CANONICAL

Ubuntu Touch Internals

Embedded Linux Conference 2014

Ricardo Salveti de Araujo <ricardo.salveti@canonical.com>
IRC: rsalveti

Ubuntu Engineering

Agenda

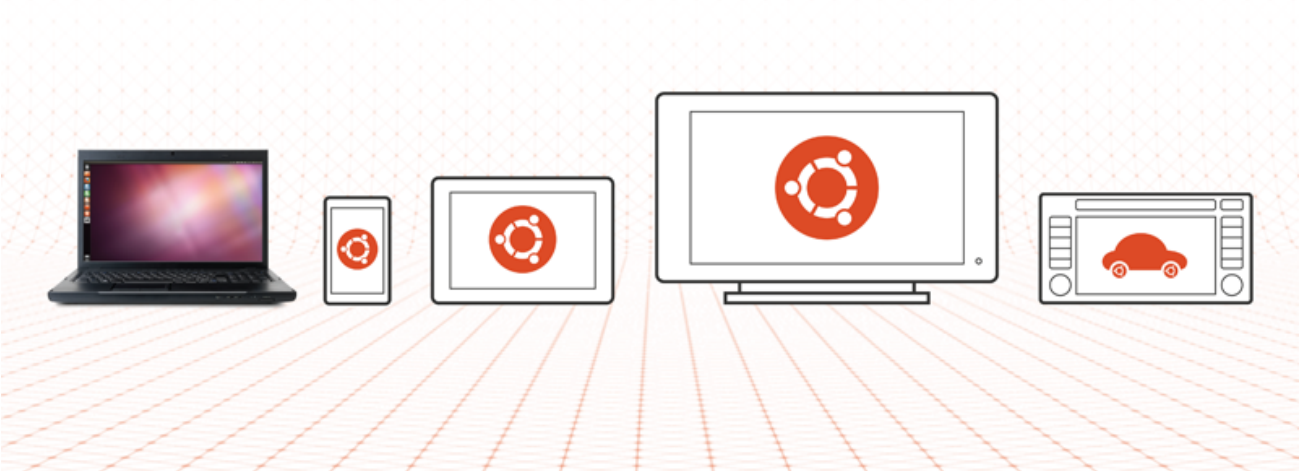
- Background
- Challenges
- Building a new Unity
- Reusing Android drivers
  - LibHybris
- Overall architecture
- Deep dive:
  - Telephony and Connectivity
  - Multimedia
  - Camera
- Future development
- Get involved!

Ubuntu Touch Background

- Mark Shuttleworth announced that Ubuntu would support Phones and Tablets in the end of 2011
  - Ubuntu was well supported on X86 and ARM already
  - Previous experience with Ubuntu Mobile / Netbook
  - Proposing a completely new UI design (under the Unity concept)
  - Requirement to be easily supported by a wide range of devices

Ubuntu Touch Background: Challenges

- Desktop Unity using Compiz + Nux
  - Complex design
  - Lacking proper support for OpenGL ES 2.0
- Quite a few components were not optimized for mobile (battery, background processes, usability, etc)
- Hardware accelerated stack without depending on the hw vendor
- Decision to develop a new stack, and make it generic enough so it could later be also shared with Desktop (convergence)




Search
12:10
Featured (21)
Dark Shadows
Source Code
Rango
Rented (35)
Purchased (42)
Recorded (42)
Home

New Unity: one that would rule them all

- Traditional stack composed of Compiz, Nux, Unity and X11
  - Not ideal for mobile, not properly compatible with OpenGL ES 2.0
  - Nux not so developer friendly
    - And not commonly known by developers
  - X11 was also not ideal, but a replacement was already on the way
    - Wayland and/or MIR

- Experience with Ubuntu Netbook (EFL) and later Unity 2D (Qt)
  - EFL fast and small, but API not that stable and issues with lack of development tools and documentation
  - Qt already supported and used by different targets and products
    - Great development tools and documentation
    - QML

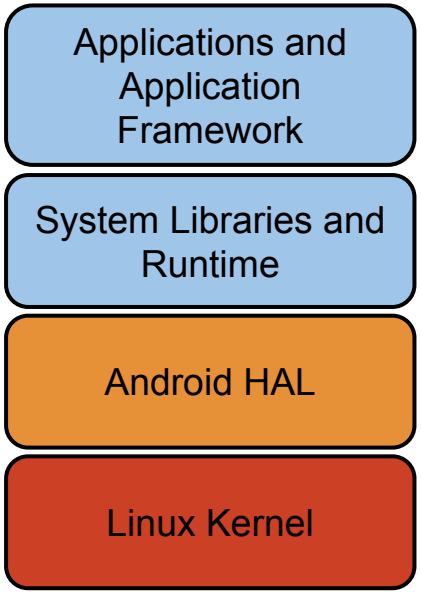
Unity8: Built with Qt and QML

- Decision to create a new Unity from scratch, using Qt 5.0
  - Qt was already quite well supported and known by developers
  - QML proved to be an easy and straightforward language/tool
  - Fully compatible with OpenGL ES 2.0
  - Different APIs and abstractions for many core components
  - Great Software Development Kit
  - Convergence in mind

- Only issue was finding hardware with decent drivers

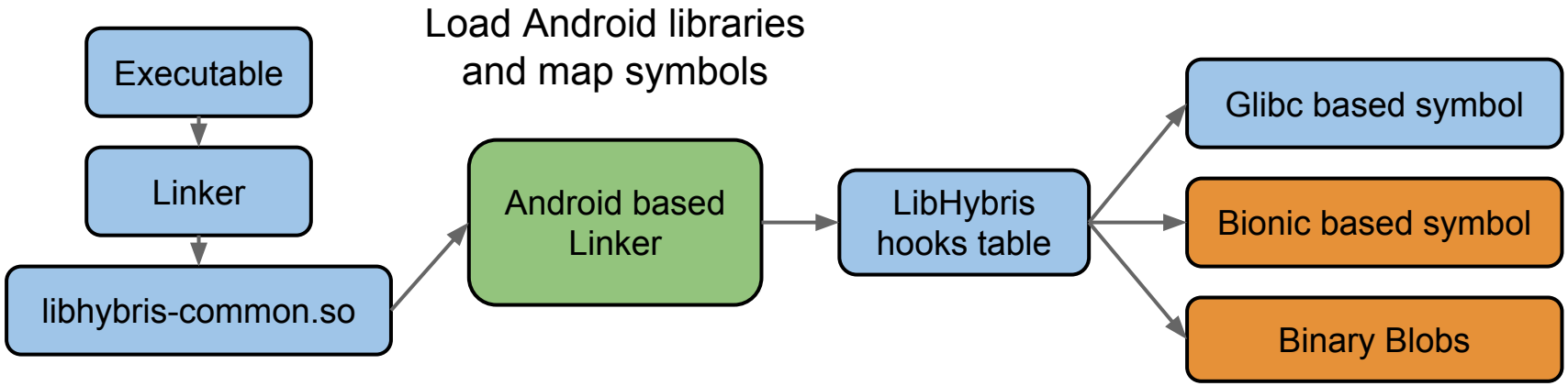
Reusing Android Drivers

- Android based devices largely available:
  - Decent drivers, but mostly closed source
  - Open Source code base, allowing us to read and modify it as needed
- Issues:
  - Highly connected with the Kernel version used by Android (along with the usual tons of vendor-specific modifications)
  - Android is built with Bionic instead of Glibc, types not necessarily compatible
  - Android core API/ABI is not necessarily stable, need to stick with a specific version (e.g. 4.4.2)



Reusing Android Drivers: LibHybris

- Compatibility layer for systems based on Glibc that allows Bionic based binaries to be used
- Created by Carsten Munk on August 2012
- Conceptually libhybris is a custom version of the Bionic linker, with hooks replacing Bionic symbols with Glibc compatible ones
- Main differences and issues with libhybris:
  - Android uses fixed TLS slots that can override glibc's TLS
  - Bionic pthreads implementation differs from glibc



LibHybris

hybris/common/jb/linker.c:

```
static int reloc_library(soinfo *si, Elf_Rel *rel, unsigned count)
{
    Elf_Sym *symtab = si->symtab;
    const char *strtab = si->strtab;
    (...)

    for (idx = 0; idx < count; ++idx) {
        (...)
        if (sym != 0) {
            sym_name = (char *)(strtab + symtab[sym].st_name);
            INFO("HYBRIS: '%s' checking hooks for sym '%s'\n", si->name,
                                                    sym_name);
            sym_addr = get_hooked_symbol(sym_name);
            if (sym_addr != NULL) {
                INFO("HYBRIS: '%s' hooked symbol %s to %x\n", si->name,
                                                sym_name, sym_addr);
            } else {
                s = _do_lookup(si, sym_name, &base);
            }
            (...)
}
```

LibHybris

hybris/common/hooks.c:

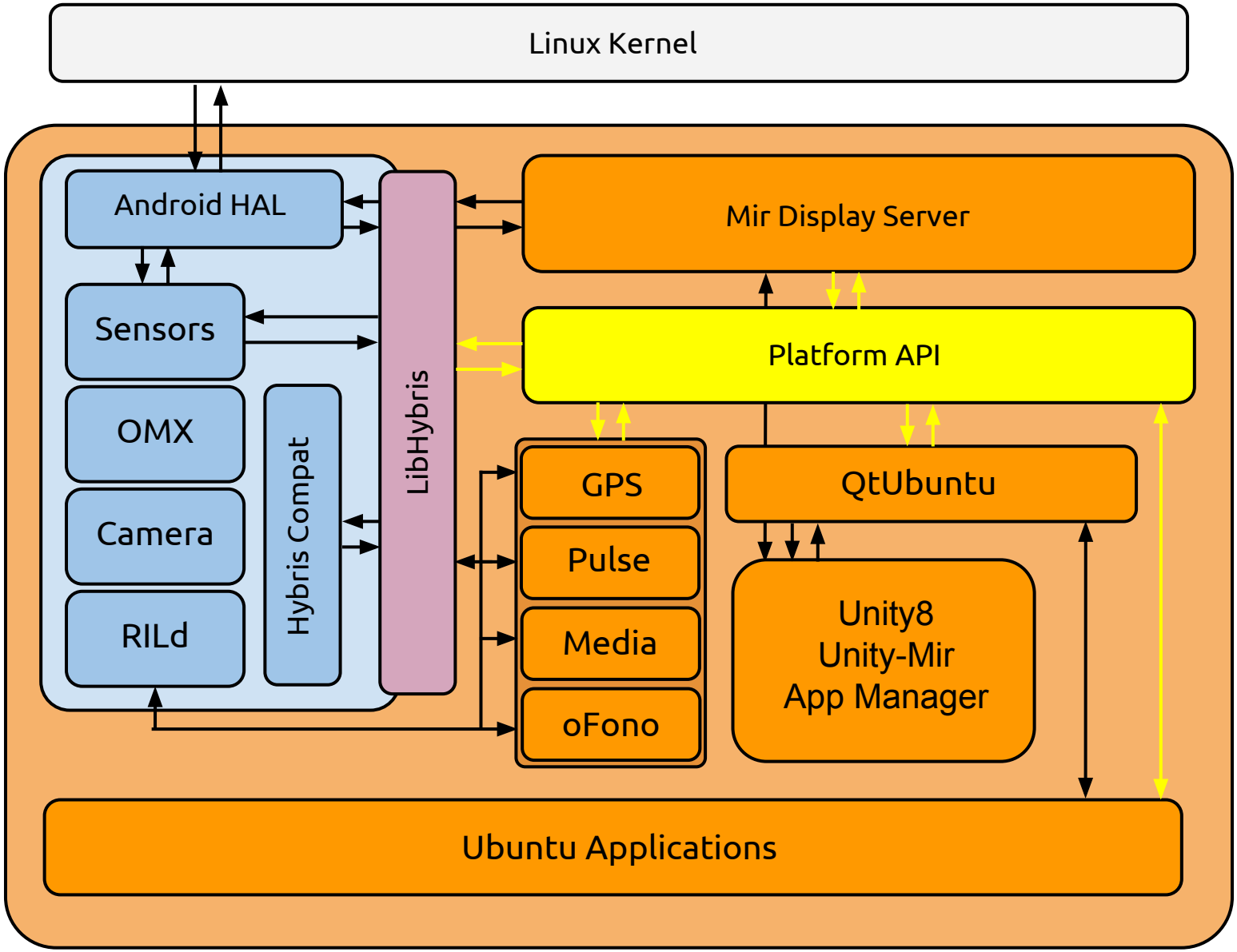
```
static struct _hook hooks[] = {
    {"property_get", property_get },
    {"property_set", property_set },
    {"printf", printf },
    {"malloc", my_malloc },
    (...)
}

void *get_hooked_symbol(char *sym)
{
    struct _hook *ptr = &hooks[0];
    static int counter = -1;

    while (ptr->name != NULL) {
        if (strcmp(sym, ptr->name) == 0) {
            return ptr->func;
        }
        ptr++;
    }
    (...)
}
```

Abstracting the Android Drivers

- Android system image isolated in a LXC container
  - Minimal image with only drivers and core system services
- LibHybris used to access and use the drivers
- API is specific to Android, not integrated with the desktop stack
  - Issue when thinking about convergence
  - When possible, create an Android abstraction for common components, such as:
    - Sensors
    - Multimedia (encode and decode)
    - Camera
    - Telephony

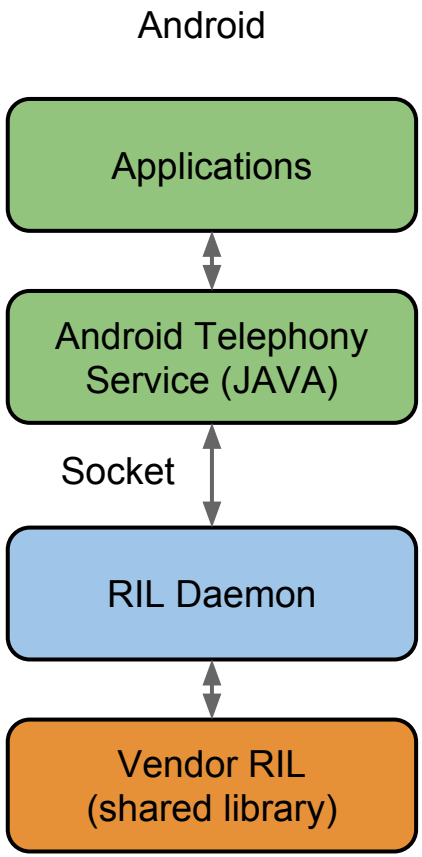
Architectural diagram of the overall system

- Platform API
  - Sensors
  - GPS
  - Multimedia
- Mir
  - Display Server
  - Abstraction for the OpenGL ES 2.0 drivers
  - Hardware Composer
- QtUbuntu
  - Qt Platform Abstraction plugin
  - Based on Platform API


Linux Kernel
Android HAL
Sensors
OMX
Camera
RILd
Hybris Compat
LibHybris
Mir Display Server
Platform API
GPS
Pulse
Media
oFono
QtUbuntu
Unity8
Unity-Mir
App Manager
Ubuntu Applications
Ubuntu 14.04
Android (LXC Container)

Telephony

- Hard to convince vendors to publish enough documentation to build an Open Source driver
- Android proposes an abstraction by providing a HAL and a specific protocol (Radio Layer Interface) for solicited and unsolicited commands
- Each vendor provides a binary blob that talks the RIL protocol
- RIL is separated in two layers:
  - Base layer that talks with the binary modem
  - Upper layer that talks to the base layer using the RIL protocol, over a socket



Telephony and Connectivity: Ubuntu Touch

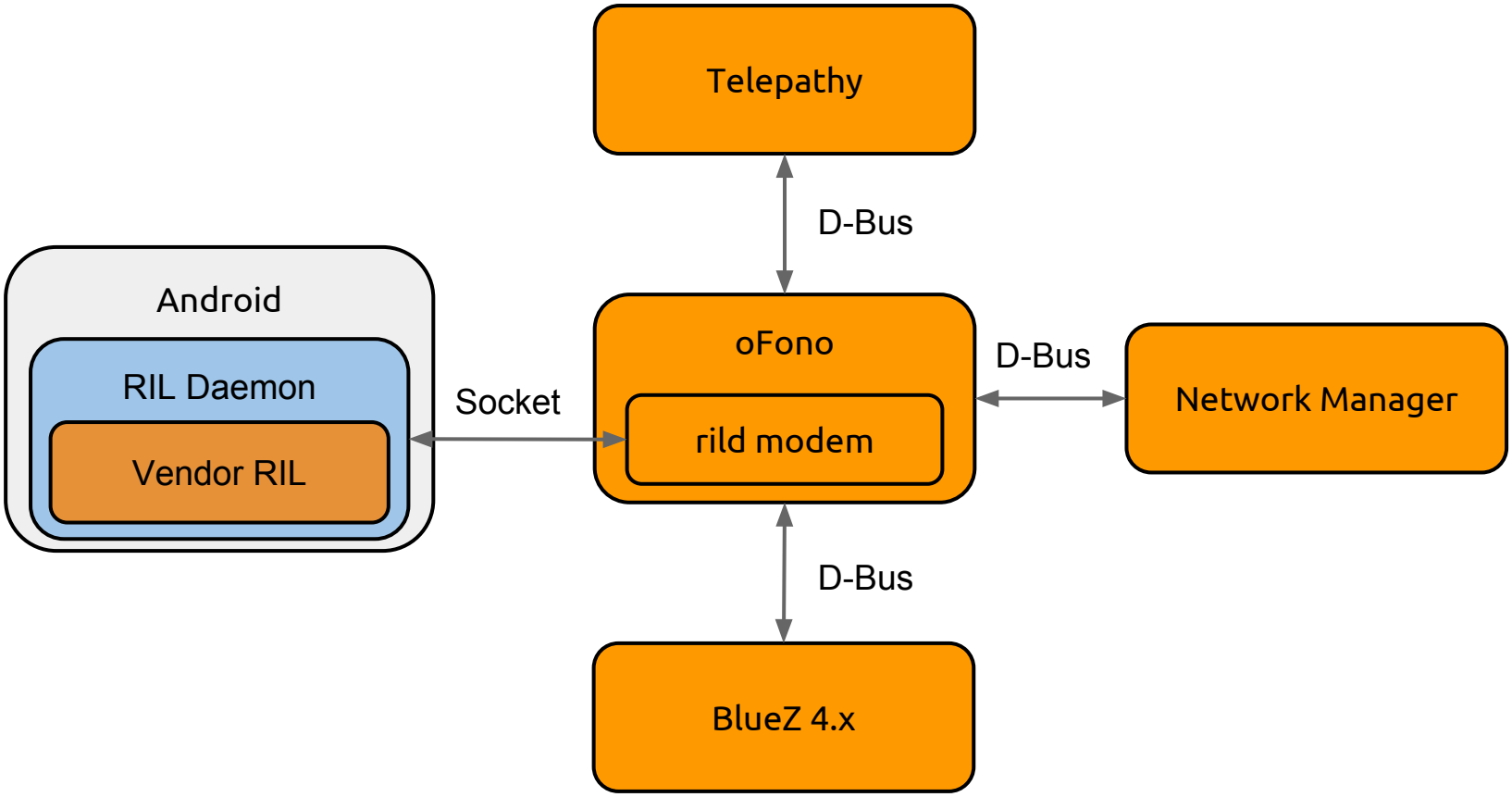
- oFono as the main telephony service
  - In order to reuse the Android modem drivers, a new oFono specific modem was created that talks with the RIL daemon
  - Communication via Socket, LibHybris not involved

- Network Manager as the default connectivity manager
  - No support to talk with oFono (oFono was only compatible with ConnMan)
  - New plugin created that talks to oFono and helps setting the data connection

- BlueZ 4.x (no issues here)

- Telepathy (and telepathy-ofono) used as the main communication framework

Telephony and Connectivity: Ubuntu Touch



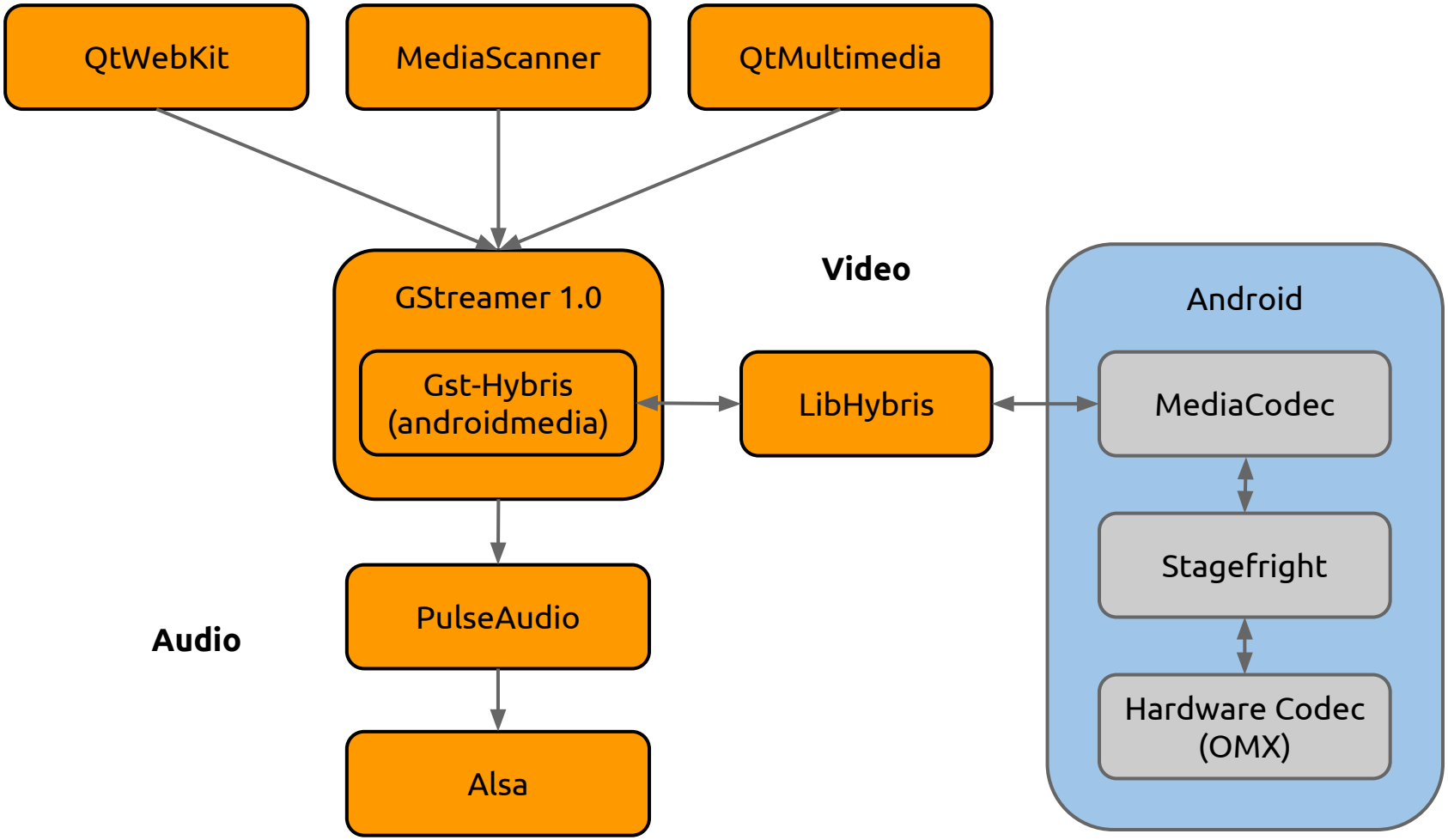
Multimedia

- GStreamer commonly used as the default multimedia framework on the Desktop
  - Used by QtWebkit, QtMultimedia and others
  - Supports a wide range of plugins
  - Abstraction for the Android multimedia stack, but only covering the JNI layer (android.media.MediaCodec)

- Android JNI (and Java) not used by Ubuntu Touch
  - New abstraction on top of stagefright and libmedia was created
  - Using LibHybris
  - Texture streaming

```
$ gst-inspect-1.0 androidmedia
Plugin Details:
  Name                     androidmedia
  Description              Android Media Hybris plugin
  Filename                 /usr/lib/arm-linux-gnueabihf/gstreamer-1.0
/libgstandroidmedia.so
  Version                  1.2.4
  License                  LGPL
  Source module            gst-plugins-bad
  Source release date      2014-04-18
  Binary package           GStreamer Bad Plugins (Ubuntu)
  Origin URL               https://launchpad.net/distros/ubuntu/+source/gst-
plugins-bad1.0

  amcviddec-omxqcomvideodecoderh263: OMX.qcom.video.decoder.h263
  amcviddec-omxqcomvideodecodermpeg4: OMX.qcom.video.decoder.mpeg4
  amcviddec-omxqcomvideodecodermpeg2: OMX.qcom.video.decoder.mpeg2
  amcviddec-omxqcomvideodecoderavc: OMX.qcom.video.decoder.avc

$ gst-launch-1.0 filesrc location=Sintel-1080p.mp4 ! qtdemux ! queue !
h264parse ! amcviddec-omxqcomvideodecoderavc ! filesink location=Sintel.raw
```

Multimedia


QtWebKit
MediaScanner
QtMultimedia
GStreamer 1.0
Gst-Hybris
(androidmedia)
Video
LibHybris
Android
MediaCodec
Stagefright
Hardware Codec
(OMX)
Audio
PulseAudio
Alsa

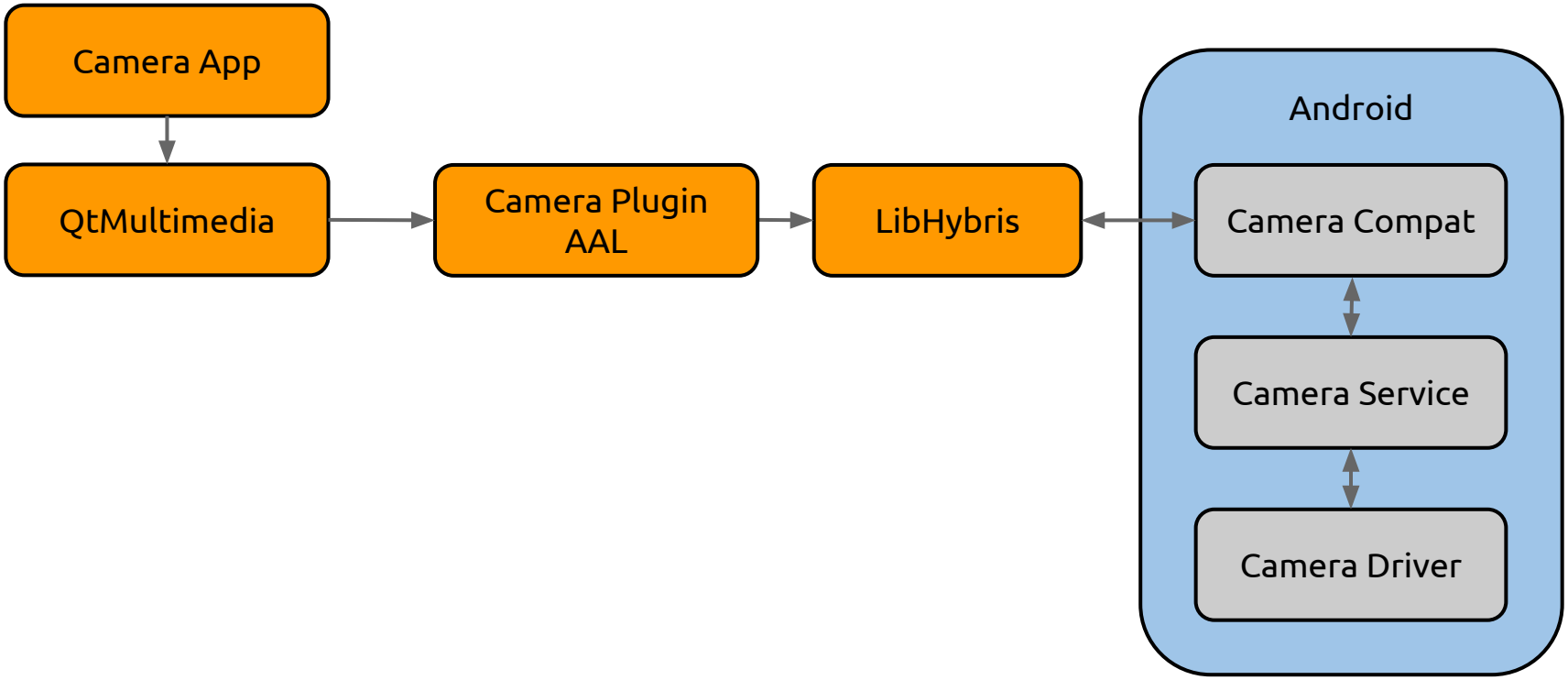
Camera: Android

- Multiple HAL versions (1.0, 2.0, 3.0, 3.1)
  - ABI breakage
  - API differences
  - Hard to abstract

- Android Camera Service
  - Part of media service
  - Abstracts the Camera HAL in a simple API
  - Texture used for both output and preview
  - Not deeply connected to any other Android subsystem

Camera: Ubuntu Touch

- Camera Service running inside the container
  - API abstracted by a compat library living on Android
  - LibHybris used to interact with the compat library
  - QtMultimedia plugin that talks with the compat library



Future Development

- Telephony and Connectivity
  - MMS
  - Bluez 5
- Multimedia
  - Encode support
  - Upstreaming
- Camera
  - Video Recording
- And many more!

Get Involved!

- Freenode:
  - #ubuntu-touch
- Mailing List:
  - https://launchpad.net/~ubuntu-phone
  - Daily updates
- Virtual UDS

CANONICAL

Questions

Thank you

Ricardo Salveti de Araujo
ricardo.salveti@canonical.com

canonical.com
ubuntu.com