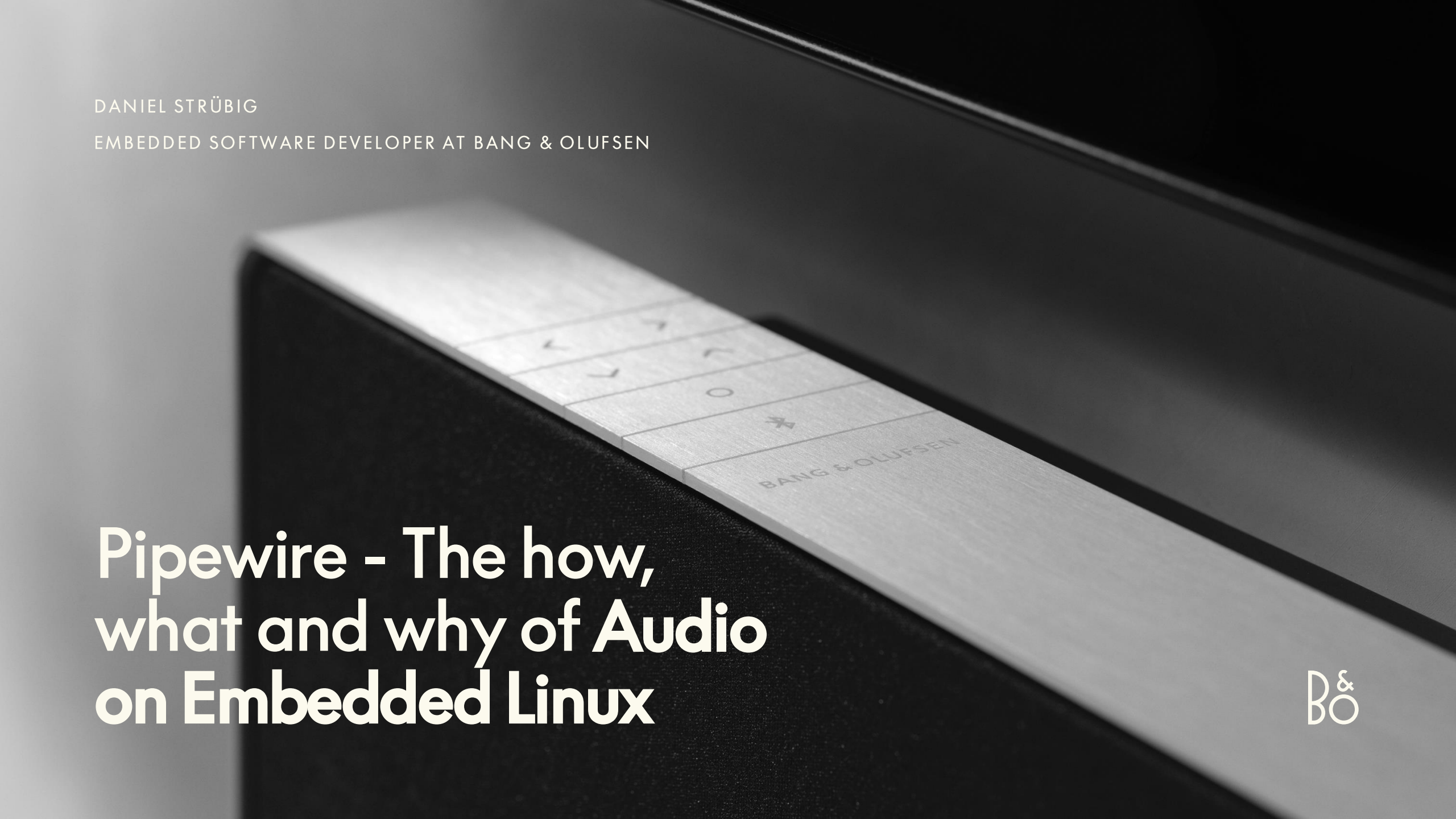
DANIEL STRÜBIG

EMBEDDED SOFTWARE DEVELOPER AT BANG & OLUFSEN



Pipewire - The how, what and why of Audio on Embedded Linux

Agenda

- How – Using the Advanced Linux Sound Architecture
- What – Facilitating audio I/O in userspace
- Why – Pipewire is a fit for our current platform

WHO WE ARE

Bang & Olufsen

Audio company, founded in 1925 in Denmark

Technologies: audio over (wireless) networks, Bluetooth, UWB, embedded Linux

Daniel

Embedded Software Developer

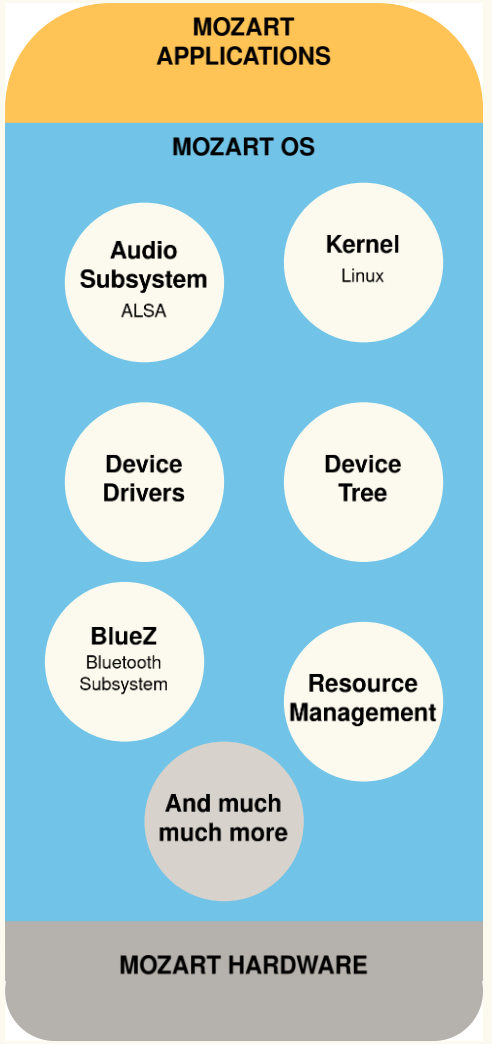
Previously worked on speech quality research, published libraries, mostly writing rust on the side

Mozart

Loudspeaker platform launched in 2019

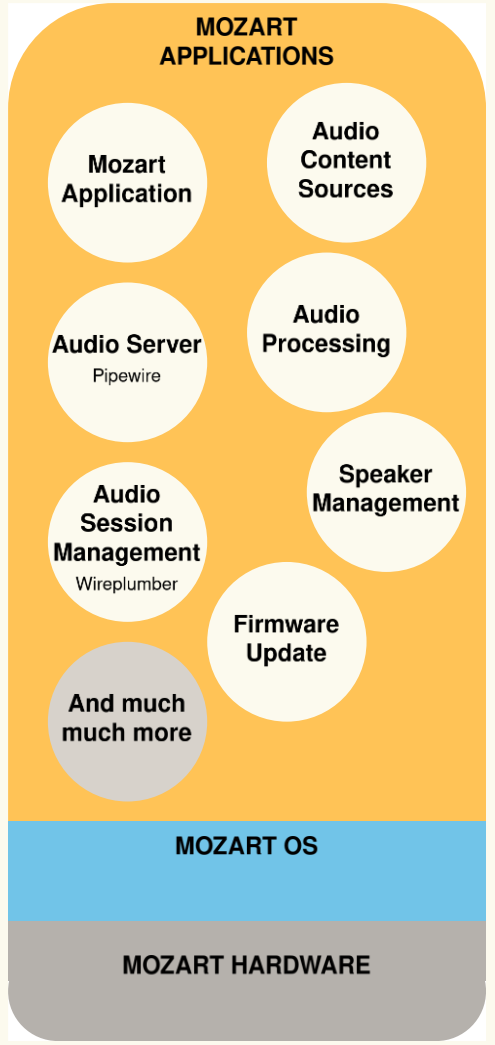
A little insight into our platform

- Custom Linux distribution
- Built with **Yocto**, Based on **Mainline Linux Kernel**
- Abstraction between the hardware and our code
- **Device drivers** and **device trees** to control and describe hardware
- Multiple subsystems, each responsible for handling their specific domain
- An ecosystem to facilitate both longevity and innovation



A little insight into our platform

- Userspace Applications written in C++17 and beyond
- Handles behaviour: State, configuration, User IO, http requests etc
- Heavy use of **FOSS** for hygiene functionality
  - **Systemd** for service management
  - **Iwd** for wifi management
  - **Bluez** for bluetooth
  - **Networkd** for network interfaces

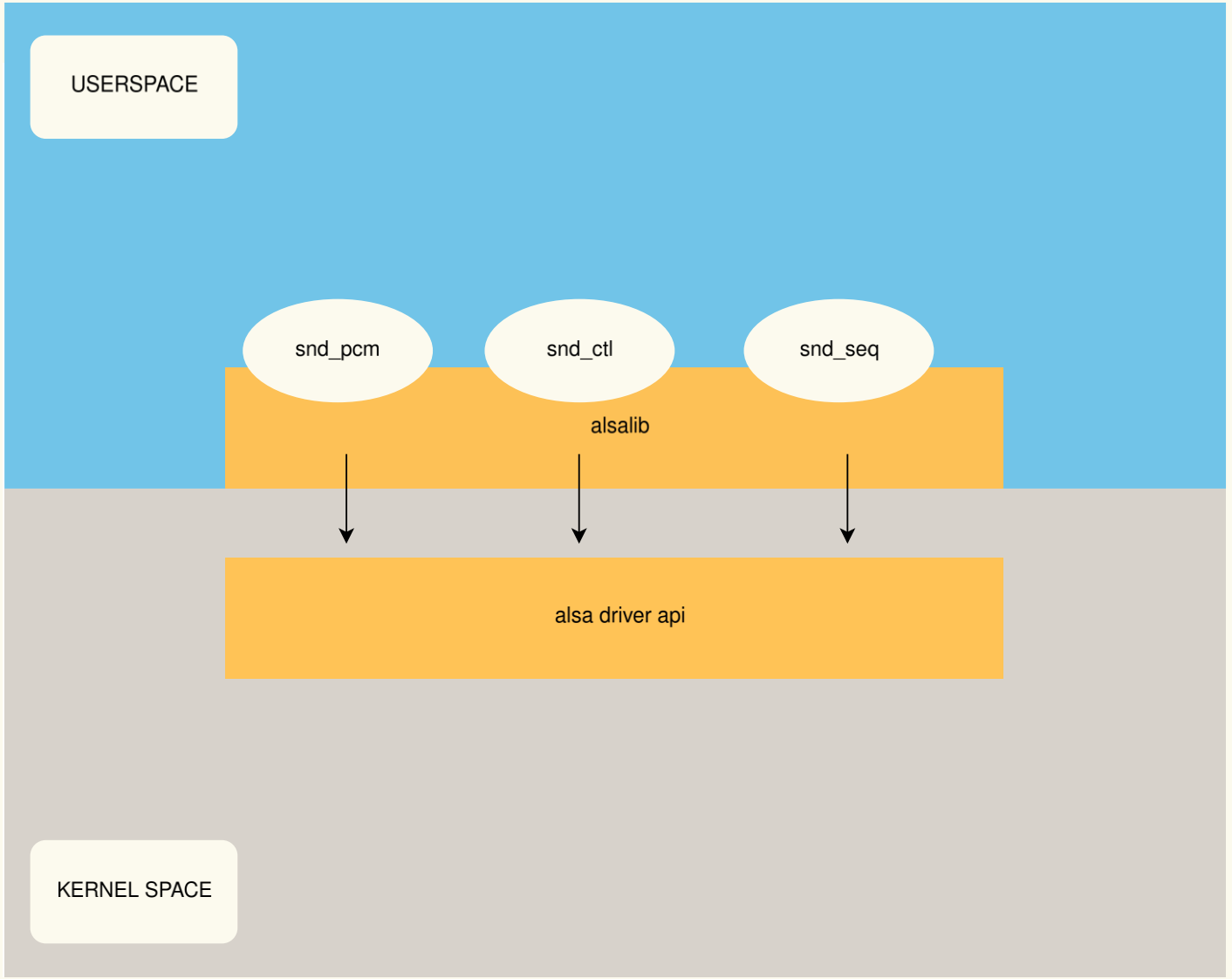


About today

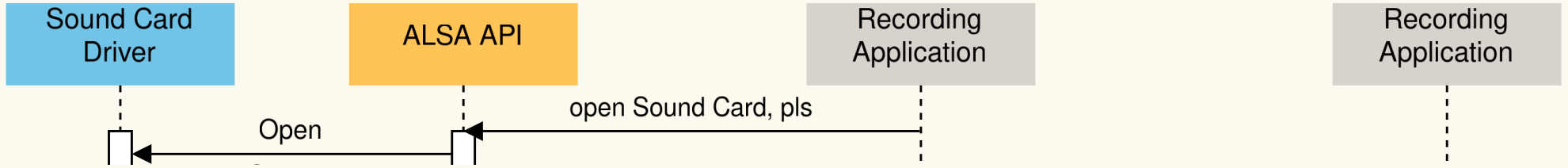
- Introduction to the Audio Subsystem on Linux
- Gradual transition from accessing a sound card to managing multiple applications dealing with audio
- Beware - Lots of terminal action
- We hope you are:
  - Familiar with the basics of audio programming
  - Eager to learn about the ecosystem in which your application runs

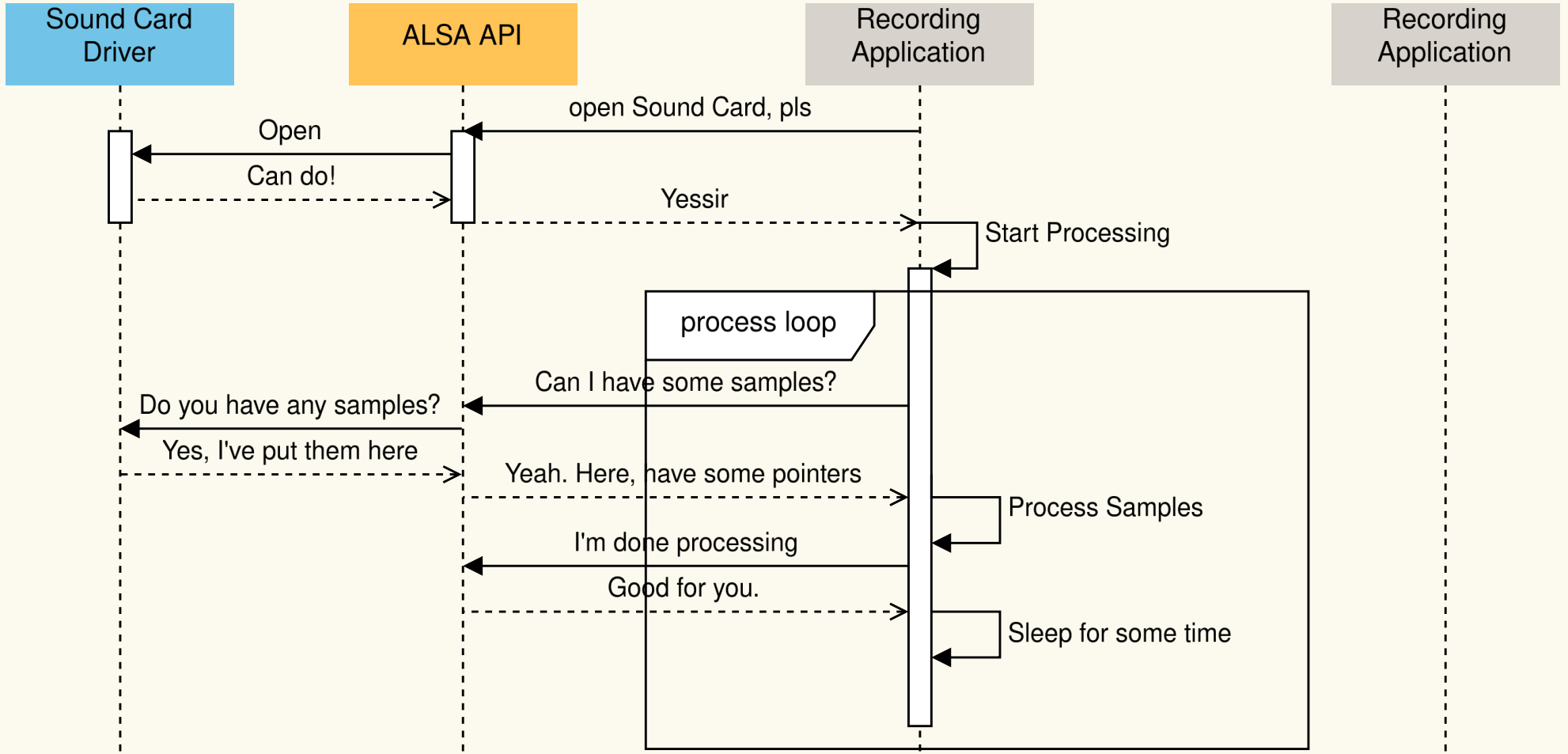
Alsa

- Linux is composed of subsystems: ALSA is one of them
- Provides a kernel api for writing a driver for a sound card
- Provides a userspace api for accessing peripheral sound cards
- Let's look at an example: capture audio from a sound card and write it to a file



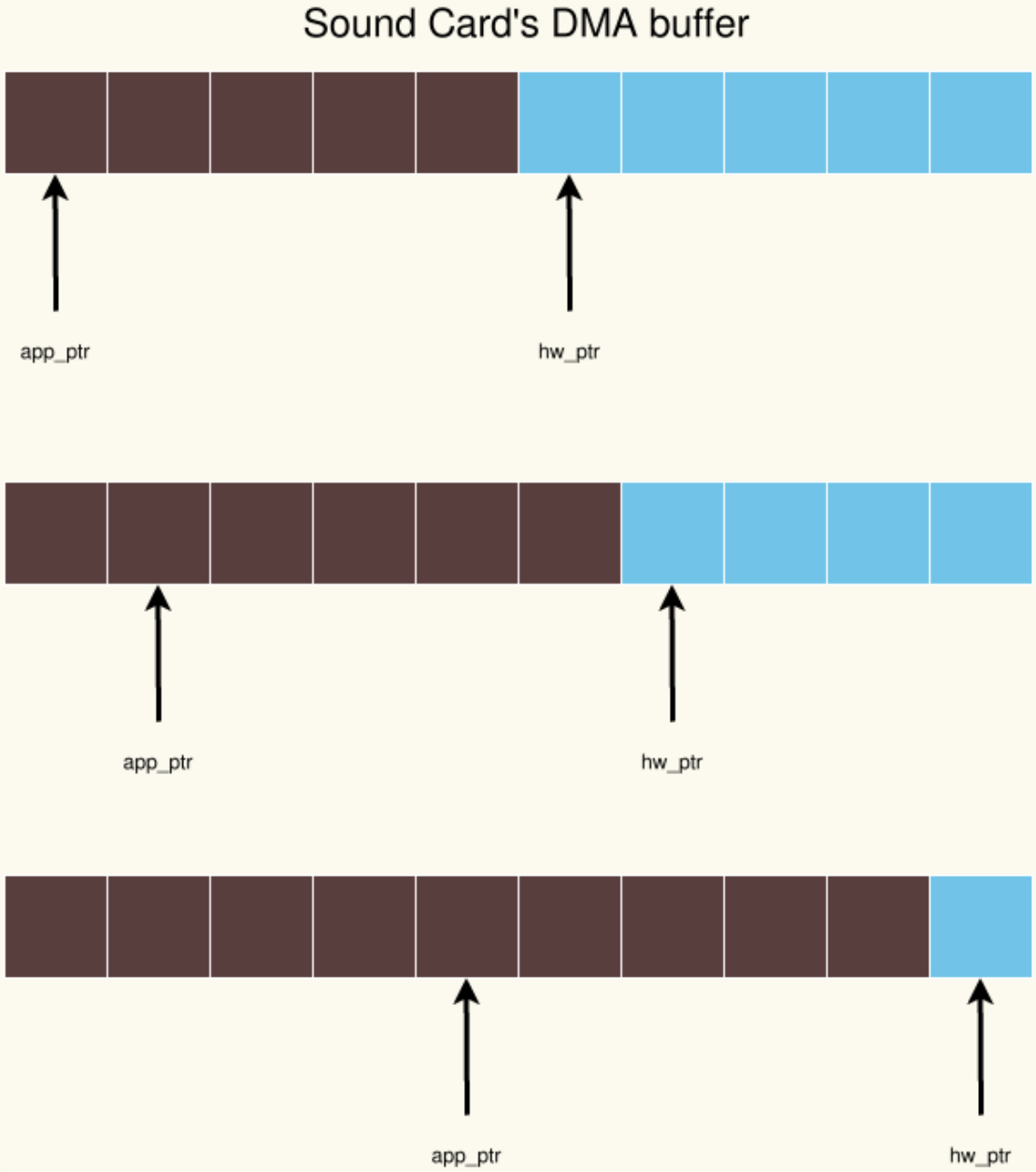

Sound Card Driver
ALSA API
Recording Application
Recording Application


Sound Card Driver
ALSA API
Recording Application
Recording Application
open Sound Card, pls
Open


Sound Card Driver
ALSA API
Recording Application
Recording Application
open Sound Card, pls
Open
Can do!
Yessir
Start Processing
process loop
Can I have some samples?
Do you have any samples?
Yes, I've put them here
Yeah. Here, have some pointers
Process Samples
I'm done processing
Good for you.
Sleep for some time

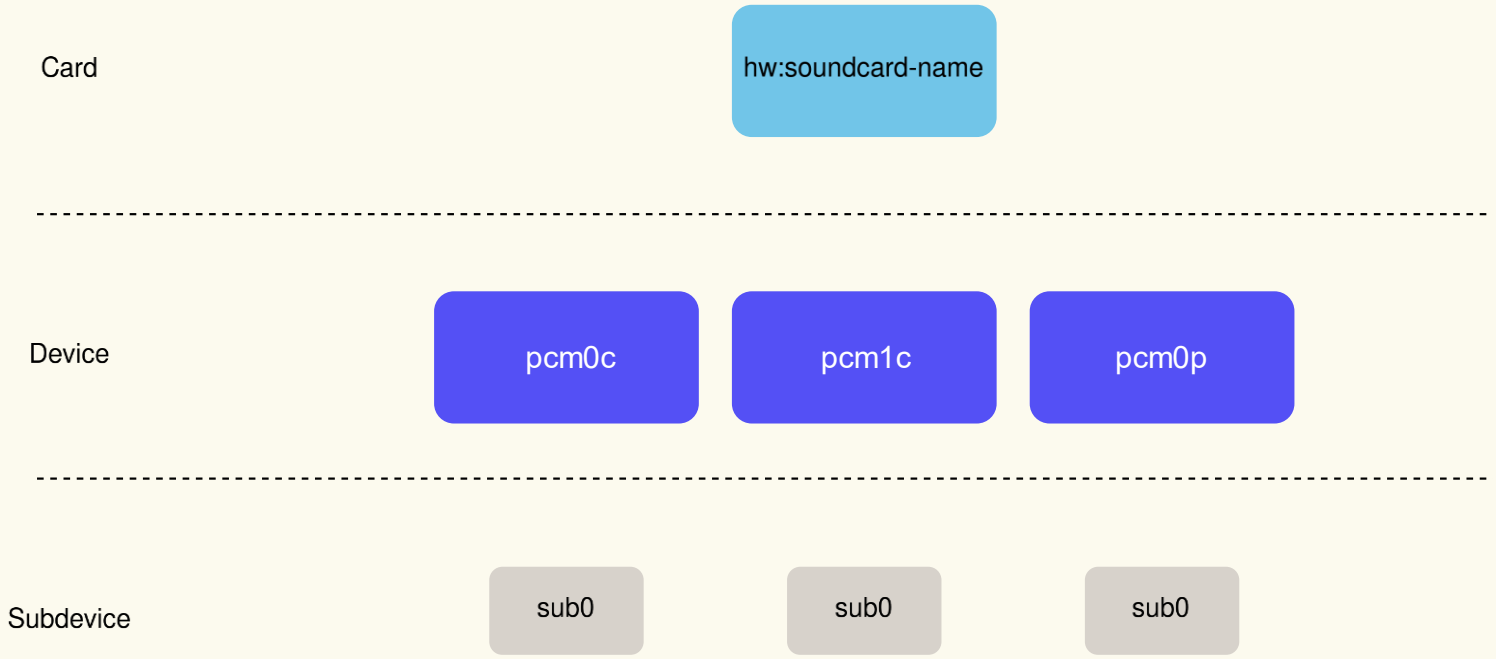
Alsa

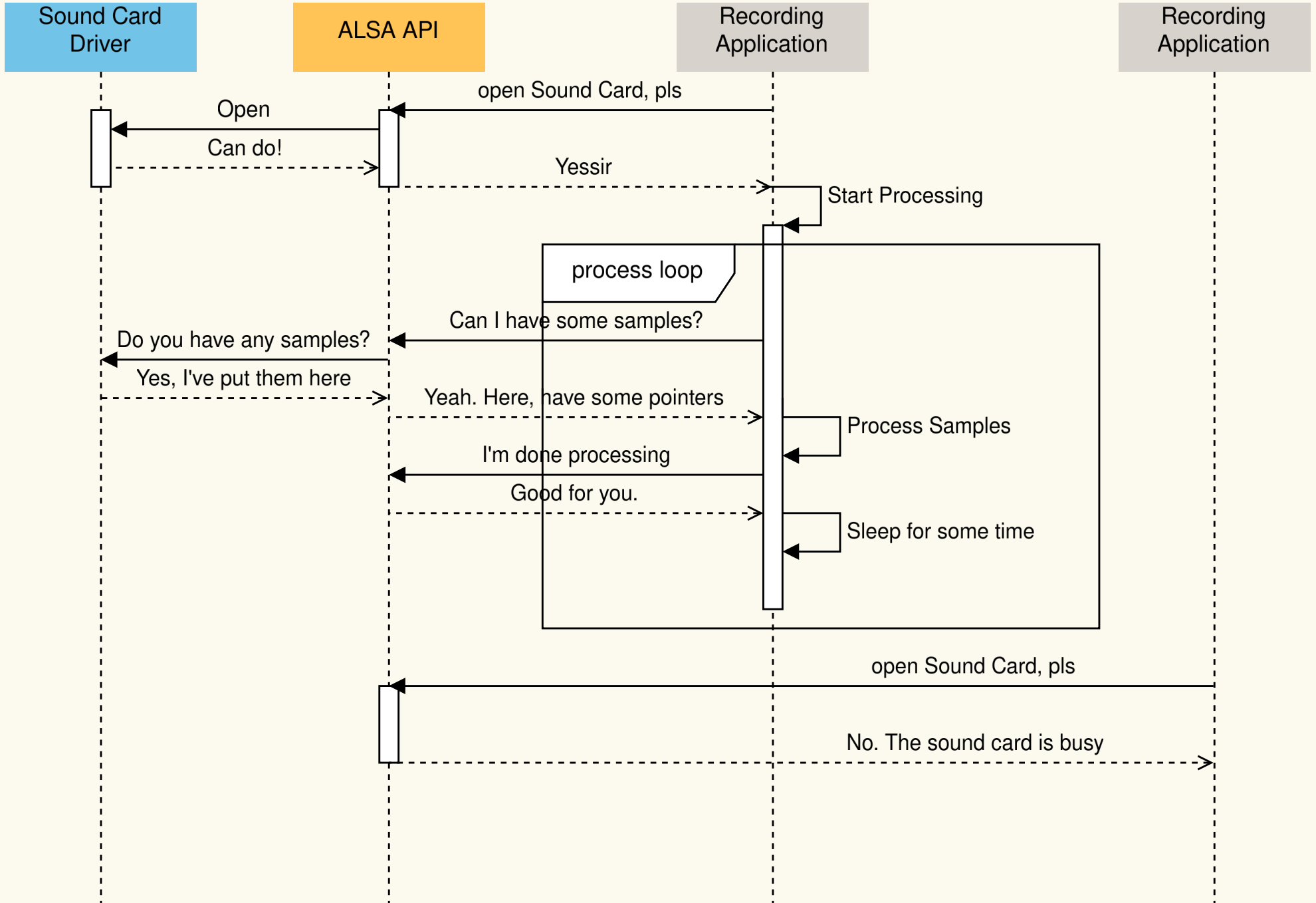
- Common config options:
  - Sample rate
  - Period size – interrupt frequency
  - Format
  - [...]



Alsa in action

- Application: arecord
- Monitor: /proc/asound subdirectories and files




Sound Card Driver
ALSA API
Recording Application
Recording Application
open Sound Card, pls
Open
Can do!
Yessir
Start Processing
process loop
Can I have some samples?
Do you have any samples?
Yes, I've put them here
Yeah. Here, have some pointers
Process Samples
I'm done processing
Good for you.
Sleep for some time
open Sound Card, pls
No. The sound card is busy

Limitations of ALSA

- Devices can only be opened by 1 process at a time
- Why should our application have to care about where the input signal comes from?
- Desirable: A single process that manages our sound cards and provides their signals to our applications
- Enter: The Audio Server

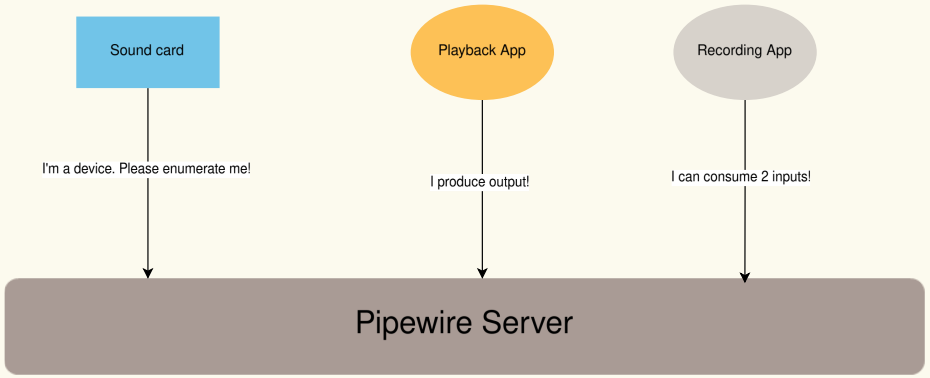
Brief Overview: Audio Servers on Linux

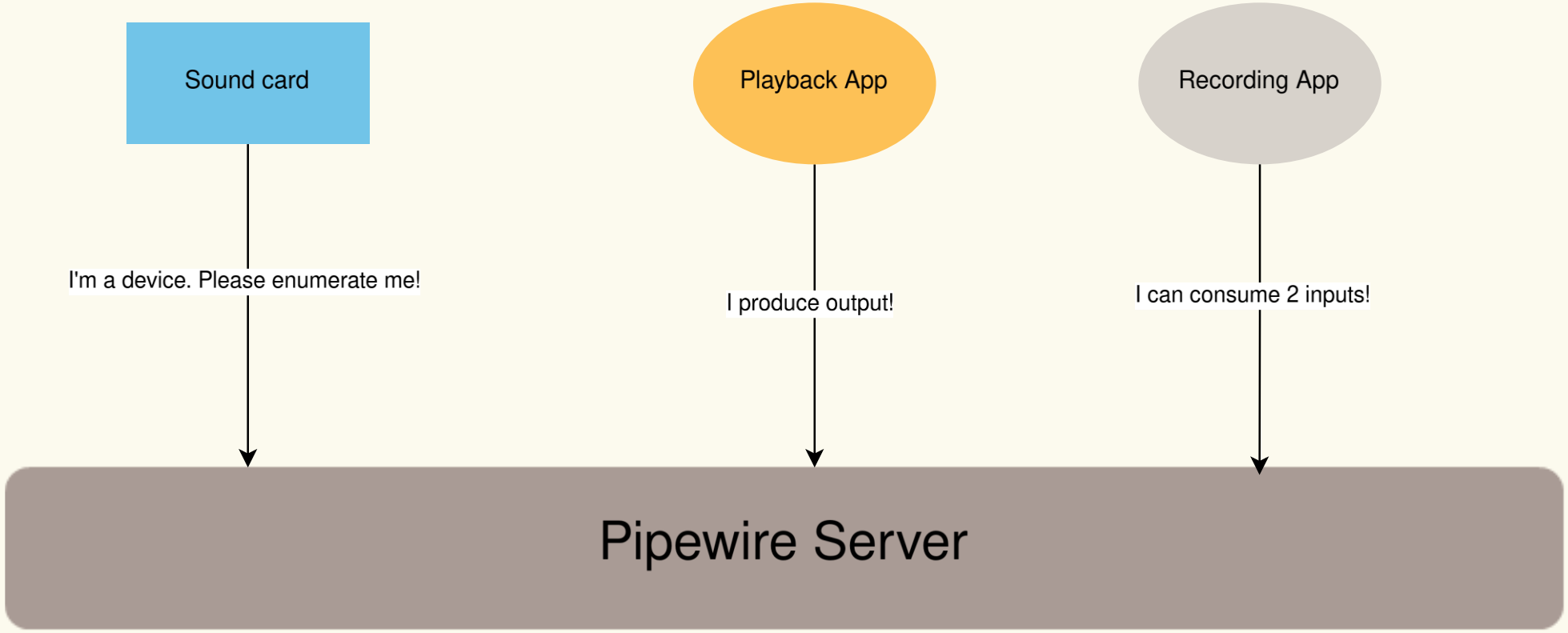
- PulseAudio: Popular solution for distributions with a desktop environment as a primary use case
- Jack: cross-platform audio server solution
- GStreamer: cross-platform GLib2.0 based media framework to programmatically spin up processing chains
- Pipewire: Developed by Wim Taymans at Red Hat, server to handle both audio and video processing
- We have recently migrated to Pipewire and it is now running on all mozart-based loudspeakers

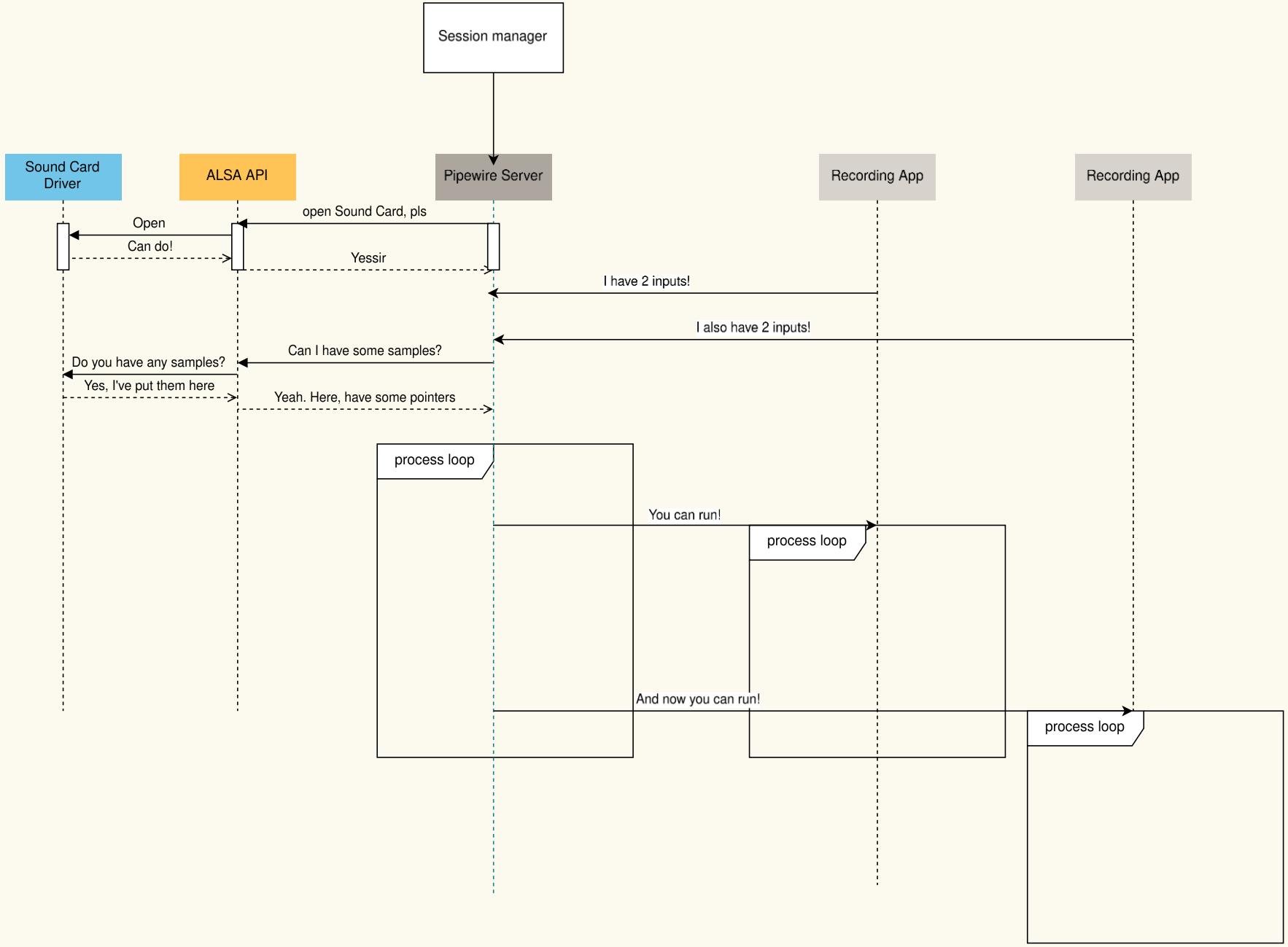
B&O

Some lingo

- Server:
  - background process, single instance on a system
  - **Not** a physical server rack
- Client:
  - Any application that can produce/consume audio
- Session Manager
  - Pipewire is mostly reactive, not proactive. It exposes an API to induce agency via a session manager. The most popular one right now is Wireplumber. We will not go into detail here.




Sound card
Playback App
Recording App
I'm a device. Please enumerate me!
I produce output!
I can consume 2 inputs!
Pipewire Server


Session manager
Sound Card Driver
ALSA API
Pipewire Server
Recording App
Recording App
open Sound Card, pls
Open
Can do!
Yessir
I have 2 inputs!
I also have 2 inputs!
Can I have some samples?
Do you have any samples?
Yes, I've put them here
Yeah. Here, have some pointers
process loop
You can run!
process loop
And now you can run!
process loop

Pipewire – What it solves

- Abstracts hardware into queryable objects
- Exposes functionality for applications to share audio signals
- Performs automatic resampling for 2 connected nodes running on different sampling rates
- Is responsible for scheduling the audio graph (node a runs before node b)
- Helvum, qpwgraph and coppwr can help you introspect Pipewire

B&O

Batteries included

- Resampler
- LADSPA / LV2 host
- Builtin primitives such as
  - Biquads
  - Convolver
  - Sine generator
  - Delay
- Bluetooth I/O

Backwards compatibility

- Pipewire provides ABI-compatible interfaces to interact with a jack client
- This means: If you have an existing application based on e.g. jack, it can communicate with the pipewire daemon

You know now

- How you can access sound cards on Linux
- What an audio server is and what problem it solves
- When it makes sense to use an audio server

DANIEL STRÜBIG
EMBEDDED SOFTWARE DEVELOPER AT BANG & OLUFSEN

Thank you!

Pipewire - The how, what and why of Audio on Embedded Linux