

WirePlumber

Present Challenges And Future Directions

Julian Bouzas

Senior Software Engineer

GStreamer Conference 2025

London, UK

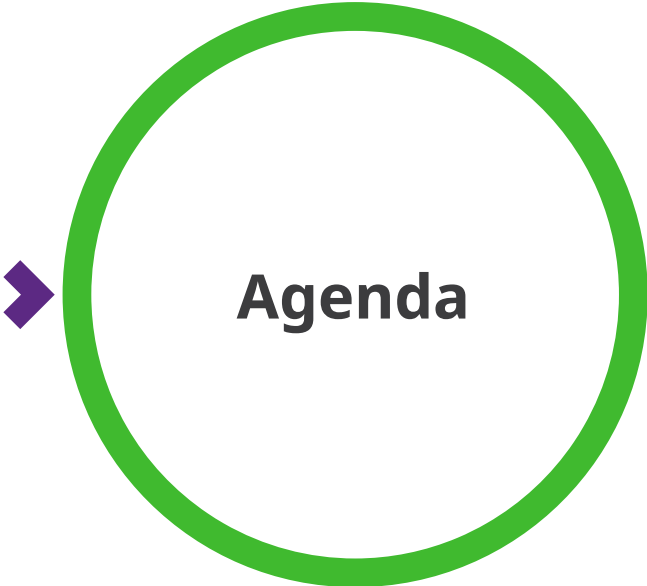
October 23rd - 24th



COLLABORA

Hi, I am Julian

- From Spain
- Multimedia Team at Collabora since 2019
- GStreamer, PipeWire and WirePlumber developer
- julian.bouzas@collabora.com



Agenda

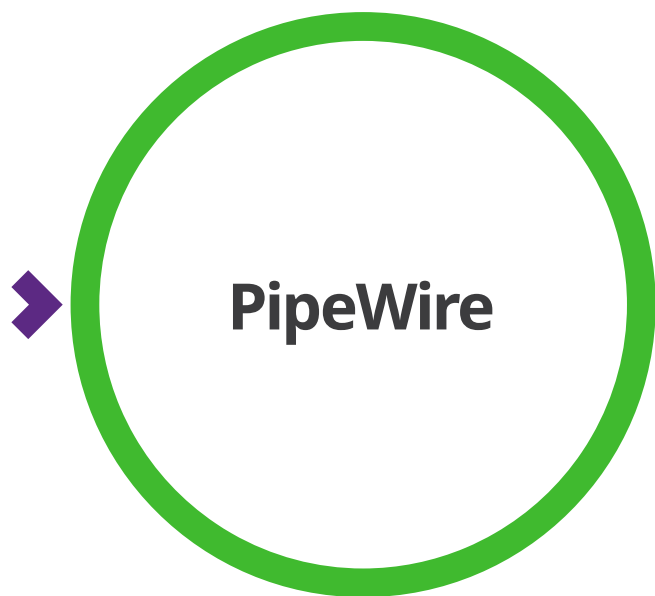
- Quick Introduction
 - PipeWire
 - WirePlumber
- Challenges and Next Steps
 - Permission
 - Performance
 - Transition to Stream Support
 - Future





COLLABORA

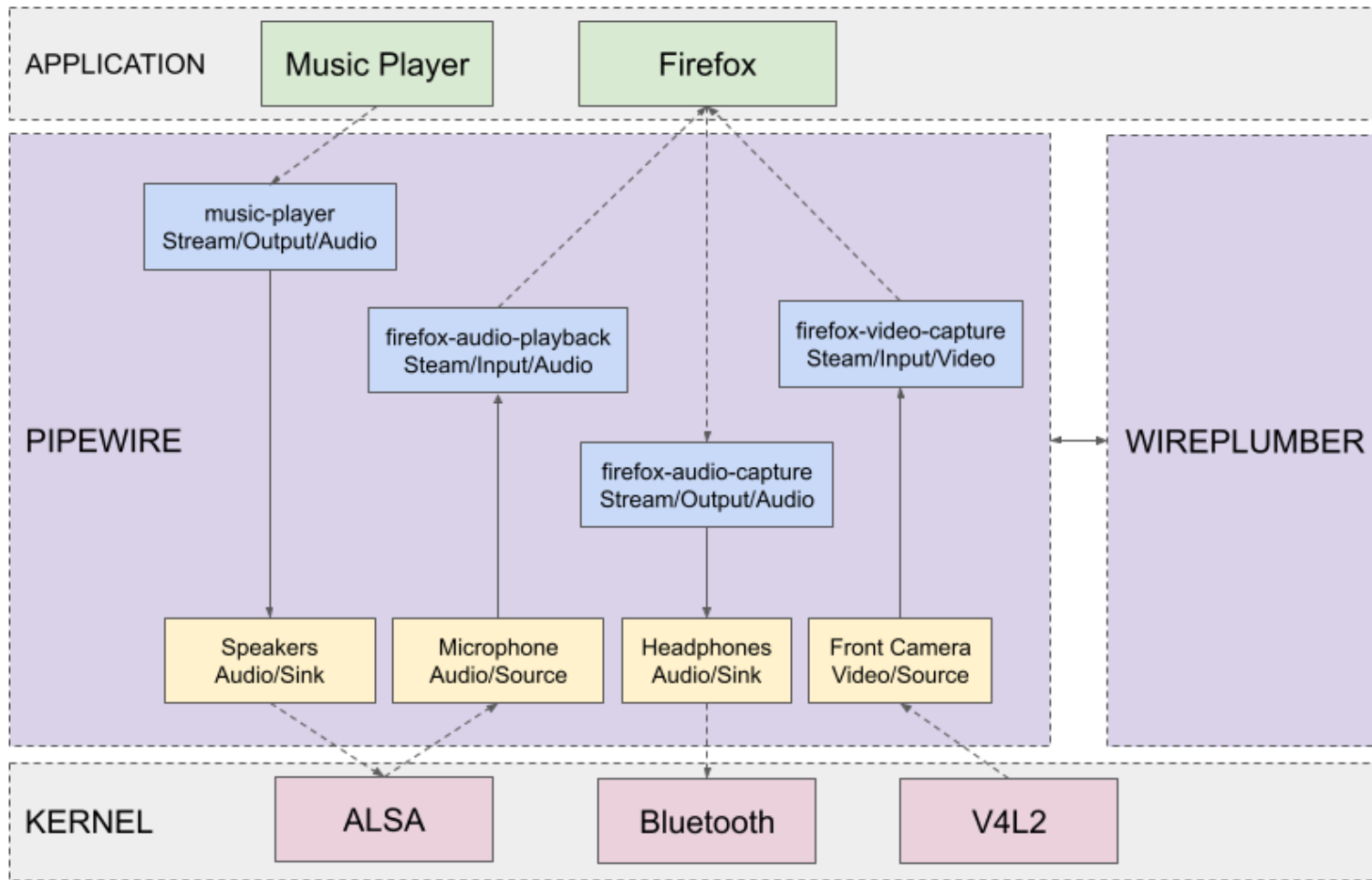
Introduction to PipeWire and WirePlumber



- Next generation multimedia daemon for Linux to handle Video and Audio devices
 - V4L2, Libcamera, ALSA, Bluetooth...
- Replaces PulseAudio
- Graph based design, like GStreamer
 - Nodes (Elements) and Ports (Pads)
- External Session Manager
 - Adaptable for any use-case



- Default Session Manager for PipeWire
 - Without WirePlumber, PipeWire does not do anything
- Daemon that instructs PipeWire how to operate
 - Configures devices: Ports, Profiles, Volumes...
 - Grants permissions to applications
 - Links objects to form the processing graph (Policy)
- Extensible, modular and extremely configurable
 - Lua scripts
 - JSON configuration



Current Status

- Included by default in major Linux distributions
 - Fedora, Ubuntu, Debian, Arch Linux...
- Also used in SteamOS from Valve Corporation for the Steam Deck
- Latest stable versions
 - PipeWire 1.4.8 (Initial release 2017)
 - WirePlumer 0.5.12 (Initial release 2019)
- Still maturing...



COLLABORA

Challenges and Next Steps



COLLABORA

Permission

Permissions in PipeWire

- Can be set on any object for any Application
- Permissions Types:
 - Read: Clients can see and receive data from objects
 - Write: Clients can send data to objects
 - Execute: Clients can configure objects

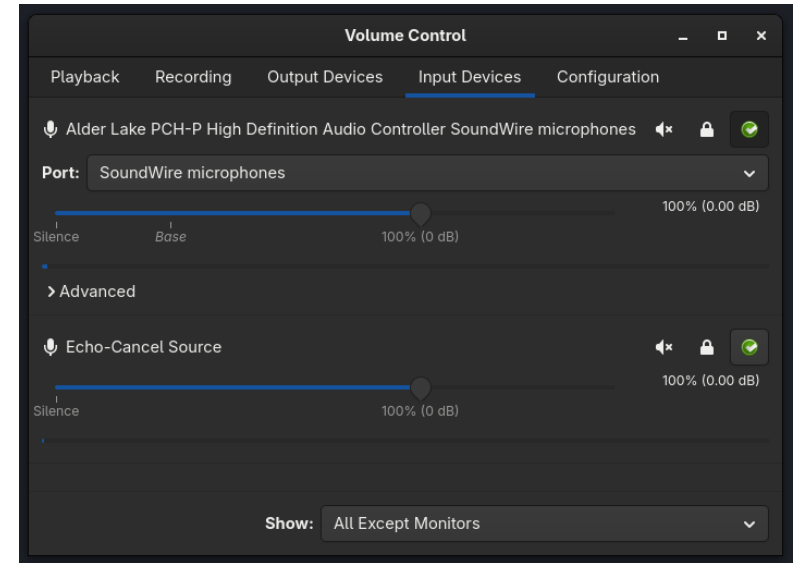
Permissions Configuration

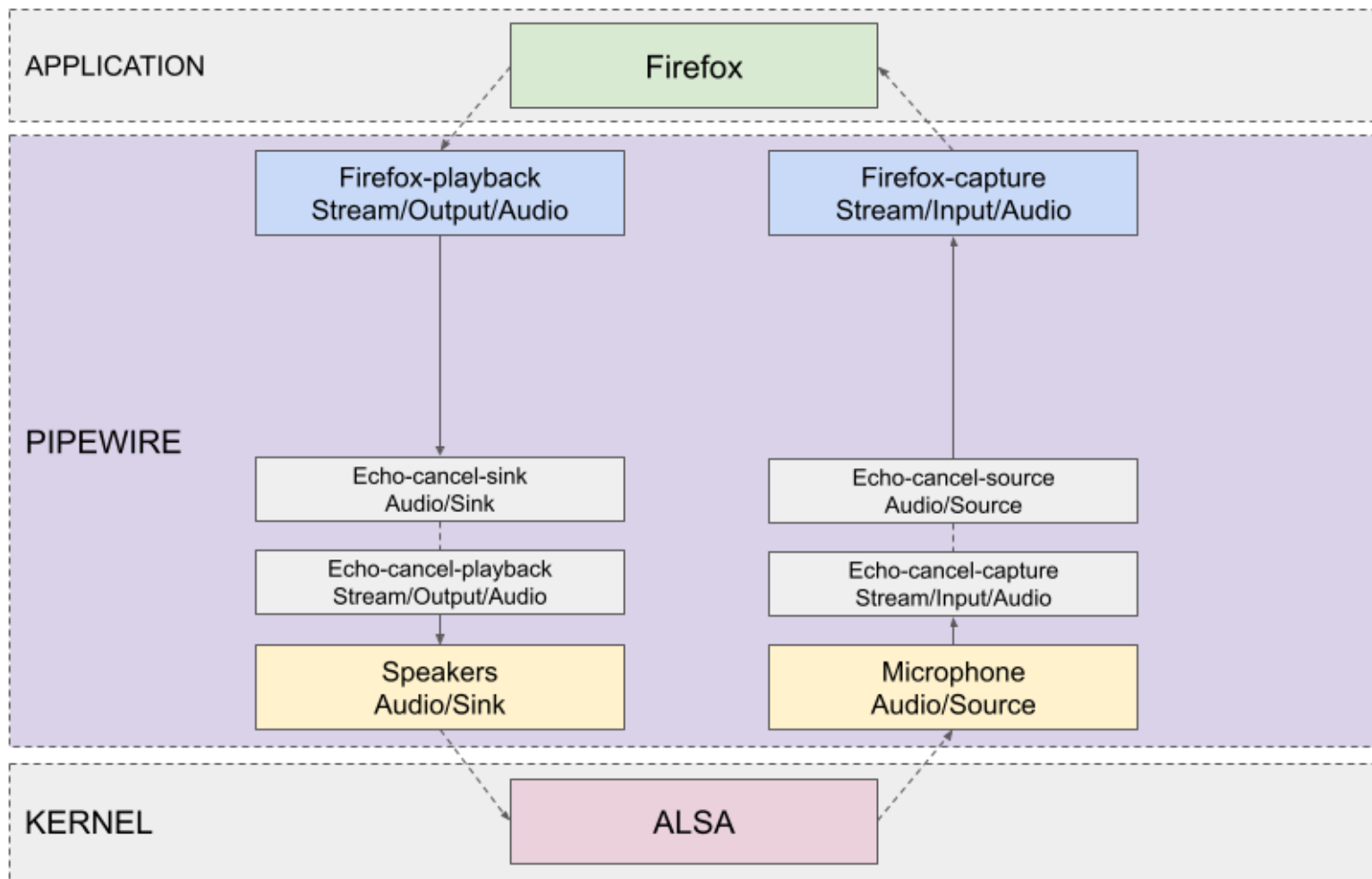
- Basic Configuration
 - Same for all objects per client
- Lua scripts for special cases:
 - Flatpak: Based on portal permission store
 - Snap: Based on client properties
- Can conflict with configuration
- Limited for complex use-cases, such as...

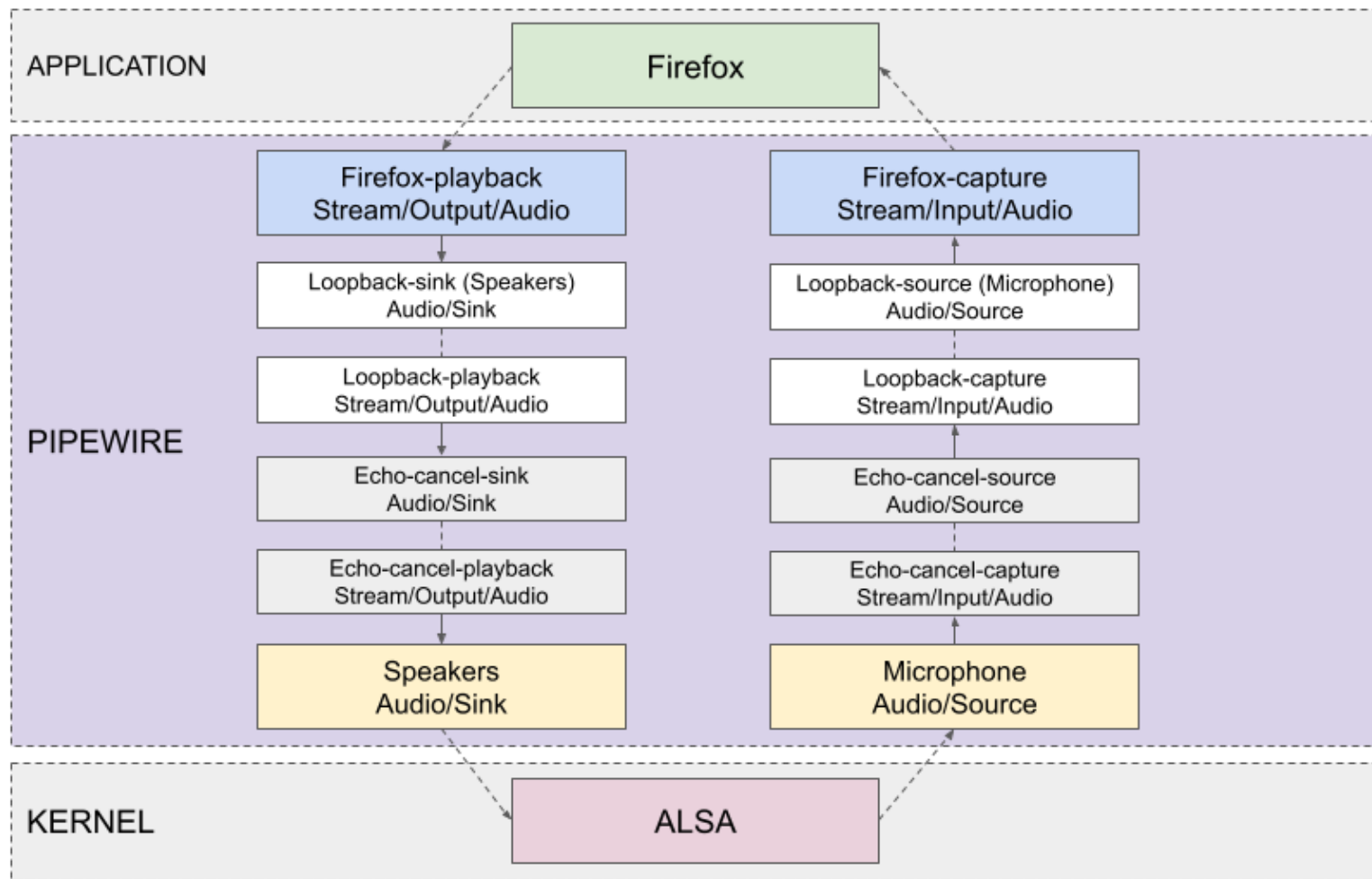
```
1 ## WirePlumber configuration example showing
2 ## how to set read-only permissions on all
3 ## object for Firefox clients
4
5 access.rules = {
6   matches = [
7     {
8       node.name = "firefox"
9     }
10  ]
11  actions = {
12    update-props = {
13      default_permissions = "r"
14    }
15  }
16 }
```

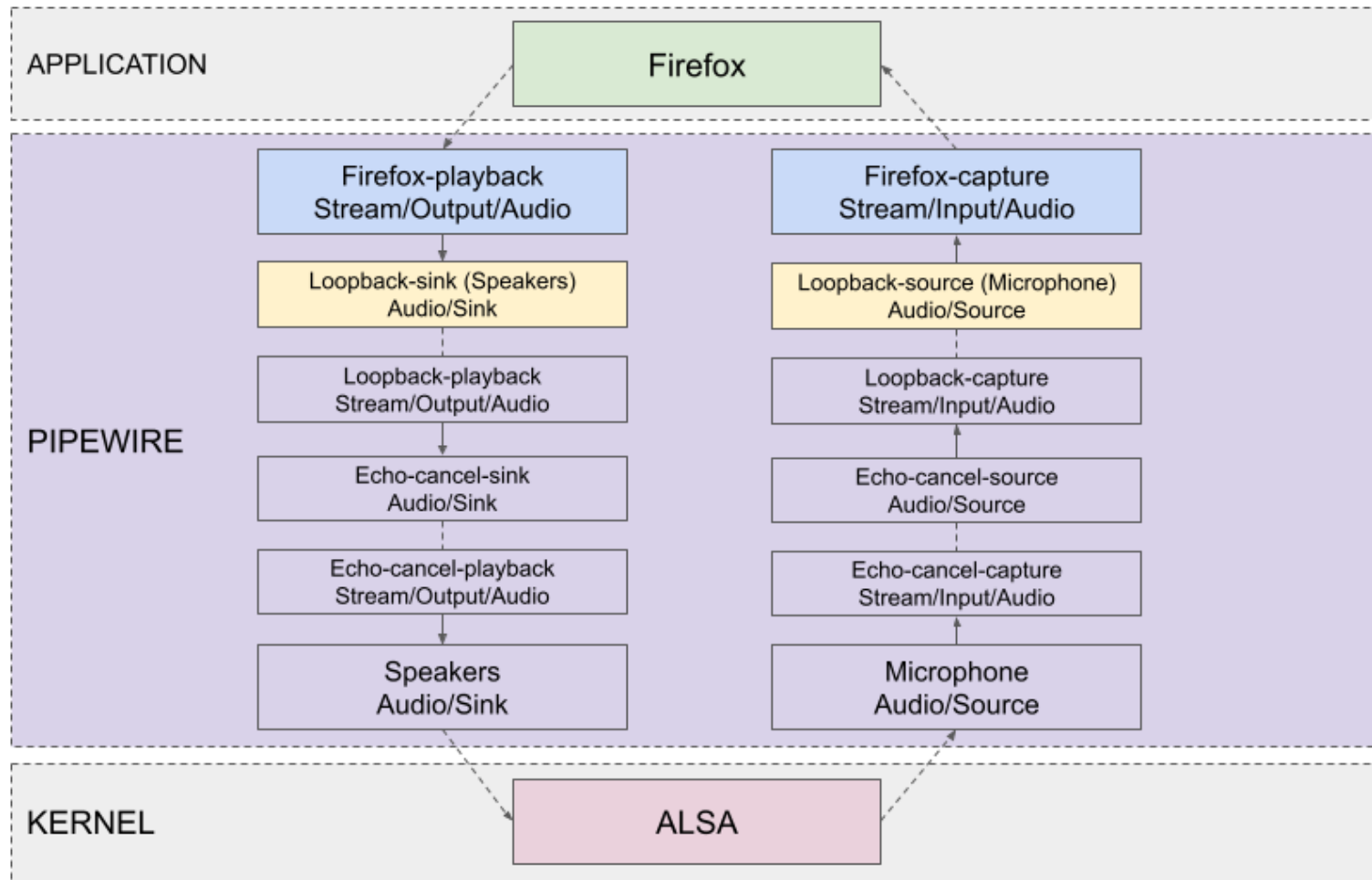
Hide Echo Cancellation

- Echo Cancel module shows up as both Input and Output filters
- Confusing for some users









New Permission Manager API

- Can be attached to clients
 - Only 1 permission manager per client
- Updates permissions on attached clients every time an object of interest appears
- Extend configuration
- Fallback mechanism to avoid conflicts using Event Hooks
 - Configuration → Flatpak → Snap → Default

```
1 ## WirePlumber configuration example showing
2 ## how to remove permissions on echo-cancell
3 ## and loopback nodes for Firefox clients
4
5 access.rules = {
6   matches = [
7     {
8       node.name = "firefox"
9     }
10  ]
11  actions = {
12    update-props = {
13      default_permissions = "rxv"
14    }
15    object-permissions = [
16      {
17        matches = [
18          { node.name = "echo-cancel-sink" }
19          { node.name = "echo-cancel-source" }
20          { node.name = "echo-cancel-playback" }
21          { node.name = "echo-cancel-capture" }
22          { node.name = "loopback-playback" }
23          { node.name = "loopback-capture" }
24        ]
25        actions = {
26          permissions = ""
27        }
28      }
29    ]
30  }
31 }
```



COLLABORA

Performance

Performance limitations

- ~25ms since a stream is added until the target device is found
 - More than the default PipeWire quantum ~20ms
- Short event sounds might not play well
 - Example: libcambera if the PipeWire graph has monitor nodes (pavucontrol)
- Bottlenecks found with perf:
 - Serialization of object properties into Lua tables
 - Event hooks collection

Properties into Lua Tables

- Properties are automatically converted into Lua tables
 - Lots of conversions from C WpProperties into Lua Tables back and forth
- Not needed in many Lua scripts
 - We only want to access 1 or 2 properties in most cases
- Solution: Add new Lua API for WpProperties to avoid conversions

```
1 -- This serializes the properties twice (bad performance)
2 local node_name = node.properties["node.name"]
3 local media_class = node.properties["media.class"]
4
5 -- This serializes the properties once (better performance)
6 local node_properties = node.properties
7 local node_name = node_properties["node.name"]
8 local media_class = node_properties["media.class"]
9
10 -- This never serializes the properties (best performance)
11 -- Using new Lua Properties API
12 local node_properties = node:get_properties ()
13 local node_name = node_properties:get ("node.name")
14 -- Direct access
15 local media_class = node:get_property ("media.class")
```

- ~4ms improvement (~18% faster)

Event Hook Collection

- Event Hooks are small pieces of code called on specific events
 - Example: Link a stream node with the default device when a stream is added
- They can have dependencies on other hooks
- Events collect and reorder hooks every time they are sent
 - Too many unnecessary reordering
- Solution: Reduce overhead by only reordering the hooks on registration

Event Hook Stack (Registration Order)

Send-notification-hook
After: link-stream, unlink-stream
Events: stream-added, stream-removed

Link-stream-hook
After: n/a
Events: stream-added

Unlink-stream-hook
After: n/a
Events: stream-removed

Event Hook Collection

Link-stream-hook
After: n/a
Events: stream-added

Send-notification-hook
After: link-stream, unlink-stream
Events: stream-added, stream-removed

Link-stream-hook
After: n/a
Events: stream-added

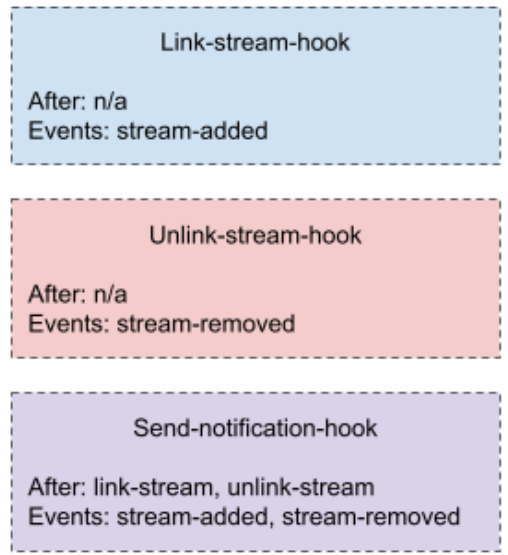
Send-notification-hook
After: link-stream, unlink-stream
Events: stream-added, stream-removed

stream-added

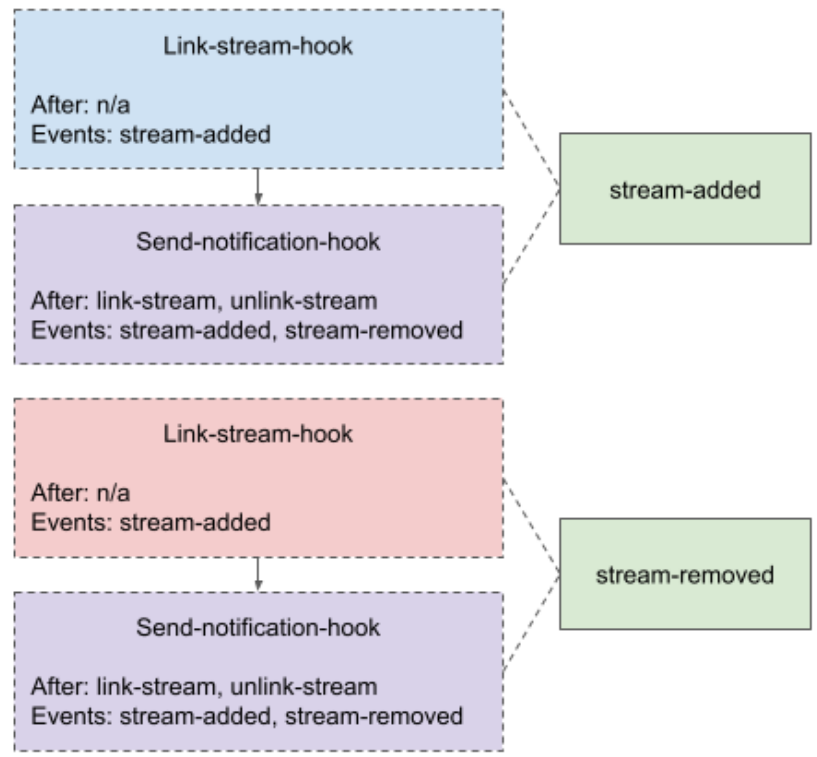
stream-removed



Event Hook Stack (Dependency Order)



Event Hook Collection



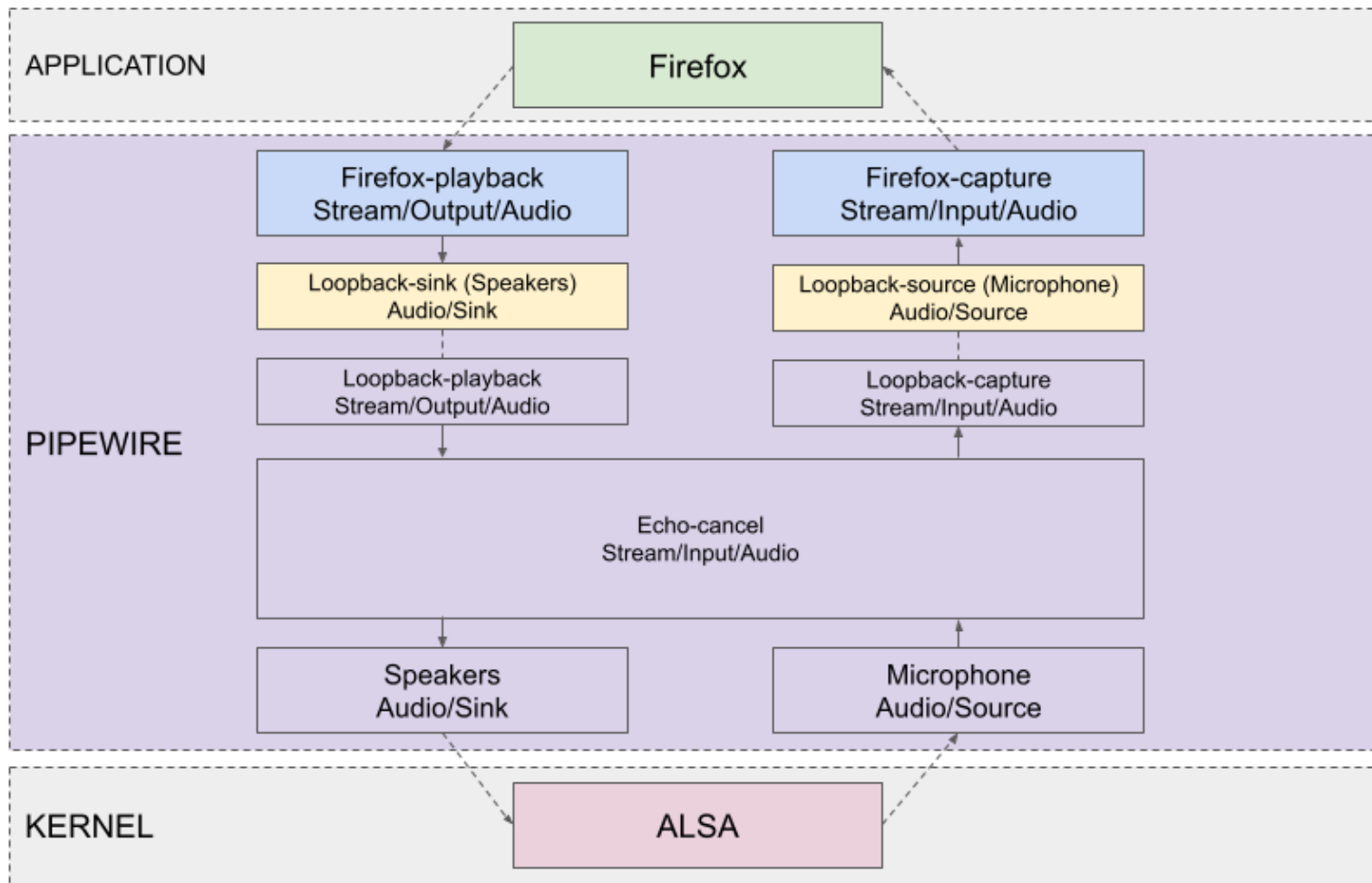
Transition to Steam Support

Current Node Concept

- No concept of Stream
 - Node == Stream
- Nodes can only receive or send data, but not both at the same time
- The whole PipeWire graph is formed by multiple pairs of 2 Nodes linked together

New Node Concept

- Nodes can support multiple streams
 - Node != Stream
- WirePlumber needs to work with streams instead of nodes
- More similar to GStreamer
 - Node → Element
 - Stream → Pad
- Easier to group streams together and treat them as a single unit
- Easier to apply permission on objects of the same type



Future

Future

- Work on the above in the next coming months
- Aiming for a 1.0 release next year (hopefully)
- Small API changes if any
- Considering re-writing some things in Rust after 1.0 release

Contributions

- Hosted on GitLab
 - <https://gitlab.freedesktop.org/pipewire/wireplumber.git>
- Documentation
 - <https://pipewire.pages.freedesktop.org/wireplumber>
- You are welcome to join the community



Thank you!