

CESNET Technical Report 20/2009

Universal Transcoder to Convert FLAC Streaming Audio to Other Formats

MILOŠ WIMMER

Received 15.12.2009

Abstract

It was our goal to design and implement a universal transcoder capable of real-time conversion of loss-less FLAC streams to other formats, making them available to other streaming servers or end-user clients. We have succeeded in implementing a system with an open, modular architecture, whose components may be freely combined or replaced with alternatives. This is very important, especially in the case of output stream producers, whose choice is not limited to a single encoding application. Any program capable of processing the input stream and generating output per specifications may be used as a producer. The implementation of the transcoder relies exclusively on free technologies.

Keywords: FLAC, audio streaming, universal transcoder, Ogg Vorbis

1 Introduction

CESNET is a long-term partner of Czech Radio¹ with a history of mutual cooperation in live broadcasting of its programs over the Internet. The primary focal point is high-quality audio broadcasting. We have implemented custom streaming system based on the loss-less FLAC² compression format as well as a lossy Ogg Vorbis format [1], [2], [3], [4].

This study aims at designing and implementing a universal transcoder to receive a FLAC stream in real time, decode it into PCM signal and generate streams in other formats.

2 Description of the System and the Environment

A year ago, we have implemented an encoding server [4] to process audio signal of the D-dur station generated in the Czech Radio center in Vinohrady, Prague, by converting it into the FLAC compression format and using Ogg transport container to transfer it to Cesnet cluster of streaming servers³, which then makes the stream available to listeners.

This system allowed us to achieve uncompromising quality, offering users a unique chance to receive audio signal in a form identical to that generated by the Czech Radio center. Unlike high-speed Ogg Vorbis streams⁴ or DVB-S/DVB-T

¹ <http://www.rozhlas.cz/>

² <http://flac.sourceforge.net/>

³ <http://radio.cesnet.cz/>

⁴ <http://www.xiph.org/>

digital broadcasting formats, FLAC production chain is free of lossy compression algorithms and audio quality is not degraded.

Neither outages nor any other problems occurred in the course of the year-long operation of the system, proving that the solution is stable and suitable for further use.

Czech Radio internet broadcasting environment uses a wide range of producers/encoders generating streams in various formats and bitrates for live broadcasting of Czech Radio stations. Input signal is typically received from DVB-S or DVB-T broadcasts, making it partially degraded. Resolving that by connecting all encoders directly to the output of the broadcasting center is, however, not possible.

While in operation, the FLAC encoding server proved its quality and stability, showing its potential for internal use. Its lossless output stream may serve as an input for other encoders, taking advantage of the fact that the decoded signal is identical with the original, not distorted by lossy compression and normalization typically taking place within the transmission chain.

A universal transcoder converting the FLAC stream in real time to other formats and bitrates seems to be an alternative to the present-day solution. Such a universal producer could transmit its output streams to broadcasting servers currently used by end-user clients.

We have designed and implemented such a transcoder.

2.1 General Model

The general model of a real-time transcoder consists of three components installed in one network-enabled computer. The first component is a client/decoder receiving the FLAC stream on the input, producing PCM signal and passing it onwards. Second in line comes a system component making the signal available for reading simultaneously by multiple applications (producers/encoders). The third stage consists of multiple producers processing the PCM data, generating compressed streams in various formats and bitrates, and forwarding them to their respective streaming servers.

In its simplest form, the solution could only consist of two components – a client to decode the signal and a producer to generate the output stream. Such a pair could be established for each output format required. However, it was our goal to develop a universal transcoder able to process a wide range of input and output streams. Each FLAC stream generates approximately 800 kb/sec of traffic and requires certain amount of processor time to decode the original PCM data. That is why we want to process the FLAC stream for each station only once, relying on a single client to decode the stream and provide the PCM data simultaneously to all producers generating their respective output streams. This approach will save transmission capacity, processor time and overall processing capacity of the computer.

The task of making the decoded PCM signal available simultaneously to multiple producers cannot be entrusted to the input client. System tools must be employed to achieve that goal. Jack⁵, a low-latency application sound system, is

⁵ <http://jackaudio.org/>

considered the best solution for the following reasons. It is a high-quality free product capable of (among other features) combining audio channels and making them available to multiple applications at once, while consuming only a minimum amount of system resources with no latency and no need for a sound card.

The input client must be able to receive and decode a FLAC audio stream and forward its output (PCM data) to the jack server. This can be done by mplayer⁶ or VLC⁷ among others.

The actual implementation of the transcoder depends on our preferences concerning various parameters of the solution as they govern our choice of system tools. Essentially, we need to choose between low latency or high reliability of the transcoder.

Models presented below represent transcoders designed to process a single input stream. With multiple inputs, each one will be served by its own instance of the given model. Certain components could still be shared by both models.

2.2 Low Latency Model

The lowest latency has been achieved by implementing a model shown in Figure 1.

The input client (VLC or mplayer) receives the FLAC stream, decodes it, produces PCM data and forwards them to a registered port (channel) of the jack server to which it connects to.

Encoding – production of the required output format – is done by a producer, e.g. a VLC instance running in an encoder mode. Producers connect to other ports of the jack server. Administration tools are used to connect the receiving client's output to the input of the producers.

The VLC encoder forwards its output stream directly to a selected TCP port. It may also be easily combined with an ices2 client, passing its output stream to the client, who takes over and transports it to a remote Icecast⁸ server.

Certain difficulties can occur once the input client is required to restart processing its input stream following an outage. This is something unlikely to occur in Czech Radio's environment; however, it must not be omitted in our considerations. The client needs to run in a loop implemented either internally, or as an infinite loop in the system. In case of the client terminating due to the lack of input signal, the loop will cause it to delay for a pre-set period and then re-initiate.

2.3 Maximum Reliability Model

The transcoder designs given in Figure 2 show most promise with respect to the transcoder's maximum resistance to failures and outages.

The input FLAC stream is being received by a local icecast server. The FLAC stream may originate directly from the source; another option is running the icecast server in a relay mode, and taking over the input stream from a remote primary icecast server. A local icecast server is used for its ability to handle occasional outages

⁶ <http://www.mplayerhq.hu/>

⁷ <http://www.videolan.org/>

⁸ <http://www.icecast.org/>

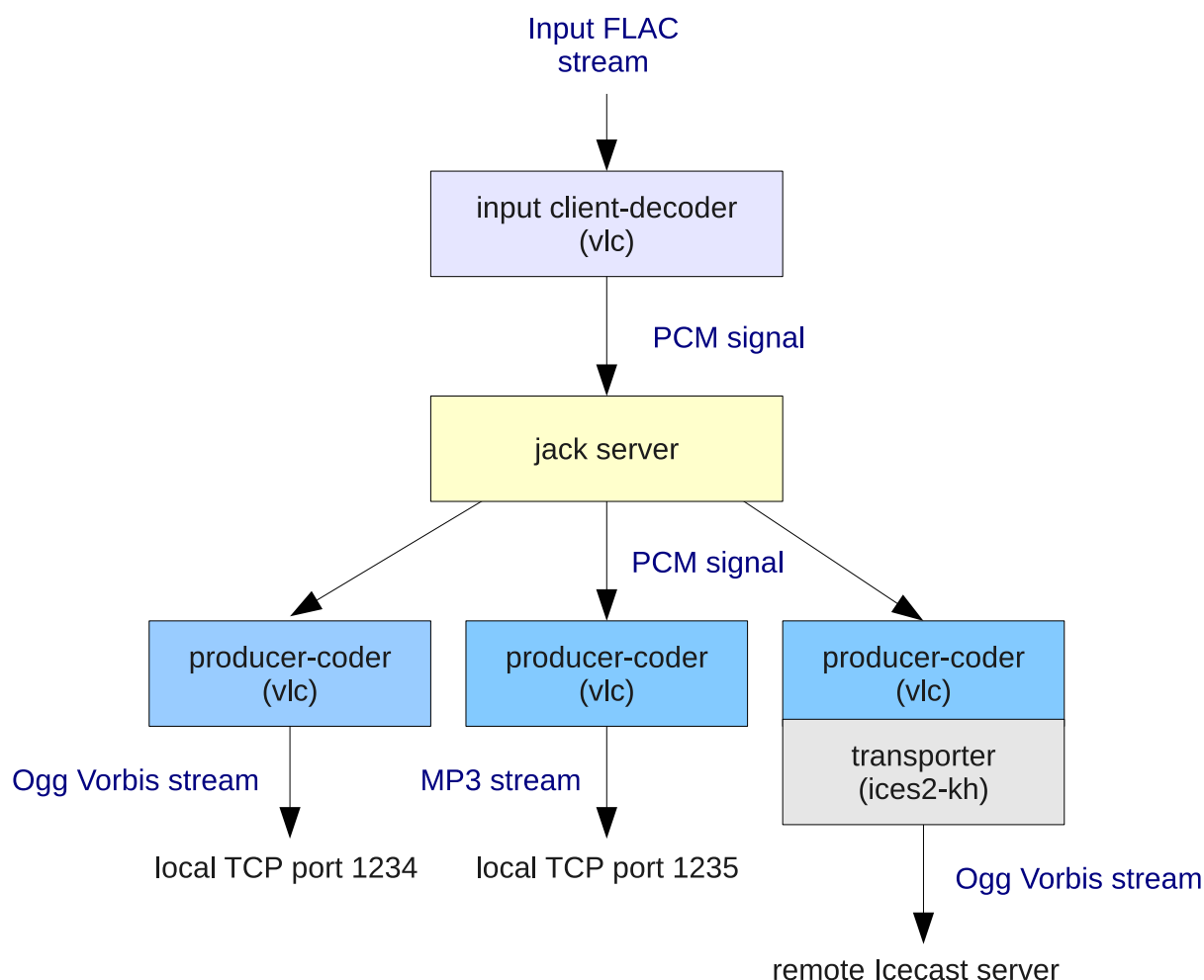


Figure 1. Topology of a low latency model.

of the input stream. Besides that, it introduces no cumulative latency to the streams it processes, even after a prolonged period of operation. There will be a latency of a few seconds but it will remain fixed all the time.

The input client (e.g. mplayer) then receives its input FLAC stream from the local icecast server, decodes it, produces PCM data and forwards them to the registered port of the jack server it connects to.

Producers (e.g. Darkice or a combination of VLC and ices2) also connect to the jack server, reading the PCM data from its ports, generating their respective output streams (using various encoding formats such as Ogg Vorbis, MP3, AAC or AAC Hev2), and forwarding them to a remote icecast server or the local icecast server that is also receiving the original input. Additional stability may be achieved by forwarding the output streams of the producers to the local instance of the icecast server, making the whole production process local and trusting the task of transmitting the output streams to a remote icecast server to the well-proven relay function implemented by icecast servers.

It is possible to overcome trouble caused by a possible outage of the input FLAC stream by sending the same input to the icecast server from another source and configuring icecast to regard that as backup. Once the primary stream fails, the icecast server will automatically switch to the secondary one, making it available to

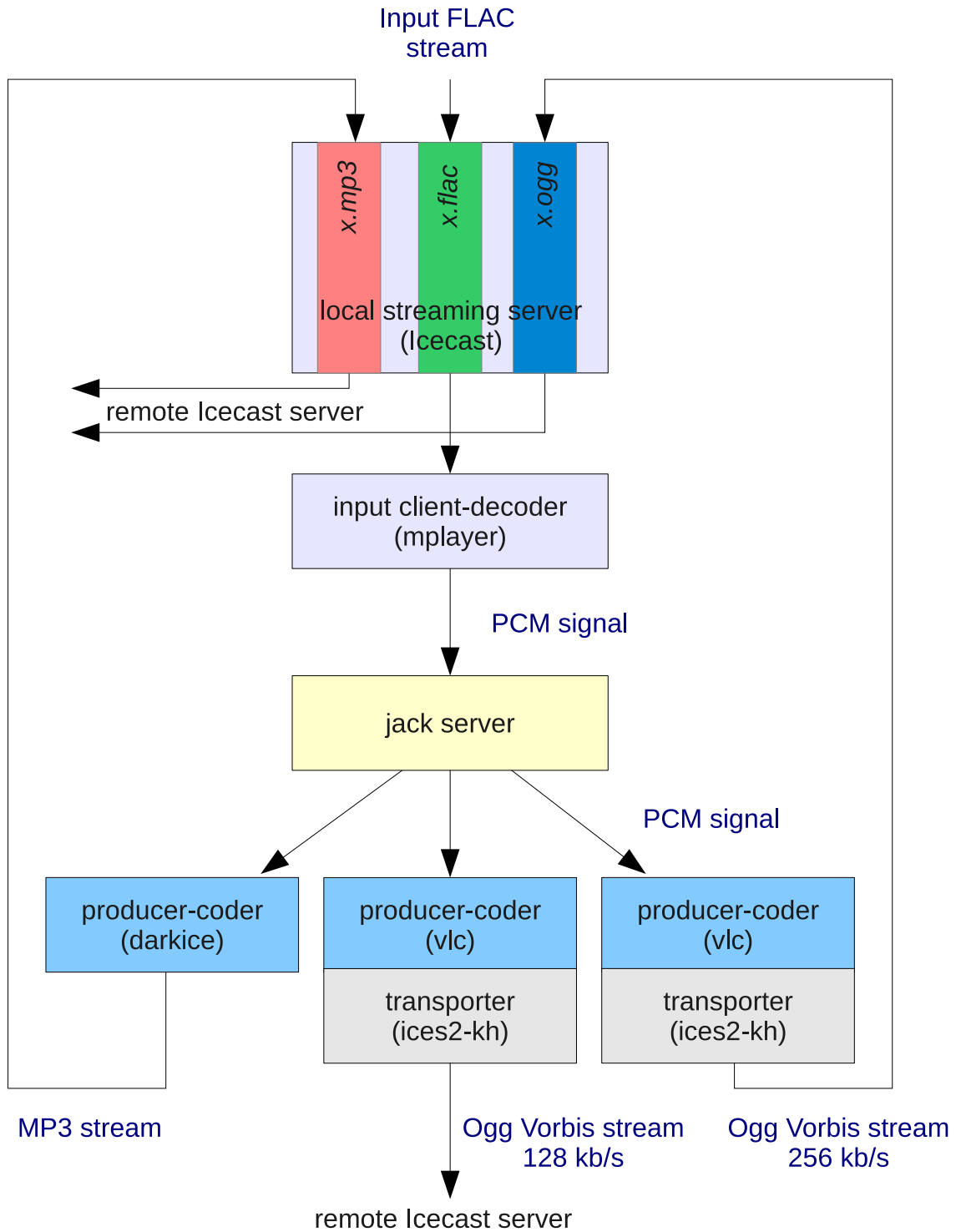


Figure 2. Topology of a maximum reliability model.

all input clients connected to its output. Thanks to that, decoding clients do not lose their input and do not fail. Once the primary stream gets back up, the icecast server switches back to it, restoring the original state. With no backup stream available, another stream may be used as an alternative containing either a voice recording notifying listeners of the outage, or silence.

2.4 Combining Both Models

When implementing the actual transcoder, component modules discussed above may be used and both designs combined.

One option worth considering is a combination of the low-latency model with a local icecast server receiving output streams from individual encoders, making the transcoder more reliable as a tradeoff for a slight increase in latency.

2.5 Recommendations Based on Practical Experiences

VLC's ability to process a single input stream and produce multiple outputs in a single instance may seem promising. Unfortunately, the current implementation of that feature (specifically in version vlc-1.0.3) does not yield satisfactory results. Even with sufficiently powerful hardware, outages occur in the output streams and VLC becomes rather unstable.

Our experience with transferring VLC's output directly to the icecast server was negative as well. There were outages, which is why we recommend combining each vlc encoder with another tool – preferably ices2, which has provided the best results.

Processing a single input stream and using it to generate multiple outputs cannot be implemented through forked system pipes as trouble arises every time one of the components within the setup restarts.

3 Implementation

We have implemented and successfully operated both models outlined above. Our universal transcoder was running on a DELL PowerEdge 1950 sever equipped with two Intel Dual Core Xeon/3.0GHz processors, 4 GB or RAM, two integrated gigabit Ethernet adapters, and no sound card.

The whole solution relies exclusively on free software. As is the case with all the other servers in our streaming system, the server runs Debian GNU/Linux⁹. Given the dynamic nature of development in the area of codecs and streaming support, we wanted to include the latest versions of key components (VLC-1.0.3, mplayer-1.0.rc2svn20091118, jack-0.116.2+svn3592 and darkice-0.19) in our testing. That is why we have been using Debian Testing version, which makes all of them available.

Processor load is generated mostly by encoders producing the output streams. On the other hand, the process of decoding the FLAC stream is not very demanding. We have not encountered any lack of processing power in our testing setup, transcoding six input FLAC streams and producing 14 output streams in Ogg Vorbis and MP3. There were no outages or distortions on the output.

Final models resulting from the implementation process and our choice of software tools meet our demands for functionality and stability of the transcoder.

⁹ <http://www.debian.org/>

4 Technical Details

The previous section dealt with an overall model of the solution. The text below is going to cover components in greater detail and present examples of configuration files.

4.1 Jack Server

Jack, a low-latency sound system, has been chosen for its ability to make identical PCM signal available to multiple applications by interconnecting audio channels.

We have been using jack server version 0.116.2+svn3592 available in the OS distribution.

The following arguments are passed to the daemon at startup:

```
/usr/bin/jackd --realtime --driver dummy --rate 48000 --period 2048 &
```

All operations the audio signal is subject to are digital in nature. Input is received from a computer network, eliminating the need of a sound card. That is why the jack server runs with a dummy driver for a virtual sound adapter.

4.2 Low Latency Transcoder

Transcoder implementation based on the low latency model takes the form of the following script:

```
#!/bin/sh
SOURCE="http://tun4.cesnet.cz:8000/cro-d-dur.flac"
```

A loss-less FLAC stream of Czech Radio's D-dur station, originating from the Czech Radio center, is the primary source.

```
# Input:
cvlc --rt-priority --aout jack --quiet --loop $SOURCE vlc://pause:2 \
> /dev/null 2>&1 &
pidA="$!"
```

The VLC input client runs in console mode with no user interface, sending its output to a jack server. It runs in an internal loop, alternating between two sources: the source FLAC stream and a 2-second pause.

```
# Producer VLC:
cvlc --rt-priority --sout-transcode-high-priority jack:// \
--sout "#transcode{acodec=vorbis,ab=256,samplerate=48000}:\
std{access=http,mux=ogg,dst=:1234}" >/dev/null 2>&1 &
pidB="$!"
```

Production of an output stream with selected encoding is handled by another cvlc instance. Its input connects to the jack server and generates a stream with lossy Ogg Vorbis compression at 256 kb/sec with a sampling frequency of 48 kHz. It forwards the output stream to TCP port 1234, making it available to other applications or servers.

```
sleep 1
# Connects the two named ports:
jack_connect "vlc_$pidA:out_1" "vlc-input-$pidB:vlc_in_1"
jack_connect "vlc_$pidA:out_2" "vlc-input-$pidB:vlc_in_2"
```

Both cvlc instances have registered with named audio ports of the jack server. Two commands shown above will connect the output port of the decoder to the input port of the producer. Connection is established separately for the left and right channel. VLC does not allow a free specification of the name of its output port. It uses a fixed form consisting of a prefix and the PID of its current instance.

Should we wish to transfer the transcoded stream to an icecast server (local or remote), it is better to include the ices2-kh application into the chain, achieving reliable transport of the stream over the network.

```
# Producer VLC | ices-kh:
cvlc --rt-priority --sout-transcode-high-priority jack:// \
--sout "#transcode{acodec=vorbis,ab=256,samplerate=48000}:\
std{access=file,mux=ogg,dst=-}" | \
/usr/local/bin/ices-kh /usr/local/etc/cro-d-dur#256.xml >/dev/null 2>&1 &
```

The VLC producer forwards its output stream to a standard output and, through a system pipe, to the standard input of ices2-kh. To achieve correct connection of the input client's audio port to that of the producer through the jack server, vlc producer's PID must be found among the list of running processes.

The following examples show excerpts of ices2-kh configuration files relevant to the transport of the output stream:

```
#Fragment of cro-d-dur#256.xml configuration file
<stream>
<input>
  <module>playlist</module>
  <param name="type">basic</param>
  <param name="file">/usr/local/etc/stdin.playlist</param>
  <param name="random">0</param>
  <param name="restart-after-reread">1</param>
</input>
<runner>
  <instance>
<shout>
  <hostname>amp1.cesnet.cz</hostname>
  <port>8000</port>
  <password>xxx_passwd_xxx</password>
  <mount>/cro-d-dur-256.ogg</mount>
  <reconnectdelay>5</reconnectdelay>
  <reconnectattempts>-1</reconnectattempts>
  </shout>
  <passthru>1</passthru>
  </instance>
</runner>
```

```
</stream>
#File /usr/opt/ices/etc/stdin.playlist
-
```

The stdin.playlist file only contains the "-" character denoting standard input – a source of data read by ices-kh. The resulting stream is then forwarded to a remote icecast server at amp1.cesnet.cz, and made available at address

<http://amp1.cesnet.cz:8000/cro-d-dur-256.ogg>.

4.3 Maximum Reliability Transcoder

Transcoder implementation based on the high reliability model takes the form of the following script:

```
#!/bin/sh
SOURCE=http://localhost:8000/cro-d-dur.flac
```

A loss-less FLAC stream of Czech Radio's D-dur station is the primary source provided by a local instance of the icecast server, which takes it over from a remote icecast server. This so-called relay technique provides extraordinary tolerance to outages, assuring smooth recovery. The local icecast server defines the stream in the following manner:

```
#Fragment of icecast.xml configuration file
<relay>
<server>tun4.cesnet.cz</server>
<port>8000</port>
<mount>/cro-d-dur.flac</mount>
<local-mount>/cro-d-dur.flac</local-mount>
<relay-shoutcast-metadata>1</relay-shoutcast-metadata>
</relay>
```

with the stream being taken over from a primary server at address

<http://tun4.cesnet.cz:8000/cro-d-dur.flac>.

```
# Input:
mplayer -ao jack:name=mplayer_cro-d-dur -quiet $ZDROJ >/dev/null 2>&1
&
```

Stream provided by the local icecast server is being received and decoded by mplayer, which sends it to named ports mplayer_cro-d-dur:out_0 and mplayer_cro-d-dur:out_1 (left and right channel).

```
# Producer darkice:
/usr/bin/darkice -c cro-d-dur-256.cfg > /dev/null 2>&1 &
```

PCM data made available by the jack server are read by darkice, capable of producing streams in multiple formats and sending them to a designated streaming server.

Governed by the configuration file shown bellow, darkice connects its input to a designated named port of the jack server (darkice_cro-d-dur_ogg-256), generates an Ogg Vorbis stream with a bitrate of 256 kb/sec, and forwards it to the local icecast server, which makes it available to other icecast servers for broadcasting, or even to individual clients.

```
sleep 1
# Connects the two named ports:
jack_connect mplayer_cro-d-dur:out_0 darkice_cro-d-dur_ogg-256:left
jack_connect mplayer_cro-d-dur:out_1 darkice_cro-d-dur_ogg-256:right
```

Now, the jack server needs to be configured by the commands shown bellow, to connect the port of the input client (mplayer) to that of the producer (darkice).

```
# Fragment of cro-d-dur-256.cfg configuration file
# this section describes general aspects of the live streaming session
[general]
duration = 0 # duration of encoding, in seconds. 0 means forever
bufferSecs = 3 # size of internal slip buffer, in seconds
reconnect = yes # reconnect to the server(s) if disconnected
# this section describes the audio input that will be streamed
[input]
device = jack
jackClientName = darkice_cro-d-dur_ogg-256
sampleRate = 48000
bitsPerSample = 16 # bits per sample. try 16
channel = 2 # channels. 1 = mono, 2 = stereo
# this section describes a streaming connection to an IceCast2 server
[icecast2-0]
bitrateMode = abr # average bit rate
format = vorbis # format of the stream: ogg vorbis
bitrate = 256 # bitrate of the stream sent to the server
server = 127.0.0.1 # host name of the server
port = 8000 # port of the IceCast2 server
password = xxxpasswdxxx # source password to the IceCast2 server
mountPoint = test-256.ogg # mount point of this stream on the IceCast2 server
name = Test # name of the stream
description = Test # description of the stream
url = http://www.rozhlas.cz
```

4.4 Icecast2 server

In a setup with the input client receiving FLAC streams from a local icecast server, reliability and tolerance of the transcoder to an outage of the input stream may be further improved by providing a backup input stream from an alternative source. The icecast server must be configured to regard the secondary stream as fallback.

```
#Fragment of icecast.xml configuration file:
<mount>
<mount-name>/cro-d-dur.flac</mount-name>
<fallback-mount>/cro-d-dur-fallback.flac</fallback-mount>
```

```
<fallback-override>1</fallback-override>
</mount>
```

In case of an outage of the primary stream, the server switches to the fallback option. It switches back to the primary stream as soon as it reappears. There will be no interruptions in streams provided to clients – an occurrence which might cause them to terminate or otherwise fail to reestablish the connection and continue receiving.

5 Conclusion

It was our goal to design and implement a universal transcoder capable of real-time conversion of loss-less FLAC streams to other formats, making them available to other streaming servers or end-user clients.

We have succeeded in implementing a system with an open, modular architecture, whose components may be freely combined or replaced with alternatives. This is very important, especially in the case of output stream producers, whose choice is not limited to a single encoding application. Any program capable of processing the input stream and generating output per specifications may be used as a producer. The implementation of the transcoder relies exclusively on free technologies.

References

- [1] WIMMER, M. *Vysílání a přenos audio signálu ve velmi vysoké kvalitě do sítě Internet – rozvoj projektu*. [In Czech: Broadcasting and Transmission of Audio Signal in a Very High Quality to the Internet – Project Development.] Technical Report 16/2004¹⁰, Praha: CESNET, 2004.
- [2] WIMMER, M. *Vysílání a přenos audio signálu ve velmi vysoké kvalitě*. [In Czech: Broadcasting and Transmission of Audio Signal in a Very High Quality.] Technical Report 28/2005¹¹, Praha: CESNET, 2005.
- [3] WIMMER, M. *Improving Reliability of a Streaming System*. Technical Report 24/2006¹², Praha: CESNET, 2006.
- [4] WIMMER, M. *Broadcasting and Transmitting Audio Signals Using the Lossless FLAC Compression Encoding*. Technical Report 18/2008¹³, Praha: CESNET, 2008.

¹⁰ <http://www.cesnet.cz/doc/techzpravy/2004/audio/>

¹¹ <http://www.cesnet.cz/doc/techzpravy/2005/audio/>

¹² <http://www.cesnet.cz/doc/techzpravy/2006/vps/>

¹³ <http://www.cesnet.cz/doc/techzpravy/2008/using-flac-encoding/>