When to use which framework?

• Android (easiest)
  – Customer wants a full featured media player with minimal trouble and no prior knowledge about our stack.
  – Doesn’t mind adopting Android/Java

• GStreamer (moderate)
  – Customer wants a Linux distribution, Android will not do.
  – Customer has complex use cases that Android does not support.
  – Hides complexity of TI stack, but needs work to develop a product either by building pipelines or by optimizing elements.

• OpenMAX (most work)
  – No other framework available on chosen OS.
  – Customer can design a system around the lowest level components for least overhead and most flexibility
  – Customer must be willing to create necessary OMX parsers, avsync, and other system components
OpenMAX benefits

• Designed by a consortium of companies including TI.
• Cross platform, industry standard Codec API.
• Standard producer/consumer interface as standard (SNT – Standard Non Tunneled)
• Android frameworks directly interface with OpenMAX under the hood.
• For environments that don’t have a de-facto MM framework OpenMAX is a good choice.
GStreamer vs OpenMax

- GStreamer is defacto filter graph based framework for Linux.
- GStreamer has an impressive number of demuxers/container parsers/network transports.
- GStreamer includes AVSync as standard and many additional capabilities. We use OpenMAX purely as a codec acceleration interface.
- GStreamer is extremely flexible in the kinds of media pipelines you can create.
- GStreamer integrated cleanly with Qt through either Qt sink or use of Phonon Qt API
- GStreamer integrates with X11 through xvsink (although there is currently not a solid schedule for this)
- UI Frontends for GStreamer are available (e.g.. Totem)
- **BUT** – GStreamer has a desktop heritage not an embedded heritage. This means optimization of components may be required.
- GStreamer is very loosely integrated into Linux
Android vs OpenMAX

- Android ≠ Linux
- Android uses OpenMAX under the hood as its codec interface.
- Android builds on the OMX codecs using a multi-media framework called Stagefright.
- Stagefright has AVSync, parsers, capture, resize, display, network streaming capabilities.
- Stagefright is exposed to Android applications through the MediaPlayer Java classes. Support for seek, pause, etc built right in.
- Stagefright is tightly integrated into the Android environment.
- Stagefright was built from the ground up for embedded devices.
GStreamer vs Android

• GStreamer has a desktop heritage
  – PC level performance is assumed
  – the gst community cares more about features than performance.

• Android is designed from the ground up with mobile and embedded devices in mind.

• GStreamer has flexible (programmable) pipelines – ideal for complex scenarios such as transcoding.
  – Android has a small number of optimized fixed function pipelines.
    • This number is growing with each new Android release
  – Android has a much more limited selection of network transports/demuxers than gst.
Android Pre-requisites

• Developers - fluent in Java
• Multimedia Experience Needed - None
• Minimum Memory Size - 512Megs (more is better)
• Application Performance - interpreted byte code (slowest)
• Codecs support - H.264, MPEG4, H.263, MP3, AAC
• Containers - MP4, 3GPP
• Development Environment - Mac OS X, Linux, Windows
GStreamer Pre-requisites

- Developers - fluent in C, C++ and Linux development
- Multimedia Experience Needed - Medium
- Minimum Memory Size - 256Megs (more is better)
- Codecs support - Accelerated: H.264, MPEG4, MPEG2 (Planned for first release)
  Non-accel: Everything known to man
- Containers - Everything known to man
- Development Tools needed - TI EZSDK
- Development Environment - Linux
OpenMAX IL Pre-requisites

• Developers - fluent in C, understands the OpenMAX data flow, can write parsers and networking code.
• Multimedia Experience Needed - High
• Minimum Memory Size - 256Megs (more is better)
• Application Performance – Native Machine Code (fastest).
• Codecs support - Accelerated: H.264, MPEG4, MPEG2 (Planned for first release)
• Containers – Very Limited (Bellagio has a 3gpp container component)
• Development Tools needed - TI EZSDK
• Development Environment – Linux
Android Quick Info
Benefits of Android (1/2)

• More than just another flavor of Linux.
  – Android integrates middleware, applications and frameworks into a complete system instead of leaving that up to the end user as is the case with traditional Linux distros.

• Attractive Licensing – business friendly, no copy-left or viral licensing.
  – Android is very attractive because all core packages are open sourced under the terms of the Apache 2.0 license,
  – Allow the use of the source code for both commercial and free open source applications.
  – Modified version of the source code need not be licensed under the terms of the original license.

• Open Source Software
  – Android leverages existing open-source projects, maintains and manages the sources
  – Many hardware-component vendors have decided to provide source code for specific drivers.

• Large eco-system
  – Android has a larger and growing developer community, driving not only application layer content (more than 200,000 -as of Dec 2010- applications are available).

• Committed Roadmap
  – Android has a relatively frequent major releases and a well maintained roadmap.

• Large Pool of Documentation
  – The Android community offers a wide variety of instructional content, helps reducing learning curve.
Benefits of Android (2/2)

• **Application development language**
  – Java is a popular programming language with a large pool of trained engineers
  – Android is based on the Java programming language, but uses its own Java Virtual Machine (Dalvik) to avoid licensing issues with Sun/Oracle.

• **Range of reference hardware platforms**
  – Android has wide availability of hardware platforms for prototyping and benchmarking purposes.

• **Adopting and consistently improving traditional frameworks**
  – Both Google and its partner community are consistently investing in frameworks that enable specific application needs
  – Android includes a complete multimedia framework and associated media player designed for a touch screen environment.

• **Supports embedding C/C++ components**
  – Android gives flexibility to developers to include their favorite C/C++ libraries and code into Android framework using the Native Development Kit – toolset to embed native C/C++ components, Gingerbread extends this to application development.

• **Development and Debug Tools**
  – Eclipse offers a dedicated plug-in for Android (ADT). This allows setting up new Android projects, create application-specific user experiences and user interfaces, adding components, debugging, and then exporting the .apks
Features of Android

- **Application framework** enabling reuse and replacement of components
- **Dalvik virtual machine** optimized for mobile devices
- **Integrated browser** based on the open source WebKit engine
- **Optimized graphics** powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
- **SQLite** for structured data storage
- **Media support** for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- **GSM Telephony** (hardware dependent)
- **Bluetooth, EDGE, 3G, and WiFi** (hardware dependent)
- **Camera, GPS, compass, and accelerometer** (hardware dependent)
- **Rich development environment** including a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE
- **GoogleTV** is built on top of the Android platform.
How is Android different from Linux (1/2)

- Android is built around a modified Linux kernel – however this does not make it Linux in the traditional sense.
- Linux we all know and love is really GNU/Linux – the userspace components are supplied by GNU and the kernel is Linux.
  - GNU software components are inevitably licensed under the restrictive LGPL or GPL.
- In Android the GNU components are replaced with Android components - so it is really Android/Linux. This allowed Google to create a Linux distribution around the more permissive BSD or Apache licenses.
  - The Linux kernel remains GPLv2.
How is Android different from Linux (2/2)

- X11 is not present in Android, and is replaced with the SurfaceFlinger/Skia libraries.
- glibc is replaced with Bionic – a small/efficient libc that is licensed under BSD
  - This change does make porting GNU/Linux C/C++ applications to Android more involved as there are some differences and some glibc features are just flat out not supported in Bionic (Bionic is NOT fully Posix compliant).
  - Gory technical details are available here
Android Versioning

- Android uses a traditional version number for its releases but during development it is allocated a name that is a tasty treat and a progression of the alphabet.
- The version number that is allocated to a name is typically not known until the public release of the Android version.
- Cupcake
- Donut
- Éclair – (DevKit 1.x)
- *Froyo* – 2.2 *(DevKit 2.x)*
- *Gingerbread* - 2.3 *(DevKit 3.x)*
- Honeycomb - 3.0
- Ice Cream Sandwich
- “J” – any guesses?
Android Application Development

- Android offers two options for application development
  - Developing in Java through the use of the Android SDK.
  - Developing in C/C++ through the use of the Android NDK (Native Development Kit)
- Developing in Java has the advantage that it is write once run anywhere.
- Developing with the NDK limits you to a single CPU architecture - most Android devices today are ARM so not a huge limitation.
- Best way to learn about Android is to Google for it (not Bing 😊) Here are a few pointers to some of the best information:
  - Plenty of useful how-to videos:
    - http://www.youtube.com/user/androiddevelopers
  - Excellent development information blog:
    - http://android-developers.blogspot.com/
Debugging Android with CCSv5

- CCSv5 combines the advantages of the Eclipse software framework with advanced embedded debug capabilities like Linux Aware Debug, Hardware Debugging, Simulation, Profiling, etc from TI.
- Eclipse ADT (Android Development Tools) plugin is supplied by Google
- With CCSv5 + ADT - it is possible to debug Java (Android SDK), C/C++ (Android NDK), Linux Kernel, DSP, etc all from within the same IDE!
- ADT adds other extensions to the CCS IDE making it a very powerful tool allowing you to easily create, debug and deploy Android applications on TI Embedded Platforms
- You can even debug apps in the Android emulator using CCSv5 + ADT.
GStreamer Quick Info
Introduction

• GStreamer is based around a Pipeline Media Architecture
• The GStreamer frameworks allows creation of functional plugins, and a mechanism to control/link these plugins together.
• It is not a media player, rather a foundation framework for building an extensible, flexible media system on top of.
Anatomy of Pipeline Media Architecture

- The basic building block is called a **filter**
- Filters are linked together to make a **filter graph**
- An example of a filter graph:
GStreamer

• Also a pipeline media architecture
• Open Source answer to DirectShow and QuickTime.
• Comes out of a research project done at Portland University.
  – Claims to be ‘loosely’ modeled on DirectShow
  – Actively developed for over 3 years
• Core API is media agnostic and GUI independent
• Licensed under LGPL
• Designed specifically with embedded systems in mind.
  – Small core size (less than 150KB, about 10K lines of code)
• Unix Centric but runs on a variety of flavors of Unix
  – FreeBSD
  – Linux
  – others
GStreamer includes a utility called gst-editor to graphically build its filter graphs. Like MS GraphEdit it also allows you to run your filter graphs directly from within the editor.