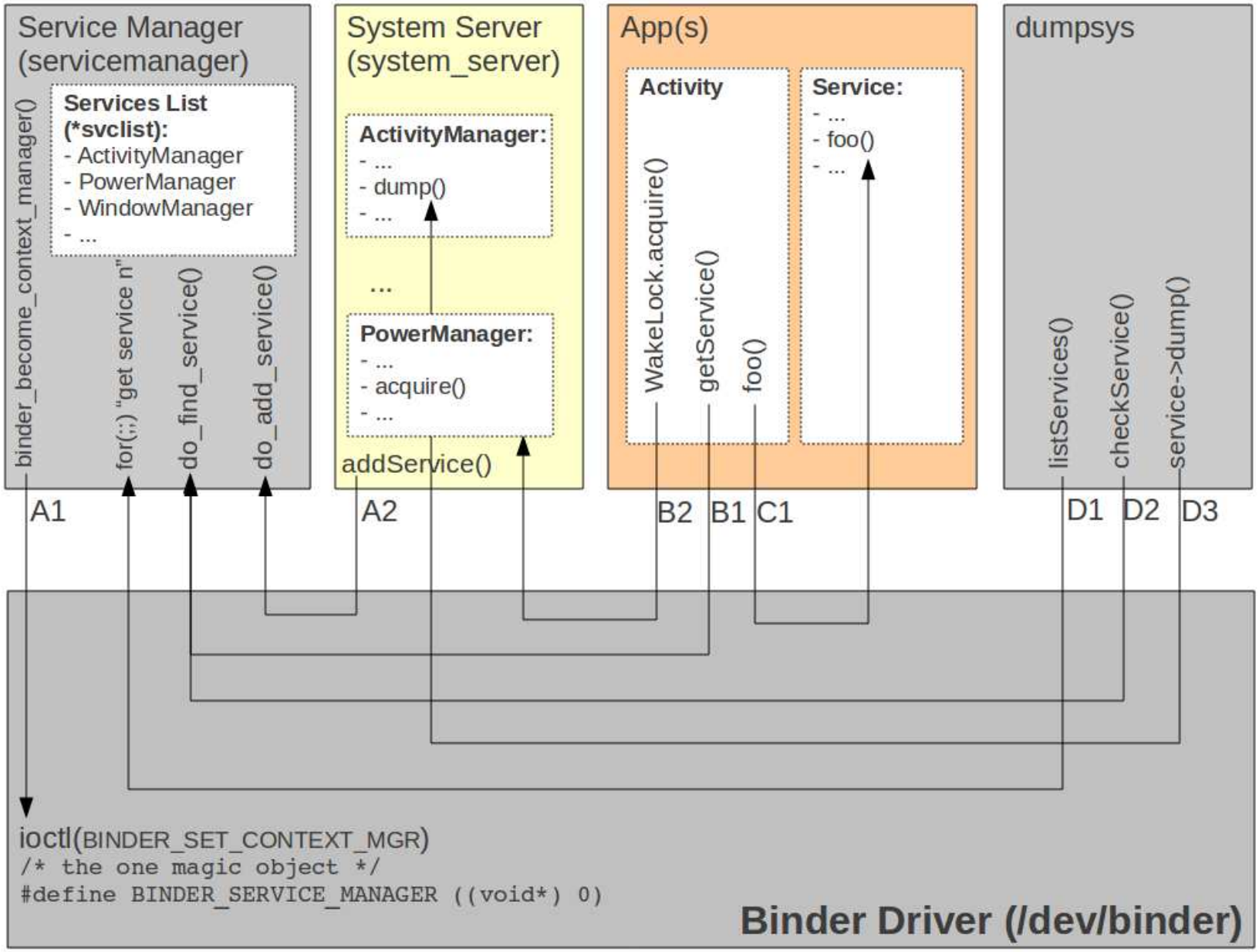
ActivityManager

- Pornește Activities, Services
- “găsește” Content Providers
- Intent broadcasting
- Application Not Responding
- Permisiuni
- Task management
- Lifecycle management

- **Exemplu:**
- Startăm app din Launcher:
- onClick(Launcher)
- startActivity(Activity.java)
- *<Binder>*
- ActivityManagerService
- startViaZygote(Process.java)
- *<Socket>*
- Zygote

Binder

- Datele trimise prin“parcels”, folosind “transactions”
- Implementat în kernel
- /dev/binder
- /sys/kerne/debug/binder/*
- android.* API se conectează la System Server prin binder


Service Manager
(servicemanager)
Services List
(*svclist):
- ActivityManager
- PowerManager
- WindowManager
- ...
binder_become_context_manager()
for(;;) "get service n"
do_find_service()
do_add_service()
System Server
(system_server)
ActivityManager:
- ...
- dump()
- ...
...
PowerManager:
- ...
- acquire()
- ...
addService()
App(s)
Activity
WakeLock.acquire()
getService()
foo()
Service:
- ...
- foo()
- ...
dumpsys
listServices()
checkService()
service->dump()
A1
A2
B2
B1
C1
D1
D2
D3
ioctl(BINDER_SET_CONTEXT_MGR)
/* the one magic object */
#define BINDER_SERVICE_MANAGER ((void*) 0)
Binder Driver (/dev/binder)

Gestiunea bateriei

- Bazat pe “Suspend to RAM” din Linux
- modulul PowerManager
  - Display, backlight, keyboard
  - Monitorizează bateria
  - Coordonează circuitul de încarcare
  - Necesită permisiunea WAKE_LOCK *
- Kernel Wake Locks: wlan_wake, alarm_rtc, suspend_backoff
- Partial Wake Locks

Flag	CPU	Screen	Keyboard
PARTIAL_WAKE_LOCK	On	Off	Off
SCREEN_DIM_WAKE_LOCK	On	Dim	Off
SCREEN_BRIGHT_WAKE_LOCK	On	Bright	Off
FULL_WAKE_LOCK	On	Bright	Bright

* **https://forum.xda-developers.com/showthread.php?t=1827676**