ALSA Project Status Update

Takashi Iwai <tiwai@suse.de>

SUSE Linux Products GmbH, Nuremberg, Germany

Gstreamer Conference Aug. 28, 2012, San Diego

Outline

- Introduction for ALSA
- Pain points
- Recent and new updates

ALSA: Introduction

ALSA: Myths

I see an announcement of ALSA grand conference.

- True... but not ours. We aren't that cut.
  - Alpaca Llama Show Association

ALSA: Myths

I heard ALSA implementation on D-Bus. Is it true?

- Sort of... but not ours. It's driving cars.
  - Automóviles Luarca, S.A

ALSA: Myths

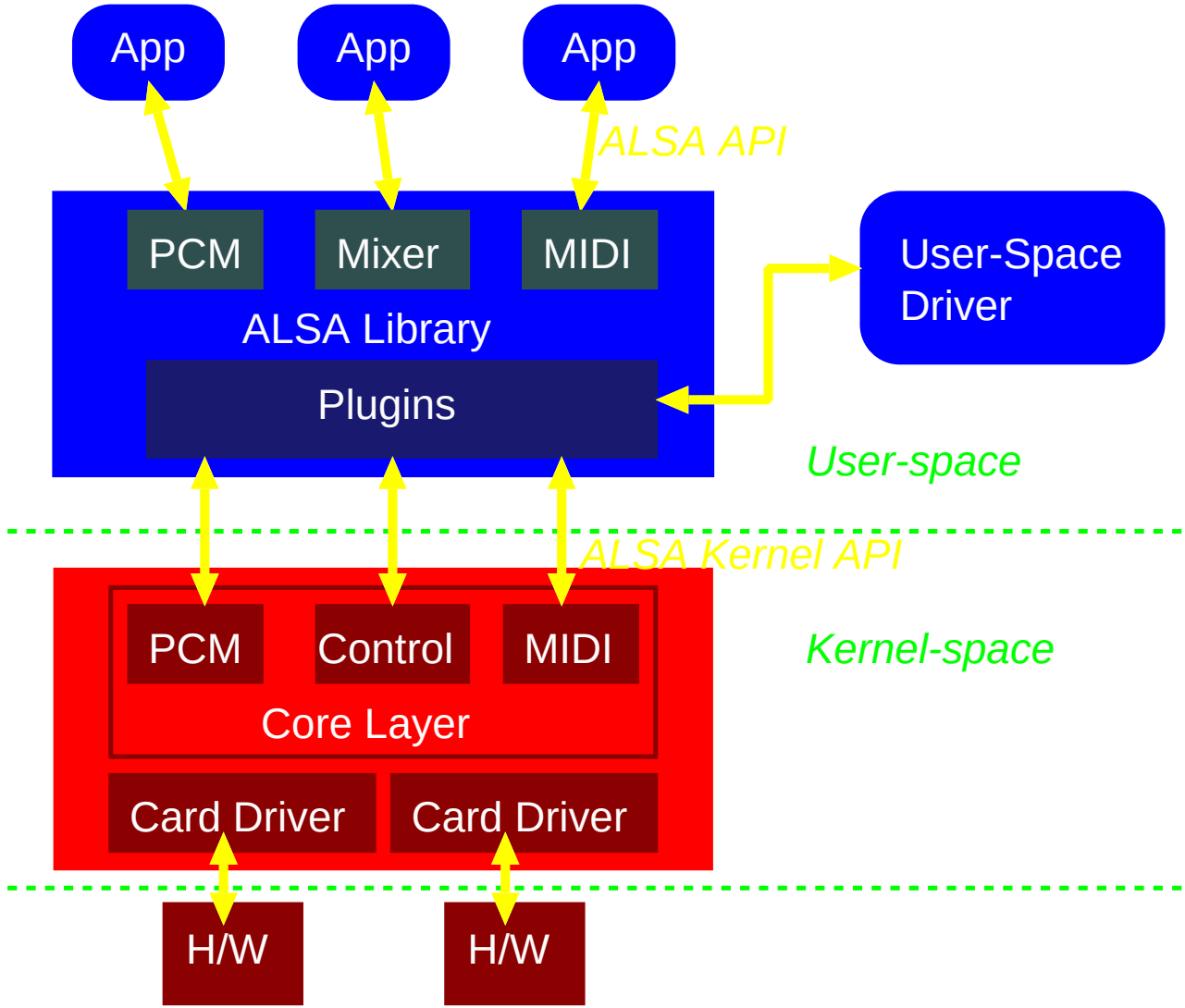
So, ALSA is about soft things?

- Sort of... but not like below.
  - We are dealing only with device driver software.

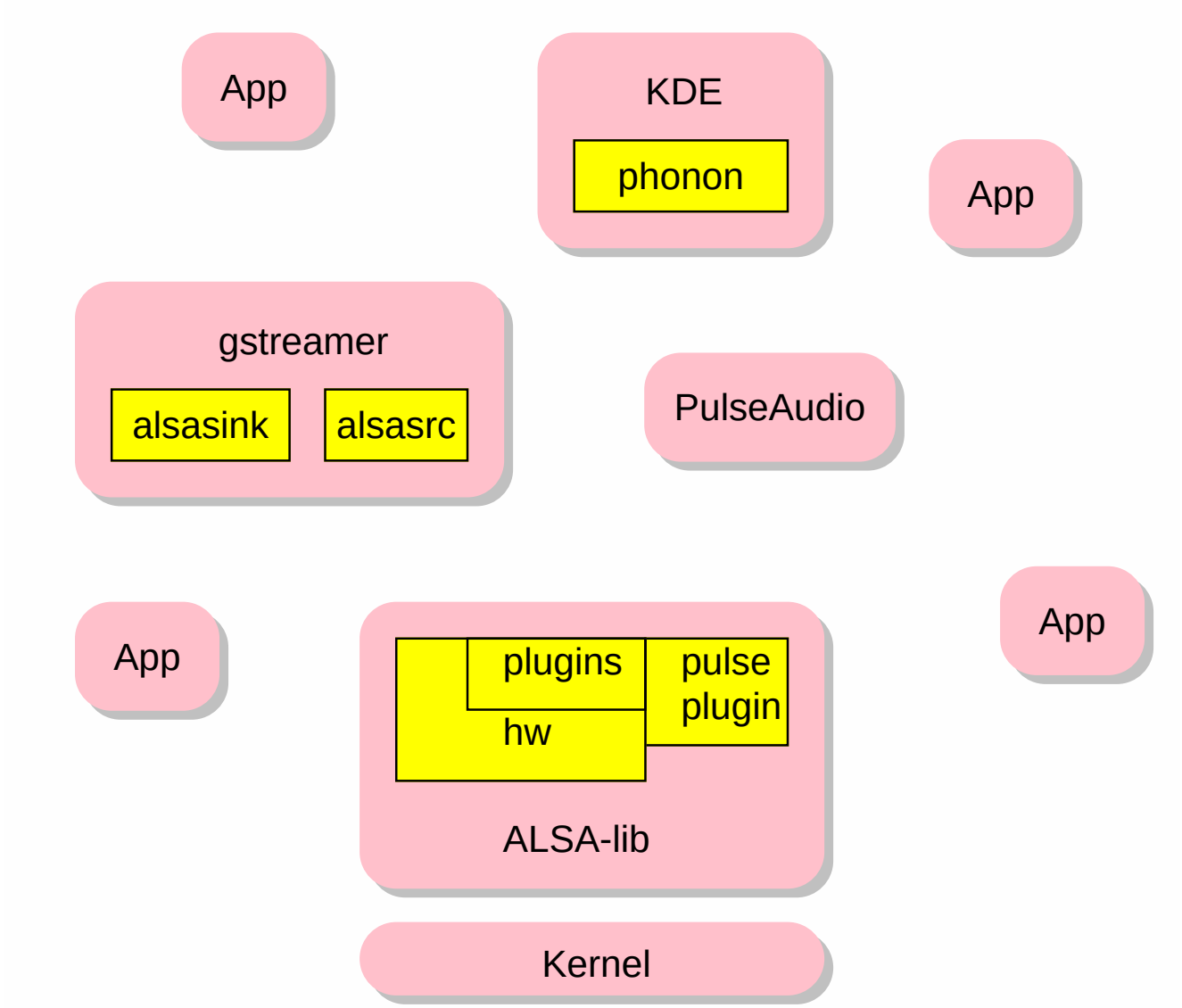
ALSA: Brief History

- ALSA = Advanced Linux Sound Architecture
- Project started by Jaroslav Kysela in 1999
- Major code change in ALSA 0.9.x series
  - Vicious alsa-lib API was defined at this moment
- Merged to Linux 2.5 kernel, replacing OSS
  - HD-audio since 2.6.12
  - ASoC merged in 2.6.21

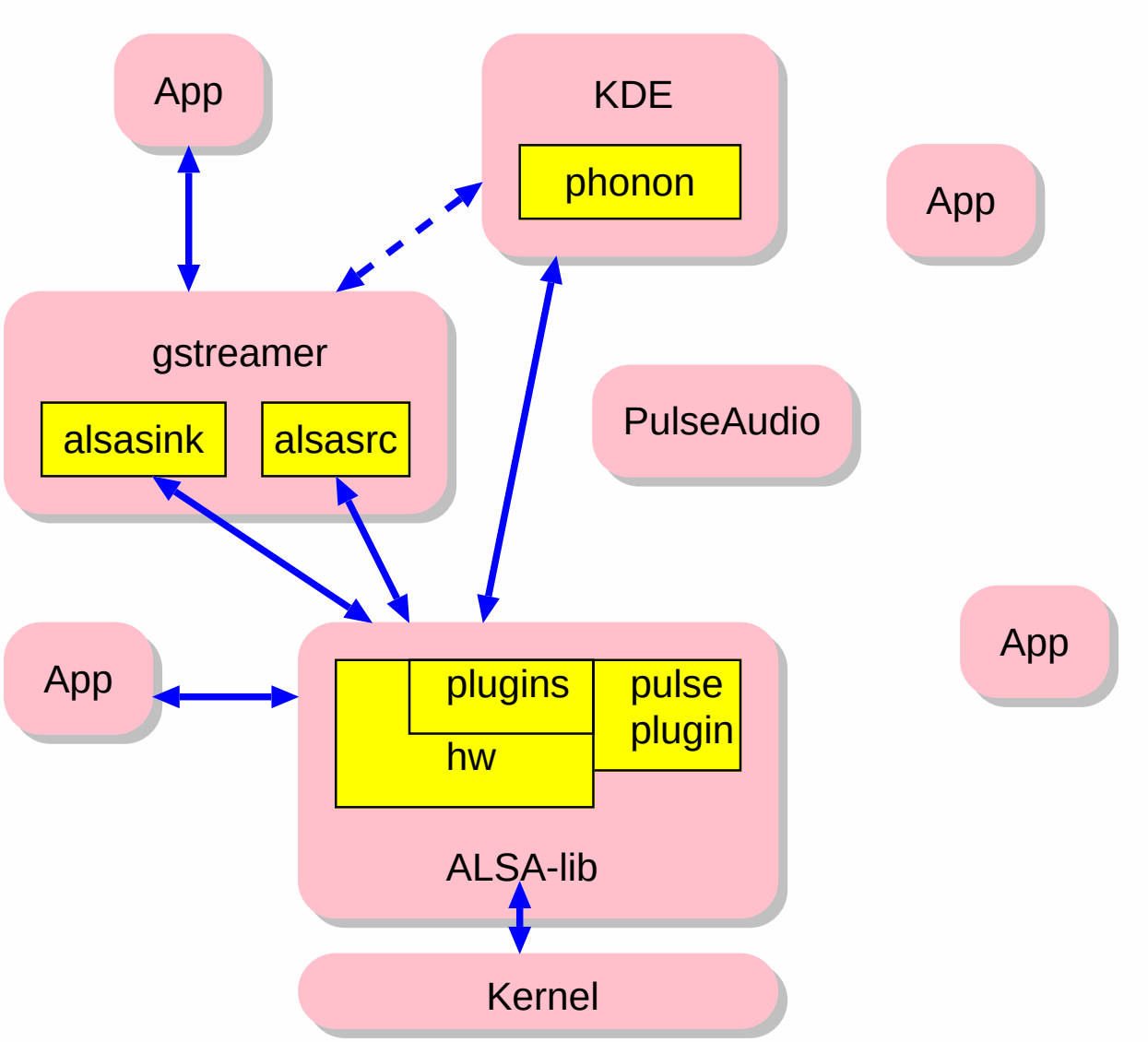
ALSA Big Picture



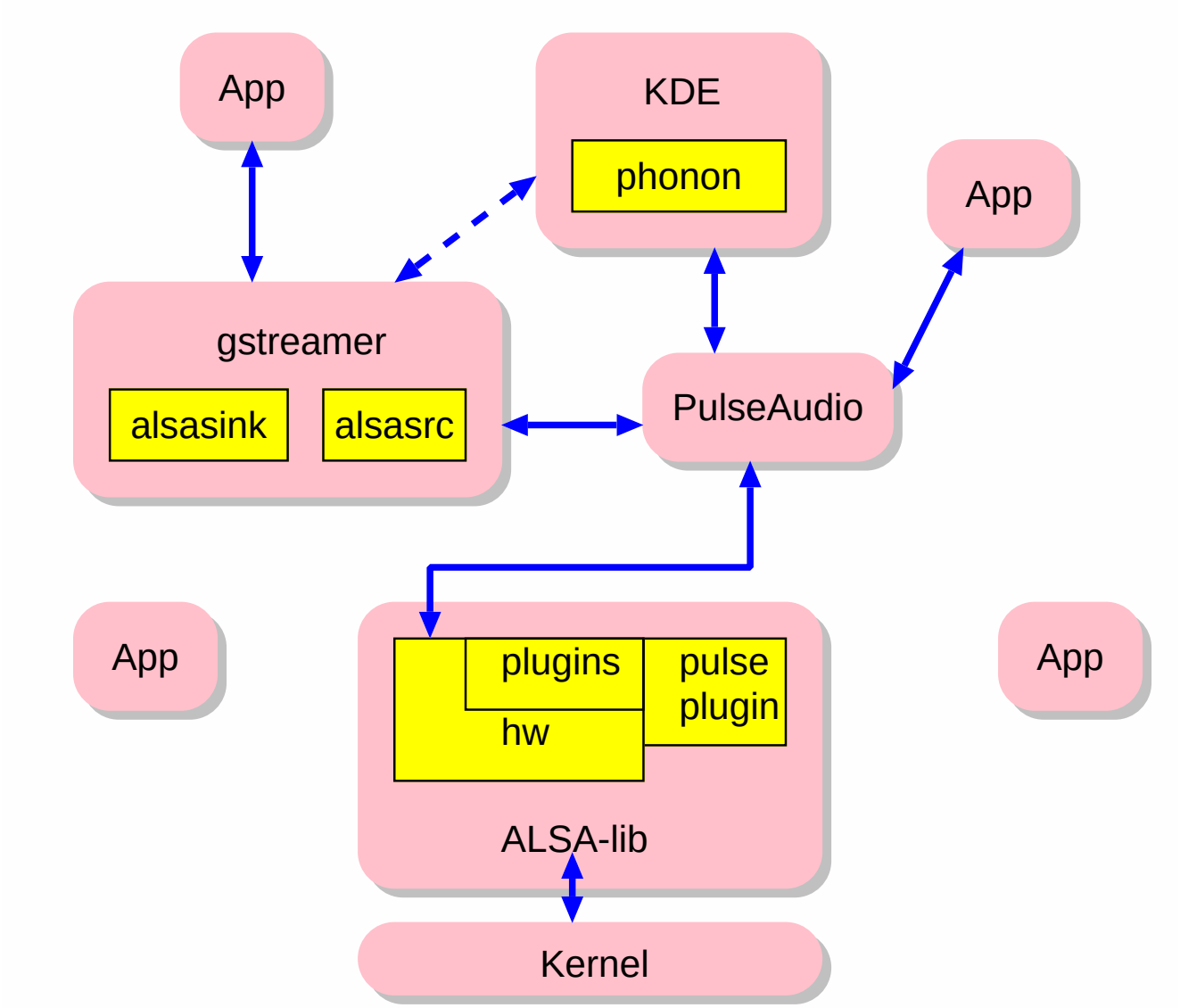
Bigger Picture



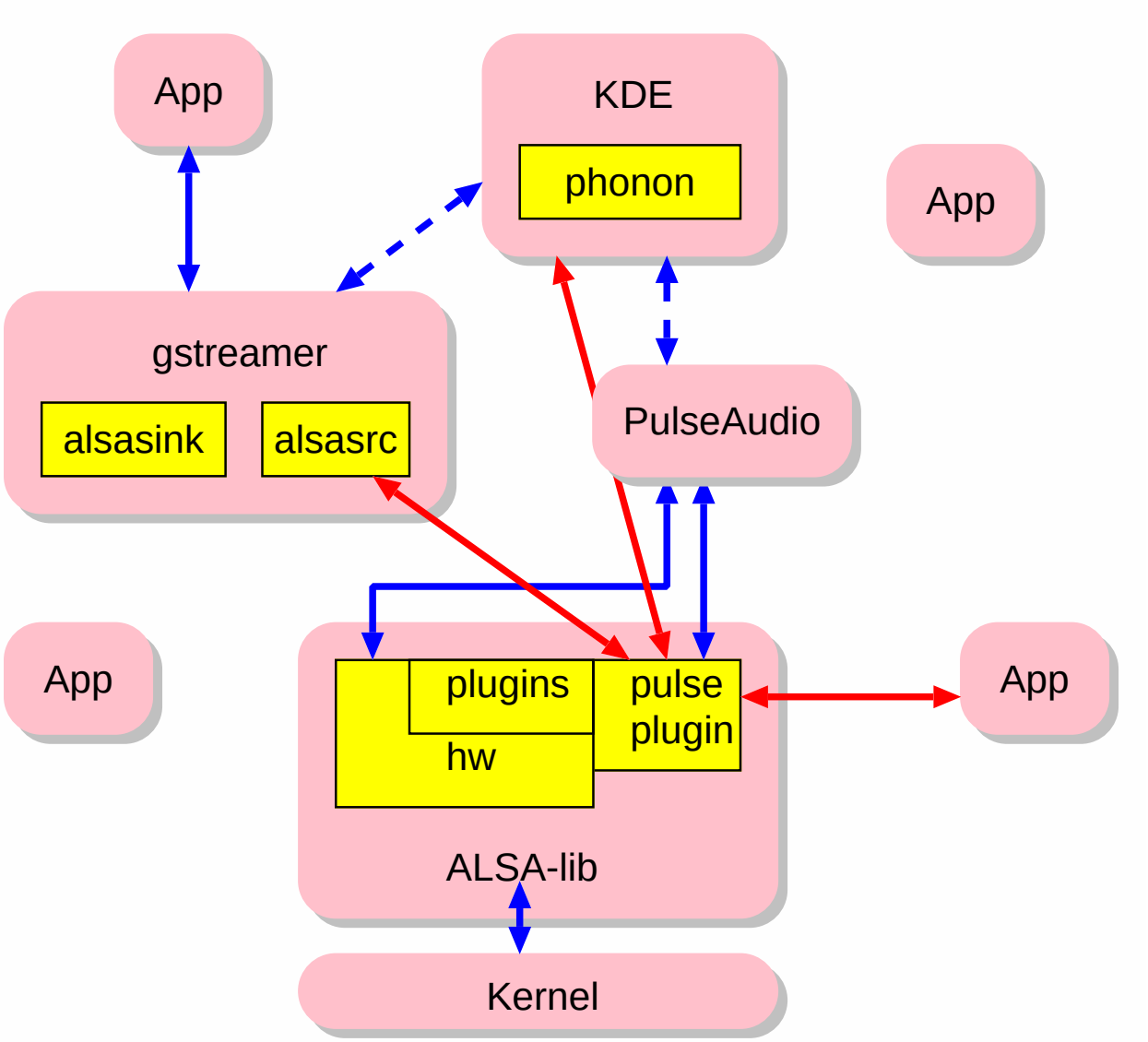
Bigger Picture: ALSA-native Routing



Bigger Picture: PA-native Routing



Bigger Picture: Indirect PA Routing



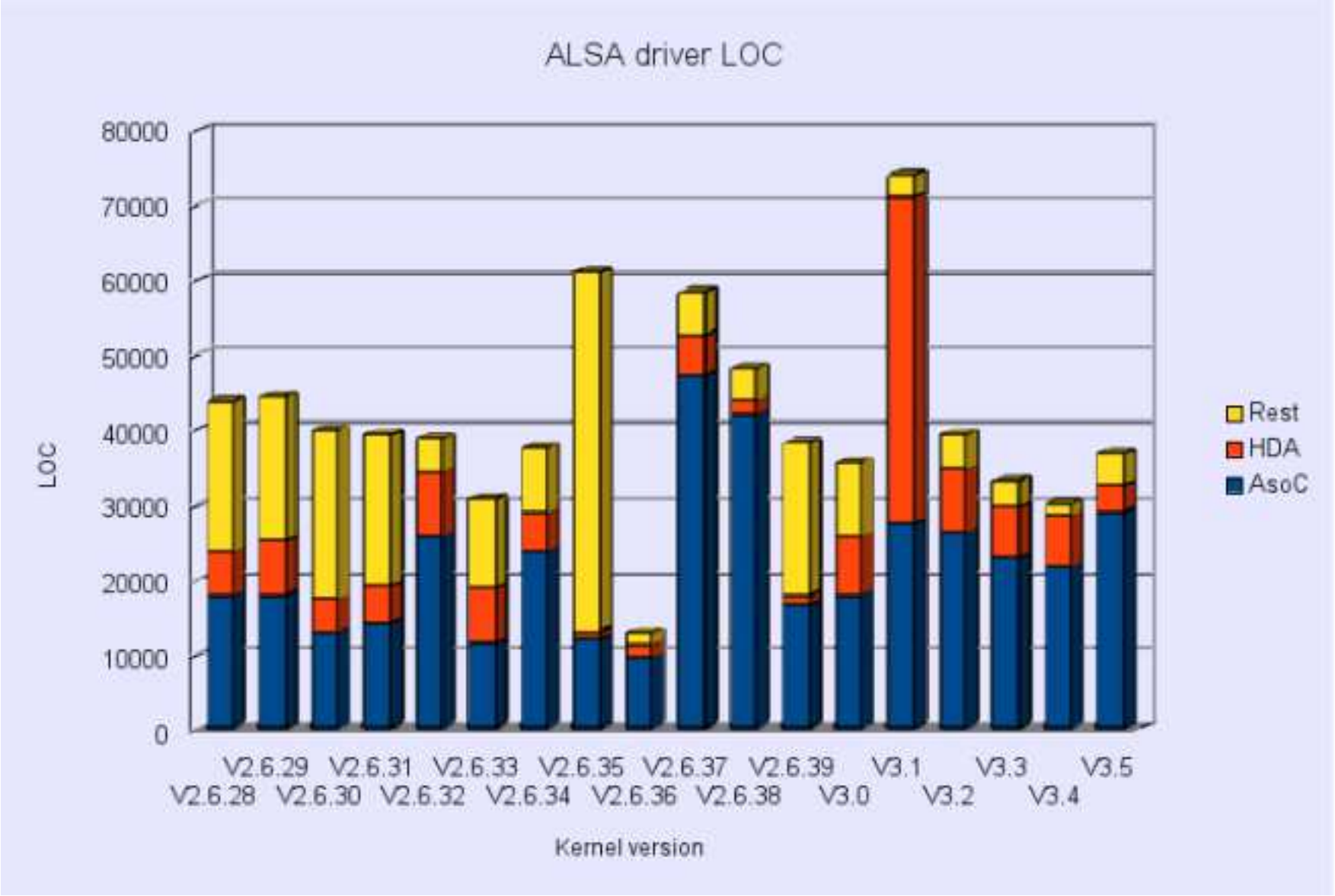
ALSA Kernel Driver

- Highly modularized
- Core parts
  - Card: the toplevel management
  - Control: control elements for mixer, etc
  - PCM: you know it
  - Timer, rawmidi, hwdep, seq, ...
  - OSS emulation modules
- Driver parts
  - PCI, USB, ASoC, legacy drivers...

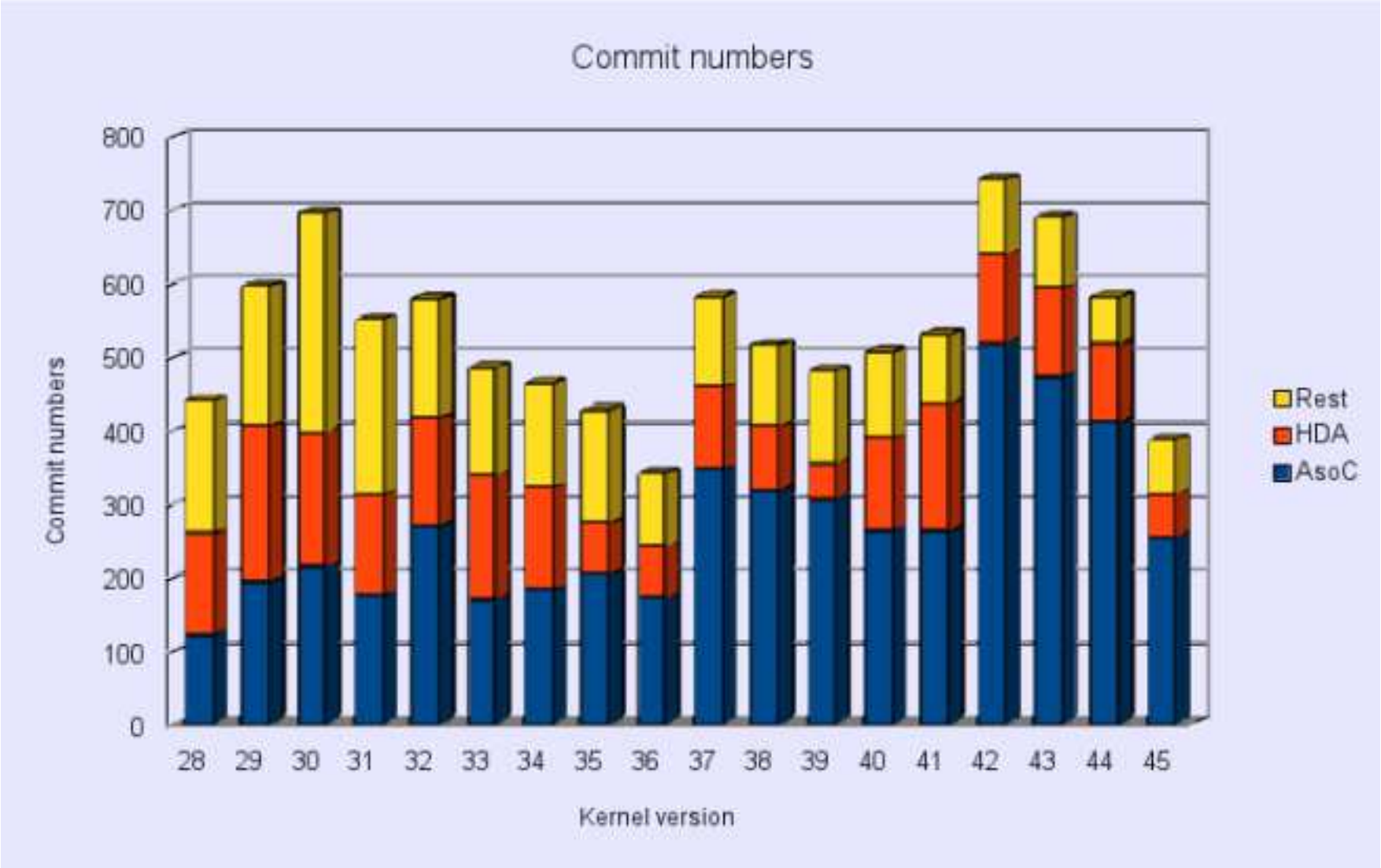
ALSA Kernel Driver Statistics

- Steadily active development over years
  - Most active part: ASoC
  - HD-audio tends to small commits, one large cleanup

ALSA Kernel Driver Statistics



ALSA Kernel Driver Statistics



Major Driver Components

- HD-audio
  - Controller driver (snd-hda-intel)
  - Codec library module (snd-hda-codec)
  - Codec drivers (snd-hda-codec-*)
- ASoC
  - ALSA sub-layer, targeted for embedded devices
  - ASoC core: PCM, DAPM, using regmap
  - Individual codec drivers (over 100)
  - Individual machine drivers
- USB-audio
  - Single generic module
    - For both USB audio v1 and v2

ALSA-Library - User-Space Layer

- API entry point
- Plug-ins
  - Absorbs the hardware incompatibility
    - Format, sample rate conversion, down/up-mixing
  - Soft-mixing and multiplexing from/to multiple streams
  - Software volume control
  - Real-time encoding
  - Communication with user-space drivers
    - JACK, PulseAudio, Bluetooth, ...
- Alternatives
  - Android’s own implementation: tinyALSA
  - SALSA-library for embedded devices

ALSA-lib API functions

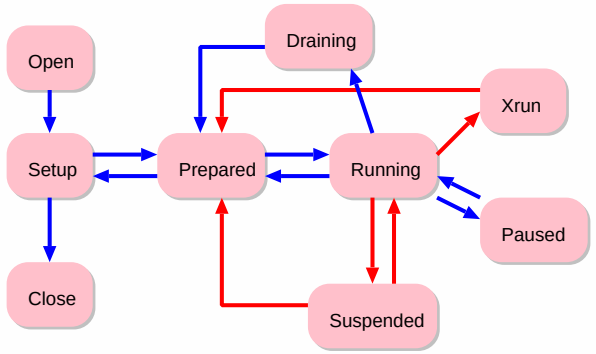
- Bold, gothic and subtle
  - Represent almost 1:1 for the driver implementation
  - Pretty stable over years
- Most data types are not exported to outside
  - Only accessor functions are provided

snd_pcm_hw_params_get_buffer_size_near()
snd_pcm_hw_params_set_buffer_size()
....

- Documentation still in a poor quality
  - Volunteer?

Abstraction Model: PCM

- Card / device / stream / direction / substream
  - A device file per stream direction level
  - A device may contain multiple substreams
- Buffer / period model
- PCM states
  - setup, prepared, running, paused, xrun, suspended



PCM (cont'd)

- Two staged parameter setups
  - hw_params
    - format, channels, rates, buffer/period sizes, etc
  - sw_params
    - start/stop threshold, alignment, etc
- Pause, suspend/resume
  - H/W-dependent implementation
  - Apps need to handle fallback cases
- Mmap support
  - sequence: begin / modify / commit
  - Channel information for non-interleaved streams
    - first offset & step size for each channel

Abstraction: Mixers

- Control API
  - An array of control elements
    - integer, boolean, enum list, byte array, IEC958
  - ID: name string, iface type, dev#, index#
- Mixers are a group of control elements
  - Kernel-level: no mixer abstraction
    - Grouping done in alsa-lib
  - Standard naming rules
    - "... Playback Volume", "... Capture Switch"
  - Standard name components
    - "Master", "Front", "Mic"

Extra Information

- Control elements are not only for mixers
  - Provide also card, PCM and other stuff
    - IFACE_CARD, IFACE_PCM, ...
- A TLV data assigned to each control element
  - Usually representing dB information
  - Can be extended to any type in theory
- ALSA-lib dB data handle
  - Better abstracted
  - For raw TLV, control and mixer APIs

snd_tlv_get_dB_range(), snd_tlv_get_dB(), ...

snd_ctl_get_dB_range(), snd_ctl_convert_to_dB(), ...

snd_mixer_selem_get_playback_dB(), ...

Configuration files

- System-wide or user config files
  - /etc/asound.conf, ~/.asoundrc
- A flexible (but cryptic) configuration syntax

```
pcm.mypcm {
  type hooks
  slave.pcm "hw:0"
  hooks.0 {
    type ctl_elems
    hook_args [
      { name "IEC958 Playback Switch"
        value true }
....
```

- Can override the "default" PCM, control, etc

```
pcm.!default "pulse"
```

Gstreamer Integration

- Fairly straightforward, simple and good
- Handles PCM and mixer
- Direct lookup of h/w devices for discovery

Pain Points

PCM Configuration

- hw_params dependencies
  - Params: period, buffer, format, channels, rate
  - Units: bytes, frames, time
    - They restrict (and conflict) with each other
  - Define preferred things first
    - For a larger buffer size, set buffer size first
- Available parameters depend on H/W & setup
  - Hardware: period-base and timer-base updates
    - Different periods and wake-up accuracy

Mess About Mmap

- A big contig. pages for audio buffer
  - No small page map/unmap like others (e.g. network)
- Cache coherency problem
  - x86: easy one, coherent architecture
  - ARM, MIPS, etc: non-coherent
    - Overhead in kernel handling
- Drivers with vmalloc buffer
  - e.g. USB-audio driver
  - Lack of proper coherent page allocations

Xrun?

- Sound glitches
  - Buffer underrun/overrun (xrun)
  - CPU scheduling latency by kernel
    - RT-task priority
  - Classical approach: bigger buffer, more periods
    - More periods -> more CPU wakeups
- PulseAudio
  - Own timer-based scheduling
  - Heavily relying on the accurate stream position
    - Often problems on HD-audio
    - Many workarounds in the driver code

S/PDIF, HDMI, DP

- IEC958 status bits
  - Non-audio, category, copyright, etc.
  - Managed by control elements in kernel driver
    - No good mapping to PCM stream -> TLV?
  - Passed via arguments of PCM open
- N:M connections for recent hardware
  - More pins (sinks) than converters (sources)
  - Connections are set dynamically at open
  - No direct connection with video: EDID check?
- Non-PCM streams
  - IEC958 status bits must be set properly
  - HBR: need fixed (eight) channels and maps

Device Management

- Indexed device registration
  - Doesn’t fit with udev well
  - index=0 isn’t always the best choice
    - HDMI can be on the earlier PCI slot
- Device listing API exists
  - e.g. "aplay -L"
  - Rarely used in the end
- More mess by fiddling with ~/.asoundrc

Multi-stream Mixing

- PulseAudio is there, isn’t it?
  - Dmix is usable, not doesn’t cover all H/W

Inconsistent Mixer Elements

- Mixer names are ambiguous
  - Master: does it really control over all volumes?
  - Front: is a front speaker or a front channel?
- Too many controls
  - Many drivers provide own controls
- Simplification possible, but not fully standardized
  - HD-audio: mostly consistent now
  - ASoC: doesn’t care, take use-case approach

Recent and New Developments

Use Case Manager (UCM)

- A high level device management abstraction
  - Originally targeted mobile phones
  - Since alsa-lib 1.0.24
    - Formerly "ALSA Scenario" by Liam Girdwoord & co
- Hardware routing and controls per use case
  - e.g. "phone call" vs "music"
  - Source/sink discovery
  - Master volume control definition
- Availability
  - alsaucm tool in alsa-utils package
  - Integration to PulseAudio

UCM Example

```
SectionUseCase."HiFi" {
   SectionVerb {
     EnableSequence [
        cdev "hw:0"
        cset "name='Headphone Playback Switch', on"
        cset "name='Headphone Playback Volume', 20,20"
     ]
     DisableSequence [
        ....
     ]
     Value {
       TQ "Music"
       PlaybackPCM "hw:0,0"
       PlaybackVolume "name='PCM Playback Volume' 90,90"
     }
  }
}
```

UCM Example (cont’d)

```
SectionUseCase."Voice" {
  File "voice.conf"
}

ValueDefaults {
  PlaybackCTL "hw:0"
}

SectionDefaults [
  exec "echo This is an example UCM config"
  cdev "hw:0"
  cset "name=’Master Playback Switch’ on"
]
```

Compression Offload

- For offloaded DSP handling
  - Playing compressed format by DSP without CPU load
  - Kenrel API similar like PCM
  - Developed by Vinod Koul, Pierre-Louis Bossart
- Availability
  - Kernel core API integrated since 3.3
    - ASoC integration (3.7)
    - Intel Medfield driver implementation (3.7)
  - Library code in separate git tree

Jack Detection

- Multiple ways for jack detections
  - No library API is provided yet
- Via input device
  - As switch to read / notify
  - No association with card#
- Via control API
  - Only with ALSA API
    - ".... Jack" control with IFACE_CARD
  - ALSA control read / notify
- Via extcon?
  - Originated from Android switch class

More PCM Time Adjustments

- Monotonic timestamp mode support
  - Required by PulseAudio
- Upcoming: wallclock timetstamp support
  - For precise hardware sample time
  - Talk at LPC by Pierre-Louis Bossart
- Open question: embed timestamp in stream?
  - Asked by V4L guys for long time
  - Separate timestamp sync stream?

Improved Power Management

- "No period" PCM mode
  - Used by PA, only for certain devices (HD-audio)
- HD-audio power-saving improvements
  - Fixed races (since 3.5 kernel)
  - Explicit power-saving trigger by parameter change (3.7)
  - Runtime PM integration (3.7)

More HD-audio Features

- Improved BIOS auto-parser
  - Most bugs can be fixed by defining pins correctly
  - Codes have been drastically reduced
  - Better jack retasking
- Robust and accurate position reporting
  - More workarounds specific to each controller
    - Adjustable position_fix option
  - Should give better results for PA
- Firmware "patch" loading
  - Changing pin config or others without recompiling

More HD-audio Features (cont'd)

- Debug / QA with emulator
  - alsa-info.sh output as base data
  - Can track codec registers and control elements
  - Automuated QA test by David Henningsson
- Non-snoop mode (non-cached memory)
  - Requirement by recent controller chips
  - No support for non-x86 platforms yet

Channel Mapping API

- So many different standards:
  - ALSA: FL / FR/ RL / RR / C / LFE ...
  - MS: FL / FR / C / LFE / RL / RR ...
- Kernel/user access via control API
  - One control per PCM substream
    - "Playback Channel Map" with IFACE_PCM
  - Query all channel maps via TLV
  - Get & set the current channel map via read & write
- ALSA-lib implementation
  - Simple: handle via int array
  - Transparent for plugins
    - Automatic correction for route & multi plugins

Channel Mapping API (cont'd)

- Proposed API functions
  - Return the list of available channel maps

```
int **snd_pcm_query_chmaps(snd_pcm_t *pcm);
```

  - Return the current channel map

```
int *snd_pcm_get_chmap(snd_pcm_t *pcm);
```

  - Set the channel map (optionally, only if h/w supports)

```
int snd_pcm_set_chmap(snd_pcm_t *pcm, const int *map);
```

- Still in discussions
  - How to manage multiple outputs per channel?
  - Different outputs (e.g. speaker, HP) in a single PCM
    - Allow to define channel map value in TLV?

Routing Exposure

- Expose connections among H/W components
  - For retasking multi-purpose I/O, etc
- Media controller API
  - API implementation by V4L guys
  - ioctl-based kernel API
    - Patch proposed by Clemens Ladisch
- ALSA control API
  - Embed connections and info in TLV
  - Might be too complex, ALSA-centric?
- What else?
  - Topic at LPC by Marc Brown

Resources

- ALSA kernel and build trees

git://git.kernel.org/pub/scm/linux/kernel/git/tiwai/sound.git
git://git.kernel.org/pub/scm/linux/kernel/git/tiwai/alsa-driver-build.git

- ALSA driver snapshot tarball

ftp://ftp.suse.com/pub/people/tiwai/snapshot/alsa-driver-snapshot.tar.gz

- ALSA library, utils, firmware, tools tree

git://git.alsa-project.org/alsa-lib
git://git.alsa-project.org/alsa-utils
git://git.alsa-project.org/alsa-firmware
git://git.alsa-project.org/alsa-tools

- SALSA library

git://git.kernel.org/pub/scm/linux/kernel/git/tiwai/salsa-lib.git