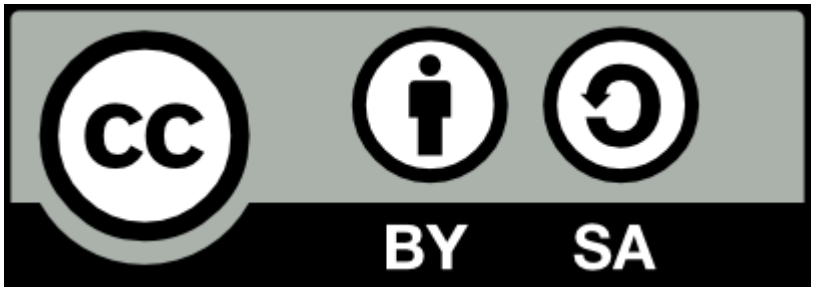
Android Platform Debugging and Development

AnDevCon Boston 2015

Karim Yaghmour
@karimyaghmour
karim.yaghmour@opersys.com

>OPERSYS™



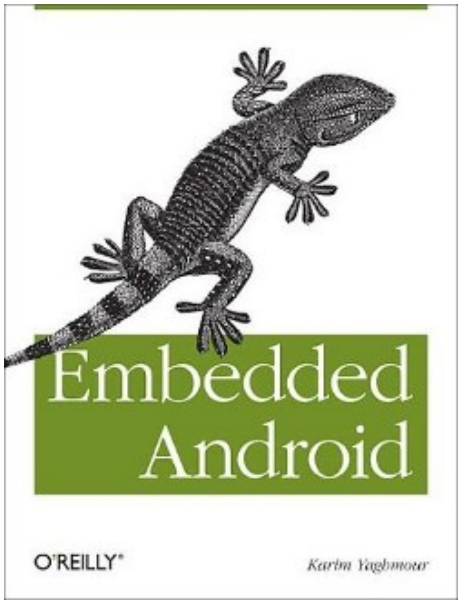
These slides created by: Karim Yaghmour

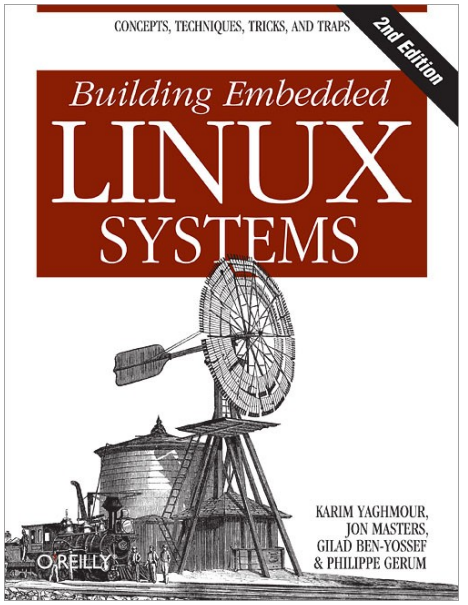
Originals at: www.opersys.com/community/docs

Delivered and/or customized by

About

- Author of:





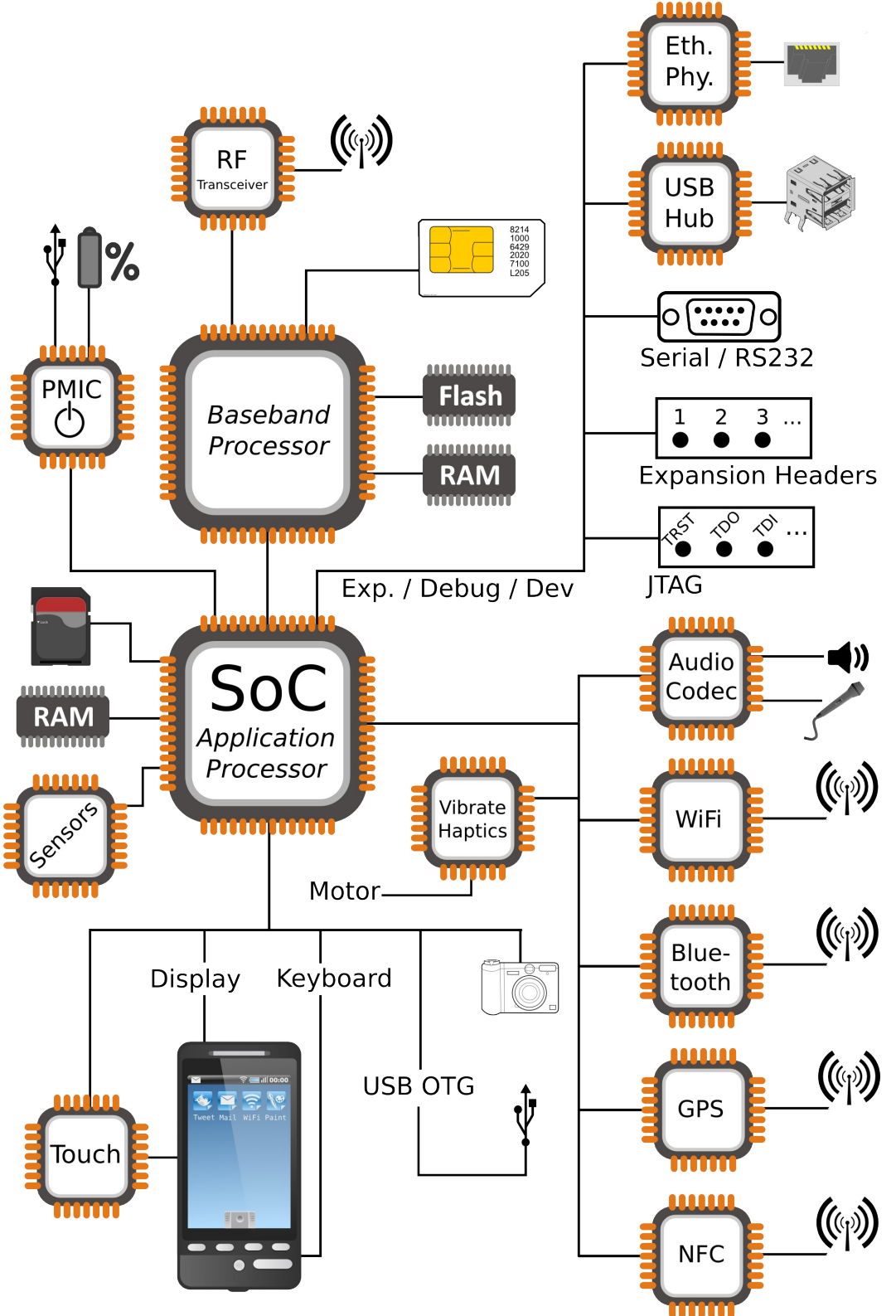
- Introduced Linux Trace Toolkit in 1999
- Originated Adeos and relayfs (kernel/relay.c)
- Ara Android Arch Oversight
- Training, Custom Dev, Consulting, ...

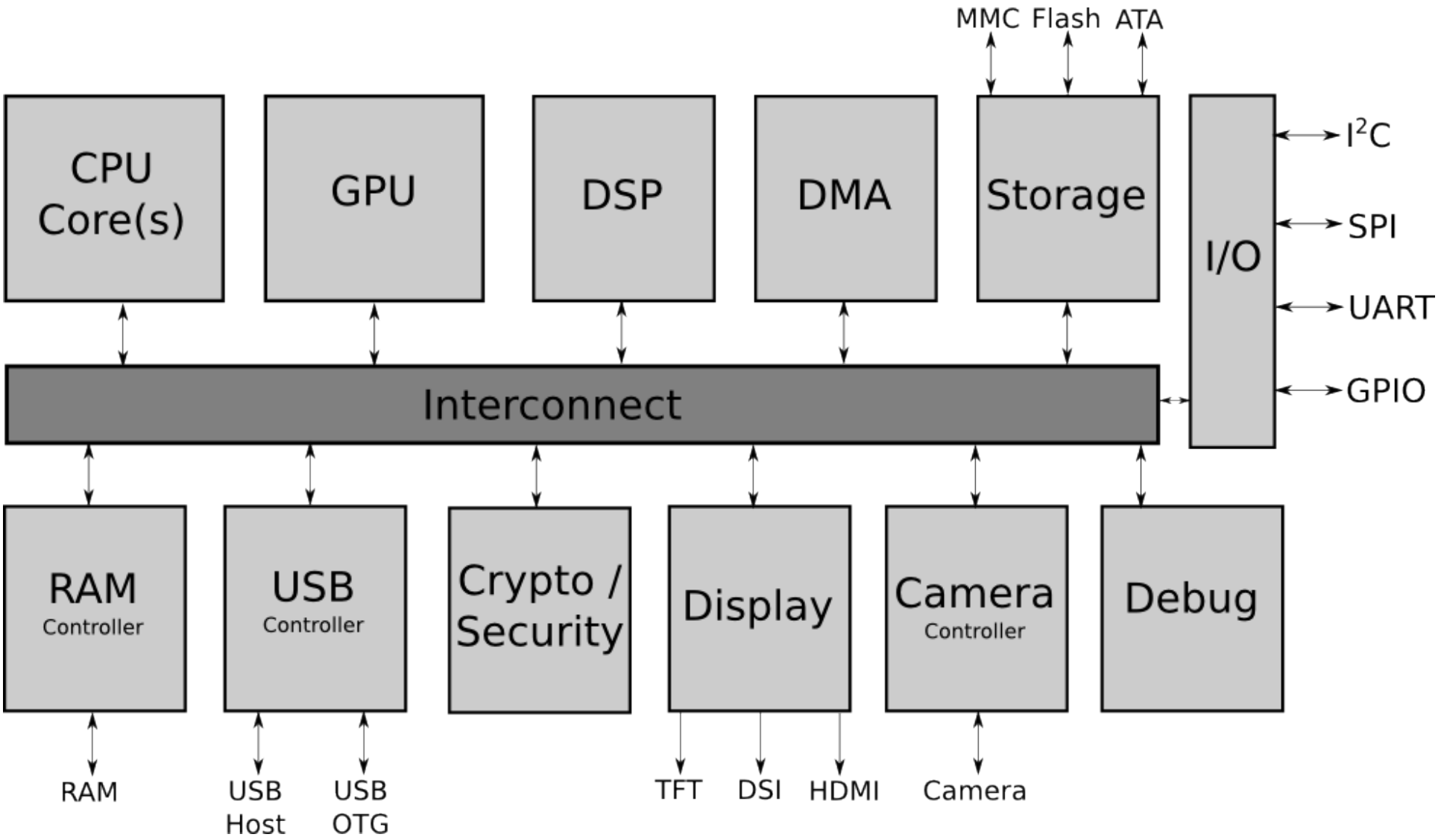
Agenda

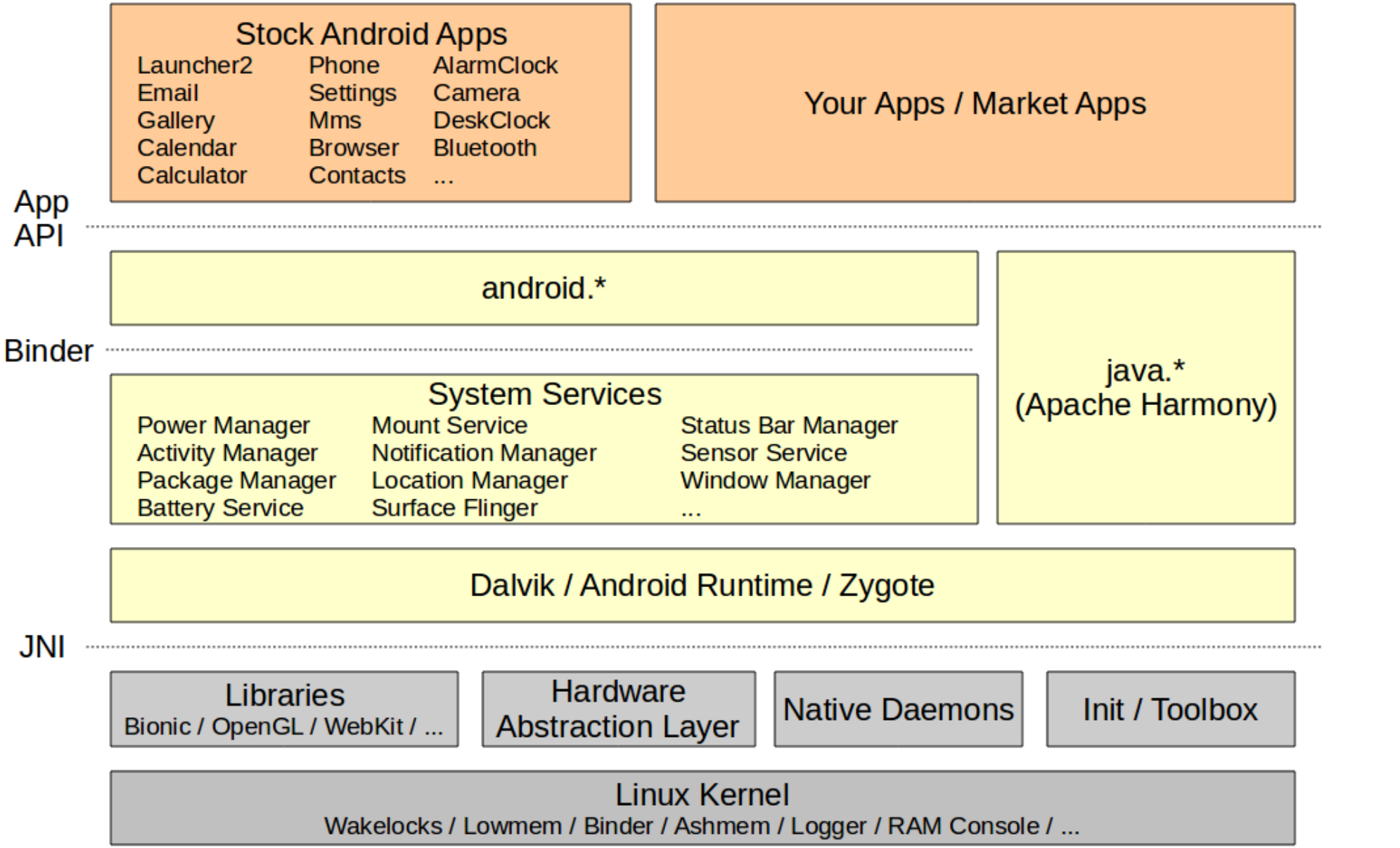
1. Architecture Basics
2. Development environment
3. Observing and monitoring
4. Interfacing with the framework
5. Working with the AOSP sources
6. Symbolic debugging
7. Detailed dynamic data collection
8. Benchmarking
9. Summing up

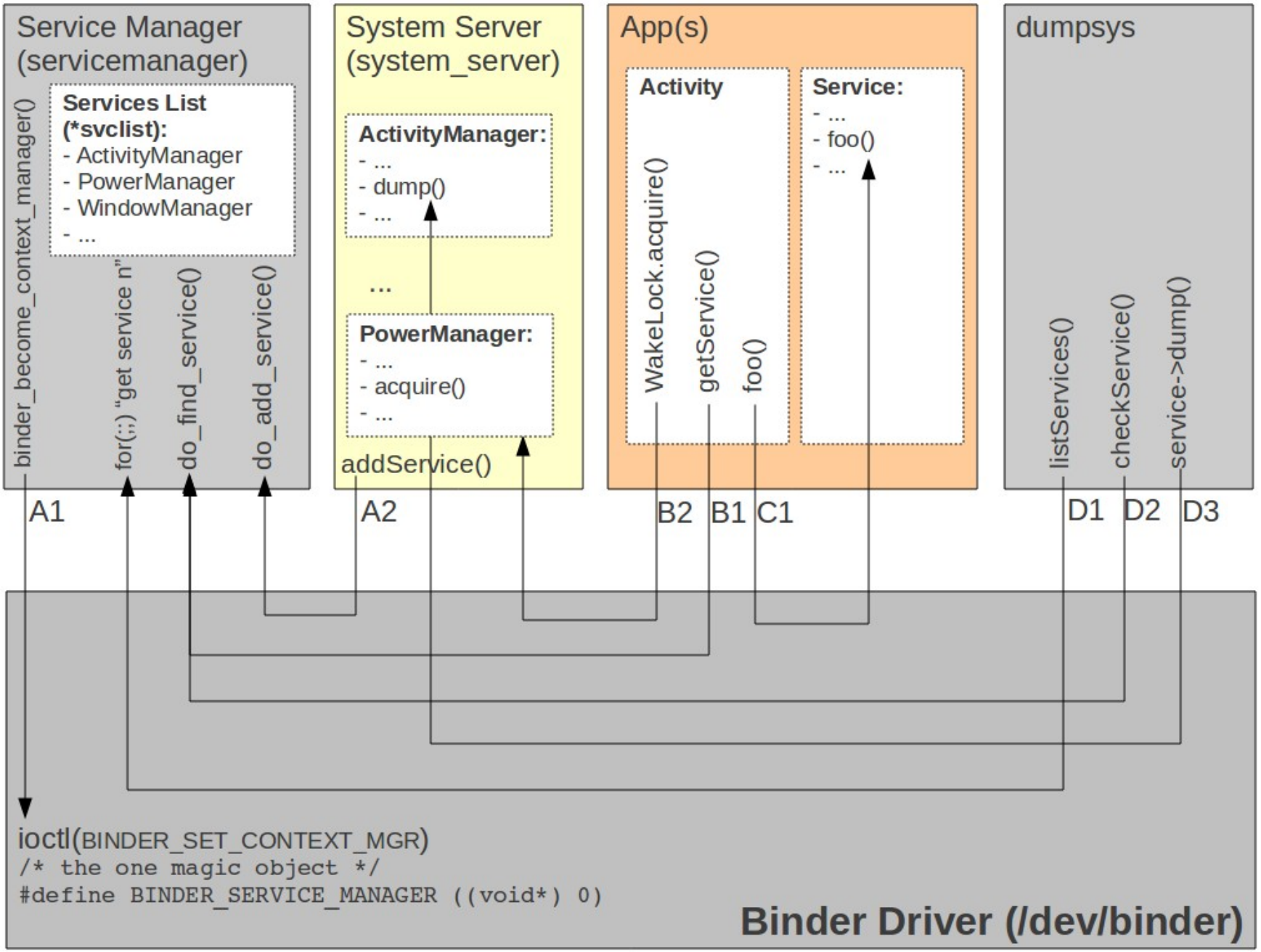
1. Architecture Basics

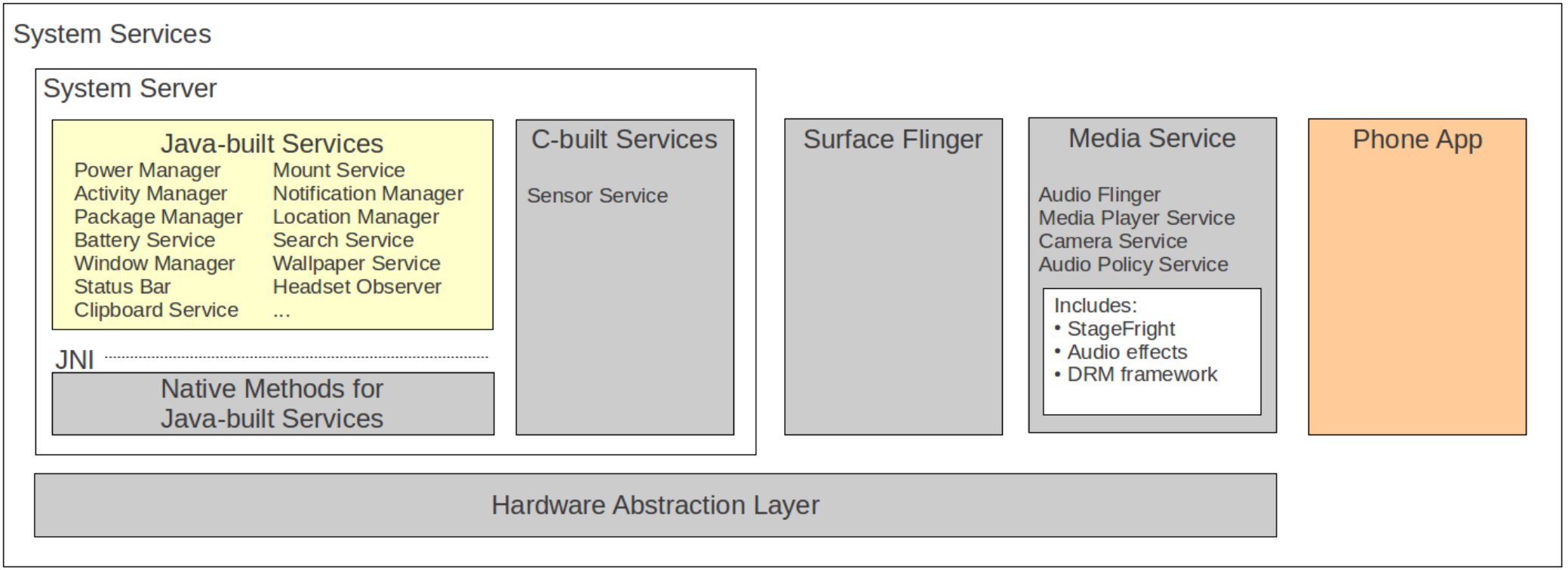
- Hardware used to run Android
- AOSP
- Binder
- System Services
- HAL

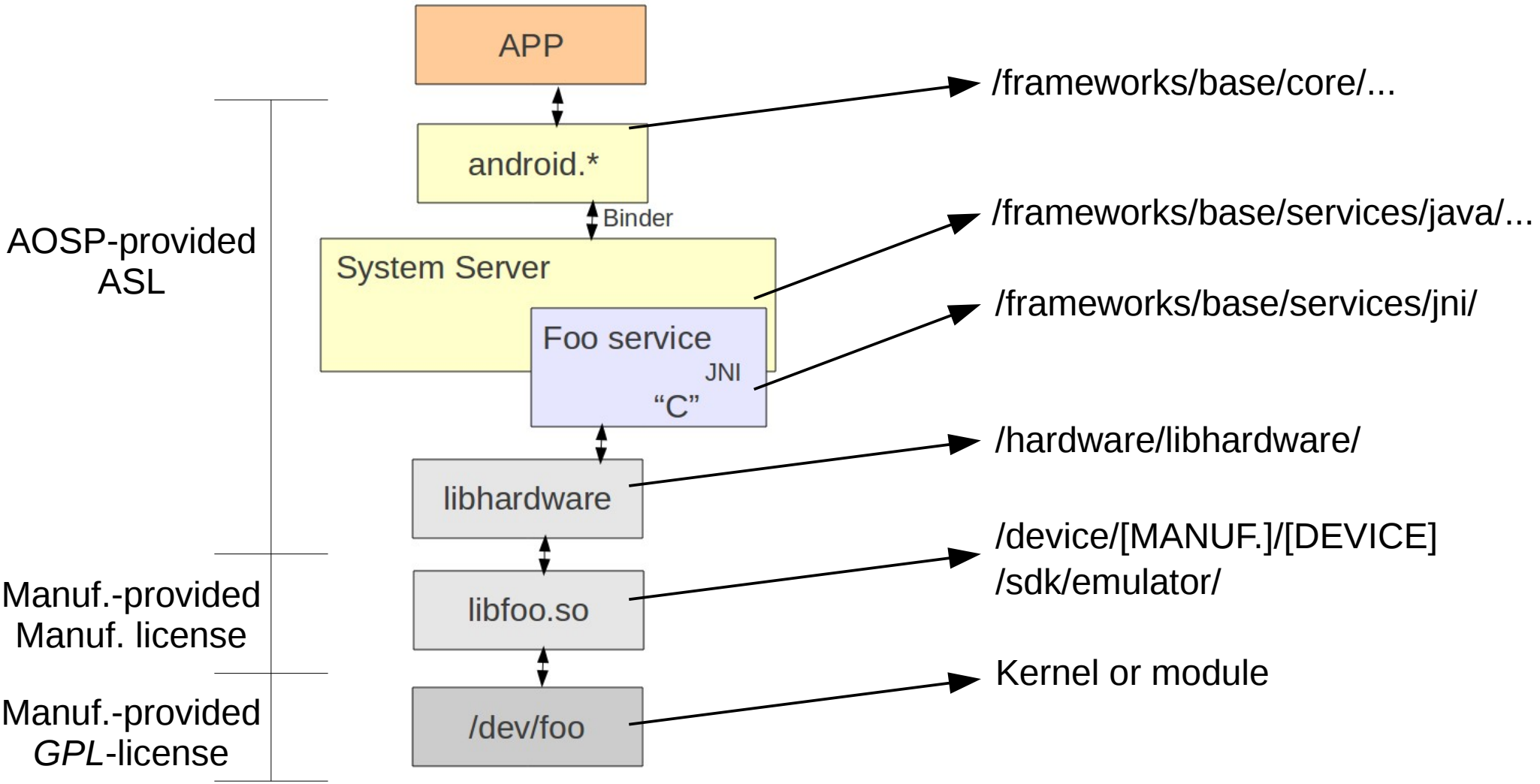

Eth.
Phy.
RF
Transceiver
USB
Hub
8214
1000
6429
2020
7100
L205
Serial / RS232
PMIC
Baseband
Processor
Flash
RAM
1 2 3 ...
Expansion Headers
TRST TDO TDI ...
JTAG
Exp. / Debug / Dev
SoC
Application
Processor
RAM
Audio
Codec
Sensors
Vibrate
Haptics
WiFi
Motor
Blue-
tooth
Display
Keyboard
GPS
Touch
USB OTG
NFC


MMC
Flash
ATA
CPU Core(s)
GPU
DSP
DMA
Storage
I/O
I²C
SPI
UART
GPIO
Interconnect
RAM Controller
USB Controller
Crypto / Security
Display
Camera Controller
Debug
RAM
USB Host
USB OTG
TFT
DSI
HDMI
Camera


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


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)

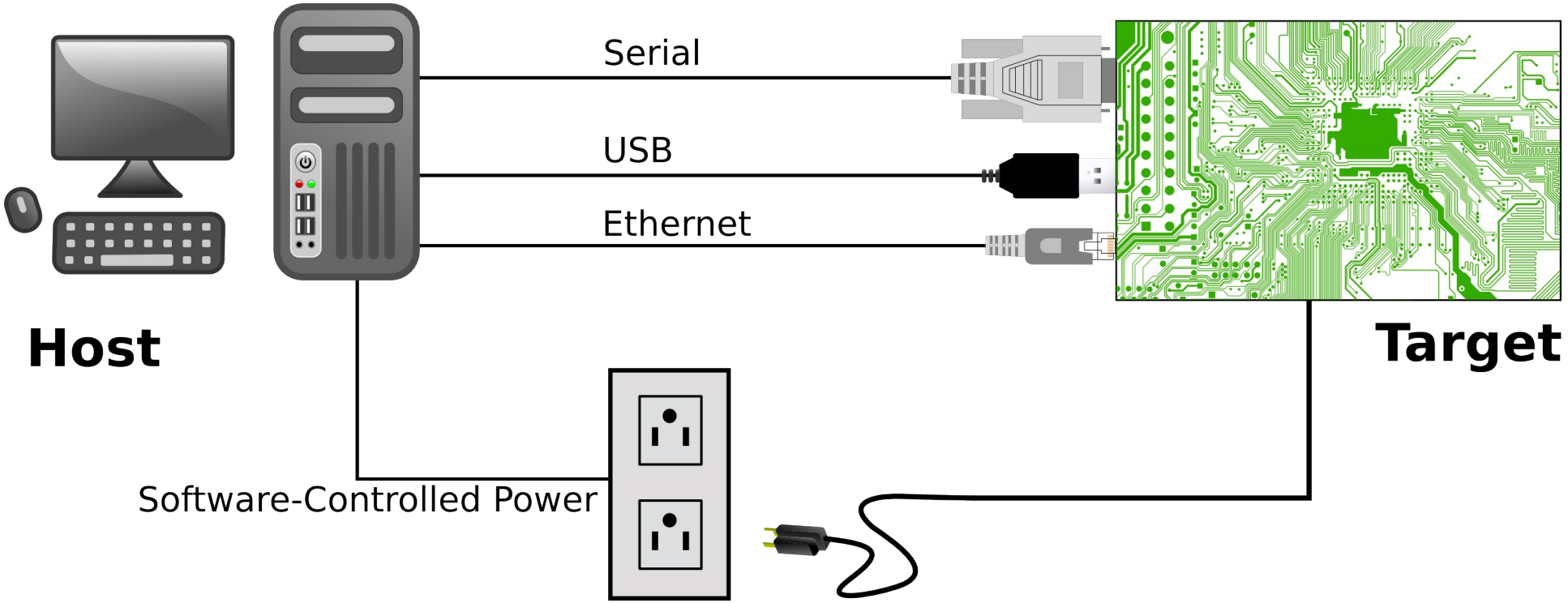

System Services
System Server
Java-built Services
Power Manager
Activity Manager
Package Manager
Battery Service
Window Manager
Status Bar
Clipboard Service
Mount Service
Notification Manager
Location Manager
Search Service
Wallpaper Service
Headset Observer
...
JNI
Native Methods for Java-built Services
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

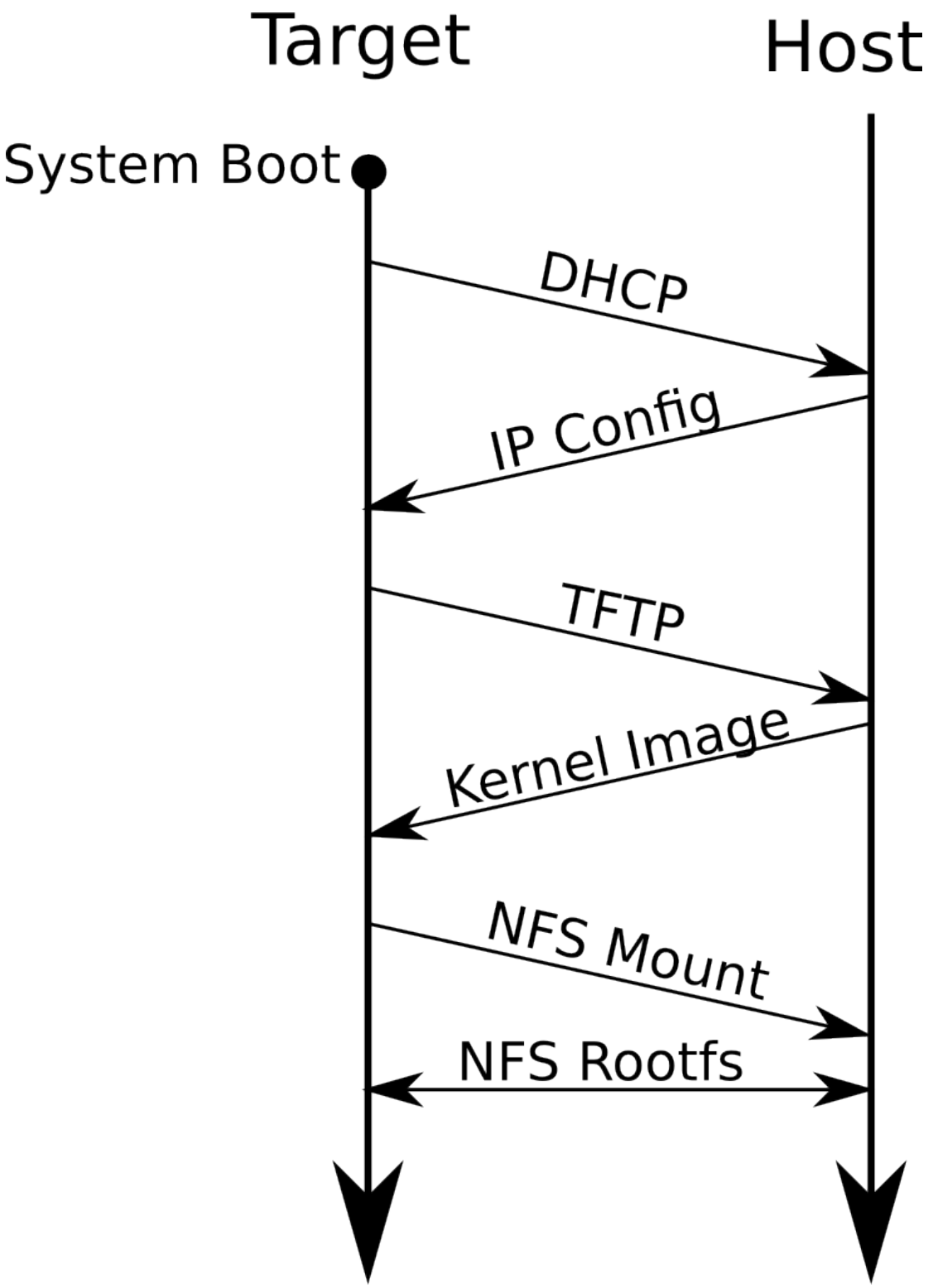

APP
android.*
Binder
System Server
Foo service
JNI
"C"
libhardware
libfoo.so
/dev/foo
/frameworks/base/core/...
/frameworks/base/services/java/...
/frameworks/base/services/jni/
/hardware/libhardware/
/device/[MANUF.]/[DEVICE]
/sdk/emulator/
Kernel or module
AOSP-provided
ASL
Manuf.-provided
Manuf. license
Manuf.-provided
GPL-license

2. Development Environment

- Host / Target setup
- IDE / Editor
- Android Studio setup

2.1. Host / Target setup




Target
Host
System Boot
DHCP
IP Config
TFTP
Kernel Image
NFS Mount
NFS Rootfs

2.2. IDE / Editor

2.3. Android Studio Setup

- Preparation
- Project importing
- Browsing the sources

2.3.1. Preparation

- AOSP Basics:
  - Get AOSP ... from Google or otherwise
  - Extract if needed
  - Configure, build, etc.
- Android Studio:
  - Get Android Studio from developer.android.com
  - Extract
  - Start and update and if needed

- Creating AOSP project files for Studio:

```
[aosp]$ make idegen && development/tools/idegen/idegen.sh
```

- Sometimes you also need to fix an issue with "res.java":

```
[aosp]$ cd out/target/product/generic/obj/GYP/shared_intermediates

[aosp]$ mv res.java res.j && croot
```

2.3.2. Project importing

- Start Android Studio:
  - Choose "Open an Existing Android Studio Project"
  - Select android.ipr from AOSP
  - Let it finish indexing
- To force framework detection -- if no auto-detect:
  - Close Studio
  - Restart Studio
  - Click on "Framework Detected" bubble

- Edit packages/apps/Launcher/src/com/android/launcher2/DragLayer.java and modify:

```
private boolean isLayoutRtl() {
```

- to

```
public boolean isLayoutRtl() {
```

- **Now: right-click on project and select "Refresh"**
- It might still show "x" on some parts until it's done rebuilding the project

2.3.3. Browsing the sources

- Right-click object type to be taken to declaration
- Browse classes through “Structure”
- Right-click "Find Usages"
- Toggle open files (Alt-left, Alt-right)
- Many other shortcuts, see:

  https://developer.android.com/sdk/installing/studio-tips.htm
- Issues:
  - Can't compile with Studio ... still need “make”
  - For Java only

3. Observing and Monitoring

- Native
- Framework
- Overall
- Apps / Add-ons

3.1. Native

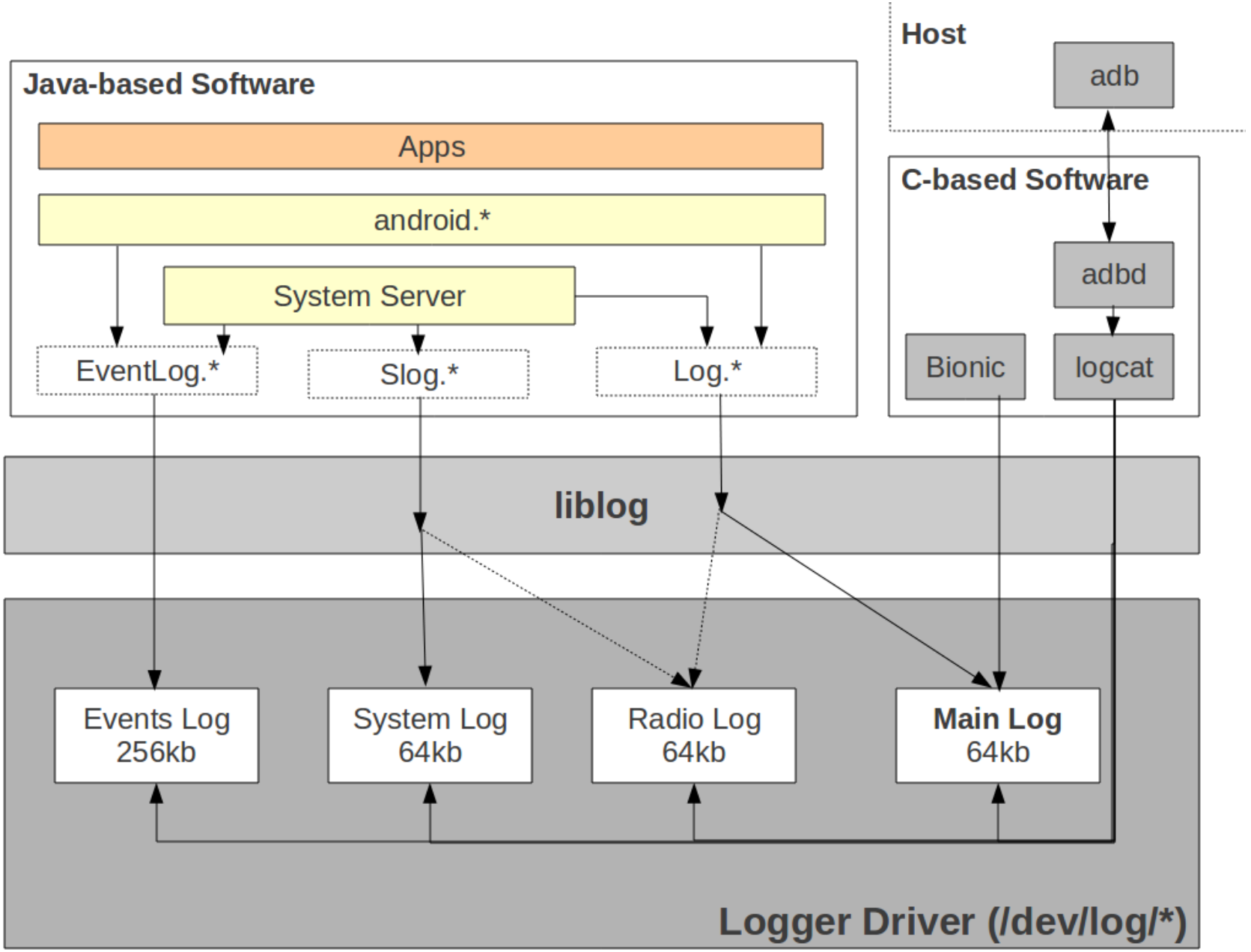
- schedtop
- librank
- procmem
- procrank
- showmap
- latencytop

3.2. Framework

- dumpsys
- service

3.3 Overall

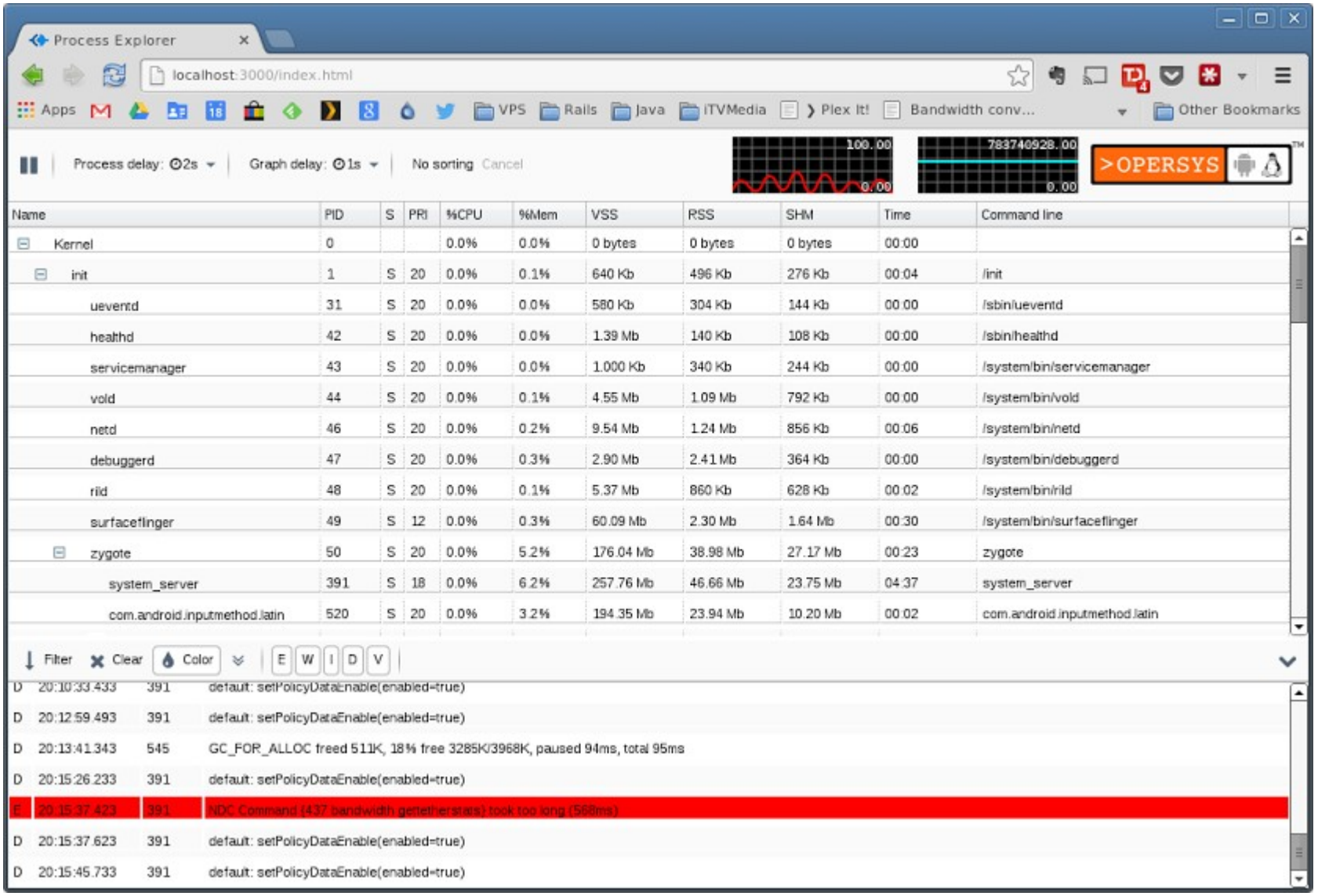
- logcat
- dumpstate / bugreport
- watchprop / getprop


Host
adb
Java-based Software
Apps
android.*
System Server
EventLog.*
Slog.*
Log.*
C-based Software
adbd
Bionic
logcat
liblog
Events Log
256kb
System Log
64kb
Radio Log
64kb
Main Log
64kb
Logger Driver (/dev/log/*)

3.4. Apps / Add-ons

- Google Play:
  - Process Manager
  - Process Monitor
  - Task Manager
  - Process Tracker
  - ...

3.5. Process Explorer



github.com/opersys

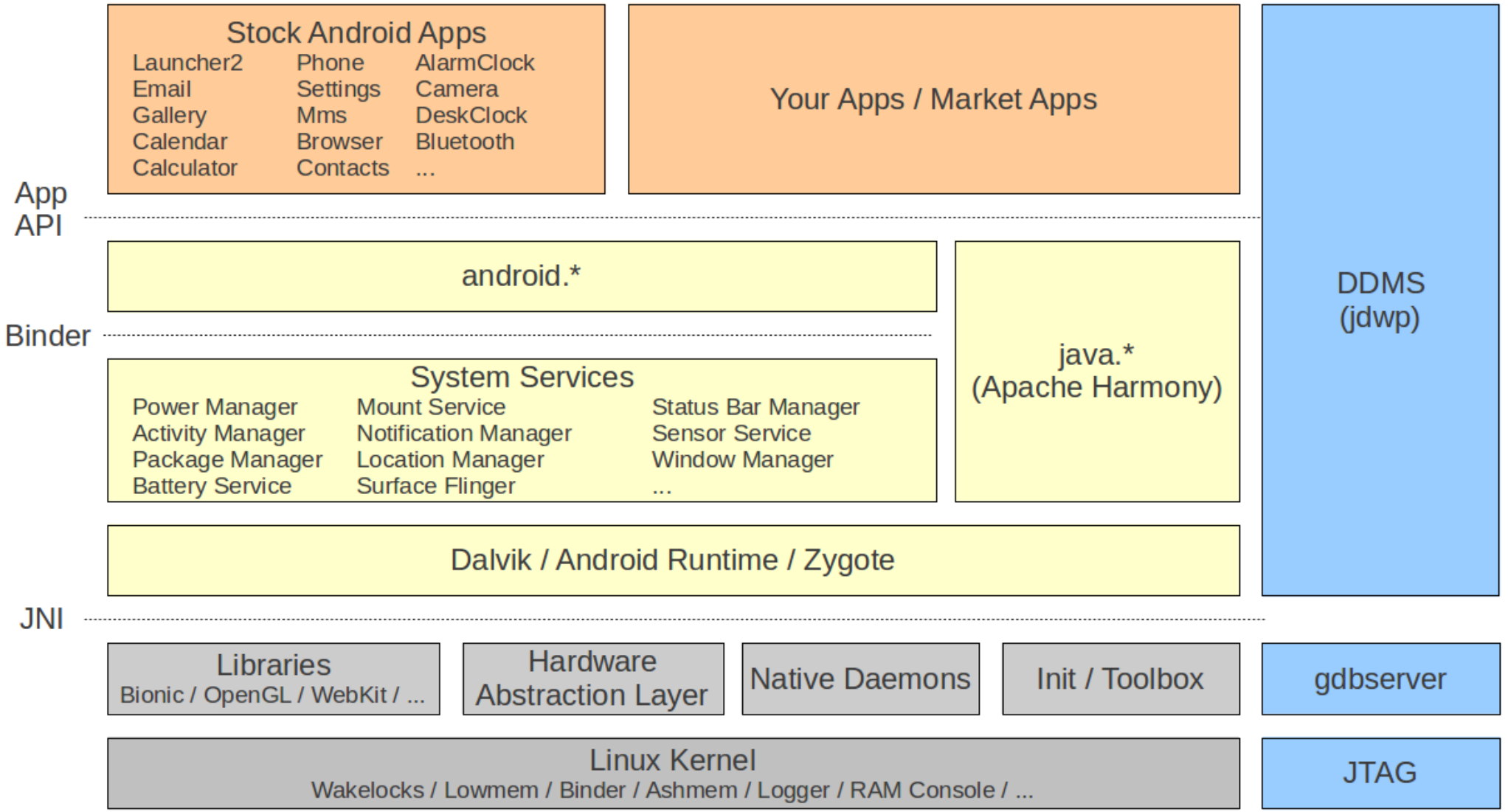
4. Interfacing With the Framework

- start / stop
- service call
- am
- pm
- wm
- svc
- monkey
- setprop
- raidl

5. Working with the AOSP Sources

- You really need to check build/envsetup.sh
- Some tricks:
  - godir
  - croot
  - mm
  - m
  - jgrep
  - cgrep
  - resgrep
- It takes time to wrap your head around the tree

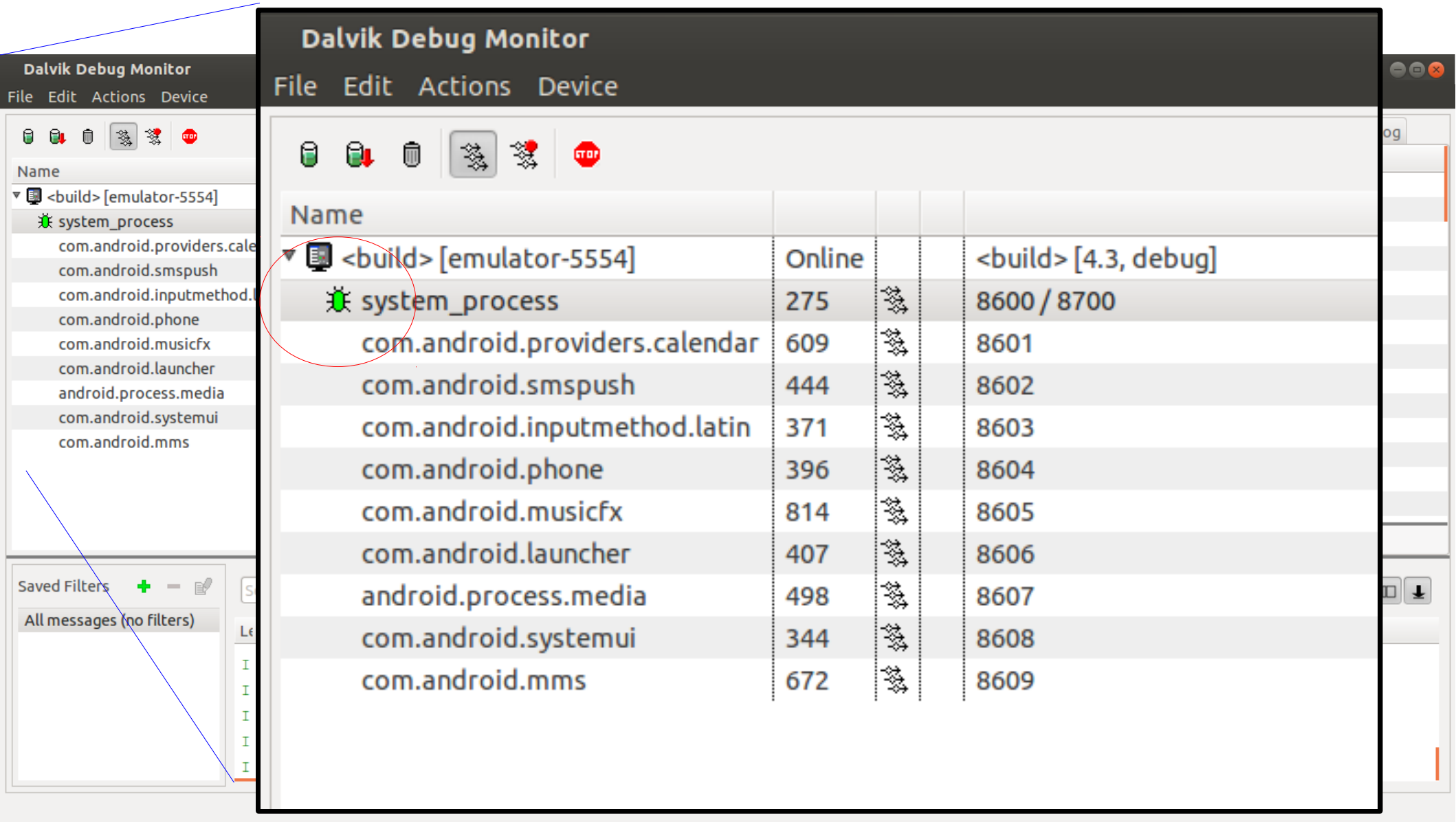
6. Symbolic Debugging - basics



6.1. Studio / Monitor integration

- Beware of libgail18 in Ubuntu
- Start Studio
- Start Monitor
- ("Android" icon on toolbar)
- Each process has a separate host-side socket
- Select the process you want to debug:
  - It'll get port 8700

- Go back to Studio:
  - Run->Edit Configurations->"+"
  - Remote->Port: 8700
  - Apply & Debug
- Go back to Monitor:
  - Check that the little green bug is beside your process in ddms
- You're now ready to debug


Dalvik Debug Monitor
File Edit Actions Device
Name
<build> [emulator-5554]
system_process
com.android.providers.cale
com.android.smspush
com.android.inputmethod.l
com.android.phone
com.android.musicfx
com.android.launcher
android.process.media
com.android.systemui
com.android.mms
Saved Filters
All messages (no filters)
Dalvik Debug Monitor
File Edit Actions Device
Name
<build> [emulator-5554] Online <build> [4.3, debug]
system_process 275 8600 / 8700
com.android.providers.calendar 609 8601
com.android.smspush 444 8602
com.android.inputmethod.latin 371 8603
com.android.phone 396 8604
com.android.musicfx 814 8605
com.android.launcher 407 8606
android.process.media 498 8607
com.android.systemui 344 8608
com.android.mms 672 8609

6.2. Debugging multiple processes

- Select process in Monitor
- Go back to Studio and start a new debugging session
- Each process will now have a green bug beside it

6.4. gdbserver - target side

- AOSP already takes care of debug:
  - “-g” flag added to all native binaries
  - Unstripped binaries in out/target/product/.../symbols/...
- Attaching to running process

```
# gdbserver --attach locahost:2345 30
```

- Start app for debugging with gdbserver prepended

```
# gdbserver localhost:2345 service list
```

- Forward the port on the host:

```
$ adb forward tcp:2345 tcp:2345
```

6.5. gdb - host side

- Load file **FIRST** and then attach on host side

```
$ prebuilts/gcc/linux-x86/arm/arm-eabi-4.7/bin/arm-eabi-gdb
GNU gdb (GDB) 7.3.1-gg2
Copyright (C) 2011 Free Software Foundation, Inc.
...
(gdb) file out/target/product/generic/symbols/system/bin/service
(gdb) target remote localhost:2345
(gdb) b main
Cannot access memory at address 0x0
Breakpoint 1 at 0x2a00146c: file frameworks/native/cmds/service/service.cpp, line 59.
(gdb) cont
Continuing.
warning: Could not load shared library symbols for 11 libraries, e.g. /system/bin/linker.
...

Breakpoint 1, main (argc=2, argv=0xbe882b74) at frameworks/native/cmds/service/service.cpp:59
59  {
(gdb) n
60      sp<IServiceManager> sm = defaultServiceManager();
(gdb) n
59  {
(gdb) n
60      sp<IServiceManager> sm = defaultServiceManager();
(gdb) n
61      fflush(stdout);
```

6.6. JNI debugging

```
$ prebuilts/gcc/linux-x86/arm/arm-eabi-4.7/bin/arm-eabi-gdb
(gdb) target remote localhost:2345
(gdb) file out/target/product/msm8960/symbols/system/bin/app_process
(gdb) set solib-absolute-prefix out/target/product/msm8960/symbols/
(gdb) set solib-search-path out/target/product/msm8960/symbols/system/lib/
(gdb) b com_android_server_OpersysService.cpp:70
(gdb) cont
Continuing.
-----------------------------
root@android:/ # service call opersys 2 s16 adfasd
-----------------------------
[New Thread 576]
[Switching to Thread 576]

Breakpoint 1, write_native (env=0x5c94ad40, clazz=<value optimized out>,
    ptr=<value optimized out>, buffer=0xa4f00005)
    at frameworks/base/services/jni/com_android_server_OpersysService.cpp:72
72      if (dev == NULL) {
(gdb)
```

6.7. JTAG

- Requires hardware device
- Sometimes interfaces with gdb
- Not Android specific
- Some allow transparent kernel/user-space debug
- Don't know of any that go all the way up to Dalvik

7. Detailed Dynamic Data Collection

- Logging
- strace
- ftrace
- perf

7.1. Logging

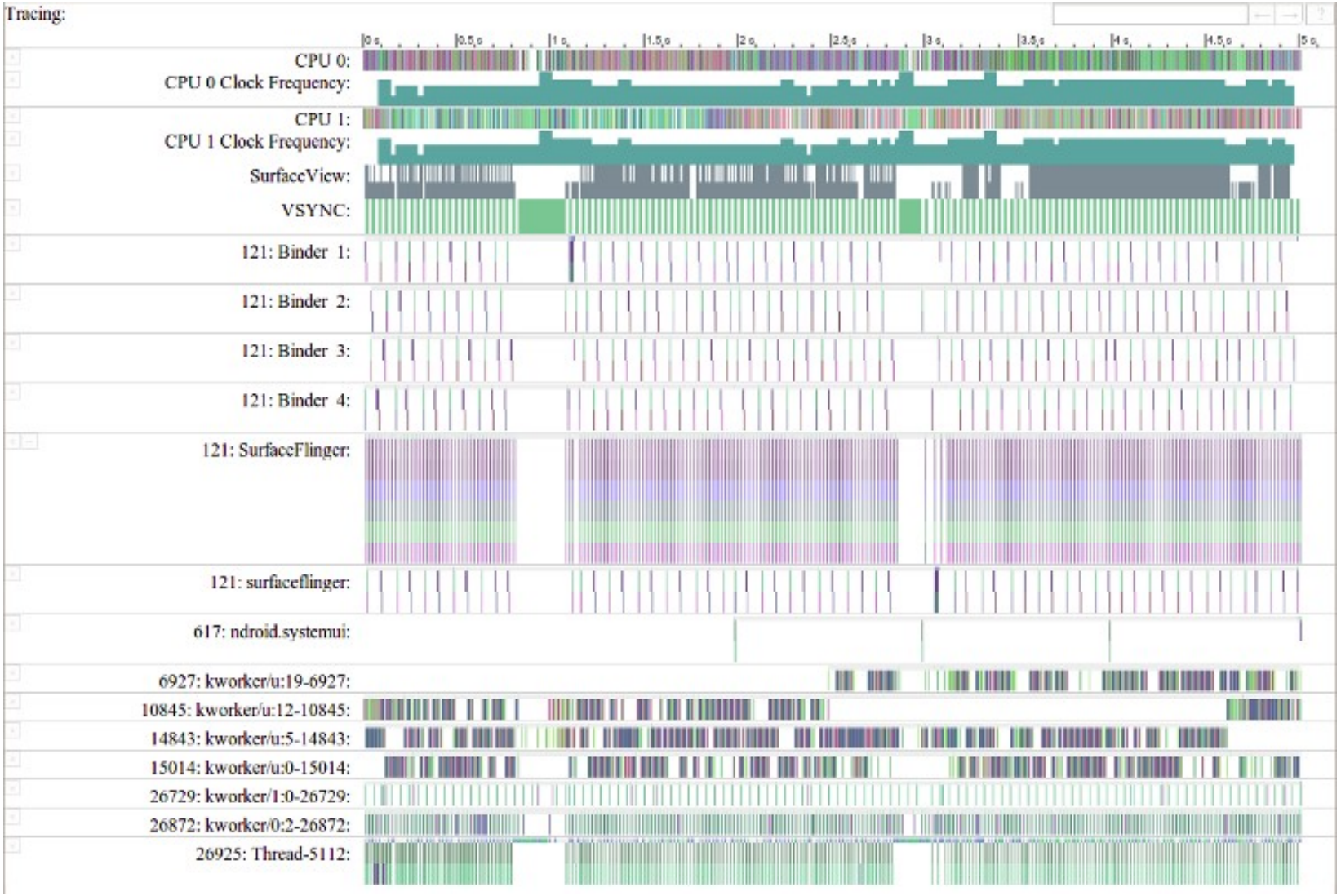
- logcat is the most rapid/consistent way to observe dynamic behavior.
- Trivial to add instrumentation points
- **It just works ...**

7.2. strace

- Same as Linux
- Use man page if need be

7.3. ftrace

- With 4.1, Google introduced systrace/atrace
- systrace is a Python script running on host side
- atrace is native Android binary
- systrace calls atrace via ADB
- atrace uses ftrace to capture kernel events
- Stack instrumented to feed events to ftrace
- Google's doc:
  - https://developer.android.com/tools/help/systrace.html
  - https://developer.android.com/tools/debugging/systrace.html


Tracing:
0 s
0.5 s
1 s
1.5 s
2 s
2.5 s
3 s
3.5 s
4 s
4.5 s
5 s
CPU 0:
CPU 0 Clock Frequency:
CPU 1:
CPU 1 Clock Frequency:
SurfaceView:
VSYNC:
121: Binder 1:
121: Binder 2:
121: Binder 3:
121: Binder 4:
121: SurfaceFlinger:
121: surfaceflinger:
617: ndroid.systemui:
6927: kworker/u:19-6927:
10845: kworker/u:12-10845:
14843: kworker/u:5-14843:
15014: kworker/u:0-15014:
26729: kworker/1:0-26729:
26872: kworker/0:2-26872:
26925: Thread-5112:

... trouble is ...

- Finicky -- notes from my attempts with 4.3:
  - I can't get it to work !*!@#$&!#*$!
  - Default goldfish kernel doesn't have ftrace
  - Able to build ftrace-enabled kernel for goldfish
  - Can trace that system ... so long as I **don't use** atrace/systrace ... WTF[1]?
- Not all Android kernels have ftrace enabled
- Generates HTML file that can only be read by Chrome ... **it doesn't work in Firefox**. NIH?

1: The AOSP sources define WTF as “What a Terrible Failure”. We trust they've done their research.

... still ...

- Have a look at these files:
  - /external/chromium-trace/systrace.py
  - /frameworks/native/cmds/atrace
  - /frameworks/base/core/java/android/os/Trace.java
  - /erameworks/native/include/utils/Trace.h
  - /system/core/include/cutils/trace.h
  - /frameworks/native/libs/utils/Trace.cpp
- Look for:
  - ATRACE* in c/cpp files
  - Trace.traceBegin()/trace.traceEnd() in Java files

```
# atrace --help
usage: atrace [options] [categories...]
options include:
  -a appname      enable app-level tracing for a comma separated list of
cmdlines
  -b N            use a trace buffer size of N KB
  -c              trace into a circular buffer
  -k fname,...    trace the listed kernel functions
  -n              ignore signals
  -s N            sleep for N seconds before tracing [default 0]
  -t N            trace for N seconds [defualt 5]
  -z              compress the trace dump
  --async_start   start circular trace and return immediatly
  --async_dump    dump the current contents of circular trace buffer
  --async_stop    stop tracing and dump the current contents of circular
                    trace buffer
  --list_categories
                  list the available tracing categories
```

```
# atrace --list_categories
          gfx - Graphics
        input - Input
         view - View System
      webview - WebView
           wm - Window Manager
           am - Activity Manager
        audio - Audio
        video - Video
       camera - Camera
          hal - Hardware Modules
          res - Resource Loading
       dalvik - Dalvik VM
```

7.3. perf on Android on ARM

8. Benchmarking

0xbench
AnTuTu
Passmark
Vellamo
Geekbench2
SunSpider
GLBenchmakr
Quadrant Standard Edition
Linpack
Neocore
3DMark
Epic Citadel
Androbench
CF-bench
SD Tools
RL Benchmark: SQL
Benchmark & Tunning
A1 SD Bench
Quick Benchmark Lite
3DRating benchmark
Smartbench 2011
NenaMark
Rightware Browsermark
An3DBenchXL
CaffeineMark
NBench
Methanol
AndEBench
SmartBench 2012
RealPi

9. Summing Up

- Works relatively well:
  - logcat
  - Studio / Monitor
  - Framework tools
- Works ok:
  - gdb/gdbserver
  - native tools
  - ftrace
- Finicky:
  - systrace/atrace
  - perf

10. Loose ends

- debuggerd
- tombstones
- anr traces

Thank you ...

karim.yaghmour@opersys.com