GStreamer on Android



Who are we?


GStreamerSDK

A short Introduction to GStreamer



- Pipeline based multimedia framework
- Cross platform, open source
- Bindings for many languages
- Stable API/ABI



- Flexible and extensible design
- Plugin-based architecture
- Easy to integrate with other software
- Active developer and user community

Multimedia applications

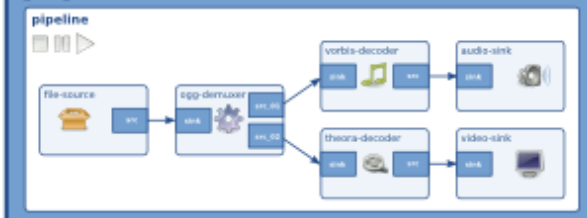
media player | VoIP & video conferencing | streaming server | video editor | (...)

gstreamer

gstreamer core framework

media agnostic
base classes
message bus
media type negotiation
plugin system
data transport
synchronization

pipeline architecture



gstreamer tools

gst-inspect
gst-launch
gst-editor
...

protocols
- file:
- http:
- rtsp:
- ...

sources
- alsa
- v4l2
- ...

formats
- avi
- mp4
- ogg
- ...

codecs
- mp3
- mpeg4
- vorbis
- ...

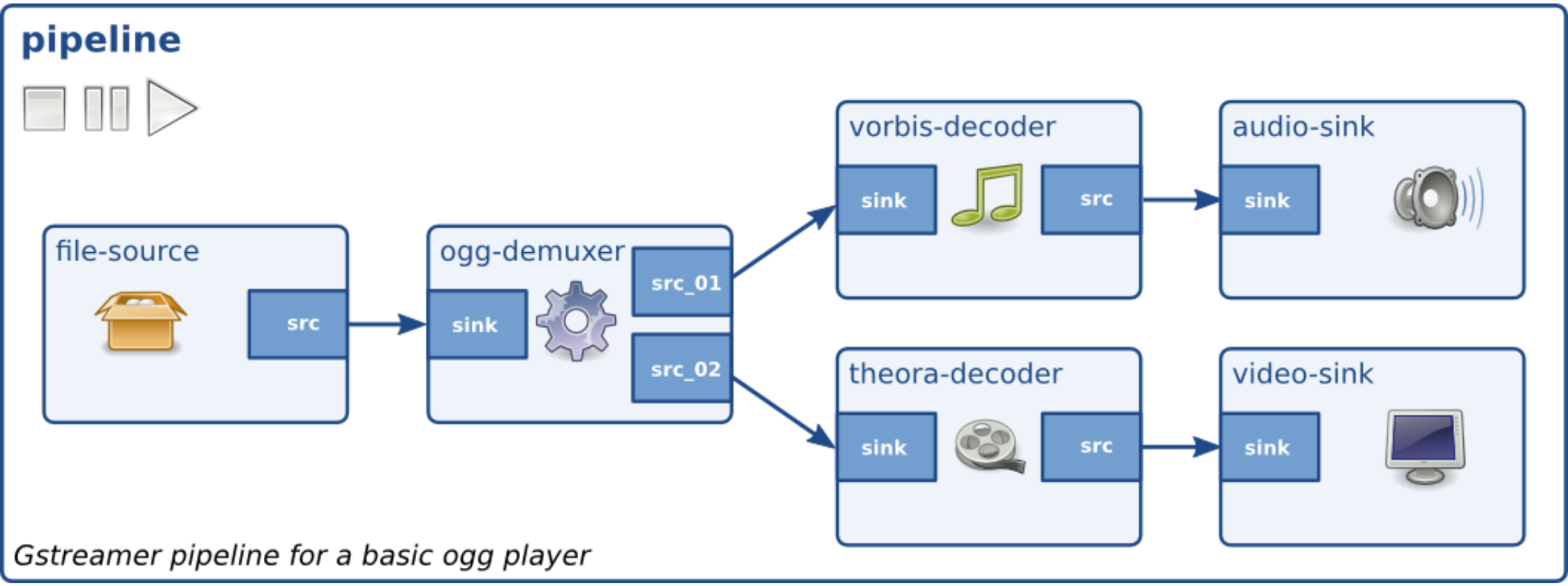
filters
- converters
- mixers
- effects
- ...

sinks
- alsa
- xvideo
- ...

gstreamer plugins

gstreamer includes over 250 plugins

3rd party plugins


pipeline
file-source
src
ogg-demuxer
sink
src_01
src_02
vorbis-decoder
sink
src
audio-sink
sink
theora-decoder
sink
src
video-sink
sink
Gstreamer pipeline for a basic ogg player

- Plugins for all important codecs and containers
- Proprietary plugins for patented codecs
- Plugins for different filters
- Hardware support
- Support for many different use cases

- Used in many different applications on desktop platforms
- Used on many different devices by different companies

GStreamer SDK



- Distribution of GStreamer with dependencies
- Available for Windows, OS X, Linux, Android
- IDE integration
- Starter documentation and tutorials
- Commercial support



Why use GStreamer on Android?

Android Multimedia stack

- Good multimedia API for playback and capture.
- Support for most common audio and video formats
- And a few streaming protocols

But...

- We want much more than just playback or capture
- Some codecs and formats are not supported:
  ASF, DTS, or new codecs like Opus
- Other are device specific:
  WMA and WMV
- Only a few streaming protocols are supported:
  DASH, Smooth Streaming, RTP ?

GStreamer has almost everything we need:

- Supports a very large number of formats.
- Support for more uses cases
- Multimedia backend re-usable across platforms.

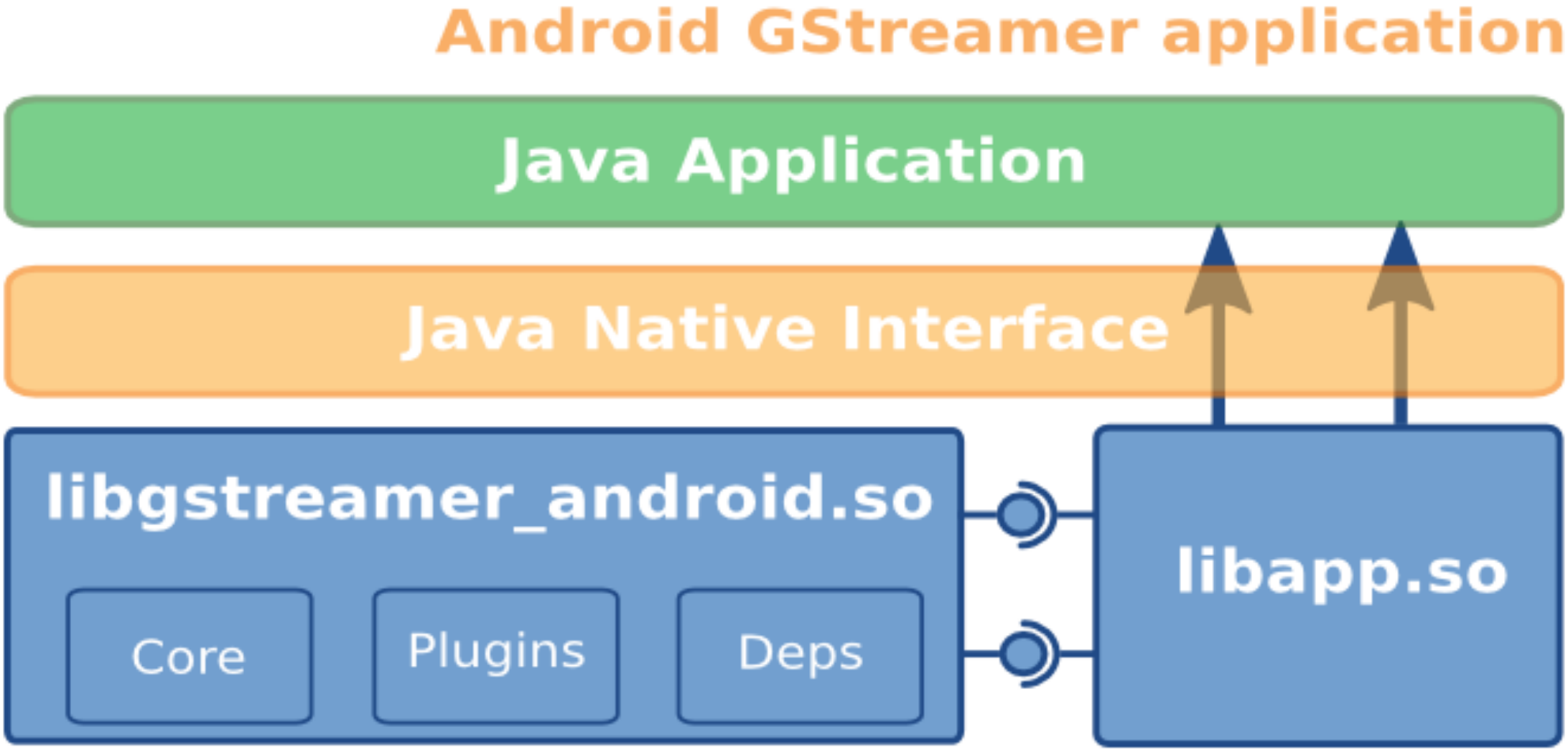
Problems with using GStreamer on Android

- Plugin-based architecture -> too many shared libraries
- Android's dynamic linker limits the number of shared libraries per process.
- We have more than 262 shared libraries 😥
- Hard to easily distribute it in the Market
- Legal constraints with the LGPL and static linking.
- The NDK is limited: C library (BIONIC) and other libraries like OpenSL.



How we solved it..

- Static linking with re-locatable archives.
- A single shared library with everything:
  libgstreamer_android.so
- Integration with ndk-build to link this shared library:
  - Complies with the LGPL requirement.
  - Allows selecting only the plugins being used.


Android GStreamer application
Java Application
Java Native Interface
libgstreamer_android.so
Core
Plugins
Deps
libapp.so

Building the SDK

- We use a build system called Cerbero.
- Same build system used to build the SDK in all platforms.
- Re-use of upstream packaging system.
- Native packaging:
  Windows .msi, OS X .pkg, RPM and DEB
- Easy to maintain
- Easy to add new packages or 3rd party plugins



```
$ git clone git://anongit.freedesktop.org/gstreamer-sdk/cerbero

$ cerbero -c config/cross-android.cbc bootstrap

$ cerbero -c config/cross-android.cbc package gstreamer-sdk
```

Static plugins and modules

- GStreamer plugins and GIO modules must be handled in a different way.
- We are trying to get these changes upstream
- Static plugins need to be registered manually.

Integration with ndk-build

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

LOCAL_MODULE    := hello-jni
LOCAL_SRC_FILES := hello-jni.c
include $(BUILD_SHARED_LIBRARY)

include $(CLEAR_VARS)
```

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

LOCAL_MODULE    := hello-jni
LOCAL_SRC_FILES := hello-jni.c
LOCAL_SHARED_LIBRARIES := gstreamer_android
include $(BUILD_SHARED_LIBRARY)

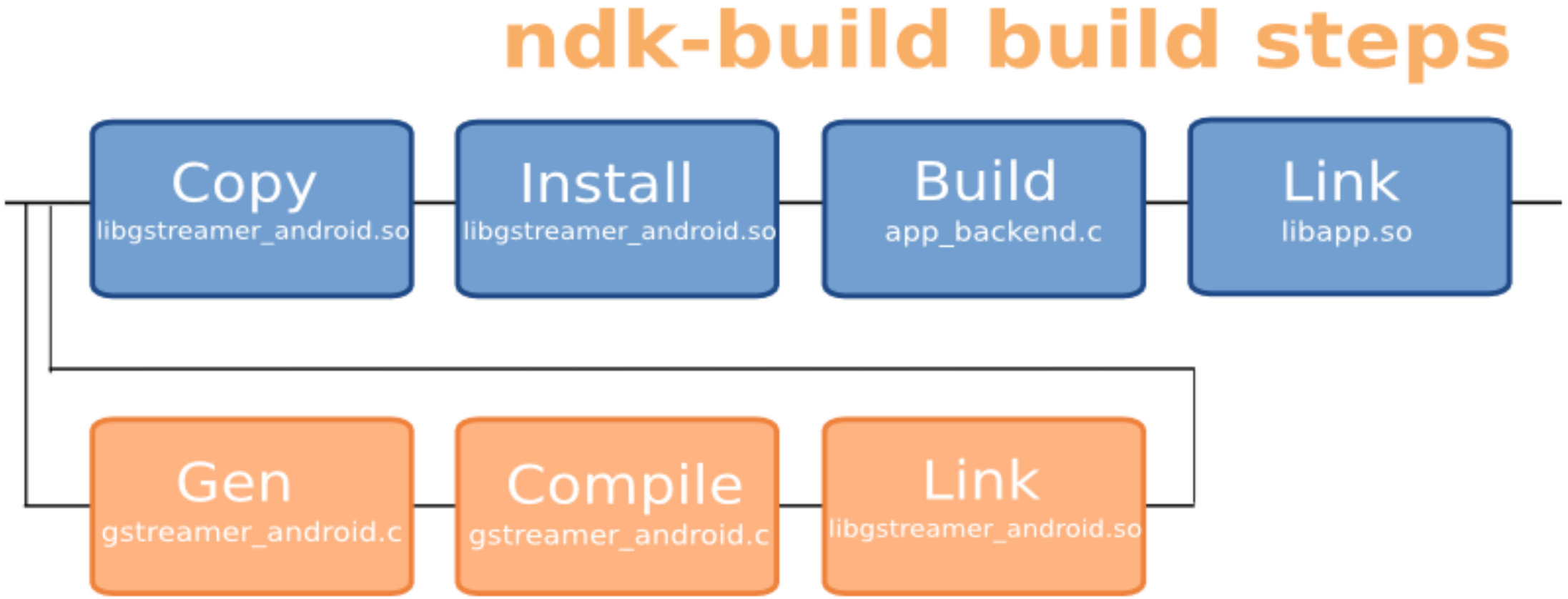
include $(CLEAR_VARS)

include $(GSTREAMER_NDK_BUILD_PATH)/plugins.mk

GSTREAMER_SDK_ROOT := /home/cerbero/android_arm
GSTREAMER_PLUGINS =  $(GSTREAMER_PLUGINS_CORE)
                     $(GSTREAMER_PLUGINS_CODECS)
GSTREAMER_EXTRA_DEPS := json-glib-1.0

include $(GSTREAMER_NDK_BUILD_PATH)/gstreamer.mk
```




ndk-build build steps
Copy
libgstreamer_android.so
Install
libgstreamer_android.so
Build
app_backend.c
Link
libapp.so
Gen
gstreamer_android.c
Compile
gstreamer_android.c
Link
libgstreamer_android.so

- We use libtool libraries to resolve link deps
- Libtool can't be used for portability issues
- A small libtool replacement in makefiles + sed
  - Portable (works on Windows too)
  - Supports relocations of .la files
  - Much faster than libtool

And some stats...

- 171 plugins (same as for other platforms)
- Size of libstreamer_android.so
  - not stripped: 60 MB
  - stripped: 15 MB
  - not stripped without GStreamer debug: 55 MB
  - stripped without GStreamer debug: 13 MB

New plugins

OpenGL ES / EGL Video Sink



- OpenGL ES/EGL only public, native API for video on Android
- Supports hardware accelerated colorspace conversion, scaling
- Usable on all Android devices
- Works like any other GStreamer video sink
- Allows embedding into Android applications
- Small and simple codebase



OpenSL ES Audio Sink/Source

- OpenSL ES only public native API for audio on Android
- Very limited implementation available on Android
- Usable on all Android devices
- Uses Android-specific API extensions
- Could support compressed formats later

android.media.MediaCodec Wrapper

- Be able to use device's codecs
- Uses Java API via JNI
- Java/JNI not performance problem
- Usable on all Android devices

- Implemented: audio/video decoders
- Encoders easy to add if necessary
- 1080p h264 easily possible, impossible in software
- Supported video codecs:
  h264/AVC, MPEG4, h263, MPEG2 and VP8
- Supported audio codecs:
  AAC, MP3, AMR-NB/WB, A-Law, µ-Law,
  Vorbis and FLAC

Developing applications with the SDK



- GStreamer projects can be built using the regular tools
- For Eclipse: using the wizard and project→Android Tools→Add Native Support
- Command line: using the standard Ant build command
- jni/Android.mk must be updated for GStreamer

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

LOCAL_MODULE    := hello-jni
LOCAL_SRC_FILES := hello-jni.c
LOCAL_SHARED_LIBRARIES := gstreamer_android
include $(BUILD_SHARED_LIBRARY)

include $(CLEAR_VARS)

include $(GSTREAMER_NDK_BUILD_PATH)/plugins.mk

GSTREAMER_SDK_ROOT := /home/cerbero/android_arm
GSTREAMER_PLUGINS =  $(GSTREAMER_PLUGINS_CORE)
                     $(GSTREAMER_PLUGINS_CODECS)

include $(GSTREAMER_NDK_BUILD_PATH)/gstreamer.mk
```



- No Java bindings yet
- Multimedia backend is written in C
- Bind the backend API to use it in the application through JNI

- Bind backend registering dynamic methods with RegisterNatives
- Declare this new methods as dynamic in the Jave side

```
/* List of implemented native methods */
static JNINativeMethod native_methods[] = {
  { "nativeInit", "()V", (void *) gst_native_init},
  { "nativeFinalize", "()V", (void *) gst_native_finalize},
  { "nativePlay", "()V", (void *) gst_native_play},
  { "nativePause", "()V", (void *) gst_native_pause},
  { "nativeClassInit", "()Z", (void *) gst_native_class_init}
};

/* Library initializer */
jint JNI_OnLoad(JavaVM *vm, void *reserved) {
  JNIEnv *env = NULL;

  java_vm = vm;

  if ((*vm)->GetEnv(vm, (void**) &env, JNI_VERSION_1_4) != JNI_OK) {
    __android_log_print (ANDROID_LOG_ERROR, "tutorial-2", "Could not retrieve JNIEnv");
    return 0;
  }
  jclass klass = (*env)->FindClass (env, "com/gst_sdk_tutorials/tutorial_2/Tutorial2");
  (*env)->RegisterNatives (env, klass, native_methods, G_N_ELEMENTS(native_methods));

  pthread_key_create (&current_jni_env, detach_current_thread);

  return JNI_VERSION_1_4;
}
```

- Glib's main loop is run in a separate thread
- Use Thread-Local Storage (TLS) for storing the JNI env
- Load libgstreamer_android.so in the application

- 5 tutorials to introduce developers:
  - Linking against GStreamer
  - A running pipeline
  - Video
  - A basic media player
  - A complete media player

GStreamer SDK

http://www.gstreamer.com

Documentation and tutorials

http://www.docs.gstreamer.com





¿Questions?







Thanks!



