The PulseAudio Sound Server

lajux.conf.au 2007

Lennart Poettering
lennart@poettering.net

Universität Hamburg, Department Informatik
University of Hamburg, Department of Informatics
Hamburg, Germany

January 17th, 2007
1 Introduction
Contents

1 Introduction

2 What is PulseAudio?
Contents

1. Introduction
2. What is PulseAudio?
3. Usage
Contents

1 Introduction

2 What is PulseAudio?

3 Usage

4 Internals

5 Recipes

6 Outlook
Who Am I?

Student (Computer Science) from Hamburg, Germany
Core Developer of PulseAudio, Avahi and a few other Free Software projects
http://0pointer.de/lennart/
lennart@poettering.de
IRC: mezcalero
Introduction

What is PulseAudio?

Usage

Internals

Recipes

Outlook

Lennart Poettering

The PulseAudio Sound Server
Current State of Linux Audio

Lennart Poettering

The PulseAudio Sound Server
Current State of Linux Audio

It’s a mess! There are just too many widely adopted but competing and incompatible sound systems.
Current State of Linux Audio

It’s a mess! There are just too many widely adopted but competing and incompatible sound systems.

Too many low-level APIs: Open Sound System (OSS), ALSA, EsounD, aRts, Phonon, JACK
Current State of Linux Audio

It’s a mess! There are just too many widely adopted but competing and incompatible sound systems.

Too many low-level APIs: Open Sound System (OSS), ALSA, EsounD, aRts, Phonon, JACK

Incompatible APIs: The OSS API is the *only* API that is widely understood and at least a half-way compatible with other sound systems. All other APIs are incompatible and exclusive to each other, e.g. you cannot run an ALSA application on top of EsounD, etc. (Notable exception: the JACK plugin for ALSA)
It’s a mess! There are just too many widely adopted but competing and incompatible sound systems.

Too many low-level APIs: Open Sound System (OSS), ALSA, EsounD, aRts, Phonon, JACK

Incompatible APIs: The OSS API is the *only* API that is widely understood and at least a half-way compatible with other sound systems. All other APIs are incompatible and exclusive to each other, e.g. you cannot run an ALSA application on top of EsounD, etc. (Notable exception: the JACK plugin for ALSA)

OSS emulation is kludgy: $LD_PRELOAD! (esddsp, artsdsp, aoss)
Current State of Linux Audio II

Sound systems fight a constant battle which one gets access to the sound device.
Current State of Linux Audio II

Sound systems fight a constant battle which one gets access to the sound device.

If Esound wins only Esound clients can play audio.
Current State of Linux Audio II

Sound systems fight a constant battle which one gets access to the sound device.

If EsounD wins only EsounD clients can play audio.

If aRts wins only aRts clients can play audio.
Sound systems fight a constant battle which one gets access to the sound device.

If EsounD wins only EsounD clients can play audio.

If aRts wins only aRts clients can play audio.

If JACK wins only JACK (and some ALSA) clients can play audio.
Sound systems fight a constant battle which one gets access to the sound device.

If EsounD wins only EsounD clients can play audio.
If aRts wins only aRts clients can play audio.
If JACK wins only JACK (and some ALSA) clients can play audio.
If ALSA dmix wins OSS applications lose.
Current State of Linux Audio III

If Esound, aRts or ALSA dmix wins, you cannot have good, dependable, exactly measured latencies.
Current State of Linux Audio III

If EsounD, aRts or ALSA dmix wins, you cannot have good, dependable, exactly measured latencies.

If EsounD wins you cannot have surround sound.
If EsounD, aRts or ALSA dmix wins, you cannot have good, dependable, exactly measured latencies.

If EsounD wins you cannot have surround sound.

If OSS wins you can (usually) play only a single audio stream at a time.
If EsounD, aRts or ALSA dmix wins, you cannot have good, dependable, exactly measured latencies.

If EsounD wins you cannot have surround sound.

If OSS wins you can (usually) play only a single audio stream at a time

If JACK wins you can only have FLOAT32 samples - and playback needs to be started manually in JACK
If EsounD, aRts or ALSA dmix wins, you cannot have good, dependable, exactly measured latencies.

If EsounD wins you cannot have surround sound.

If OSS wins you can (usually) play only a single audio stream at a time.

If JACK wins you can only have FLOAT32 samples - and playback needs to be started manually in JACK.

...
Abstraction APIs exist, but are not widely accepted.
Current State of Linux Audio IV

Abstraction APIs exist, but are not widely accepted.
PortAudio, libao, OpenAL, SDL, CSL
Abstraction APIs exist, but are not widely accepted.

PortAudio, libao, OpenAL, SDL, CSL

Phonon?
Abstraction APIs exist, but are not widely accepted.
PortAudio, libao, OpenAL, SDL, CSL
Phonon?
Is excessive abstraction a good idea?
Current State of Linux Audio IV

Abstraction APIs exist, but are not widely accepted.
PortAudio, libao, OpenAL, SDL, CSL
Phonon?
Is excessive abstraction a good idea?
Amarok on Phonon on GStreamer on ALSA on PulseAudio on ALSA?
Current State of Linux Audio V

Free desktops lack a “Compiz for sound”:

- Allowing different volumes for each running application
Current State of Linux Audio V

Free desktops lack a “Compiz for sound”:

- Allowing different volumes for each running application
- Automatically remembering the volume for each application
Current State of Linux Audio V

Free desktops lack a “Compiz for sound”:

- Allowing different volumes for each running application
- Automatically remembering the volume for each application
- Interrupting music playback when a VoIP call takes place
Current State of Linux Audio V

Free desktops lack a “Compiz for sound”:

- Allowing different volumes for each running application
- Automatically remembering the volume for each application
- Interrupting music playback when a VoIP call takes place
- Automatically increasing volume of the application in foreground, decreasing volume of window in background
Current State of Linux Audio V

Free desktops lack a “Compiz for sound”:

- Allowing different volumes for each running application
- Automatically remembering the volume for each application
- Interrupting music playback when a VoIP call takes place
- Automatically increasing volume of the application in foreground, decreasing volume of window in background
- Automatically remembering the output device of an application
Current State of Linux Audio V

Free desktops lack a “Compiz for sound”:

- Allowing different volumes for each running application
- Automatically remembering the volume for each application
- Interrupting music playback when a VoIP call takes place
- Automatically increasing volume of the application in foreground, decreasing volume of window in background
- Automatically remembering the output device of an application
- Doing “hot” switching of playback streams between devices on USB headset hotplug
- ...
Current State of Linux Audio VI

Professional audio and desktop audio on Linux are currently two completely separate worlds.
Professional audio and desktop audio on Linux are currently two completely separate worlds.

The only common infrastructure seems to be ALSA.
Current State of Linux Audio VI

Professional audio and desktop audio on Linux are currently two completely separate worlds.

The only common infrastructure seems to be ALSA
- but in different configuration (no dmix!)
Current State of Linux Audio VII

However, current free desktops also have unique features:
However, current free desktops also have unique features:
Network transparent sound (EsounD) for thin clients
However, current free desktops also have unique features:
Network transparent sound (EsounD) for thin clients
Wide range of high-level audio applications
However, current free desktops also have unique features:
Network transparent sound (EsounD) for thin clients
Wide range of high-level audio applications
Low-latency kernel
Current State of Linux Audio VII

However, current free desktops also have unique features:

Network transparent sound (EsounD) for thin clients
Wide range of high-level audio applications
Low-latency kernel
Well defined, accepted APIs for pro audio, such as JACK for interconnection or LADSPA for plugins
The current audio mess we have on Linux is not law of nature that cannot be overturned.
The current audio mess we have on Linux is not law of nature that cannot be overturned.

Apple has shown with CoreAudio that a unified sound system for both desktop and professional use is achievable.
The current audio mess we have on Linux is not law of nature that cannot be overturned.

Apple has shown with CoreAudio that a unified sound system for both desktop and professional use is achievable.

Microsoft ships a new userspace sound system with Windows Vista.
What can we do?

We need to acknowledge that OSS is not going to go away
What can we do?

We need to acknowledge that OSS is not going to go away

We need to agree on an API that people should standardize on
What can we do?

We need to acknowledge that OSS is not going to go away
We need to agree on an API that people should standardize on
We should stop abstracting abstracted abstraction layers
We need to find a way to marry all the currently conflicting APIs or at least introduce a (temporary?) compatibility system for them.
We need to find a way to marry all the currently conflicting APIs or at least introduce a (temporary?) compatibility system for them.

We need to come up with a “Compiz for sound”.
Missing Pieces

We need to find a way to marry all the currently conflicting APIs or at least introduce a (temporary?) compatibility system for them. We need to come up with a “Compiz for sound”. We need to retain the unique features we already can offer.
We need to find a way to marry all the currently conflicting APIs or at least introduce a (temporary?) compatibility system for them.

We need to come up with a “Compiz for sound”.

We need to retain the unique features we already can offer.

Play a little catch-up with Apple, Microsoft
What is PulseAudio?
What is PulseAudio?

"The Project Formely Known as Polypaudio"

Modular sound server

Drop-in replacement for Esound; Esound done right

"Application server for sound"

"Compiz for sound"

"Window manager for sound"

Counterpart for the new Windows Vista userspace sound system

The "compatible sound system", which allows running 90% of Linux audio software simultaneously without conflicting.

Developed by Pierre Ossman and yours truly.

LGPL licensed (practically downgraded to GPL on the server side)
What is PulseAudio?

“The Project Formely Known as Polypaudio”
What is PulseAudio?

“The Project Formely Known as Polypaudio”

Modular sound server
What is PulseAudio?

“The Project Formely Known as Polypaudio”

Modular sound server

Drop-in replacement for EsounD; EsounD done right
What is PulseAudio?

“The Project Formely Known as Polypaudio”

Modular sound server

Drop-in replacement for EsounD; EsounD done right

“Application server for sound”
What is PulseAudio?

“The Project Formely Known as Polypaudio”

Modular sound server

Drop-in replacement for EsounD; EsounD done right

“Application server for sound”

“Compiz for sound”
What is PulseAudio?

“The Project Formely Known as Polypaudio”
Modular sound server
Drop-in replacement for Esound; Esound done right
“Application server for sound”
“Compiz for sound”
“Window manager for sound”
What is PulseAudio?

“The Project Formely Known as Polypaudio”

Modular sound server

Drop-in replacement for EsounD; EsounD done right

“Application server for sound”

“Compiz for sound”

“Window manager for sound”

Counterpart for the new Windows Vista userspace sound system
What is PulseAudio?

“The Project Formely Known as Polypaudio”
Modular sound server
Drop-in replacement for Esound; Esound done right
“Application server for sound”
“Compiz for sound”
“Window manager for sound”
Counterpart for the new Windows Vista userspace sound system
The “compatible sound system”, which allows running 90% of Linux audio software simultaneously without conflicting.
What is PulseAudio?

“The Project Formely Known as Polypaudio”
Modular sound server
Drop-in replacement for Esound; Esound done right
“Application server for sound”
“Compiz for sound”
“Window manager for sound”
Counterpart for the new Windows Vista userspace sound system
The “compatible sound system”, which allows running 90% of Linux audio software simultaneously without conflicting.
Developed by Pierre Ossman and yours truly.
What is PulseAudio?

“The Project Formely Known as Polypaudio”
Modular sound server
Drop-in replacement for EsounD; EsounD done right
“Application server for sound”
“Compiz for sound”
“Window manager for sound”
Counterpart for the new Windows Vista userspace sound system
The “compatible sound system”, which allows running 90% of Linux audio software simultaneously without conflicting.
Developed by Pierre Ossman and yours truly.
LGPL licensed (practically downgraded to GPL on the server side)
What is PulseAudio, really?

It’s basically a proxy for your sound device, that receives audio data from your applications, does simple, or more advanced operations on it, and passes it on to the device.
What is PulseAudio, really?

It’s basically a proxy for your sound device, that receives audio data from your applications, does simple, or more advanced operations on it, and passes it on to the device.

It’s also a proxy that receives audio data from your sound device, does simple, or more advanced operations on it, and passes it on to your applications.
What is PulseAudio, really? II

What are those *simple, or more advanced operations*?
What is PulseAudio, really? II

What are those *simple, or more advanced operations*?

- Mixing multiple streams together
What is PulseAudio, really? II

What are those *simple, or more advanced operations*?

- Mixing multiple streams together
- Adjust sample rate or format
What is PulseAudio, really? II

What are those *simple, or more advanced operations*?

- Mixing multiple streams together
- Adjust sample rate or format
- Do volume adjustments
What is PulseAudio, really? II

What are those *simple, or more advanced operations*?

- Mixing multiple streams together
- Adjust sample rate or format
- Do volume adjustments
- Apply other filters, echo cancellation
What is PulseAudio, really? II

What are those *simple, or more advanced operations*?

- Mixing multiple streams together
- Adjust sample rate or format
- Do volume adjustments
- Apply other filters, echo cancellation
- Redirect or copy to another audio device or application, possibly over the network
What is PulseAudio, really? II

What are those *simple, or more advanced operations*?

- Mixing multiple streams together
- Adjust sample rate or format
- Do volume adjustments
- Apply other filters, echo cancellation
- Redirect or copy to another audio device or application, possibly over the network
- Reroute channels (e.g. Stereo to Surround)
- ...
Modular Design

Modular System, currently shipping with 34 modules:
Modular Design

Modular System, currently shipping with 34 modules:
Driver support: OSS, ALSA, Solaris Audio, Win32 Audio
Modular Design

Modular System, currently shipping with 34 modules:

- Driver support: OSS, ALSA, Solaris Audio, Win32 Audio
- Protocol support: Native TCP, EsounD TCP, RTP
Modular Design

Modular System, currently shipping with 34 modules:
Driver support: OSS, ALSA, Solaris Audio, Win32 Audio
Protocol support: Native TCP, Esound TCP, RTP
Toys: LIRC, multimedia keyboard (Linux evdev)
Modular Design

Modular System, currently shipping with 34 modules:
Driver support: OSS, ALSA, Solaris Audio, Win32 Audio
Protocol support: Native TCP, EsounD TCP, RTP
Toys: LIRC, multimedia keyboard (Linux evdev)
Desktop integration: X11 bell, X11 credentials
Modular Design II

Integration: JACK, EsounD
Modular Design II

Integration: JACK, EsounD
Zeroconf - Avahi Rocks!
Modular Design II

Integration: JACK, EsounD

Zeroconf - Avahi Rocks!

Managing: Restore volumes, devices, move stream to other device if device vanishes
Modular Design II

Integration: JACK, EsounD

Zeroconf - Avahi Rocks!

Managing: Restore volumes, devices, move stream to other device if device vanishes

Plug’n’Play: simple device autodetection, HAL-based hotplug
Modular Design II

Integration: JACK, EsounD

Zeroconf - Avahi Rocks!

Managing: Restore volumes, devices, move stream to other device if device vanishes

Plug’n’Play: simple device autodetection, HAL-based hotplug

Synchronize output on multiple sound devices
What is PulseAudio not?

Not a competitor for JACK, GStreamer, Helix, Xine, Phonon!
Not just "Yet another audio API"!
Not a try to push yes another Esound on the people!
What is PulseAudio not?

Not a competitor for JACK, GStreamer, Helix, Xine, Phonon!
What is PulseAudio not?

Not a competitor for JACK, GStreamer, Helix, Xine, Phonon!
Not just “Yet another audio API”!
What is PulseAudio not?

Not a competitor for JACK, GStreamer, Helix, Xine, Phonon!
Not just “Yet another audio API”!
Not a try to push yes another EsounD on the people!
Features

Supports up to 32 channels.
Wide range of sample formats (PCM, μLaw, aLaw)
Realtime scheduling
Network transparent sample cacheing
Current Status

Supersedes EsounD, ALSA dmix in every way
Current Status

Supersedes EsounD, ALSA dmix in every way
Part of most major distributions
Current Status

Supersedes EsounD, ALSA dmix in every way
Part of most major distributions
Wide adoption in thin client environments - not so on normal desktops (yet)
Current Status

Supersedes EsounD, ALSA dmix in every way
Part of most major distributions
Wide adoption in thin client environments - not so on normal desktops (yet)
Commercial support from Cendio
Current Status

Supersedes Esound, ALSA dmix in every way
Part of most major distributions
Wide adoption in thin client environments - not so on normal desktops (yet)
Commercial support from Cendio
Portable: Linux, FreeBSD, Solaris, Native Win32 (no Cygwin)
**Current Status**

Supersedes EsounD, ALSA dmix in every way

Part of most major distributions

Wide adoption in thin client environments - not so on normal desktops (yet)

Commercial support from Cendio

Portable: Linux, FreeBSD, Solaris, Native Win32 (no Cygwin)

Good latency behaviour and exact latency estimations
Current Status

Supersedes EsounD, ALSA dmix in every way
Part of most major distributions
Wide adoption in thin client environments - not so on normal desktops (yet)
Commercial support from Cendio
Portable: Linux, FreeBSD, Solaris, Native Win32 (no Cygwin)
Good latency behaviour and exact latency estimations
Lots of room for improvement
PulseAudio vs. ALSA dmix

ALSA dmix has serious limitations: bad latency behaviour, unstable timing, not portable
PulseAudio vs. ALSA dmix

ALSA dmix has serious limitations: bad latency behaviour, unstable timing, not portable

Why not fix ALSA dmix? - PulseAudio is kind of a “fix” for dmix.
PulseAudio vs. ALSA dmix

ALSA dmix has serious limitations: bad latency behaviour, unstable timing, not portable

Why not fix ALSA dmix? - PulseAudio is kind of a “fix” for dmix.

ALSA dmix is a sound daemon - however far less powerful than PulseAudio, and not as portable. PulseAudio may be started automatically on demand by libasound much the same way as ALSA dmix.
PulseAudio vs. ALSA dmix

ALSA dmix has serious limitations: bad latency behaviour, unstable timing, not portable

Why not fix ALSA dmix? - PulseAudio is kind of a “fix” for dmix.

ALSA dmix is a sound daemon - however far less powerful than PulseAudio, and not as portable. PulseAudio may be started automatically on demand by libasound much the same way as ALSA dmix.

Not recommended to run it that way, though.
PulseAudio vs. ALSA dmix

ALSA dmix has serious limitations: bad latency behaviour, unstable timing, not portable

Why not fix ALSA dmix? - PulseAudio is kind of a “fix” for dmix. ALSA dmix is a sound daemon - however far less powerful than PulseAudio, and not as portable. PulseAudio may be started automatically on demand by libasound much the same way as ALSA dmix.

Not recommended to run it that way, though.

PulseAudio replaces the libasound plugin dmix, plughw, others
PulseAudio vs. EsoundD

There is practically no reason left to use EsoundD instead of PulseAudio
PulseAudio vs. EsoundD

There is practically no reason left to use EsoundD instead of PulseAudio

At least on the desktop - not necessarily on embedded devices, due to lack of FPU.
PulseAudio vs. JACK

There is no “vs.”!

Different objectives - JACK: inter-application communication for pro audio
PulseAudio vs. JACK

There is no “vs.”!

Different objectives - JACK: inter-application communication for pro audio

JACK has some limitations that makes it unusable for desktop use: reliance on FP, the need to “start” it manually, high CPU load. Incompatibility with everything else.
PulseAudio vs. JACK

There is no “vs.”!

Different objectives - JACK: inter-application communication for pro audio

JACK has some limitations that makes it unusable for desktop use: reliance on FP, the need to “start” it manually, high CPU load. Incompatibility with everything else.

Future integration with PulseAudio?
PulseAudio vs. aRts
There is no “vs.”!
PulseAudio vs. aRts

There is no “vs.”!
aRts is officially dead.
Usage
Required Dependencies

liboil: Optimized inner loops
libsamplerate: Sample rate adjustments
libsndfile: Loading sound files
Optional Dependencies

ALSA: Hardware access

GLIB: Support for integration into the GLib event loop - no hard dependency

Avahi: for Zeroconf support

JACK: for JACK integration

X11: Hook into bell event, store authentication credentials

liboil: Asynchronous name resolution

TCPWRAP: access control

LIRC: remote control
Compatibility

Plugin for libasound: most ALSA applications can access PulseAudio like a local sound card

$LD_PRELOAD based OSS compatibility

Implementation of the EsounD protocol

Integration with JACK

Plugin for libao

Plugin for GStreamer (With nice tricks!)
Compatibility II

Plugin for XMMS, Audacious

Driver for MPlayer, Xine

Driver for MPD
Compatibility II

Plugin for XMMS, Audacious
Driver for MPlayer, Xine
Driver for MPD
> 90 % of all Linux audio applications should run on it.

http://pulseaudio.org/wiki/PerfectSetup
Introduction
What is PulseAudio?
Usage
Internals
Recipes
Outlook

Screenshot

Lennart Poettering
The PulseAudio Sound Server
GStreamer Plugin

- Wraps playback, capturing, mixer, device enumeration
- Extracts song metadata from pipeline, uses it to name the stream in the PulseAudio server - independent from the application.
- Not yet part with upstream GStreamer, but will be
XMMS Plugin

- Wraps playback, mixer
- Uses song name to name stream in PulseAudio
GUI: Volume Control
GUI: Volume Control II

- Shows streams, sinks, sources
- Allows volume changing for each channel separately
- Shows song name for each stream (that’s why sliders are horizontal!)
- Allows to move stream from one sink to another without interrupting playback
GUI: Volume Meter
Introduction
What is PulseAudio?
Usage
Internals
Recipes
Outlook

GUI: Manager

Lennart Poettering
The PulseAudio Sound Server
GUI: Manager II

- Shows internals of a PulseAudio server
- Not for the regular user
Introduction

What is PulseAudio?

Usage

Internals

Recipes

Outlook

GUI: Device Chooser

PulseAudio Device Chooser

Quick access to the PulseAudio sound server

Notifications

- Show notifications for discovered servers
- Show notifications for discovered sinks
- Show notifications for discovered sources
- Don't show notifications during startup

Startup

- Start applet on session login

Lennart Poettering

The PulseAudio Sound Server
GUI: Device Chooser II

- Sits in the notification area
- Allows easy changing of the input/output device/server
- Avahi support
- Easy access to the utility programs
GUI: Preferences

PulseAudio Preferences

- Network Access
  - Enable network access to local sound devices
  - Allow other machines in LAN to browse for local sound devices
  - Don't require authentication

- Multicast/RTP
  - Enable Multicast/RTP receiver
  - Enable Multicast/RTP sender
    - Send audio from local microphone
    - Send audio from local speakers
    - Create separate audio device for Multicast/RTP
    - Loopback audio to local speakers

Close
GUI: Preferences II

- Easy access to selected configuration options
- Communicates over GConf with PulseAudio
- Instant apply
- Requires `module-gconf` loaded into the server
- Current options: Remote access, Authentication, Zeroconf, RTP multicast receive, send
Internals
Sink - A clocked output device PA writes data to (e.g. ALSA sound card)

Source - A clocked input device PA reads data from (e.g. ALSA sound card)

Monitor Source - Implicitly attached to every sink for monitoring what is currently being played
Nomenclatura

**Sink** - A clocked output device PA writes data to (e.g. ALSA sound card)

**Source** - A clocked input device PA reads data from (e.g. ALSA sound card)

**Monitor Source** - Implicitly attached to every sink for monitoring what is currently being played

**Sink Input** - An output stream whose data PA sends on to a sink; not clocked! (e.g. a client application such as Rhythmbox)

**Source Output** - An input stream where PA sends to data from a source; not clocked! (e.g. a client application such as Ekiga)
Nomenclatura

*Sink* - A clocked output device PA writes data to (e.g. ALSA sound card)

*Source* - A clocked input device PA reads data from (e.g. ALSA sound card)

*Monitor Source* - Implicitly attached to every sink for monitoring what is currently being played

*Sink Input* - An output stream whose data PA sends on to a sink; not clocked! (e.g. a client application such as Rhythmbox)

*Source Output* - An input stream where PA sends to data from a source; not clocked! (e.g. a client application such as Ekiga)

*PulseAudio does not know the notion of “pipelines”!*
Module - A shared library code blob which can be loaded into the daemon at any time which can register any number of sinks, sources, inputs or outputs.

Client - A local or networked client which can allocate any number of inputs or outputs.
**Module** - A shared library code blob which can be loaded into the daemon at any time which can register any number of sinks, sources, inputs or outputs.

**Client** - A local or networked client which can allocate any number of inputs or outputs.

Sinks and sources can be identified by a short string such as `dsp1` or `dsp1.monitor`. Name is generated automatically from the underlying device name - or may be configured manually.
Internals

Zero-Copy memory management
Internals

Zero-Copy memory management

Shared-Memory data transfer
Internals

Zero-Copy memory management

Shared-Memory data transfer

Powerful playback model
Internals

Zero-Copy memory management
Shared-Memory data transfer
Powerful playback model
Automatic underrun handling
Internals

Zero-Copy memory management
Shared-Memory data transfer
Powerful playback model
Automatic underrun handling
Accurate latency estimation for every element in the pipeline
Internals

Zero-Copy memory management
Shared-Memory data transfer
Powerful playback model
Automatic underrun handling
Accurate latency estimation for every element in the pipeline
Client-side latency interpolation
Internals

Zero-Copy memory management
Shared-Memory data transfer
Powerful playback model
Automatic underrun handling
Accurate latency estimation for every element in the pipeline
Client-side latency interpolation
Embeddable core
Internals

Zero-Copy memory management
Shared-Memory data transfer
Powerful playback model
Automatic underrun handling
Accurate latency estimation for every element in the pipeline
Client-side latency interpolation
Embeddable core
Supports multiple sinks/sources in a single daemon
Internals

Zero-Copy memory management
Shared-Memory data transfer
Powerful playback model
Automatic underrun handling
Accurate latency estimation for every element in the pipeline
Client-side latency interpolation
Embeddable core
Supports multiple sinks/sources in a single daemon
Asynchronous Client API
Internals

Zero-Copy memory management
Shared-Memory data transfer
Powerful playback model
Automatic underrun handling
Accurate latency estimation for every element in the pipeline
Client-side latency interpolation
Embeddable core
Supports multiple sinks/sources in a single daemon
Asynchronous Client API
Fully configurable during runtime
Zero-Copy Memory Management

We never copy memory around if not really necessary
Instead of queueing samples, we queue pointers to reference counted memory data blocks
Location of those memory data blocks is flexible: dynamic memory, SHM from other process, DMA buffer of sound card, mapped file
Advantages: low memory usage, low-latency
Shared-Memory data transfer

Audio data between local clients and server is transferred via shared memory.

The server and each client allocates one shared memory segment and allocates its audio buffers from it. Indexes into this segment are passed between clients and between client and server.

RW access only to local segment, RO access to other process’ segments.
Playback Model

Playback buffer is a linked list of pointers to audio data blocks and their indexes

Monotonically advancing read index

Freely seekable write index: SEEK_RELATIVE, SEEK_ABSOLUTE, SEEK_RELATIVE_ON_READ, SEEK_RELATIVE_END

If data is written “left” of the current read pointer, it is immediately dropped

If due to seeking the buffer contains “holes” silence is automatically inserted.
Playback Model II

Multiple streams can be synced together, read indexes will never deviate if enabled.

If buffer fill level becomes 0, an UNDERRUN event is sent to the client. If buffer fill level becomes smaller than a specified lower watermark a REQUEST event is sent to the client. If a specified maximum buffer length is reached an OVERRUN event is sent to the client.

Advantages: large buffers with quick reaction possible - useful especially over the network; easy to write RTP receivers; low memory usage
Playback Model III

Index = 0
Pointer to Memory Block
Index into Memory Block
Length = 1024
Next
Previous

Index = 1024
Pointer to Memory Block
Index into Memory Block
Length = 512
Next
Previous

Index = 2048
Pointer to Memory Block
Index into Memory Block
Length = 1024
Next
Previous

Index = 3072
Pointer to Memory Block
Index into Memory Block
Length = 1024
Previous

Read Index = 768
SEEKbsolute SEEK_RELATIVE_ON_READ

Buffer Length = 2560
512 Missing!

Write Index = 3328
SEEK_RELATIVE SEEK_RELATIVE_END

Total Length = 4096
Memory Allocated = 3584
Handling Underruns

Two different modes supported:

- Playback stops on underrun, starts again if “prebuf” level is reached.
Handling Underruns

Two different modes supported:

- Playback stops on underrun, starts again if “prebuf” level is reached.
- Playback never stops, read index advances monotonically, and drops enough samples so that the time the underrun lasted is compensated.
Emulating OSS

Difficult, because kernel interfaces cannot easily be emulated from userspace
Emulating OSS

Difficult, because kernel interfaces cannot easily be emulated from userspace

$LD_PRELOAD based tools: esddsp, aoss, padsp.
Emulating OSS

Difficult, because kernel interfaces cannot easily be emulated from userspace

$LD\_PRELOAD$ based tools: esddsp, aoss, padsp.

Doesn’t support: SUID binaries, static binaries, tools which unset $LD\_PRELOAD$, dlopen()

It’s slow and it’s ugly.

DMA practically impossible.
Emulating OSS

Difficult, because kernel interfaces cannot easily be emulated from userspace

$LD\_PRELOAD$ based tools: esddsp, aoss, padsp.

Doesn’t support: SUID binaries, static binaries, tools which unset $LD\_PRELOAD$, dlopen()

It’s slow and it’s ugly.

DMA practically impossible.

Quake2 - What about DMA?
Emulating OSS II

Possible solution: FUSD - emulating character devices from userspace.
Emulating OSS II

Possible solution: FUSD - emulating character devices from userspace.

DMA still a problem?
Emulating aRts

Difficult, because aRts is more a synthesizer than a sound server.
Emulating aRts

Difficult, because aRts is more a synthesizer than a sound server.
Worth it? Does anyone still use it?
Emulating aRts

Difficult, because aRts is more a synthesizer than a sound server.

Worth it? Does anyone still use it?

Probably more applications around that run exclusively on aRts than run exclusively on EsounD.
Authentication

Local: UNIX user ids
Authentication

Local: UNIX user ids

Remote: Cookie, IP ACL, X11 root window
Authentication

Local: UNIX user ids
Remote: Cookie, IP ACL, X11 root window
No encryption - no challenge response
Recipes
PulseAudio may be configured during startup and runtime with a simple imperative command line language. (Not Turing complete).
PulseAudio may be configured during startup and runtime with a simple imperative command line language. (Not Turing complete).

CLI script `.pulse/default.pa` is read on startup.
CLI - Command Line Language

PulseAudio may be configured during startup and runtime with a simple imperative command line language. (Not Turing complete).

CLI script  ./pulse/default.pa is read on startup.
Use pacmd to enter configuration commands during runtime.
PulseAudio may be configured during startup and runtime with a simple imperative command line language. (Not Turing complete).

CLI script 

```
/.pulse/default.pa
```

is read on startup.

Use `pacmd` to enter configuration commands during runtime.

Interesting Commands:

```
load-module module-oss device="/dev/dsp"
sink_name=output source_name=input channels=1
```
PulseAudio may be configured during startup and runtime with a simple imperative command line language. (Not Turing complete).

CLI script `.pulse/default.pa` is read on startup.

Use `pacmd` to enter configuration commands during runtime.

Interesting Commands:

```
load-module module-oss device="/dev/dsp"
sink_name=output source_name=input channels=1
load-sample x11-bell /usr/share/sounds/notify.wav
```
Recipe: RTP Multicast “Radio” Receiver
Recipe: RTP Multicast “Radio” Receiver

load-module module-rtp-recv
Recipe: RTP Multicast “Radio” Sender
Recipe: RTP Multicast “Radio” Sender

```makefile
load-module module-null-sink sink_name=rtp
```
Recipe: RTP Multicast "Radio" Sender

load-module module-null-sink sink_name=rtp
load-module module-rtp-send source=rtp.monitor
Recipe: RTP Multicast “Radio” Sender

load-module module-null-sink sink_name=rtp
load-module module-rtp-send source=rtp.monitor
set-default-sink rtp
Recipe: Output audio on two soundcards simultaneously

```
load-module module-oss-mmap device="/dev/dsp"
sink
name=output0
load-module module-oss-mmap device="/dev/dsp1"
sink
name=output1
load-module module-combine sink
name=combined
master=output0 slaves=output1
set-sink-default combined
```
Recipe: Output audio on two soundcards simultaneously

```bash
load-module module-oss-mmap device="/dev/dsp"
sink_name=output0
```
Recipe: Output audio on two soundcards simultaneously

load-module module-oss-mmap device="/dev/dsp"
sink_name=output0

load-module module-oss-mmap device="/dev/dsp1"
sink_name=output1
Recipe: Output audio on two soundcards simultaneously

```bash
load-module module-oss-mmap device="/dev/dsp"
sink_name=output0

load-module module-oss-mmap device="/dev/dsp1"
sink_name=output1

load-module module-combine sink_name=combined
master=output0 slaves=output1
```
Recipe: Output audio on two soundcards simultaneously

load-module module-oss-mmap device="/dev/dsp"
sink_name=output0

load-module module-oss-mmap device="/dev/dsp1"
sink_name=output1

load-module module-combine sink_name=combined
master=output0 slaves=output1

set-sink-default combined
Recipe: Combine two Stereo devices into a virtual 4-channel Surround device
Recipe: Combine two Stereo devices into a virtual 4-channel Surround device

load-module module-oss-mmap device="/dev/dsp"
sink_name=output0 channel_map=left,right channels=2
Recipe: Combine two Stereo devices into a virtual 4-channel Surround device

load-module module-oss-mmap device="/dev/dsp"
sink_name=output0 channel_map=left,right channels=2
load-module module-oss-mmap device="/dev/dsp1"
sink_name=output1 channel_map=rear-left,rear-right channels=2
load-module module-combine sink
name=combined
master=output0 slaves=output1
channel_map=left,right,rear-left,rear-right
channels=4
set-sink-default combined
Recipe: Combine two Stereo devices into a virtual 4-channel Surround device

```bash
load-module module-oss-mmap device="/dev/dsp"
sink_name=output0 channel_map=left,right channels=2
load-module module-oss-mmap device="/dev/dsp1"
sink_name=output1 channel_map=rear-left,rear-right channels=2
load-module module-combine sink_name=combined
master=output0 slaves=output1
channel_map=left,right,rear-left,rear-right channels=4
```

Lennart Poettering
The PulseAudio Sound Server
Recipe: Combine two Stereo devices into a virtual 4-channel Surround device

load-module module-oss-mmap device="/dev/dsp"
sink_name=output0 channel_map=left,right channels=2
load-module module-oss-mmap device="/dev/dsp1"
sink_name=output1 channel_map=rear-left,rear-right channels=2
load-module module-combine sink_name=combined
master=output0 slaves=output1
channel_map=left,right,rear-left,rear-right channels=4
set-sink-default combined
Outlook
Current Work

Move to a more threaded design
Current Work

Move to a more threaded design
Move to lock-free data structures
Current Work

Move to a more threaded design
Move to lock-free data structures
Best possible OSS compatibility
Current Work

Move to a more threaded design
Move to lock-free data structures
Best possible OSS compatibility
New abstracted sound API?
Future Work

Better integration with JACK
Future Work

Better integration with JACK
RTP Timing synchronisation
Future Work

Better integration with JACK
RTP Timing synchronisation
Compatibility with more sound APIs: PortAudio, SDL.
Future Work

Better integration with JACK
RTP Timing synchronisation
Compatibility with more sound APIs: PortAudio, SDL.
CODECs
Future Work

Better integration with JACK
RTP Timing synchronisation
Compatibility with more sound APIs: PortAudio, SDL.
CODECs
Better GUI tools
Future Work

Better integration with JACK
RTP Timing synchronisation
Compatibility with more sound APIs: PortAudio, SDL.
CODECs
Better GUI tools

http://pulseaudio.org/browser/trunk/todo
That’s all, folks.
That’s all, folks.

Any questions?
PulseAudio

http://pulseaudio.org/
https://tango.0pointer.de/mailman/listinfo/pulseaudio-discuss

#pulseaudio on irc.freenode.org

PulseAudio