PipeWire in the heart of car multimedia

George Kiagiadakis
Senior Multimedia Engineer
What is PipeWire?
Are you familiar with PulseAudio?

- Sound system / Sound card proxy for audio applications
- Transfer audio between machines
- Change formats, mix, re-position on the fly
- Device auto-configuration, including Bluetooth devices
- Echo cancellation & other effects
PipeWire

- Initial idea: PulseAudio for **video**
- Now: generic multimedia platform service
  - Video capture server
    - Camera and other video sources (ex. gnome-shell screencast)
  - Audio server
    - *PulseAudio* and JACK (pro-audio) replacement
    - Borrowing ideas also from CoreAudio, AudioFlinger, and others...
  - Now implementing the audio system in **Automotive Grade Linux**
Architecture

PipeWire daemon

Application 1
- eventfd / memfd / dmapuf

Application 2
- eventfd / memfd / dmapuf

Remote audio (Bluetooth / RTP / Airplay / DLNA) manager
- eventfd / memfd / dmapuf

Session Manager
- UNIX Socket

ALSA

V4L2
Architecture

• Multi-process, graph based processing
  - Simple JACK-like scheduler

• Extensible: types, protocols, ...

• Plugins based on SPA (Simple Plugin API)
  - Header-only C library with zero dependencies
  - Extremely lightweight data structures
  - “Like GStreamer, but not so heavy!” - Wim Taymans

• External session manager
PipeWire Session Manager

• Setup of devices
  - DSP processing
  - Mixers
  - Effects

• Management of links/nodes

• Security and access control of clients

• Policy
Performance & Efficiency

- Zero-copy with modern linux kernel APIs (memfd, dmabuf)
- eventfd & timerfd to wake up the processes
- Low-latency real-time capable + standard high-latency
  - < 1.5 milliseconds possible on desktop
- Much lower CPU usage than PulseAudio
CPU Usage Statistics

Playback of a 24bit 96kHz 5.1 channel file, downmixed to 3.1 and resampled to 48kHz

- Measurements:
  - 21.33 ms (1024 samples / buffer)
  - 1.33 ms (64 samples / buffer)
  - 1.33 ms with 2 clients
  - 1.33 ms with CPU pinned @ 800 MHz

- Measurements on Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz

- Comparatively, on 1.33 ms, PulseAudio uses 100% CPU and fails (underruns)
Security

• Fine-grained object access controls per client
  - Visible (R)
  - Write data (W)
  - Execute methods (X)

• Each client can be made to “see” an entirely different graph

• Session manager applies permissions
Who is behind this

• Author: Wim Taymans
  – Well-known old GStreamer developer & ex-maintainer
  – Sponsored by: Red Hat

• Embraced by PulseAudio developers
  – Seen as the next generation of PulseAudio

• Welcomed by ALSA and JACK developers

• License: MIT
Status

• Version 0.2 distributed in fedora
  - Used for video only

• Version 0.3 to be released soon
  - Estimated for later this year
  - A lot of audio work & refactoring done
  - Used in AGL
PipeWire in AGL
PipeWire in Automotive Grade Linux

- Audio system implemented with PipeWire, replacing 4A
  - Mixer, Media player & Radio player using the native API
  - ALSA compatibility plugin available
- Session manager: WirePlumber
- Merged in Happy Halibut (8.0.0) RC2
- In the future also: video
WirePlumber

• First session manager implementation
• Target: reusable session manager for embedded use cases
  - we’ll see about desktop...
• Modular & extensible, like PipeWire
• Based on GObject
  - To support writing modules using bindings in other languages (TODO)
WirePlumber concepts

• Graph abstraction: Endpoint
  - Closer to PulseAudio’s representation (sources, sinks, source-outputs, sink-inputs)
  - Described by name & media-class (Audio/Source, Audio/Sink, Stream/Audio/Input, Stream/Audio/Output)

• Extensibility: Modules & Factories

• Pipewire Proxies
  - GObject-ify the pipewire API (ease of use + access from bindings)
Endpoints

- Abstract the graph around the device
- Abstract controls (volume, mute, brightness, contrast, …)
- Abstract link / unlink operations
- Abstract streams
  - paths to/from the device with specific properties
- Bridge to functionality implemented in hardware
Software DSP Endpoint
Policy Management

- Policies implemented by modules
- Clients have roles with priorities:
  - Multimedia, Navigation, Emergency, ...
- Highest priority wins
- Equal priority: last one wins
- Audio in other apps is automatically paused & restored
Policy Management TODO

- Re-work the internal API
- Introduce bindings for scripting languages
  - Allow people to define custom logic with scripts
- Allow mixing streams with different volumes
- Volume ramping & cross-fading for changes
- Stop signal, in addition to pause
Other areas needing work

- Bluetooth audio support (planned for AGL 8.0.1)
- Unicens hardware support (planned for AGL 8.0.2)
- Better configuration
  - Currently limited options in wireplumber.conf
- Better security
  - Mechanisms exist but currently all clients are granted full permissions
- Documentation
PipeWire Audio APIs
Audio Stream (pw_stream)

- Nicer than PipeWire low-level API
- Takes input from client (asynchronously)
- Does conversion
  - Resampling
  - Channel mixing / volume
  - Format conversion
  - Channel splitting into DSP
  - Decouples server buffer size from client requested latency
- Flush / drain
In GStreamer

• pipewiresrc / pipwiresink
  - Available upstream
  - Built mainly for video; have issues with audio at the moment

• pwaudiosrc / pwaudiosink
  - Available in AGL
  - Smooth operation with audio
  - Upstream-Status: Submitted [https://github.com/PipeWire/pipewire/pull/140]

• All built with the streams API
Compatibility APIs

• ALSA apps
  – PipeWire PCM plugin
  – Built with streams API

• PulseAudio apps
  – Replacement libpulse.so, libpulse-mainloop-glib.so
  – Built with streams API

• JACK apps
  – Replacement libjack.so
  – Built on top of low-level PipeWire
Mixer API

- Mixer controls are implemented in the SM
  - But exposed through PipeWire
- Upstream: no support
- In AGL:
  - Mixer controls exposed through a virtual Endpoint object
  - `audiomixer` binding for easy access
  - Design subject to changes - should be upstreamed
WirePlumber API

• For wireplumber modules only
• Implement endpoints
  - add support for custom hardware, filters, etc...
• Implement policy
• **Unstable** – subject to change
Showtime
Thank you!
We are hiring
col.la/careers