Audio playback on mobile devices: challenges and lessons learned

Pierre-Louis Bossart

UMG Platform Architecture

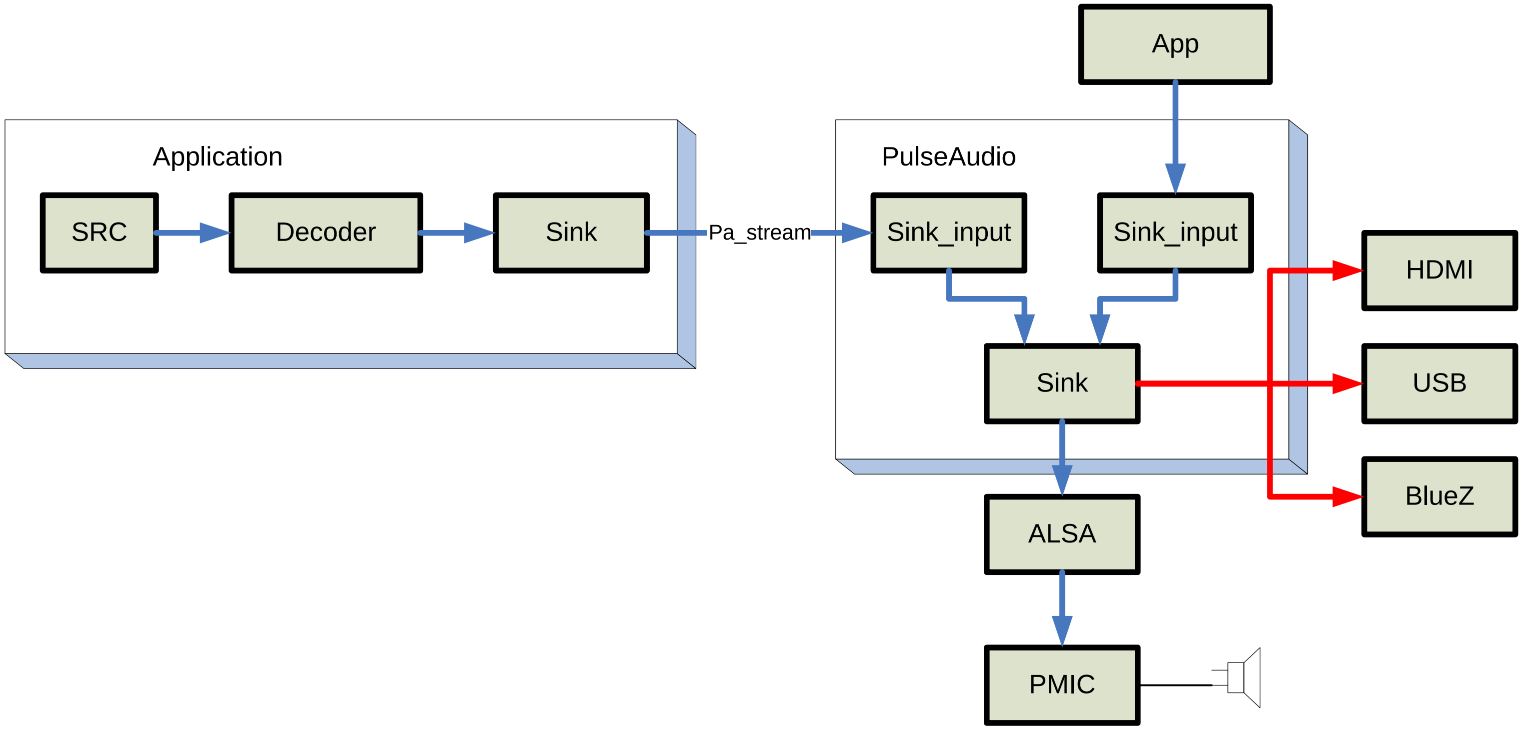
Outline

PulseAudio is a fundamental component of the Moblin distribution

Goal: share lessons learned on upcoming consumer/mobile devices

- Optimized/secure music playback
- Volume control
- Delay management with HDMI

Music playback recap



Intel relies on PulseAudio for

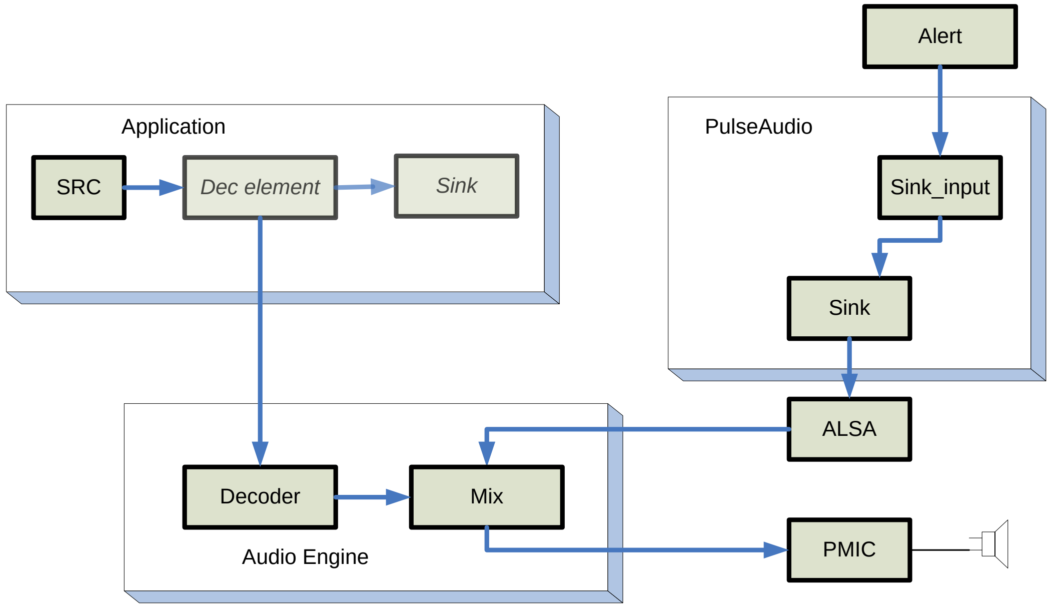
- Mixing
- Detection and transition to ‘plugable’ output.

Low-power use cases

PA Timer-based scheduling enables long sleep-time (up to 2s if the ALSA buffer is big enough)

- Music playback app can specify that latency is not an issue
- Application processor can decode a large buffer then go to sleep with PCM is rendered

- Optimal battery life is still reached with separate audio engine

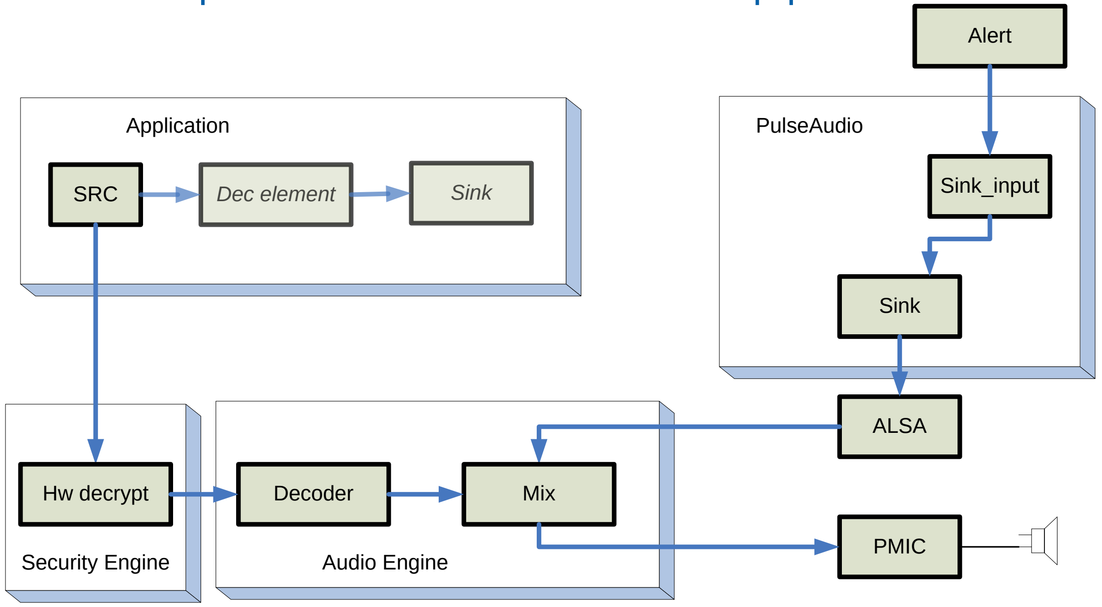


DRM use cases

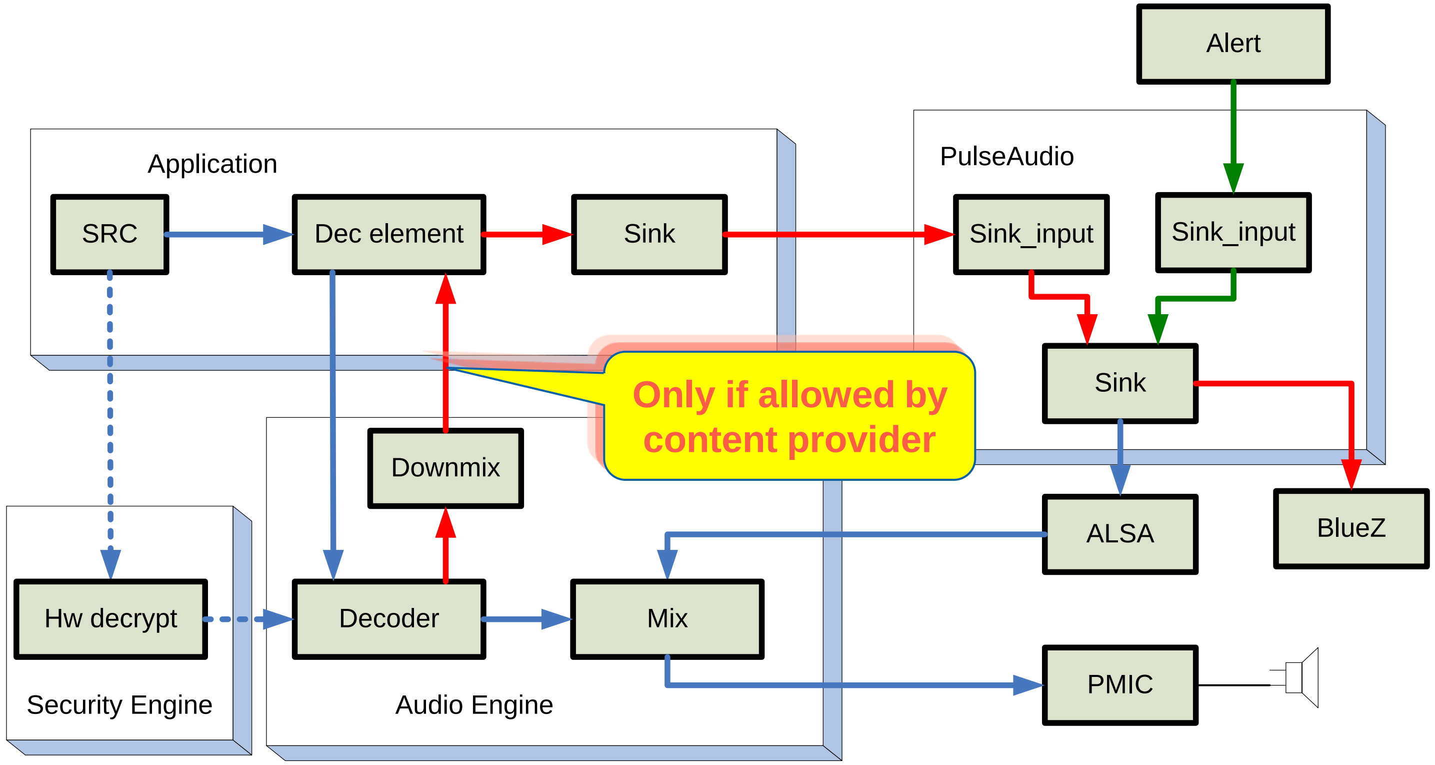
Content-protection is still a requirement

- Premium content (BluRay), Wireless providers (OMA, etc)

- Different levels of protection
  - Compressed content protected
  - High-resolution PCM protected only
  - PCM playback to certain outputs only
- Solution to meet all requirements: secure hardware pipeline



Stack evolution



Main hurdles:

Gstreamer pipeline creation

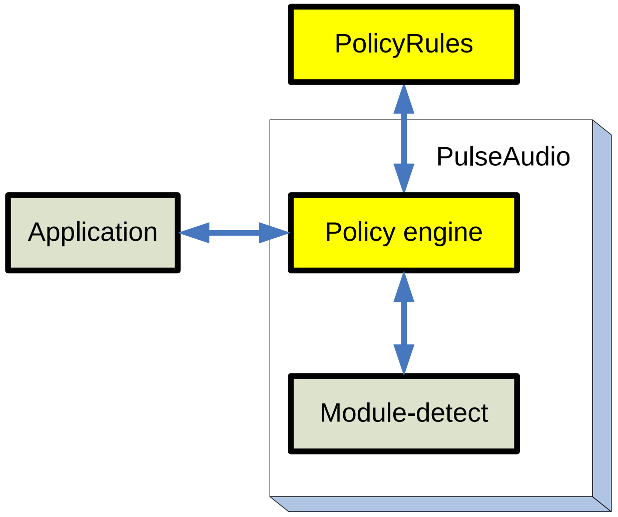
Transitions between modes

Gstreamer audio pipeline

Main ways of creating pipeline
- Explicit pipeline
- Autoplug (playbin/decodebin)
- All of the above

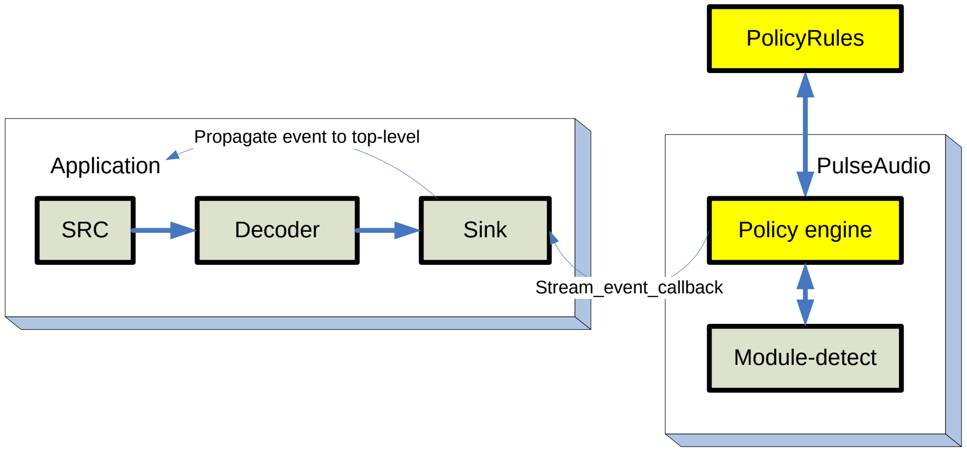
- High rank for hw-accelerated elements may not be enough.
- Too many details may prevent any kind of hardware pipeline
    - gst-launch -v filesrc location=music.ogg ! decodebin ! Volume volume=0.8 ! audioconvert ! audioresample ! alsasink
        - PCM data need to be provided back to the host…
- Playbin assumes PCM data will be provided for visualization
- Solution:
    - Clone of playbin to remove visualization part…
    - Modify apps to enable hardware pipeline
        - Code clean-up required anyway in media player (remove wake-ups, UI refresh when screen is off, etc)

Routing transitions and audio policy



- Audio policy exists in all mobile devices
- Examples:
  - BT device paired
    - Audio policy moves sink_inputs to BT A2DP output
  - Ringtone:
    - Music playback paused while ringtone plays
    - Music playback mixed with ringtone
- Audio policy needs interaction with application for state changes
  - Example: PulseAudio cannot pause a stream on its own
    - Coherency issues with UI, stream displayed as playing but not moving
    - Pause/resume requests are handled by application
- Routing requests need to be provided by Audio policy
  - New requirement to enable audio pipeline:
    - Routing transitions need to be notified to the application in addition to state transitions

PA event callback



Only make sense when a true pa_stream was opened

- Transition between BT and local playback possible
- To handle a transition between local playback and BT, would need to keep an open pa_stream connection but never send any actual data until the policy changes the routing.
  - Can be done but dangerous

Moblin Audio Manager

Audio Manager is a PulseAudio module

Defines new stream type to extend pa_stream

- am_stream can be
  - Regular pa_stream when data flows through PA
  - Hw accelerated stream

- Audio Manager relies on
  - Native hooks to trap all pa_stream events
  - DBUS messages from application when hw accelerated stream changes states.
- Requires application using hardware pipeline to register with AudioManager
  - Notifies application of policy decision (stop/pause) or routing change.

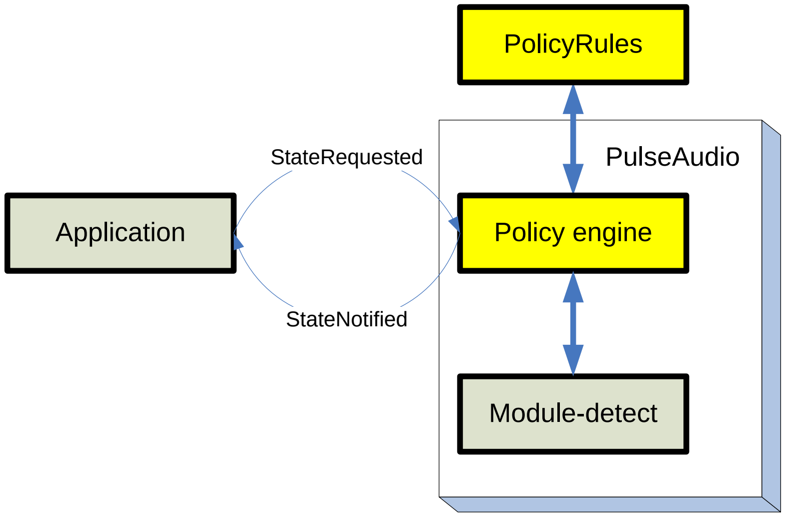
Discussion

PA event callback is only available after pa_stream was created.

- Need to open a pa_stream connection to know that you don't need the connection

- Event-propagation isn't straightforward in apps
  - Callback handled by audio back-end
  - Needs to be propagated to main event loop/UI
- Audio Manager
  - Requires Intel to modify apps&PulseAudio
- Ideally, there would be only one generic enough means of interaction between PulseAudio and applications
  - Available for mobile and desktop/laptop

Proposal



Common DBUS interface

- Application requests a state
- PulseAudio notifies the state granted by audio policy

- Can be used to convey audio policy or routing decisions
- Doesn't seem too incompatible with Maemo
- Would be fine with Moblin
- Would only require minor change in 'module-cork-music-on-phone'

Volume control

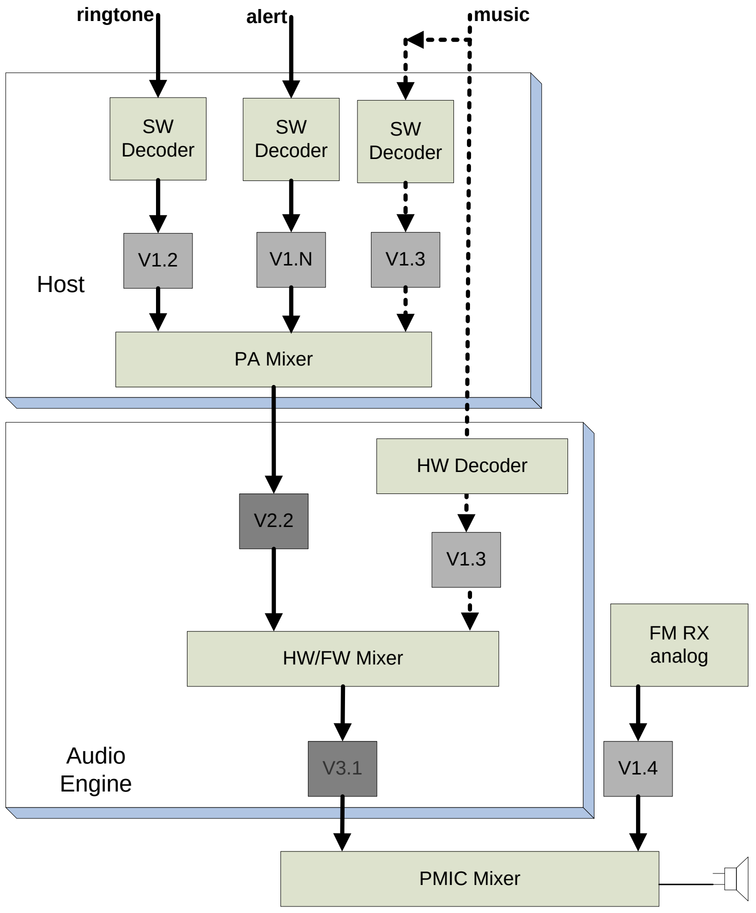
Significant evolutions in PA code in H1'09

- Flat-volumes

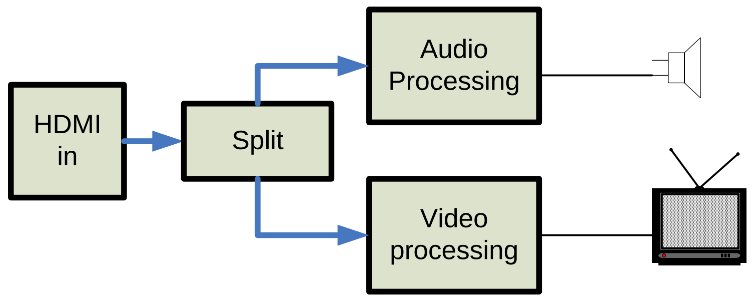
Issue:

- Not all volumes are controlled through ALSA mixer
- Different experience when using sw and hw decoder

- Solutions:
  - 'fake' pa_stream
    - Visible in pavucontrol
    - No vu-meter since no data is sent
    - Application doesn't know how the device volume was changed
  - Add ALSA mixer input
    - Odd since ALSA not really used for low-power/DRM playback
  - Provide volume change information over DBUS



HDMI lip-sync

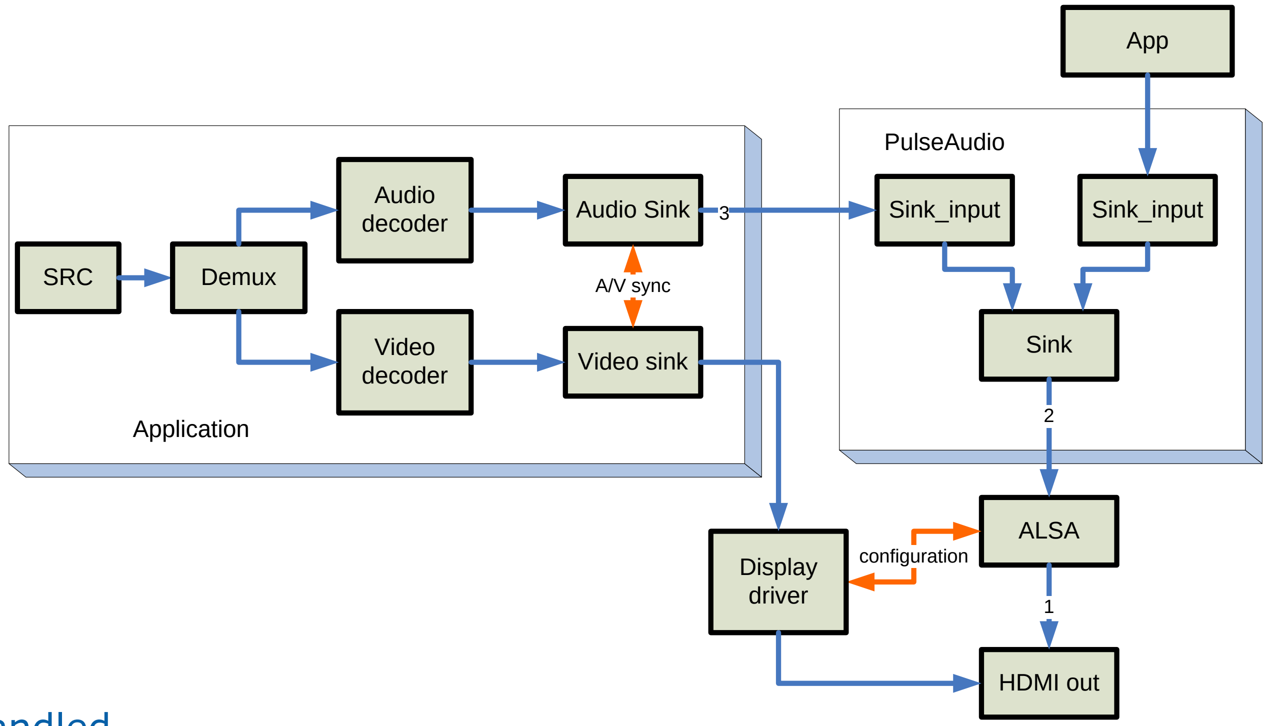


HDMI receiver splits audio and video payload

Audio and video processing have different latencies
- Video enhancements typically require 100s of ms of buffering
- Need to delay audio data to maintain alignment

- When audio and video processing are performed on different devices, delay needs to be applied at the source
- HDMI v1.3 provides audio/video latencies in EDID fields
  - Read by HDAudio driver, reported in /proc file
  - Note: Delay is dynamic
    - If video resolution changes, required audio delay needs to change
- Issue: Linux audio stack doesn't do anything with it
  - Lip-sync issue!

HDMI stack



Delay can be handled

In the driver

In PulseAudio

By the application

Lip-sync discussion

Delay at driver level

- Needs to be done for every link that handles audio/video (not just wired HDMI)
- Needs to be done for each and every HDMI driver. Not good.
- Internal delay, should not affect sample counts reported by snd_pcm_delay().

- Delay at PA level
  - Would need to obtain delay information from ALSA
  - No such interface at this time
  - May interfere with latency computations
    - Need to remove delay from sample counts/latency reports to avoid influencing A/V sync in app

Application

- A/V sync handled by application, would be possible to apply delay and take delay into account
- Would need an interface from PulseAudio to know by how much the audio is delayed

Conclusion

Minor enhancements of interfaces needed to allow for

- Seamless handling of hardware-accelerated audio
- Better HDMI experience

- This is not rocket-science
  - Mainly dependencies on others
- Let’s work together on this
  - Would benefit the desktop as well.

Questions?