FaustLive

Just-In-Time Faust Compiler... and much more

Sarah Denoux, Stephane Letz, Yann Orlarey, Dominique Fober
{sdenoux, letz, orlarey, fober} @grame.fr

LAC
03 May 2014

Table of contents

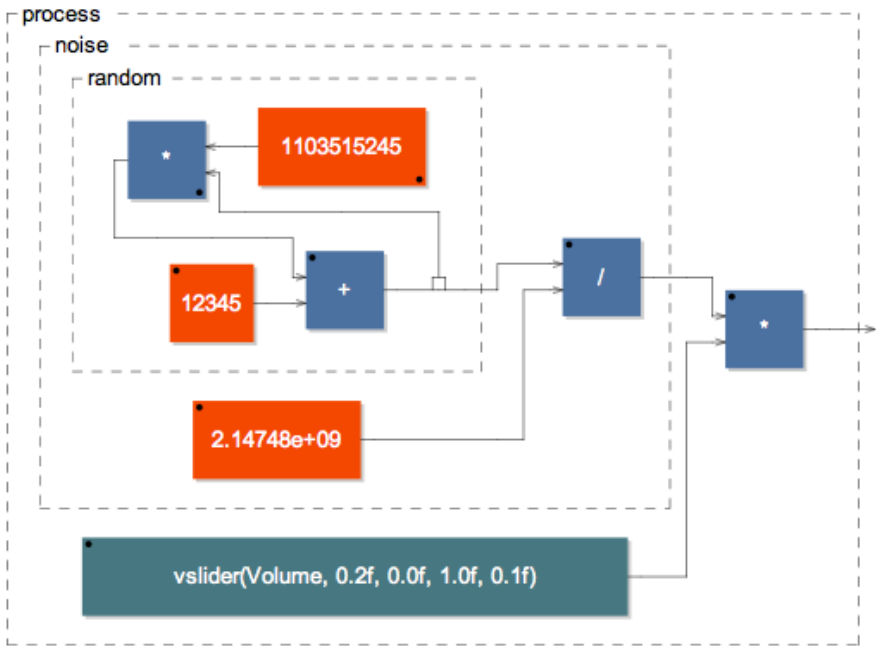
Faust Language

Faust is a *Domain-Specific Language* for real-time signal processing and synthesis. A Faust program denotes a *signal processor* :

- A *signal processor* is a mathematical function that maps a group of *n* input *signals* to a group of *m* output *signals*.
- Everything in FAUST is a *signal processor* :
  - '+'
  - '3.14'
- Programming in FAUST is essentially combining signal processors.

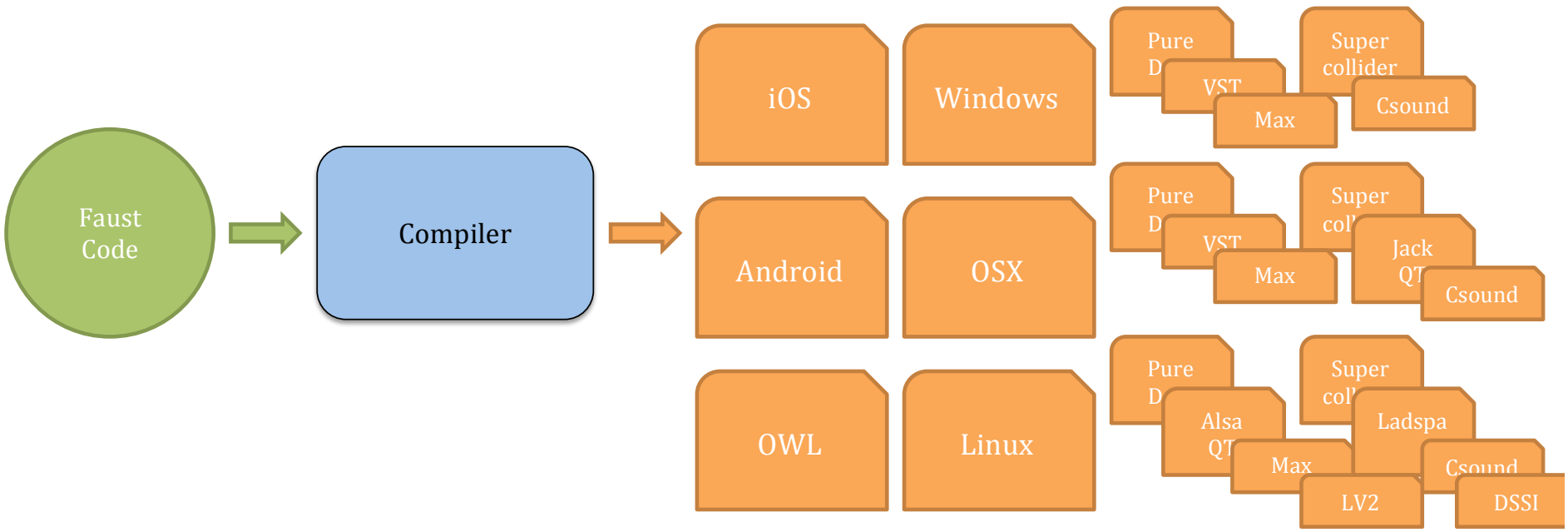
Faust - Example

```
random = +(12345) *(1103515245);
noise = random/2147483647.0;
process = noise * vslider("Volume[style:knob]", 0, 0, 1, 0.1);
```

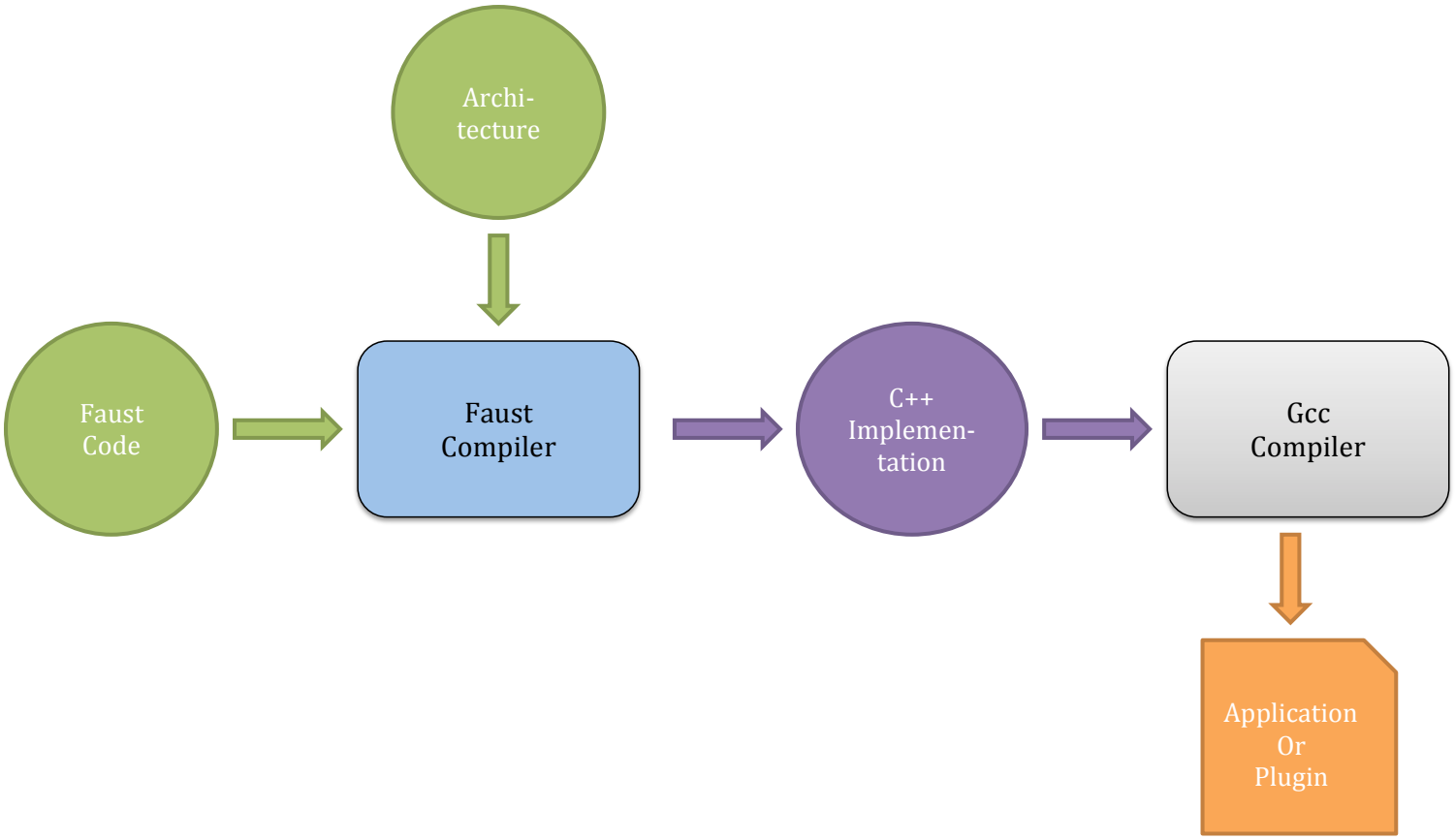


Learn more about faust programming : http://faust.grame.fr

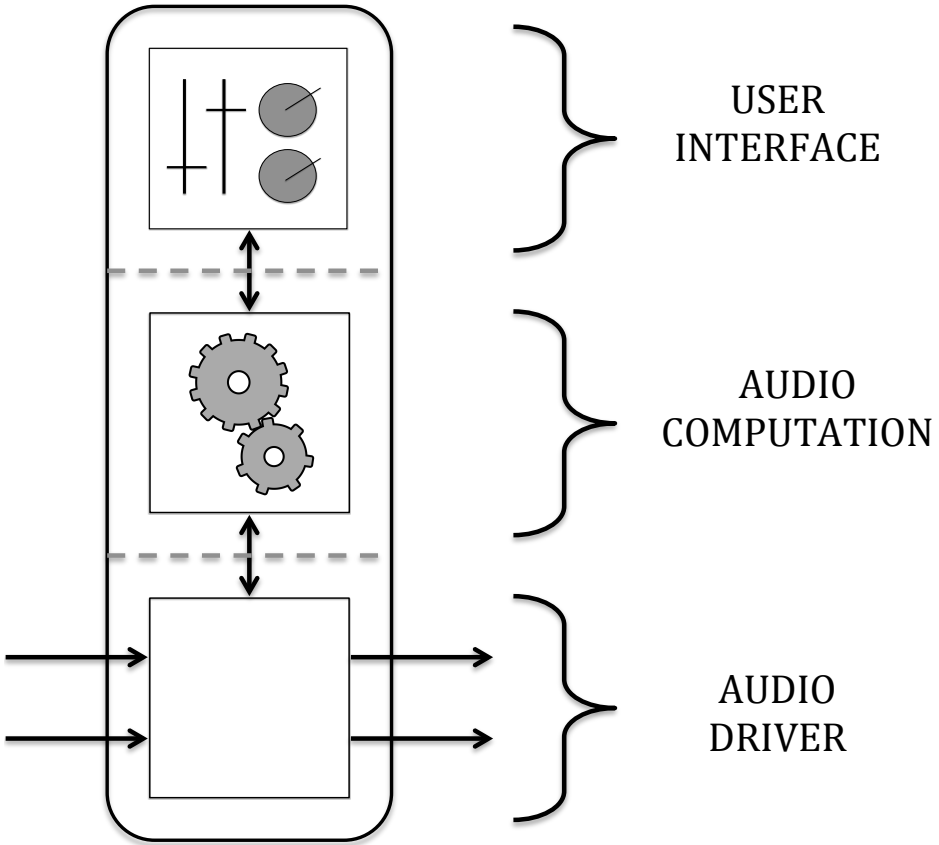
Faust Compiler - Role



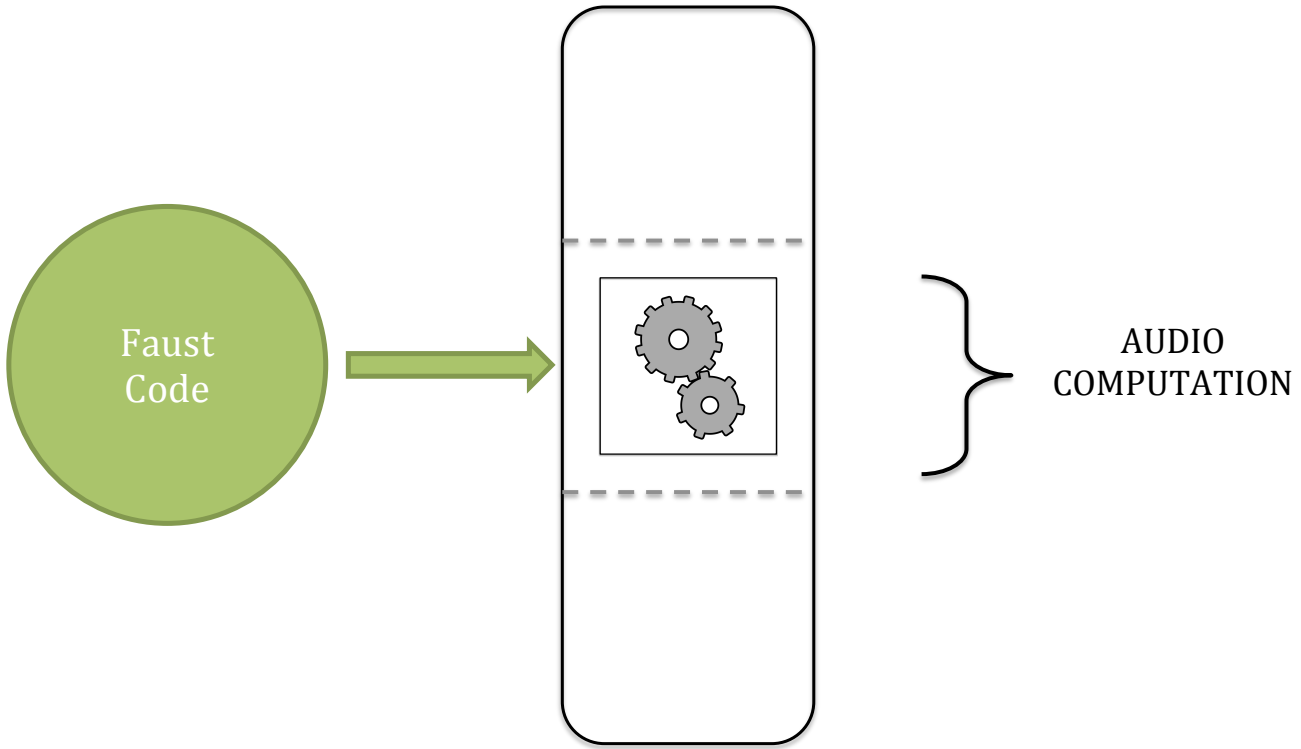
Faust Compiler - Structure



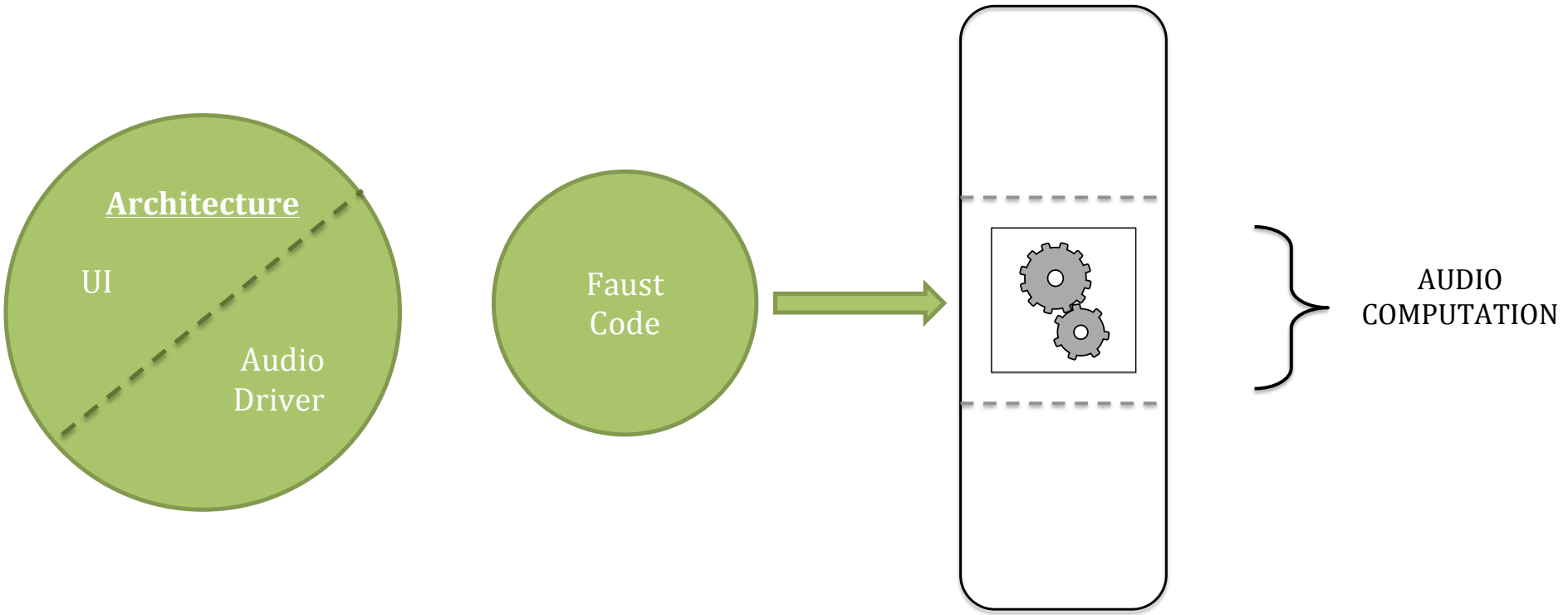
Faust Application Structure



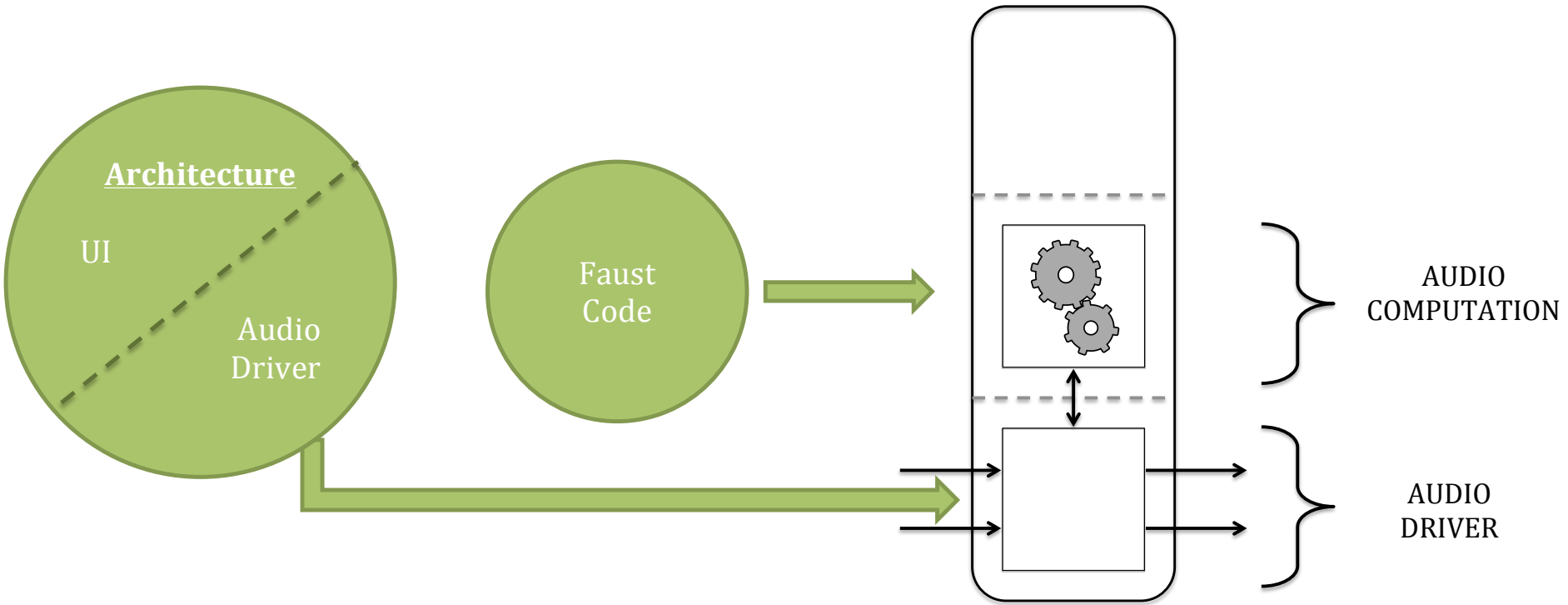
Faust Application Structure



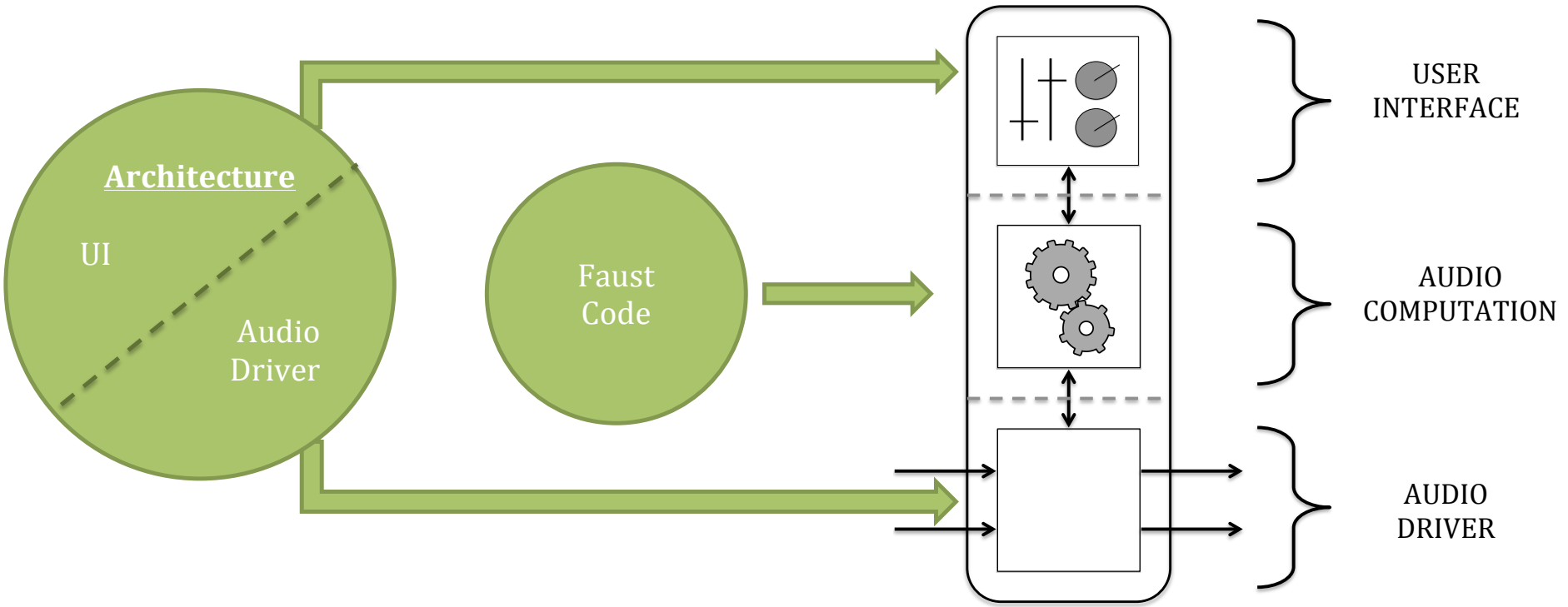
Faust Application Structure



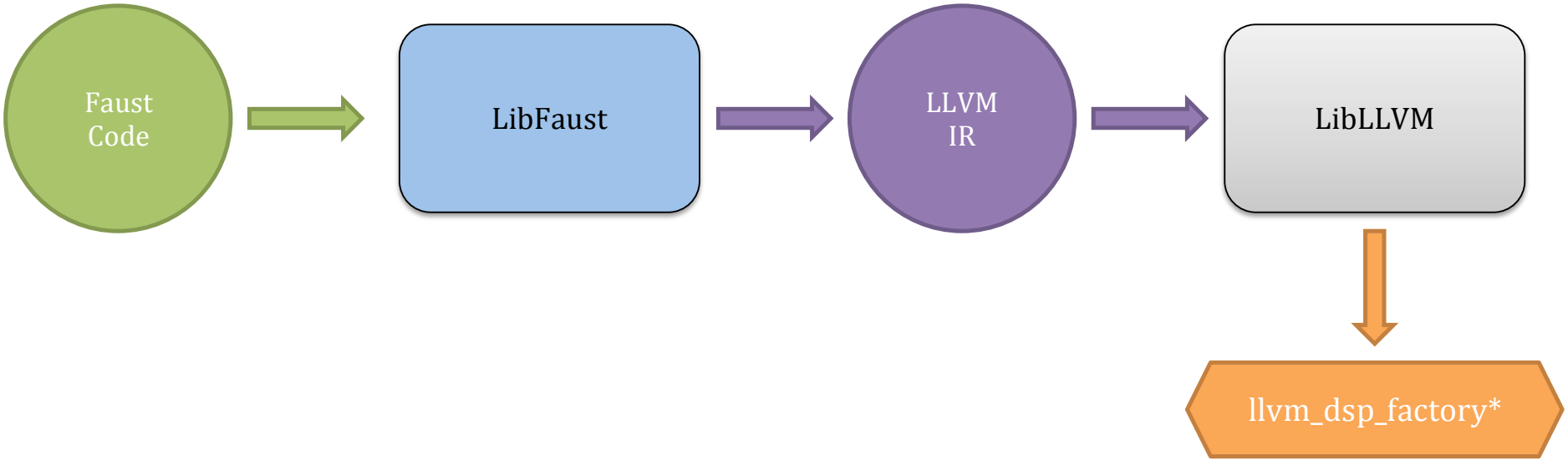
Faust Application Structure



Faust Application Structure



Embeddable Compilation Chain



Faust Compiler's API

- *createDSPFactory*(const std::string& filename,
  int argc, const char* argv[],
  const std::string& target,
  std::string& error_msg,
  int opt_level)

  Builds the prototype of the class

- *createDSPInstance*(llvm_dsp_factory* factory)

  Creates an instance of this class

Existing Embedded Faust Compilers

- Max/MSP (faustgen)
- CSound (faustcsound)
- Antescofo
- Open Music (through LibAudioStream)
- iScore
- **Your** Developement Is Coming Soon ...

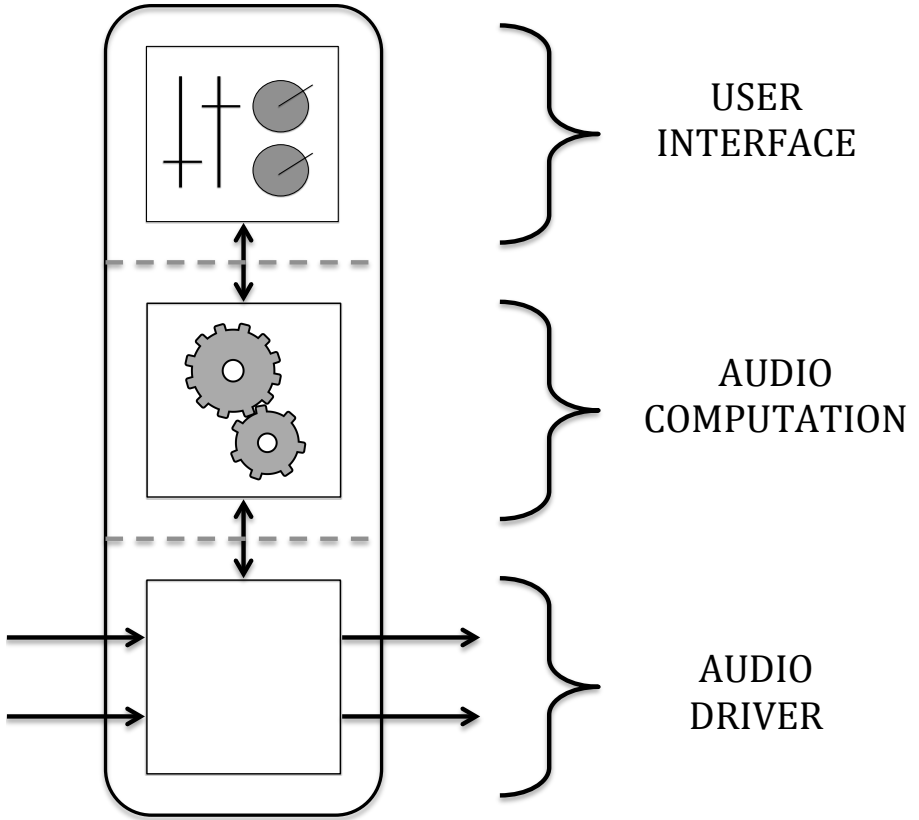
What is FaustLive ?

FaustLive is a QT-based software.

It aims to create a dynamic environment for Faust prototyping.

It includes more advanced features.

It is self-contained.



Basic Features

2 basic features ...

- Drag and Drop Compilation
- Source Edition

Basic Features

2 basic features ...

- Drag and Drop Compilation
- Source Edition

...customized with

- Crossfade
- Maintained Jack Connections

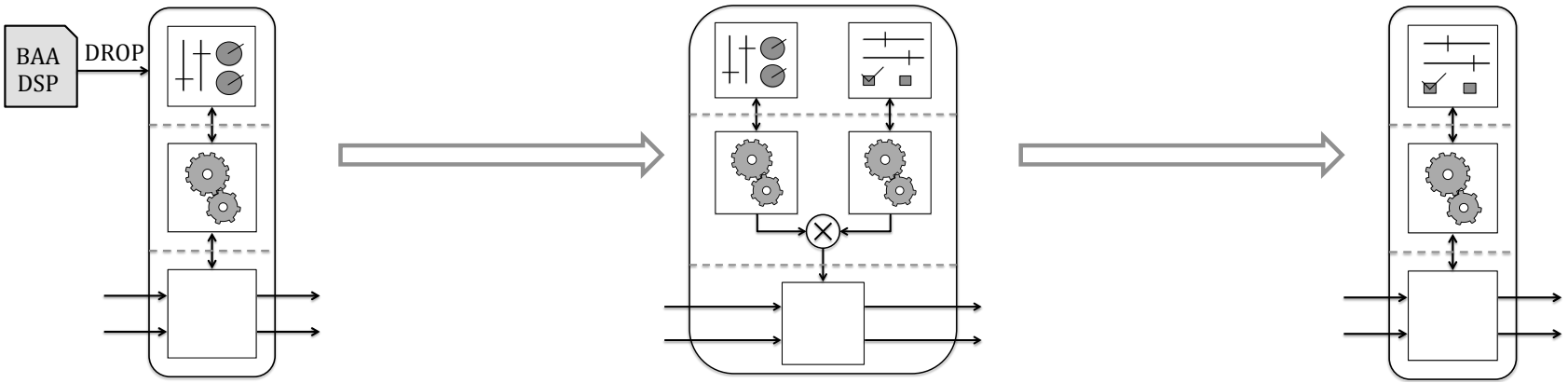
Basic Features

2 basic features ...

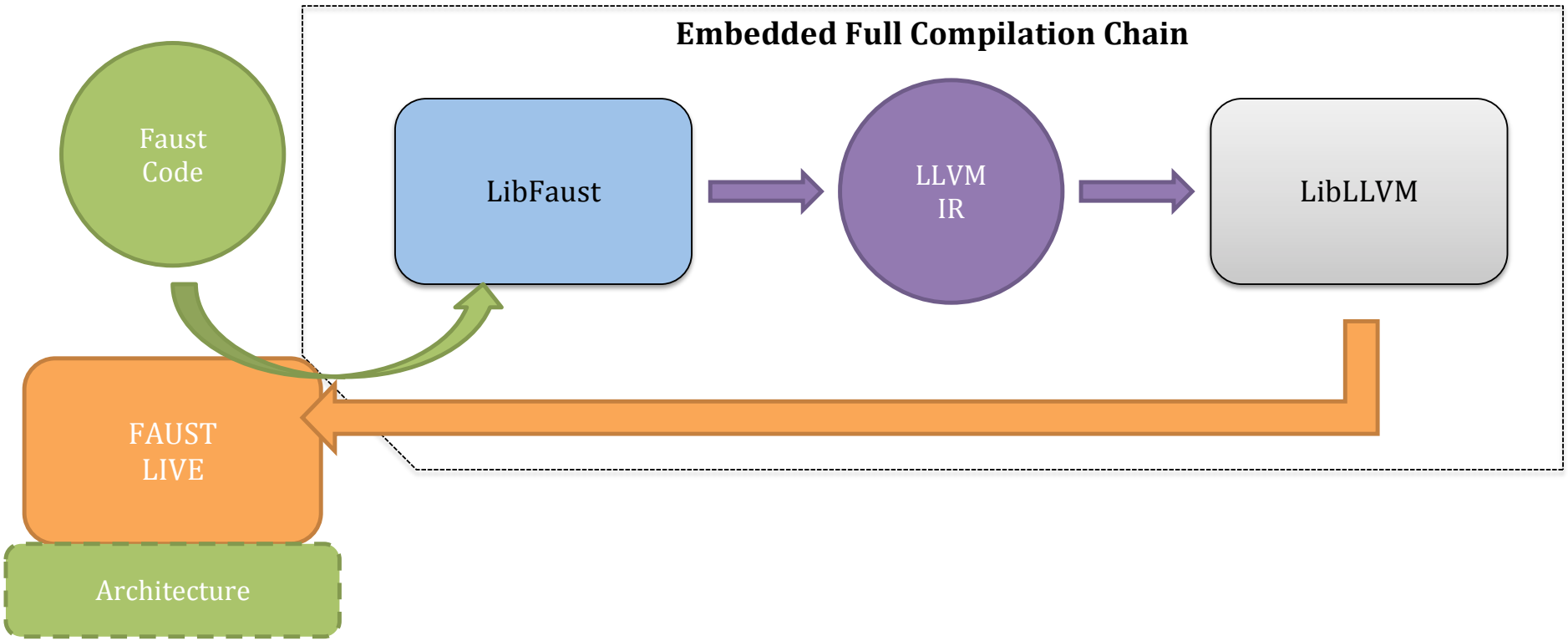
- Drag and Drop Compilation
- Source Edition

...customized with

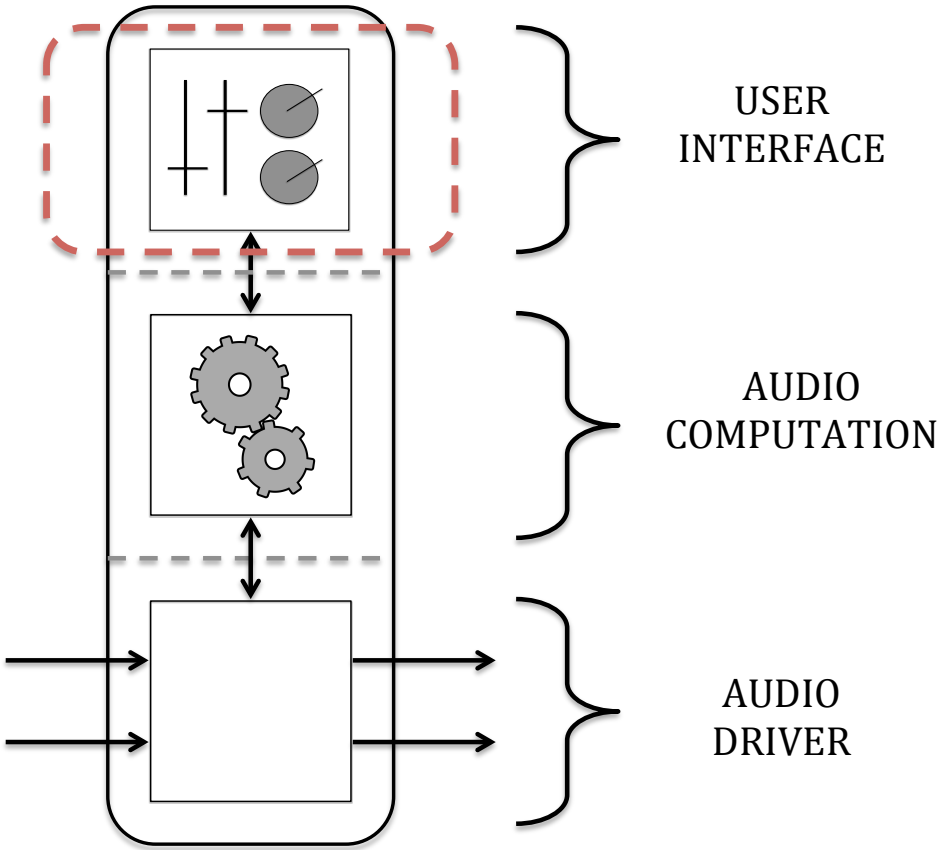
- Crossfade
- Maintained Jack Connections



Embedded Full Compilation Chain in FaustLive



Interfaces Diversity



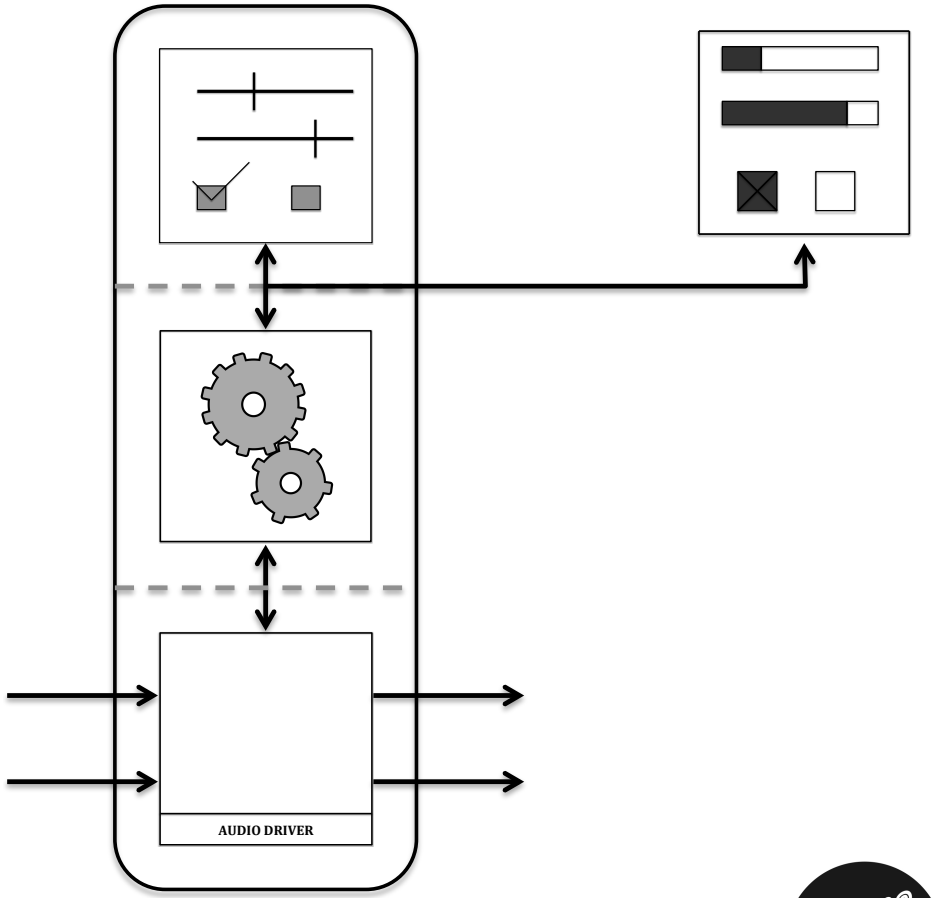
Interfacing systems

- OSC Control

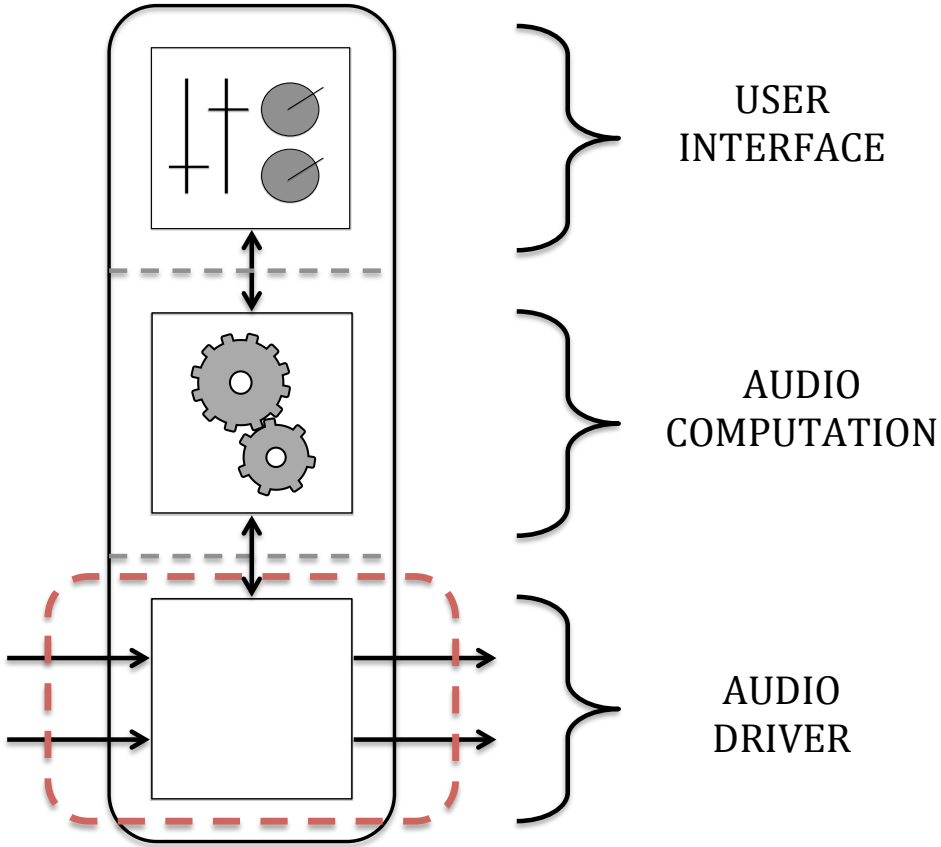
  Open Sound Control protocol is enabled on an UDP port.

- HTTP Control

  A HTTP server is started to deliver a HTML interface. The interface is available on any browser.



Audio Drivers Diversity



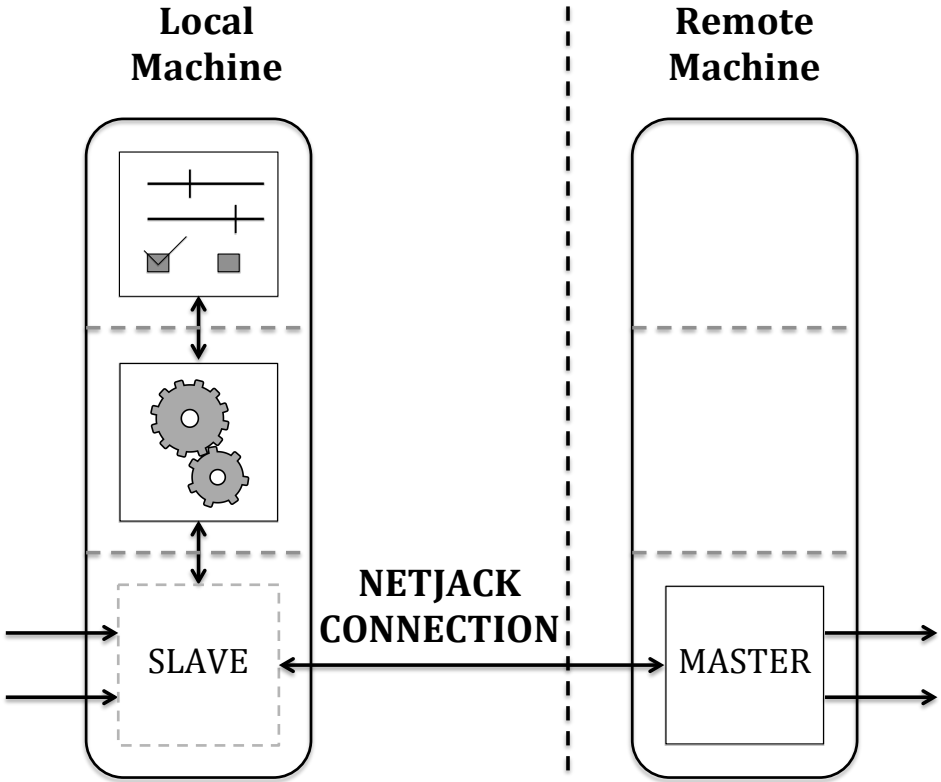
Audio Drivers Diversity

Available drivers :

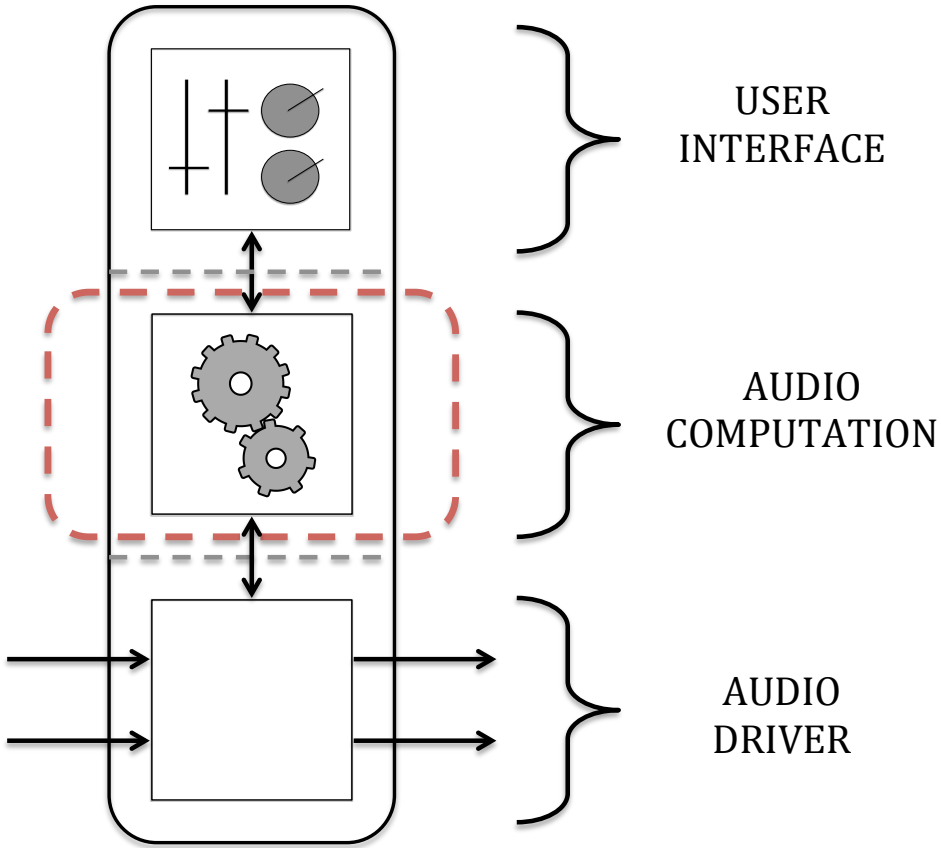
- On Linux : Jack and NetJack
- On OSX : CoreAudio, Jack and NetJack
- On Windows : PortAudio, Jack and NetJack

Remote Audio Rendering with NetJack

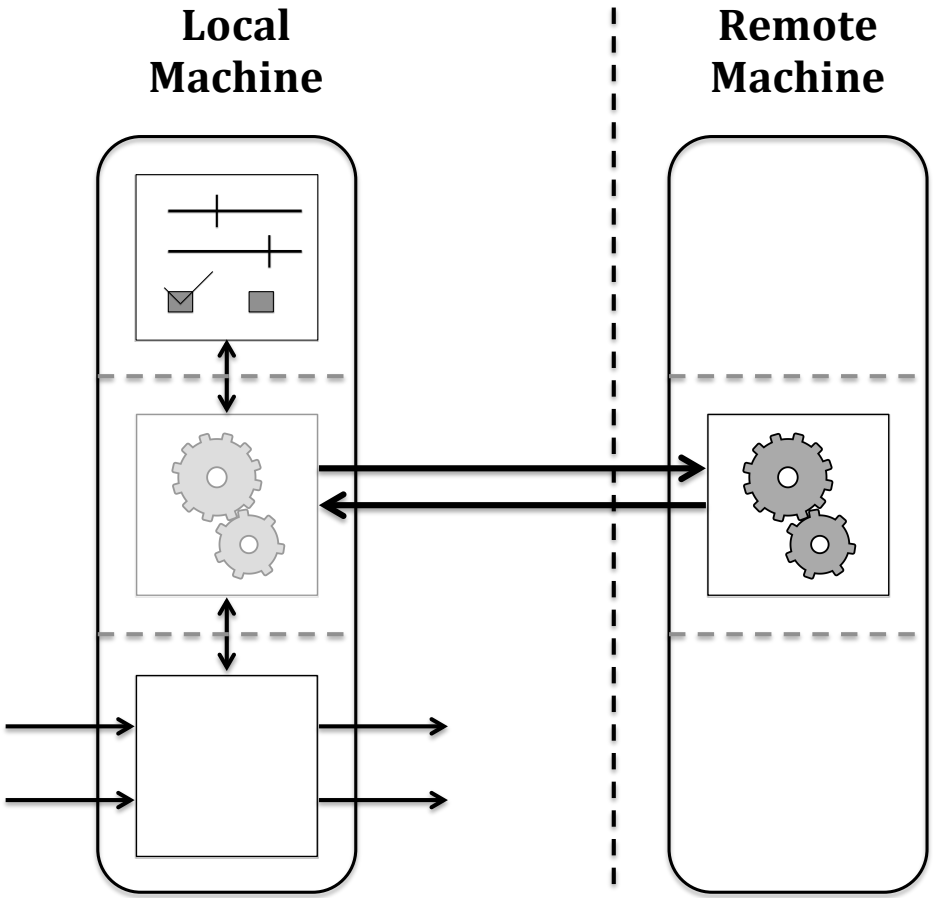
NetJack is used in a light mode, through libjacknet, instead of using Jack server. This library contains the slave/master NetJack protocol.



Remote Processing



Remote Processing



Tools for remote Processing

- Remote Server, a command line application that starts a server waiting for:
  - Compilation requests
  - Processing requests

Tools for remote Processing

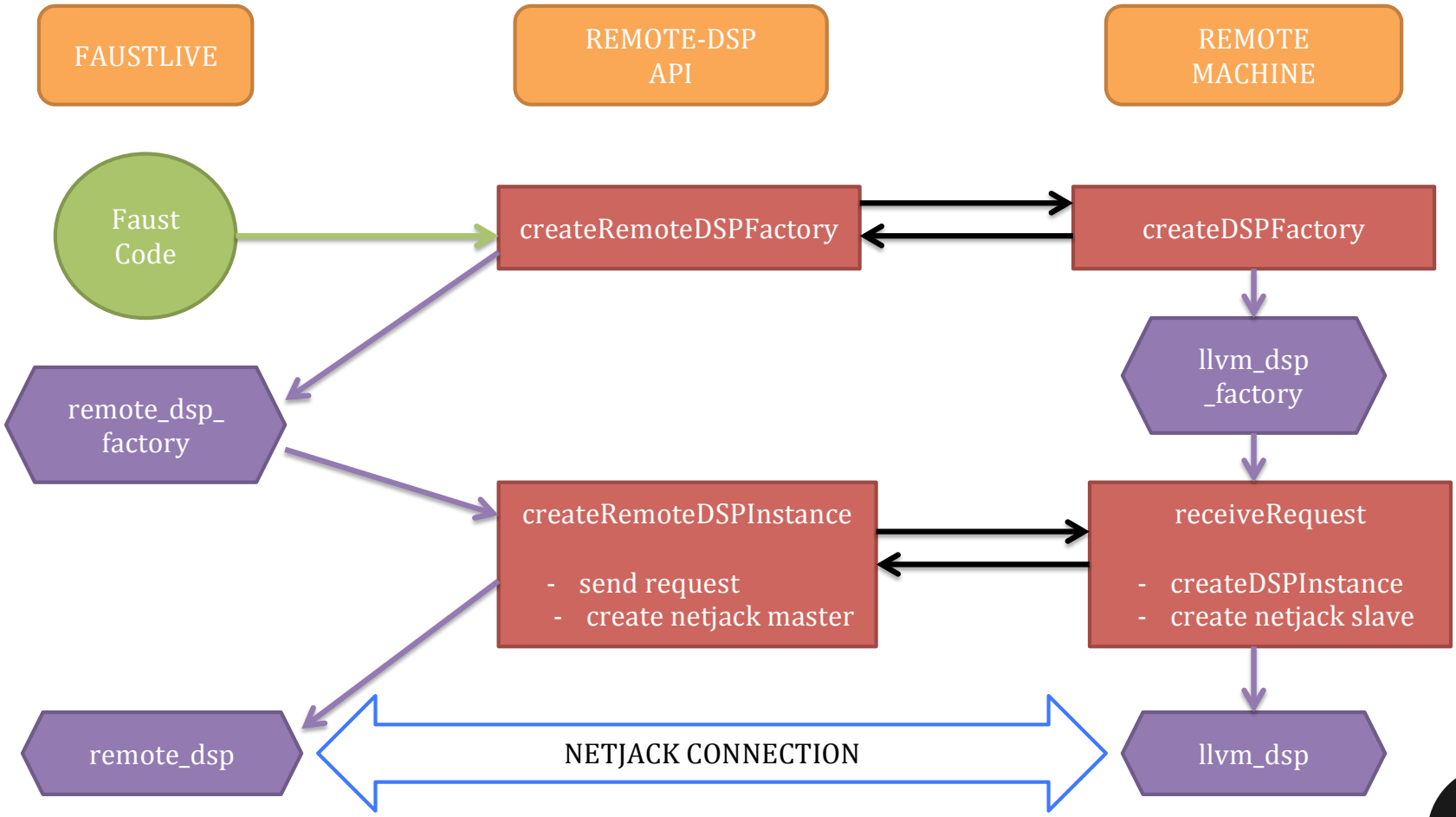
- Remote Server, a command line application that starts a server waiting for:
  - Compilation requests
  - Processing requests

- Remote Client API, a "proxy" API to make transparent the creation of remote dsp.

Tools for remote Processing

- Remote Server, a command line application that starts a server waiting for:
  - Compilation requests
  - Processing requests
- Remote Client API, a "proxy" API to make transparent the creation of remote dsp.
- FaustLive, using the API for remote processing.

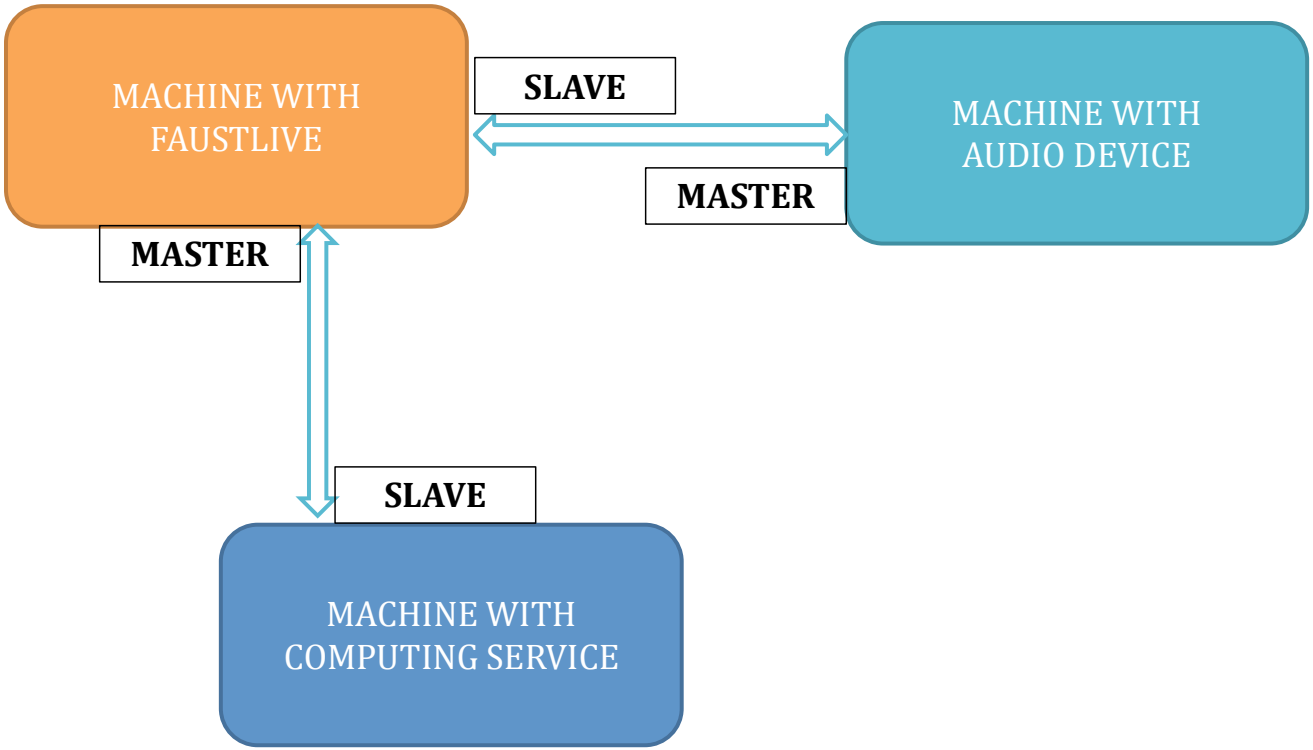
Remote Processing - Flows



Faust Remote API

- *getRemoteMachinesAvailable*(map<string,pair<string,int>* machine_list)
- *createRemoteDSPFactory*(const string& filename, int argc, const char* argv[], const string& ip_server, int port_server, string& error_msg, int opt_level)
- *createRemoteDSPInstance*(remote_dsp_factory* factory, int argc, const char* argv[], int sampling_rate, int buffer_size, RemoteDSPErrorCallback error_callback, void* error_callback_arg, int& error)
- *getRemoteFactoriesAvailable*(const string& ip, int port, vector<pair<string, string> >* factories_list)

Remote Processing - Configuration



Remote Processing - iPad Use Case

- A motivation :

  Experiment quickly the faust applications that use accelerometers.

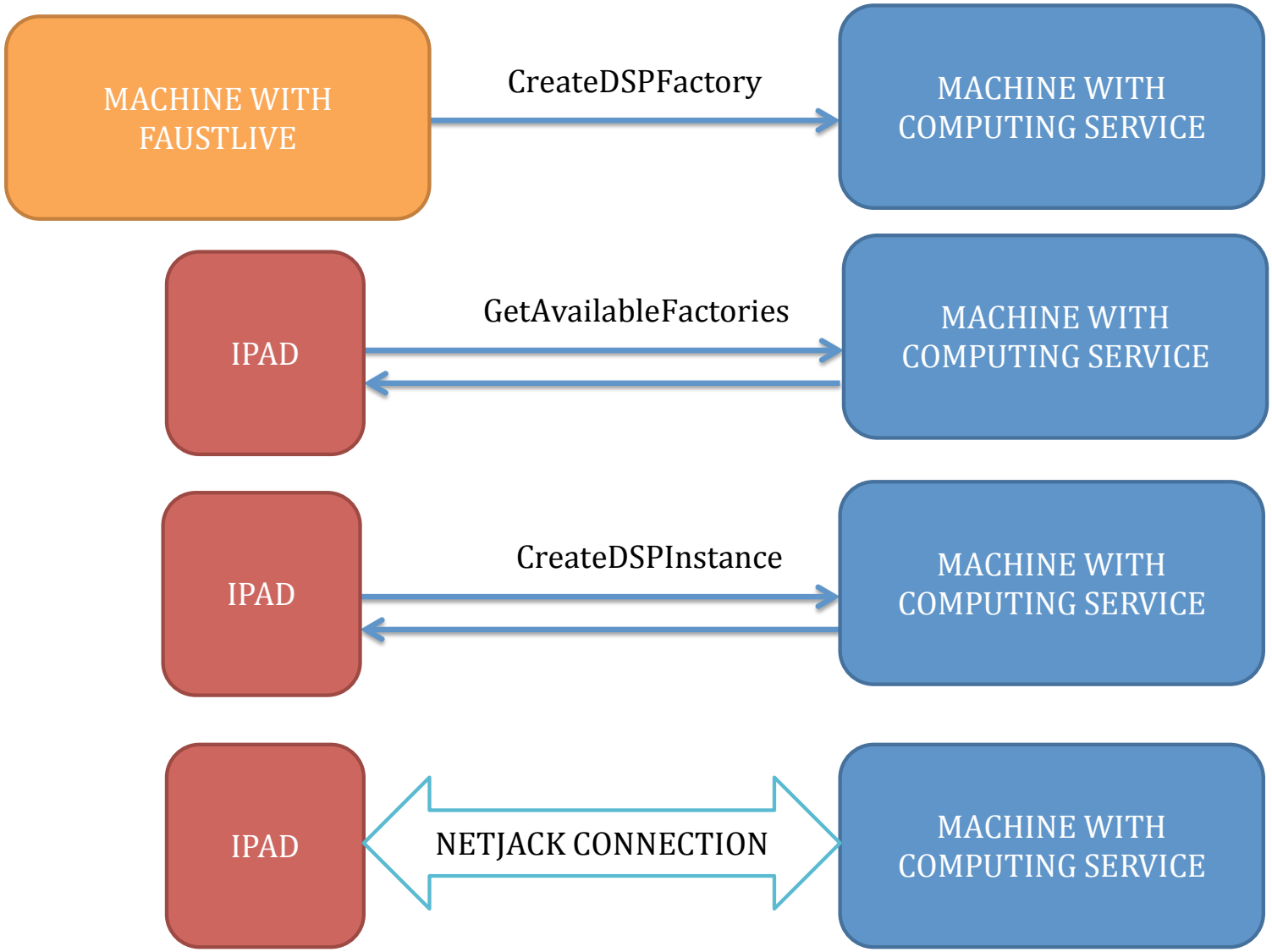
- A technical obstacle :

  The interdiction to embed compilers in iOS application.

- A solution:

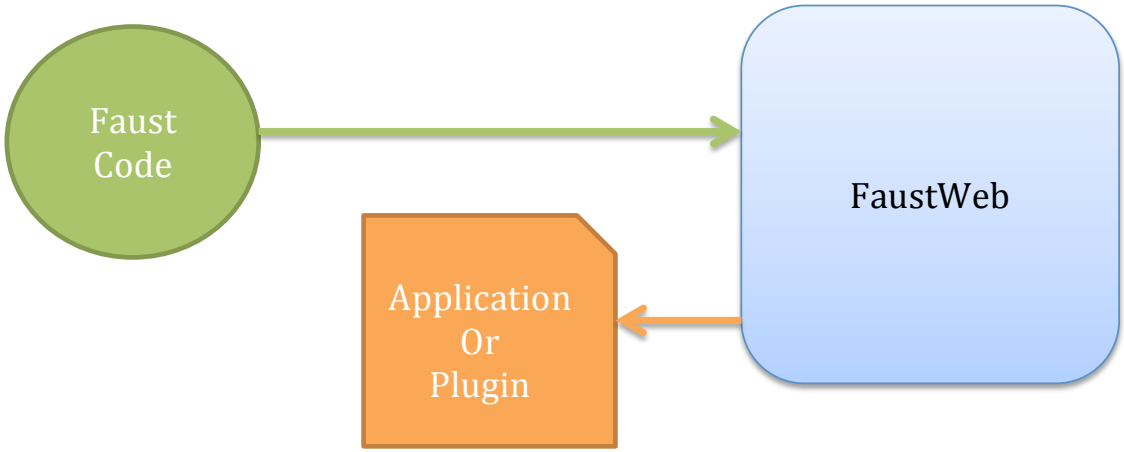
  Use the remote processing service.

Remote Processing - iPad Configuration



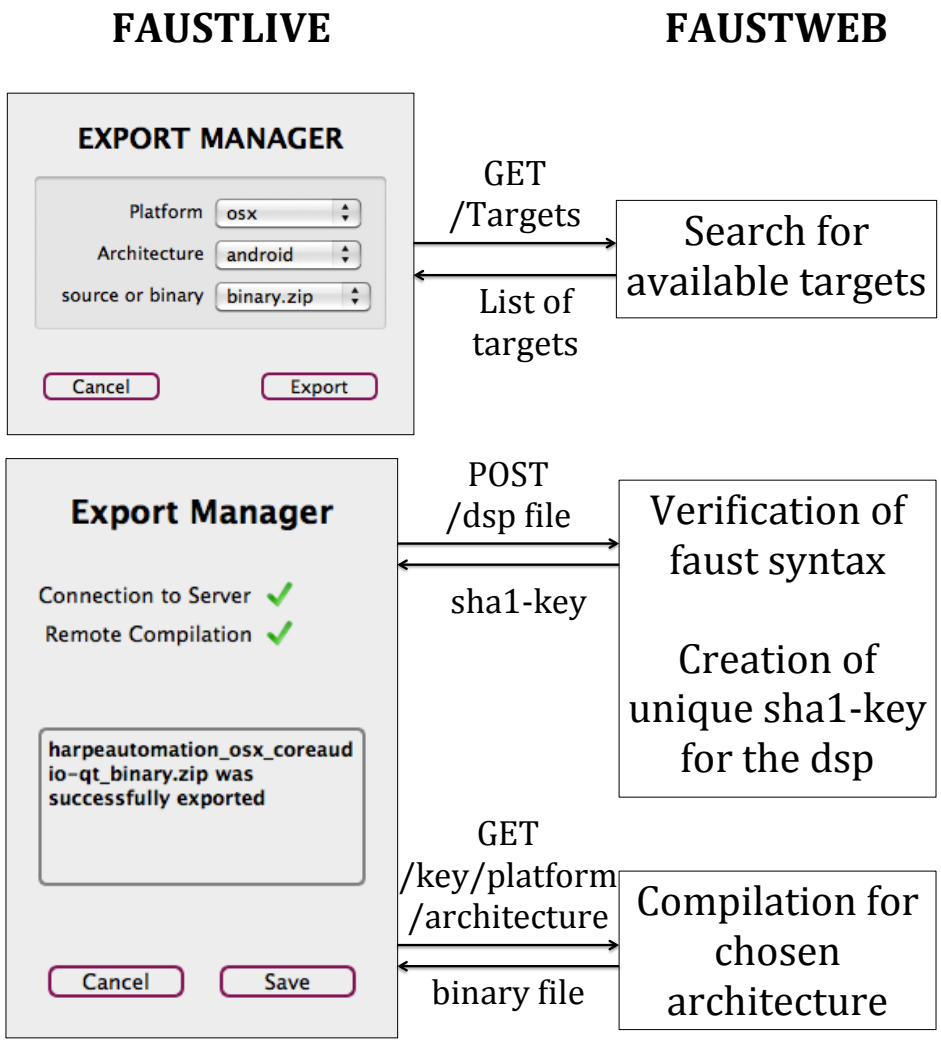
Export with FaustWeb

FaustWeb is a web service for remote compilation : http://faustservice.grame.fr.



FaustWeb can be used through a browser or programatically

Export with FaustWeb



Conclusion

- FaustLive has some perspectives
  For example : embed the remote server into FaustLive
- Binaries will be soon available on : http://faust.grame.fr/
  for OSX, Windows
- The sources are already available on :
  http://sourceforge.net/p/faudiostream/faustlive/ci/master/tree
- This research was granted by the ANR (Agence Nationale de la Recherche) [INEDIT]

Thank you for your attention