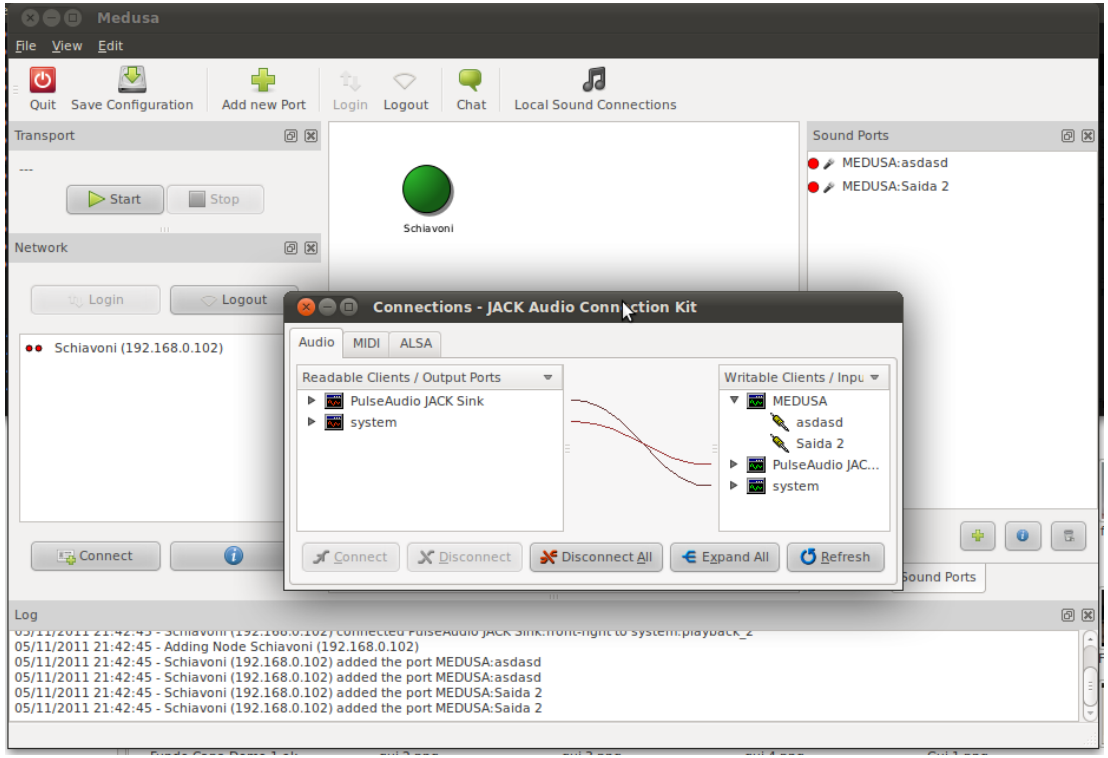
# Network distribution in music applications with Medusa

Flávio Luiz SCHIAVONI  Marcelo QUEIROZ
Computer Science Department
University of São Paulo
São Paulo
Brazil,
{fls, mqz}@ime.usp.br

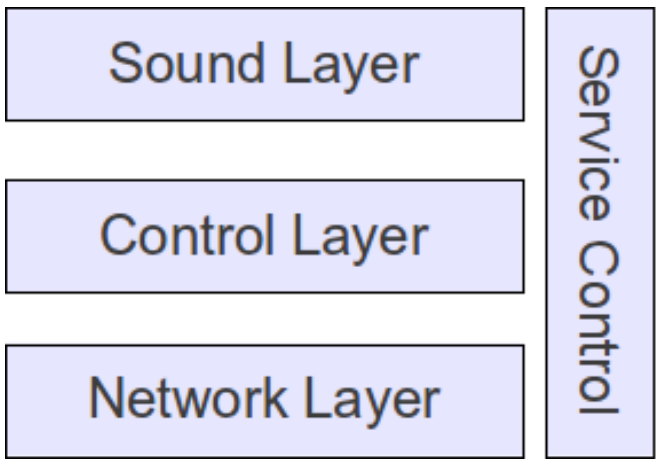
April 12, 2012

# What is Medusa?




- Distributed Audio Environment
- Focused in Usability

# Medusa Architecture




Service Control allows transparent connections using messages to publish / query networked resources.

# What is Medusa?

First Version:

- Jack as Audio API
- SCTP as network transport protocol
- Control Service to publish / query resources
- Transparency of resources and localization
- Qt GUI
- Monolithical development

# Networked Music

Scenarios:

- Recording
- Rehearsal
- Distributed DSP
- Spatialization
- **Performance**

# Networked Music

Scenarios:

- Recording
- Rehearsal
- Distributed DSP
- Spatialization
- **Performance**

**Hypothesis 1:** End users can't deal with audio infrastructure lying below the application they are running.

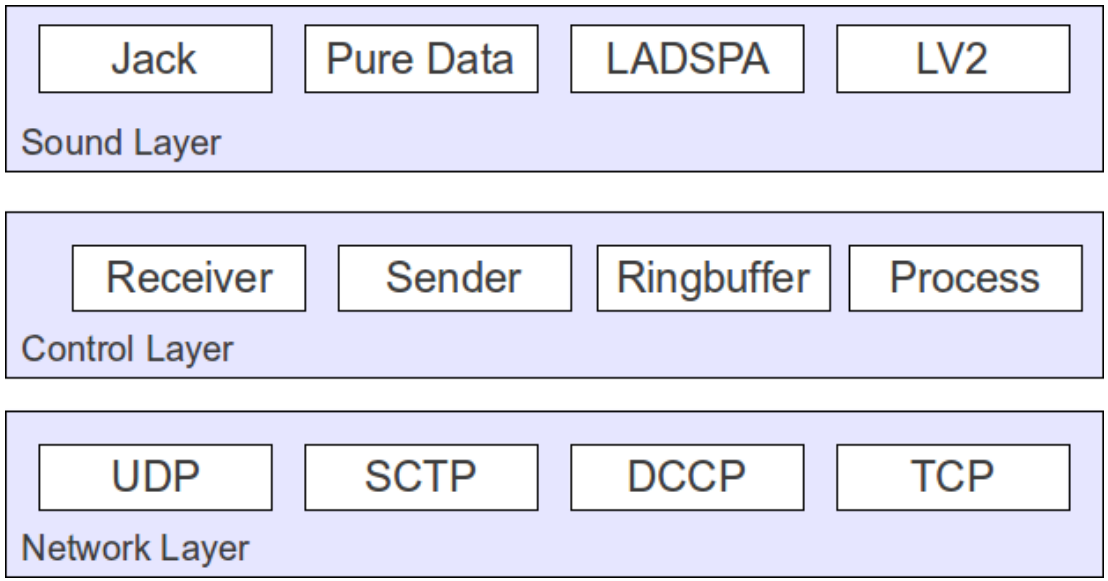
# Networked Music

Scenarios:

- Recording
- Rehearsal
- Distributed DSP
- Spatialization
- **Performance**

**Hypothesis 1:** End users can't deal with audio infrastructure lying below the application they are running.
**Hypothesis 2:** Different scenarios can need different approachs.

# Medusa Architecture




Audio API, processing and network transport protocols alternatives to different scenarios.

# Medusa Architecture

UDP : User Datagram Protocol is the classical unreliable (but faster) transport protocol.

TCP : Transmission Control Protocol is a reliable transport protocol, which ensures absence of packet losses.

SCTP : Stream Control Transmission Protocol is a connection-oriented transport protocol that provides a reliable full-duplex association. This protocol was not originally meant as a replacement for TCP, but was developed for carrying voice over IP (VoIP).

DCCP : Datagram Congestion Control Protocol is a transport protocol that combines TCP-friendly congestion control with unreliable datagram semantics for applications that transfer fairly large amounts of data.

# Medusa Architecture


Figure: LADSPA implementation

# Medusa Architecture

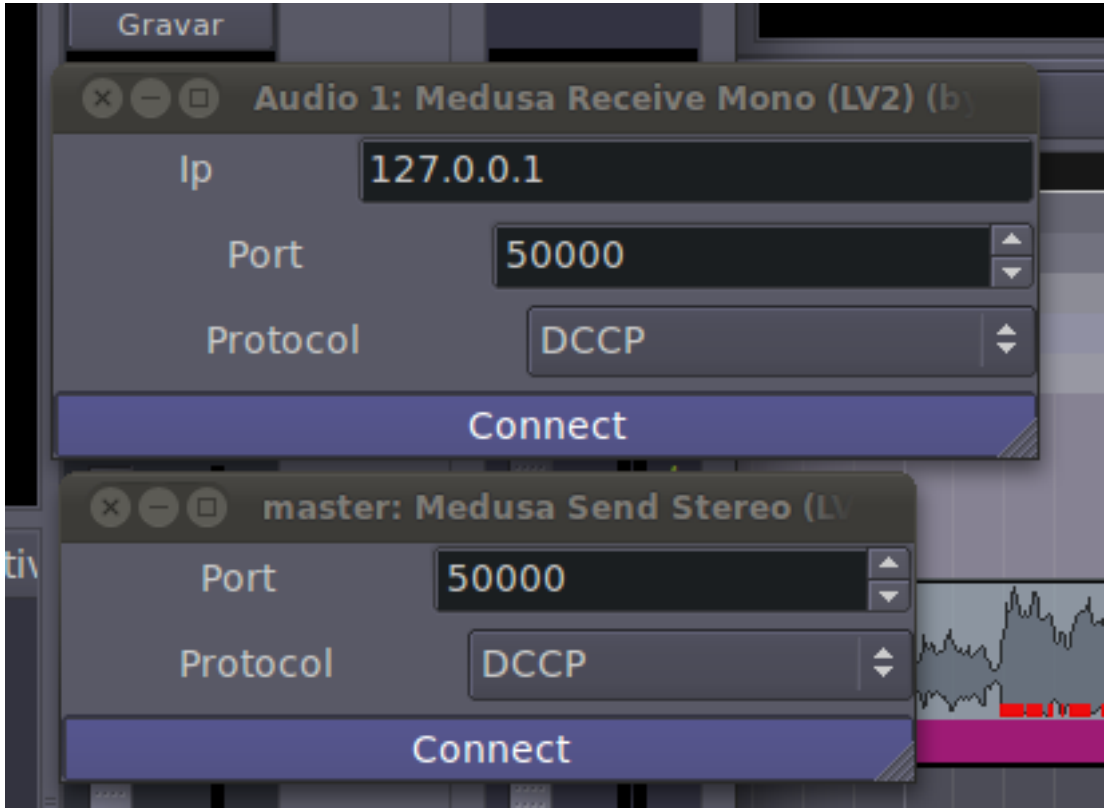



Figure: LV2 implementation

# Medusa Architecture

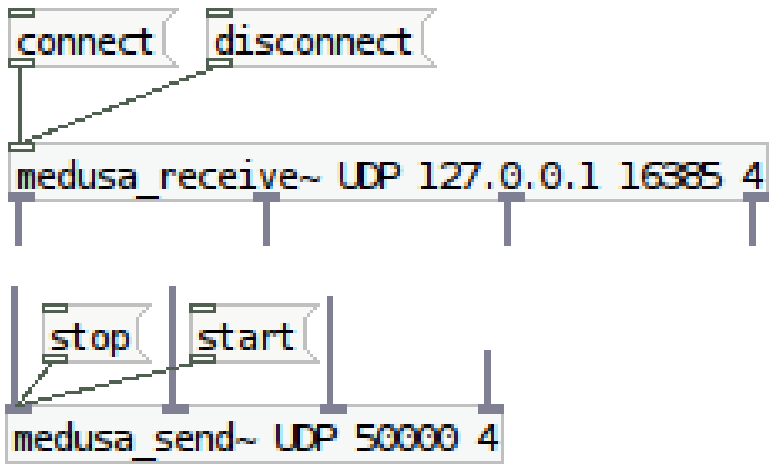



Figure: Pure Data implementation

# Medusa Architecture

Process:

- CELT
- Change sample rate
- Change bit depth
- Adjust worldclock drift

Can be integrated with Service control

# Implementation

- Control layer and network layer as a library
- The library provides a easy way to create senders and receivers
- Each audio API implementation creates senders ans receivers
- Each audio API can has it own user interface


Easy code maintenance and more connections alternatives.

# Results and Future works

- Extend the possibility of networked audio with Jack and UDP
- Try out new ways to do old things
- Better code with layered development

Future works

- Implement the service control as a separated server
- Integrate service control to implemented code
- MIDI streams
- Investigate other sounds API

# Acknowledgements

Thanks to uncountable developers of LADSPA, LV2 and Pure Data externals and their amazing open source code. Without their anonymous help this project would not have been possible.
Thanks also go to André Jucovsky Bianchi, Beraldo Leal, Santiago Davila, Antônio Goulart, Giuliano Obici, Danilo Leite and the Computer Music Group of IME/USP for their interest, feedback and support.
This work has been supported by the funding agencies CNPq (grant 141730/2010-2) and FAPESP - São Paulo Research Foundation (grant 2008/08623-8).

# Thanks!

http://sourceforge.net/projects/medusa-audionet/
fls@ime.usp.br
Questions?

Thanks!