Real-Time High Performance Multimedia Systems on Android

By: Ramachandra Pai, Santosh Holla K S, Prashanth Dixit K S
Agenda

- Android on Embedded
- Typical Real Time Multimedia System
- Key Challenges
- Designing/Integrating multimedia systems into the Android Framework and Runtime
- Android API Standardization
Agenda

Android on Embedded

Typical Real Time Multimedia System

Key Challenges

Designing/Integrating multimedia systems into the Android Framework and Runtime

Android API Standardization
Android on Embedded

➡️ Embedded System?
  ➡️ A system dedicated for a particular real time application

➡️ Hardware Capabilities (an example)
  ➡️ Processor speed - 1.5Ghz QuadCore
  ➡️ RAM – As high as 2GB
  ➡️ External Memory – Scalable up to 64GB
  ➡️ GPU – Multicore GPUs like Adreno 320
  ➡️ VISA Accelerators – Full HD / Multi-channel capable accelerators

➡️ Multi Media Applications on Embedded
  ➡️ Video Conferencing, streaming systems, Surveillance and more
Android on Embedded

Why Android for Embedded systems?

- One of the fastest growing operating system that is ougrowing the mobile and tablet space
- Open source
- Linux++ : Ideal for an embedded system with a screen
- Access to thousands of apps from the Android Market
- ...
Agenda

Android on Embedded

- Typical Real Time Multimedia System

Key Challenges

Designing/Integrating multimedia systems into the Android Framework and Runtime

Android API Standardization
Real Time System

A real-time system is any information processing system which has to respond to externally generated input stimuli within a finite and specified period – the correctness depends not only on the logical result but also the time it was delivered.
Video Conferencing: An example...

GUI

Application Controller
  State Machine

Signaling
  (RTSP/HTTP/SIP)

Media Engine

capture -> encode -> Packetize

Display/Playout

decode -> De-packetize

Network

Network
Video Conferencing: An example...

- GUI
- Application Controller
  - State Machine
- Media Engine
  - Signaling (RTSP/HTTP/SIP)

- Capture
- Encode
- Packetize
- Network

- Display/Playout
- Decode
- De-packetize
- Network
Agenda

Android on Embedded

Typical Real Time Multimedia System

Key Challenges

Designing/Integrating multimedia systems into the Android Framework and Runtime

Android API Standardization
Key Challenges

- Low Latency
- High Performance
- Low System Load => Maintain Battery Life
- Quality of Service
- Co-Existence with other Applications
- Portability
Agenda

Android on Embedded

Typical Real Time Multimedia System

Key Challenges

- Designing/Integrating multimedia systems into the Android Framework and Runtime

Android API Standardization
Although performance can be met, it results in Software/Hardware conflicts and breaks existing applications.

Some of the required functionality is not exposed and expected Performance cannot be reached due to IPC and other overheads.

Co-existence with other frameworks and also expected performance can be achieved.
Native Framework Interfaces

Audio - Open SLES / Tiny ALSA
Video – Camera Client

Application Controller
State Machine

Playout - Open SLES / Tiny ALSA
Display - Surface (Native Window)

Video - IOMX / OpenMAX AL (limited)

capture → encode → Packetize → Network

Display/Playout

decode → De-packetize → Network

Signaling (RTSP/HTTP/SIP)
Audio

Capture and Playout

- Open SLES
  - Standardized at NDK
  - Latency ~ 200ms

- Tiny ALSA
  - Not standardized at NDK, requires access of Audio HAL
  - Latency ~ 40ms => Meant for ultra low latency system

Audio encode and decode

- Software Codecs
- Requires low processing power
Camera Client

- Can provide frames that can be directly fed to the encoder – No copy overheads
- Provides tunneled preview mode
- Gives control for low level camera operations
Video - Display

SurfaceView / TextureView

Surface

NativeWindow

Decoder

Camera

Java

Java

Native
Video – Encode/Decode

iOMX Interface

- Android Native Interface to OpenMaxIL
- Applications get access to hardware accelerated codecs through an iOMX client
- Not standardized

OpenMAX AL

- Standardized NDK APIs available only for Media Player
- Supports minimalistic configuration
Are we there yet?

- Low Latency
- High Performance
- Low System Load => Maintain Battery Life
- Quality of service
- Co-Existence with other Applications
- Portability
Integrating application with Android

- **Listening for Broadcast message**
  - GUI
  - Java API Classes
  - Java Service
  - JNI layer
  - Native C/C++ APIs

- **Send Broadcast message**
  - Android Interface Definition Language - AIDL
  - Java Native Interface - JNI
  - Listening for Broadcast message: Boot Completed

- **Runs in one process**
Agenda

Android on Embedded

Typical Real Time Multimedia System

Key Challenges

Designing/Integrating multimedia systems into the Android Framework and Runtime

Android API Standardization
Android API Standardization

- **Video Capture**
  - Camera APIs can be standardized at NDK

- **Video Display**
  - Native Window APIs like queue/dequeue and so on can be standardized at NDK

- **Video Encode**
  - An independent video encode APIs can be added to NDK

- **Video Decode**
  - APIs can further be enriched by adding support for configuration like low delay settings and so on

- **Audio Capture/Playout**
  - OpenSLES - already supported in NDK. Minor enhancements required.
It is not the answer that enlightens, but the question.
- Eugene Ionesco
Thank you