Open Source

Studio to Transmitter Link

(OSSTL)

Designed and implemented by

Jason Ellison
infotek@gmail.com

http://www.jasonellison.net/

Currently in use by Goforth Media

http://www.goforth.org/

The system described in this document is in production and supplying audio to one commercial FM station and two AM stations.
It has done so without flaw for over a year.

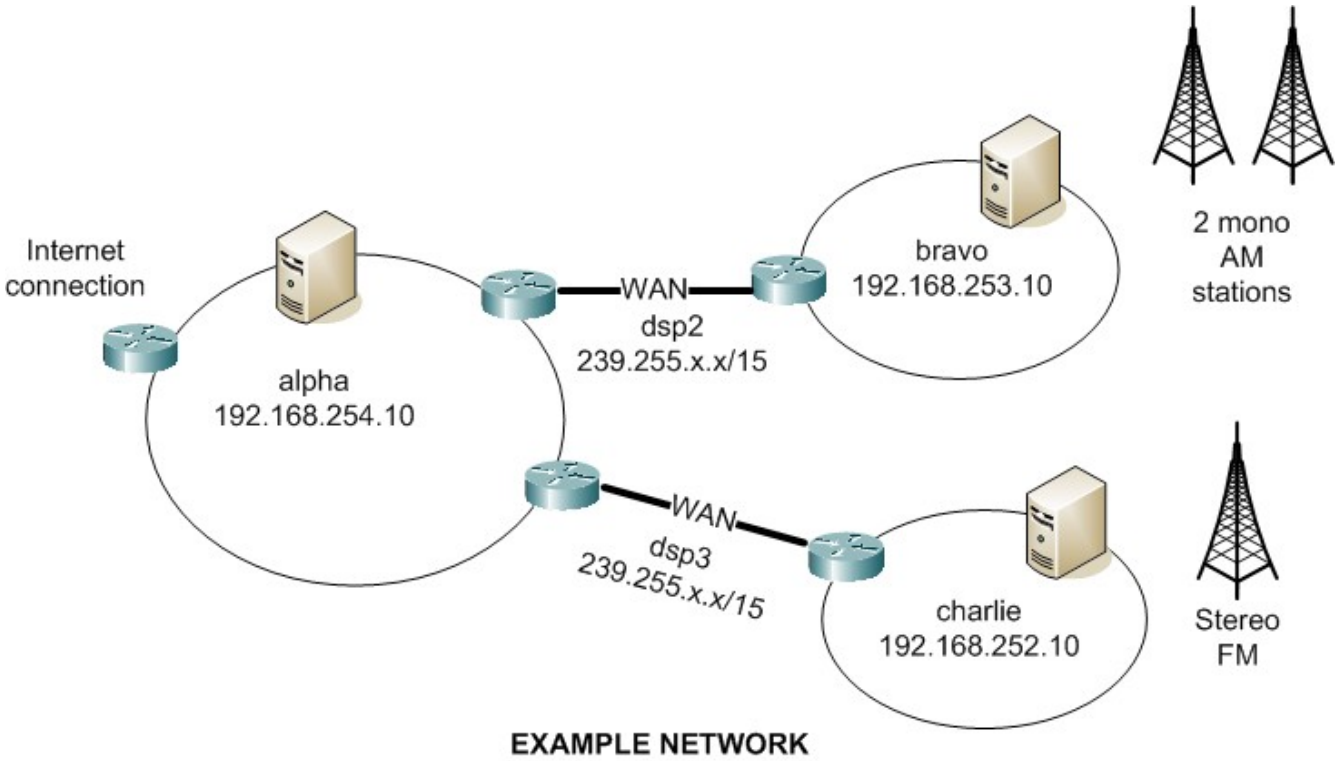
document version 0.0.3

Summary:

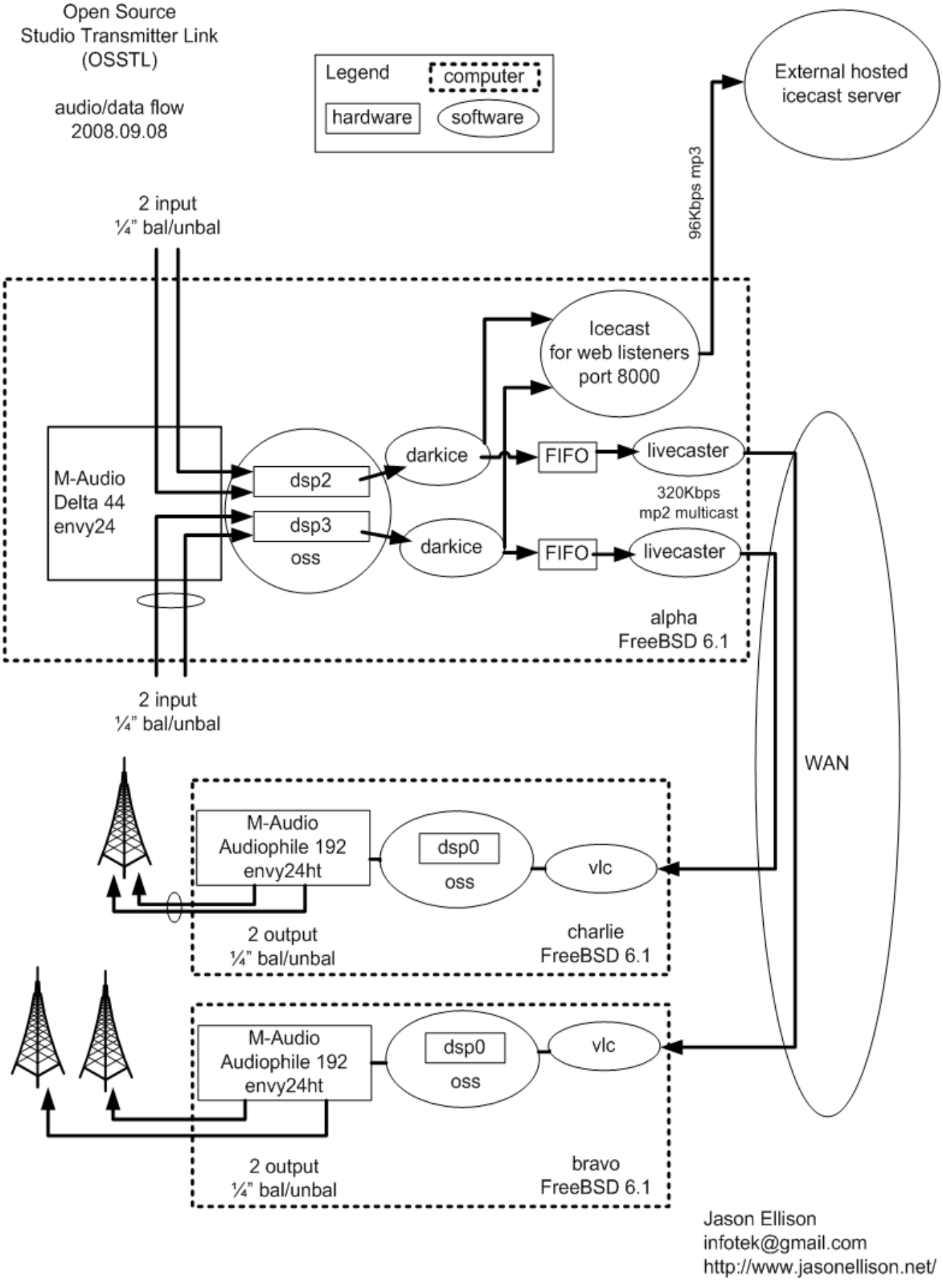
The purpose of this project is to create and document a Studio to Transmitter Link (STL) using (mostly) Open Source software. Typically commercial STL's cost between $5,000.00 and $10,000 per link in hardware costs, plus additional costs for engineering and installation. Using Open Source software, commodity computer hardware and commercial grade audio cards, one can maintain audio quality, reduce cost, and increase flexibility.

- The audio is read in from the microphone jacks on machine named alpha.
- Darkice encodes the audio into mp3 format and mp2 format. The mp3 data is send to icecast for weblisteners while the mp2 data is written to a FIFO for livecaster to read.
- livecaster reads the mp2 encoded audio and sends it over the network in multicast packets. The livecaster daemon reads a file named SELF.txt that configures how it sends the data.
- The machines named bravo and charlie receive the multicast audio data and play the audio using the local sound cards. vlc reads a file named *.sdp which like SELF.txt tells vlc what multicast stream to subscribe to.
- The local iceast server steams the mp3 data to a remote icecast server which has more bandwidth. The remote icecast server services a large number of clients over the internet who listen to the “internet radio”.

The livecaster daemon encodes the audio at a rate of 320Kbps. I found that lower settings introduced unwanted audio artifacts if the input data was previously compressed (i.e. mpeg encoded satellite feeds). Each of the servers have a webpage written in php that allows the operator to control the various daemons and to watch the audio levels of the soundcard.



EXAMPLE NETWORK



Jason Ellison
infotek@gmail.com
http://www.jasonellison.net/

Todo:

- Add multicast support directly to darkice using liveMedia. Remove dependence on liveCaster the closed source binary.
- Test FreeBSD 7.x with M-Audio soundcards. Remove dependence on commercial OpenSound Sytem (OSS) by 4Front technologies.

Project requirements:

- This setup is intended for LPFM/community radio Studio to Transmitter Link (STL). To connect a studio to a remote transmitter site using (mostly) open source components.
- Machines alpha bravo and charlie will be connected via 1.5Mbps network to simulate wireless/T1 and other low bandwidth links.
- alpha will have a professional quality soundcard M-Audio Audiophile 192 supported by oss, Open Source support N/A at this time.
- alpha will make CD quality sound available to remote machine via the low bandwidth link in a reliable fashion.
- alpha will receive audio on a soundcard "MIC" input.
- bravo and charlie shall play audio from alpha's "MIC" with consistent quality and latency.
- alpha should also make audio available to Internet radio listeners.

Hardware:

Encoder: alpha, encoding 4 channels of audio into MPEG2 plus one stereo into MPEG2 Layer 3 for the icecast stream.
2GHz CPU 512MB RAM
M-Audio Delta 44 (4 bal/unbal inputs)

Player: bravo and charlie, decoding a two channel MPEG2 stream.
800MHz CPU 128MB RAM
M-Audio Audiophile 192 (2 bal/unbal outputs)

Software:

The following Open Source software was used in this project FreeBSD 6.1, darkice 0.18, livecaster 2006 release, icecast 2.3.1, lame 3.96.1, twolame 0.3.7 and vlc 0.8.5. The sound drivers used were Open Sound System (OSS) 4.0, a commercial offering by 4Front technologies. In FreeBSD 6.1 there was no open source offering for the M-Audio sound cards. This has since changed in FreeBSD 7.0 via envy24 and envy24ht audio drivers, although I have not tested this. While liveCaster uses an Open Source library, the source code for the 'lc' binary used in this project is not available. It would be advantageous to add multicasting directly to Darkice using the same liveMedia library which liveCaster uses.

FreeBSD 6.1-RELEASE
http://www.freebsd.org/

FreeBSD is a stable and reliable server platform.

darkice-0.18
http://darkice.tyrell.hu/

Darkice reads audio from sound card (M Audio Delta 44) and encodes it into MPEG2 layer 3 format for the icecast server and MPEG2 layer 2 for a FIFO which is read by liveCaster.

liveCaster
http://www.live555.com/liveCaster/

liveCaster reads audio from a FIFO, encapsulates it in a muticast packet and delivers it to the network.

icecast-2.3.1
http://www.icecast.org/

Shares data with 3rd party hosted icecast server... this is for the internet radio.

vlc-0.8.5
http://www.videolan.org/vlc/

This software is used to play the multicast stream to the sound card (M-Audio Audiophile 192).

4Front Open Sound System (oss) 4.x
http://www.opensound.com/

These sound drivers were used because FreeBSD 6.1 did not support M-Audio sound cards. FreeBSD 7.0 support the card via the envy24 and envy24ht drivers, negating the need for this non-free commercial sound driver. The envy24 and envy24ht drivers need to be tested for compatibility with various M-Audio cards and quality.

M-Audio
http://www.m-audio.com/

Delta 44 - Professional 4-In/4-Out Audio Card
http://www.m-audio.com/products/en_us/Delta44.html

Audiophile 192 - High-Definition 4-In/4-Out Audio Card with Digital I/O and MIDI
http://www.m-audio.com/products/en_us/Audiophile192.html

lame-3.96.1
http://lame.sourceforge.net/

LAME is a high quality MPEG Audio Layer III (MP3) encoder. LAME is used to encode the audio into the standard mp3 format sent to Internet radio listeners.

twolame-0.3.7
http://www.twolame.org/

TwoLAME is an optimised MPEG Audio Layer 2 (MP2) encoder. TwoLAME is used to encode the audio stream being sent to the transmitter sites.

Controlling OSSTL

Remote access to FreBSD servers

Remote access to the servers is accomplished via ssh. Accessing the STL system via ssh from a windows operating system can be done using the program putty, available from http://www.chiark.greenend.org.uk/~sgtatham/putty/. After logging in you will need to 'su' to root.

The order in which services should start on alpha should be: darkice, livecaster, icecast. The order of shutdown on alpha should be: icecast, livecaster, darkice. Because darkice and livecaster share data via a FIFO... one may quit if the other is killed. Use 'ps axw' command to verify that each dsp has both a darkice process and an lc process associated with it. If you do not see both processes you will need to restart the one that is not running.

Below is an example response in which both processes that are required to be running to service a dsp are running:

```
alpha# ps axw | grep dsp2

80037  p0- S      2:01.26 /usr/local/bin/lc -d /usr/local/etc/dlc/dsp2/320/ -s
/usr/local/etc/dlc/dsp2/320/stream.fifo
80044  p0- S     51:12.97 /usr/local/bin/darkice -v 5 -c
/usr/local/etc/dlc/dsp2/320/darkice.cfg

# ps axw | grep dsp3

80060  p0- R      2:04.94 /usr/local/bin/lc -d /usr/local/etc/dlc/dsp3/320/ -s
/usr/local/etc/dlc/dsp3/320/stream.fifo
80067  p0- S    127:29.29 /usr/local/bin/darkice -v 5 -c
/usr/local/etc/dlc/dsp3/320/darkice.cfg
```

Determining output volume levels on bravo and charlie from the shell interface .

```
bravo# /usr/bin/ossmix | grep peak.main
peak.main <leftVU>:<rightVU>] (currently 0:117)
```

Determining input audio levels on alpha from the shell.

```
alpha# ossmix | grep peak.in
peak.in1/2 <leftVU>:<rightVU>] (currently 0:129)
peak.in3/4 <leftVU>:<rightVU>] (currently 125:123)
```

Darkice : audio encoding on alpha

```
Usage: /usr/local/etc/rc.d/darkice-a [start|stop]

Usage: /usr/local/etc/rc.d/darkice-b [start|stop]
```

stopping a darkice instance

```
 /usr/local/etc/rc.d/darkice-a stop
```

starting a darkice instance

```
 /usr/local/etc/rc.d/darkice-a start
```

liveCaster : multicast encapsulation on alpha

livecaster relies on an SELF.txt file that can be created using liveCaster.

```
Usage: /usr/local/etc/rc.d/livecaster-a [start|stop]

Usage: /usr/local/etc/rc.d/livecaster-b [start|stop]
```

stopping a liveCaster instance

```
/usr/local/etc/rc.d/livecaster-a stop
```

starting a liveCaster instance

```
/usr/local/etc/rc.d/livecaster-a start
```

vlc : playing the audio at the transmitter site on bravo and charlie

vlc relies on an sdp file that can be created using liveCaster.

```
Usage: /usr/local/etc/rc.d/020-vlc.sh [start|stop] (BPS)
  BPS is optional. Values may be 128, 160, 192, 256 or 320
```

starting vlc player on bravo or charlie

```
/usr/local/etc/rc.d/020-vlc.sh start
```

stopping vlc player on bravo or charlie

```
/usr/local/etc/rc.d/020-vlc.sh stop
```

Icecast2 : Internet radio listeners

```
Usage: /usr/local/etc/rc.d/icecast2 [fast|force|one](start|stop|restart|
rcvar|status|poll)
```

starting the icecast server on alpha

```
/usr/local/etc/rc.d/icecast2 start
```

stopping the icecast server on alpha

```
/usr/local/etc/rc.d/icecast2 stop
```

Icecast may also has a web interface available at http://server:8000/. Replace server with the devices internal or external IP address.

Performance Tuning

Increasing caching on vlc

Increasing the udp caching of the vlc program at the remote computers makes the system more resilient in networks that exhibit variable latency or network congestion (i.e. wireless 802.11). Another side effect of increasing the udp caching is the increase in the audio latency of the system. Meaning it will take longer for sound fed into alpha to play out of charlie or bravo.

/usr/local/etc/rc.d/020-vlc.sh

```
/usr/local/bin/vlc --daemon \
  --no-video \
  --audio --aout oss --aout-rate 44100 --dspdev /dev/dsp \
  --no-oss-buggy --rt-priority --loop \
  --udp-caching 1000 \
  sdp:///usr/local/etc/livecaster/320bps
```

Increase resilience to packet loss

Increase the multicast stream's resiliency to packet loss using the “ADU Frame Interleaving” feature of livecaster. . Configure darkice to deliver mp3 data to the FIFO for livecaster. Configure livecaster to use interleave the data with the '-a 96 “a0 a1 a2 a3...” option as described in the advanced options section of http://www.live555.com/liveCaster/gui.html.

RFC 5219 A More Loss-Tolerant RTP Payload Format for MP3 Audio
http://www.rfc-editor.org/rfc/rfc5219.txt

Configuration Files

/usr/local/etc/rc.d/darkice-a

```
#!/bin/sh
# darkice and livecaster

# PROVIDE: darkice-a
# REQUIRE: DAEMON oss icecast2

. /etc/rc.subr

name="darkice-a"
stop_cmd="darkice_stop"
start_cmd="darkice_start"
pidfile=/var/run/$name

#BPS=${2:-320}
#DSP=${2:-dsp2}

BPS=320
DSP=dsp2

darkice_start(){
  #darkice
  echo -e "\nstarting: $name"
  /usr/local/bin/darkice -v 5 \
    -c /usr/local/etc/dlc/$DSP/$BPS/darkice.cfg &
  echo $! > $pidfile
}
darkice_stop(){
  echo -e "\nstopping: $name"
  kill `cat $pidfile`
  rm -f $pidfile

}

darkice_restart(){
  darkice_stop;
  darkice_start;
}
rc_usage(){
  echo "Usage: $0 [start|stop]"
}

load_rc_config $name
run_rc_command "$1"
```

/usr/local/etc/rc.d/darkice-b

```
#!/bin/sh
# darkice and livecaster

# PROVIDE: darkice-b
# REQUIRE: DAEMON oss icecast2

. /etc/rc.subr

name="darkice-b"
stop_cmd="darkice_stop"
start_cmd="darkice_start"
pidfile=/var/run/$name

#BPS=${2:-320}
#DSP=${2:-dsp3}

BPS=320
DSP=dsp3

darkice_start(){
  #darkice
  echo -e "\nstarting: $name"
  /usr/local/bin/darkice -v 5 \
    -c /usr/local/etc/dlc/$DSP/$BPS/darkice.cfg &
  echo $! > $pidfile
}
darkice_stop(){
  echo -e "\nstopping: $name"
  kill `cat $pidfile`
  # 2 > /dev/null
  rm -f $pidfile

}

darkice_restart(){
  darkice_stop;
  darkice_start;
}
rc_usage(){
  echo "Usage: $0 [start|stop]"
}

load_rc_config $name
run_rc_command "$1"
```

/usr/local/etc/rc.d/livecaster-a

```
#!/bin/sh
# livecaster

# PROVIDE: livecaster-a
# REQUIRE: DAEMON oss darkice-a

. /etc/rc.subr

name="livecaster-a"
stop_cmd="livecaster_stop"
start_cmd="livecaster_start"
pidfile=/var/run/$name

#BPS=${2:-320}
#DSP=${2:-dsp2}
BPS=320
DSP=dsp2

livecaster_start(){
  #livecaster
  sleep 1
  echo "starting: $name"
  /usr/local/bin/lc \
    -d /usr/local/etc/dlc/$DSP/$BPS/ \
    -s /usr/local/etc/dlc/$DSP/$BPS/stream.fifo &
  echo $! > $pidfile
}
livecaster_stop(){
  echo "stopping: $name"
  kill `cat $pidfile` 2 > /dev/null
  rm -f $pidfile
}
livecaster_restart(){
  livecaster_stop
  sleep 1
  livecaster_start
}
rc_usage(){
  echo "Usage: $0 [start|stop]"
}

load_rc_config $name
run_rc_command "$1"
```

/usr/local/etc/rc.d/livecaster-b

```
#!/bin/sh
# livecaster

# PROVIDE: livecaster-b
# REQUIRE: DAEMON oss darkice-b

. /etc/rc.subr

name="livecaster-b"
stop_cmd="livecaster_stop"
start_cmd="livecaster_start"
pidfile=/var/run/$name

#BPS=${2:-320}
#DSP=${2:-dsp3}
BPS=320
DSP=dsp3

livecaster_start(){
  #livecaster
  sleep 1
  echo "starting: $name"
  /usr/local/bin/lc \
    -d /usr/local/etc/dlc/$DSP/$BPS/ \
    -s /usr/local/etc/dlc/$DSP/$BPS/stream.fifo &
  echo $! > $pidfile
}
livecaster_stop(){
  echo "stopping: $name"
  kill `cat $pidfile` 2 > /dev/null
  rm -f $pidfile
}
livecaster_restart(){
  livecaster_stop
  sleep 1
  livecaster_start
}
rc_usage(){
  echo "Usage: $0 [start|stop]"
}

load_rc_config $name
run_rc_command "$1"
```

/usr/local/etc/dlc/dsp2/320/darkice.cfg

```
[general]
duration        = 0        # duration of encoding, in seconds. 0 means forever
bufferSecs      = 1         # size of internal slip buffer, in seconds
reconnect       = yes       # reconnect to the server(s) if disconnected
realtime        = yes

[input]
device          = /dev/dsp2  # OSS DSP soundcard device for the audio input
sampleRate      = 44100     # sample rate in Hz. try 11025, 22050 or 44100
bitsPerSample   = 16        # bits per sample. try 16
channel         = 2         # channels. 1 = mono, 2 = stereo

[icecast2-0]
bitrateMode     = cbr
format          = mp3
bitrate         = 128
quality         = 0.5
server          = localhost
port            = 8000
password        = $3cr3t
mountPoint      = dsp2-128
name            = LPFM AM stations
description     = dsp2 44100 samples/sec 16bit 2 channels 128Kb/s bandwidth
url             = http://www.example.org/
genre           = none
public          = yes

[file-0]
bitrateMode     = cbr
format          = mp2
bitrate         = 320
quality         = 0.8
fileName        = /usr/local/etc/dlc/dsp2/320/stream.fifo
samplerate      = 44100
highpass        = 25000
lowpass         = 10
```

/usr/local/etc/dlc/dsp2/320/SELF.txt

```
info    dsp2-320bps MP3 audio
bwLimit 320
progId  P:10.0.0.109:1172752654
nickname        dsp2-320bps
outputMode      audio
SDPdir
GroupEId        239.255.7.113 54010 {15 nokey}
tunnelState     0 {} 10100 10100 1
```

/usr/local/etc/dlc/dsp2/320/dsp2-320bps.sdp

```
v=0
o=- 1172752654 3429764632 IN IP4 10.0.0.109
s=dsp2-320bps
i=dsp2-320bps MP3 audio
b=AS:320
t=3429764632 3429766252
a=type:broadcast
a=tool:lc v1.0b1
m=audio 54010 RTP/AVP 14
c=IN IP4 239.255.7.113/15
```

/usr/local/etc/dlc/dsp3/320/darkice.cfg

```
[general]
duration        = 0        # duration of encoding, in seconds. 0 means forever
bufferSecs      = 1         # size of internal slip buffer, in seconds
reconnect       = yes       # reconnect to the server(s) if disconnected
realtime        = yes

[input]
device          = /dev/dsp3  # OSS DSP soundcard device for the audio input
sampleRate      = 44100     # sample rate in Hz. try 11025, 22050 or 44100
bitsPerSample   = 16        # bits per sample. try 16
channel         = 2         # channels. 1 = mono, 2 = stereo

[icecast2-0]
bitrateMode     = cbr
format          = mp3
bitrate         = 128
quality         = 0.5
server          = localhost
port            = 8000
password        = $3cr3t
mountPoint      = power88fm.mp3
name            = Power88 FM - Radio With A Vision
description     = (C)2007 Goforth Media
url             = http://www.goforth.org/
genre           = Christian
public          = yes

[icecast2-1]
bitrateMode     = cbr
format          = mp3
bitrate         = 96
quality         = 0.5
server          = localhost
port            = 8000
password        = $3cr3t
mountPoint      = power88fm-96.mp3
name            = Power88 FM - Radio With A Vision
description     = (C)2007 Goforth Media
url             = http://www.goforth.org/
genre           = Christian
public          = yes

[icecast2-2]
bitrateMode     = cbr
format          = mp3
bitrate         = 64
quality         = 0.5
server          = localhost
port            = 8000
password        = $3cr3t
mountPoint      = power88fm-64.mp3
name            = Power88 FM - Radio With A Vision
description     = (C)2007 Goforth Media
url             = http://www.goforth.org/
genre           = Christian
public          = yes

[icecast2-3]
bitrateMode     = cbr
format          = mp3
bitrate         = 32
quality         = 0.5
server          = localhost
port            = 8000
password        = $3cr3t
mountPoint      = power88fm-32.mp3
name            = Power88 FM - Radio With A Vision
description     = (C)2007 Goforth Media
url             = http://www.goforth.org/
genre           = Christian
public          = yes
```

```
[file-0]
bitrateMode     = cbr
format          = mp2
bitrate         = 320
quality         = 0.8
fileName        = /usr/local/etc/dlc/dsp3/320/stream.fifo
samplerate      = 44100
highpass        = 25000
lowpass         = 10
```

/usr/local/etc/dlc/dsp3/320/SELF.txt

```
bwLimit 320
info    dsp3-320bps MP3 audio
progId  P:10.0.0.109:1172752749
SDPdir  local-default
outputMode      audio
nickname        dsp3-320bps
tunnelState     0 {} 10100 10100 1
GroupEId        239.255.149.81 56910 {15 nokey}
```

/usr/local/etc/dlc/dsp3/320/dsp3-320bps.sdp

```
v=0
o=- 1172752749 3429764669 IN IP4 10.0.0.109
s=dsp3-320bps
i=dsp3-320bps MP3 audio
b=AS:320
t=3429764669 3429766289
a=type:broadcast
a=tool:lc v1.0b1
m=audio 56910 RTP/AVP 14
c=IN IP4 239.255.149.81/15
```

/usr/local/etc/rc.d/020-vlc.sh

```
#!/bin/sh
# vlc
echo -n "$0 : "

BPS=${2:-320}

start(){
  echo "Starting. BPS is $BPS."

  ossmix vmix0-src Production ;
  ossmix envy24.rate 44100 ;
  ossmix envy24.ratelock on ;

  /usr/local/bin/vlc --daemon \
    --no-video \
    --audio --aout oss --aout-rate 44100 --dspdev /dev/dsp \
    --no-oss-buggy --rt-priority --loop \
    --udp-caching 1000 \
    sdp:///usr/local/etc/livecaster/320bps
}
stop(){
  echo "Stoping"
  killall vlc
  sleep 1
}
usage(){
  echo "missing parameter"
  echo "Usage: $0 [start|stop] (BPS)"
  echo "  BPS is optional. Values may be 128, 160, 192, 256 or 320"
}
case "$1" in
  start)
    start "$2" ;
  ;;
 stop)
   stop ;
  ;;
 *)
   usage ;
  ;;
esac
```

/usr/local/etc/livecaster/320bps/dsp3-320bps.sdp

```
v=0
o=- 1172752749 3381741714 IN IP4 10.0.0.109
s=dsp3-320bps
i=dsp3-320bps MP3 audio
b=AS:320
t=3381743230 3381744850
a=type:broadcast
a=tool:liveCaster v1.0b1
m=audio 56910 RTP/AVP 14
c=IN IP4 239.255.149.81/15
```

/usr/local/etc/livecaster/320bps/dsp2-320bps.sdp

```
v=0
o=- 1172752654 3381741506 IN IP4 10.0.0.109
s=dsp2-320bps
i=dsp2-320bps MP3 audio
b=AS:320
t=3381741506 3381743126
a=type:broadcast
a=tool:liveCaster v1.0b1
m=audio 54010 RTP/AVP 14
c=IN IP4 239.255.7.113/15
```

/usr/local/etc/icecast.xml

```
<icecast>
    <limits>
        <clients>10</clients>
        <sources>5</sources>
        <threadpool>5</threadpool>
        <queue-size>524288</queue-size>
        <client-timeout>30</client-timeout>
        <header-timeout>15</header-timeout>
        <source-timeout>10</source-timeout>
        <burst-on-connect>1</burst-on-connect>
        <burst-size>65535</burst-size>
    </limits>
    <authentication>
        <source-password>$3cr3t</source-password>
        <relay-password>$3cr3t</relay-password>
        <admin-user>admin</admin-user>
        <admin-password>$3cr3t</admin-password>
    </authentication>
    <hostname>localhost</hostname>
     <listen-socket>
        <port>8000</port>
     </listen-socket>
     <fileserve>1</fileserve>
    <paths>
        <basedir>/usr/local/share/icecast</basedir>
        <logdir>/usr/local/var/log/icecast</logdir>
        <webroot>/usr/local/share/icecast/web</webroot>
        <adminroot>/usr/local/share/icecast/admin</adminroot>
        <alias source="/" dest="/status.xsl"/>
    </paths>
    <logging>
        <accesslog>access.log</accesslog>
        <errorlog>error.log</errorlog>
        <loglevel>3</loglevel>
        <logsize>10000</logsize>
    </logging>
    <security>
        <chroot>0</chroot>
        <changeowner>
            <user>nobody</user>
            <group>nogroup</group>
        </changeowner>
    </security>
</icecast>
```

Web interface for sending server

/usr/local/www/data/index.php

```
<?
extract($_GET);
// darkice-a
if($cmd == 'stop_darkicea')
{
        passthru('/usr/local/bin/sudo /usr/local/etc/rc.d/darkice-a stop');
}
if($cmd == 'start_darkicea')
{
        passthru('/usr/local/bin/sudo /usr/local/etc/rc.d/darkice-a start');
}
// darkice-b
if($cmd == 'stop_darkiceb')
{ passthru('/usr/local/bin/sudo /usr/local/etc/rc.d/darkice-b stop'); }
if($cmd == 'start_darkiceb')
{ passthru('/usr/local/bin/sudo /usr/local/etc/rc.d/darkice-b start'); }
?>
<HTML>
<HEAD>
<TITLE>HOSTNAME Sending - Audio Control Panel</TITLE>
<link rel="stylesheet" href="site.css" type="text/css">
<script src="mootools.js" type="text/javascript"></script>
<script type="text/javascript">
  function dsp2()
{
    new Ajax('/cgi-bin/dsp2.sh', {method: 'get', update: 'dsp2'}).request.periodical(700);
}
  function dsp3()
{
     new Ajax('/cgi-bin/dsp3.sh', {method: 'get', update: 'dsp3'}).request.periodical(750);
}
</script>
</HEAD>
<BODY onload="dsp2();dsp3();">
<DIV id="main">
  <H2>Alpha</A>
  <p>AJAX test interface for displaying audio levels.  My foray into AJAX land..
. thanks for the information on mootools Lumis.
  <p>This small webpage was written to quick hack to intergrate audio numerical
data in a format similar to audio equipment.  This is intended to be intergrated
 into an opensource Studio to Transmitter Link (STL).  Enabling the user to see
the status of the audio input and output.
  <DIV id="dsp2">
    <TABLE>
    <TR>
      <TH> L <TD class=grey>   <TD class=grey>   <TD class=grey>  
; <TD class=grey>   <TD class=grey>   <TD class=grey>   <TD class
=grey>   <TD class=grey>   <TD class=grey>  
    <TR>
      <TH> R <TD class=grey>   <TD class=grey>   <TD class=grey>  
; <TD class=grey>   <TD class=grey>   <TD class=grey>   <TD class
=grey>   <TD class=grey>   <TD class=grey>  
    </TABLE>
  </DIV>
<a href="index.php?cmd=stop_darkicea">Stop Darkice-A</a> |
<a href="index.php?cmd=start_darkicea">Start Darkice-A</a>
<hr>
  <DIV id="dsp3">
    <TABLE>
    <TR>
      <TH> L <TD class=grey>   <TD class=grey>   <TD class=grey>  
; <TD class=grey>   <TD class=grey>   <TD class=grey>   <TD class
=grey>   <TD class=grey>   <TD class=grey>  
    <TR>
      <TH> R <TD class=grey>   <TD class=grey>   <TD class=grey>  
; <TD class=grey>   <TD class=grey>   <TD class=grey>   <TD class
=grey>   <TD class=grey>   <TD class=grey>  
    </TABLE>
  </DIV>
<a href="index.php?cmd=stop_darkiceb">Stop Darkice-B</a> |
<a href="index.php?cmd=start_darkiceb">Start Darkice-B</a>
```

```
  <p>The code is available here <a href="audio_level.tgz">audio_level.tgz</a>.
  <p>The code is available here <a href="audio_level.zip">audio_level.zip</a>.
</DIV>
```

/usr/local/www/cgi-bin/dsp2.sh

```
#!/usr/local/bin/bash
# bar.sh
# generate two random numbers to simulate left and right audio levels
# create html to represent this as an audio level indicator using tables.
DSP2=$( /usr/bin/ossmix peak.in1/2 | cut -d " " -f 10 )
echo DSP2 = $DSP2
left=`echo $DSP2 | cut -d ':' -f 1`
right=`echo $DSP2 | cut -d ':' -f 2`
echo "Content-type: text/html"
echo ""
echo "    <TABLE>"
echo "    <TR>"
echo -n "      <TH>AM-1</TH><TH WIDHT="50"> $left </TH>"
for (( i=1 ; i<=12 ; i++ )); do
  if (( $left >= $(($i*10)) )); then
    #color
    if (( $i <= 2 )); then
      echo -n "<TD class="yellow">   "
    elif (( $i >= 9 )); then
      echo -n "<TD class="red">   "
    else
      echo -n "<TD class="green">   "
    fi
  else
    echo -n "<TD class="grey">   "
  fi
  if (( $i == 12 )) ; then echo " "; fi
done
echo "    <TR>"
echo -n "      <TH>AM-2</TH><TH WIDHT="50"> $right</TH>"
for (( i=1 ; i<=12 ; i++ )); do
  if (( $right >= $(($i*10)) )); then
    #color
    if (( $i <= 2 )); then
      echo -n "<TD class="yellow">   "
    elif (( $i >= 9 )); then
      echo -n "<TD class="red">   "
    else
      echo -n "<TD class="green">   "
    fi
  else
    echo -n "<TD class="grey">   "
  fi
  if (( $i == 12 )) ; then echo " "; fi
done
echo "    </TABLE>"
```

/usr/local/www/cgi-bin/dsp3.sh

```
#!/usr/local/bin/bash
# bar.sh
# generate two random numbers to simulate left and right audio levels
# create html to represent this as an audio level indicator using tables.
DSP2=$( /usr/bin/ossmix peak.in3/4 | cut -d " " -f 10 )
#echo DSP2 = $DSP2
left=`echo $DSP2 | cut -d ':' -f 1`
right=`echo $DSP2 | cut -d ':' -f 2`
echo "Content-type: text/html"
echo ""
echo "    <TABLE>"
echo "    <TR>"
echo -n "      <TH>FM-L</TH><TH WIDTH="27">$left</TH>"
for (( i=1 ; i<=11 ; i++ )); do
  if (( $left >= $(($i*10)) )); then
    #color
    if (( $i <= 3 )); then
```

```
      echo -n "<TD class="yellow">   "
    elif (( $i >= 11 )); then
      echo -n "<TD class="red">   "
    else
      echo -n "<TD class="green">   "
    fi

  else
    echo -n "<TD class="grey">   "
  fi
  if (( $i == 11 )) ; then echo " "; fi
done
echo "    <TR>"
echo -n "      <TH>FM-R</TH><TH WIDTH="27">$right</TH>"
for (( i=1 ; i<=11 ; i++ )); do
  if (( $right >= $(($i*10)) )); then
    #color
    if (( $i <= 3 )); then
      echo -n "<TD class="yellow">   "
    elif (( $i >= 11 )); then
      echo -n "<TD class="red">   "
    else
      echo -n "<TD class="green">   "
    fi
  else
    echo -n "<TD class="grey">   "
  fi
  if (( $i == 11 )) ; then echo " "; fi
done
echo "    </TABLE>"
```

Web interface for receiving server

/usr/local/www/data/index.php

```
<?
extract($_GET);

// darkice-a
if($cmd == 'stop_player')
{
        passthru('/usr/local/bin/sudo /usr/local/etc/rc.d/020-vlc.sh stop');
}

if($cmd == 'start_player')
{
        passthru('/usr/local/bin/sudo /usr/local/etc/rc.d/020-vlc.sh start');
}

?>

<HTML>
<HEAD>
<TITLE>HOSTNAME Receiving - Audio Control Panel</TITLE>
<link rel="stylesheet" href="site.css" type="text/css">
<script src="mootools.js" type="text/javascript"></script>
<script type="text/javascript">
  function dsp()
{
    new Ajax('/cgi-bin/dsp.sh', {method: 'get', update: 'dsp'}).request.periodical(700);
}
</script>
</HEAD>
<BODY onload="dsp();">

<DIV id="main">

 <H2>Charlie</H2>

  <p>AJAX test interface for displaying audio levels.  My foray into AJAX land..
. thanks for the information on mootools Lumis.

  <p>This small webpage was written to quick hack to intergrate audio numerical
data in a format similar to audio equipment.  This is intended to be intergrated
 into an opensource Studio to Transmitter Link (STL).  Enabling the user to see
the status of the audio input and output.

  <DIV id="dsp">

    <TABLE>
    <TR>
      <TH> L <TD class=grey>   <TD class=grey>   <TD class=grey>  
; <TD class=grey>   <TD class=grey>   <TD class=grey>   <TD class
=grey>   <TD class=grey>   <TD class=grey>  
    <TR>
      <TH> R <TD class=grey>   <TD class=grey>   <TD class=grey>  
; <TD class=grey>   <TD class=grey>   <TD class=grey>   <TD class
=grey>   <TD class=grey>   <TD class=grey>  
    </TABLE>

  </DIV>

<a href="index.php?cmd=stop_player">Stop reciever</a>
<a href="index.php?cmd=start_player">Start reciever</a>


<hr>

  <p>The code is available here <a href="audio_level.tgz">audio_level.tgz</a>.
  <p>The code is available here <a href="audio_level.zip">audio_level.zip</a>.

</DIV>
```

/usr/local/www/cgi-bin/dsp.sh

```
#!/usr/local/bin/bash
# bar.sh
# generate two random numbers to simulate left and right audio levels
# create html to represent this as an audio level indicator using tables.
DSP2=$( /usr/bin/ossmix peak.main | cut -d " " -f 10 )
#echo DSP2 = $DSP2

left=`echo $DSP2 | cut -d ':' -f 1`
right=`echo $DSP2 | cut -d ':' -f 2`

echo "Content-type: text/html"
echo ""

echo "    <TABLE>"
echo "    <TR>"

echo -n "      <TH>FM-L<TD WIDHT="27"> $left </TD>"
for (( i=1 ; i<=12 ; i++ )); do
  if (( $left >= $(($i*10)) )); then

    #color
    if (( $i <= 2 )); then
      echo -n "<TD class="yellow">   "
    elif (( $i >= 9 )); then
      echo -n "<TD class="red">   "
    else
      echo -n "<TD class="green">   "
    fi

  else
    echo -n "<TD class="grey">   "
  fi
  if (( $i == 12 )) ; then echo " "; fi
done

echo "    <TR>"

echo -n "      <TH>FM-R<TD WIDHT="27"> $right</TD>"
for (( i=1 ; i<=12 ; i++ )); do
  if (( $right >= $(($i*10)) )); then

    #color
    if (( $i <= 2 )); then
      echo -n "<TD class="yellow">   "
    elif (( $i >= 9 )); then
      echo -n "<TD class="red">   "
    else
      echo -n "<TD class="green">   "
    fi

  else
    echo -n "<TD class="grey">   "
  fi
  if (( $i == 12 )) ; then echo " "; fi
done

echo "    </TABLE>"
```

References:

Similar attempts at an Open Source (STL)

prometheusradio.org STL Tools
http://flow-stl.sourceforge.net/
Now defunct. Andy Gunn andy@prometheusradio.org was contacted. Andy Gunn is using and suggesting an all commercial software solution running on a win32 platform.

German Reboot.FM 104.1 MHz Berlin, DE
http://reboot.fm/software/
using ogg icecast2, ices2 and streamswitch
http://reboot.fm/software/playout-system

Disk usage

```
alpha# df -h
Filesystem     Size    Used   Avail Capacity  Mounted on
/dev/ad4s1a    989M     91M    819M    10%    /
devfs          1.0K    1.0K      0B   100%    /dev
/dev/ad4s1e    989M    1.4M    909M     0%    /tmp
/dev/ad4s1f     68G    1.6G     61G     3%    /usr
/dev/ad4s1d    989M     66M    844M     7%    /var
```

Apache logging

Disable access logging in apache. The AJAX interaction is make the access logs get huge very quickly. I commented out the all of the "CustomLog" lines.

Balanced / Unbalanced audio

Balanced audio signals are used in professional audio environments. Balanced audio means the the audio signal is sent on two conductors, 180 degrees out of phase. After the transmission the audio data is converted back into the same phase. If the cable picked up any noise, the process of putting the audio back into phase should make the noise 180 degrees out of phase, which when summed should cancel out. The end effect is that noise introduced to the audio cable is removed from the signal.

http://en.wikipedia.org/wiki/Balanced_audio_connector

sox to generate sounds

sox was used to generate various frequiencies to check the low pass and high pass filters. And genera quality of the various encoding settings.

```
$SOX $SOXOPT $NAME synth $LEN sine $f $FADE $VOL
SOXOPT="-t nul -c 2 -r 44100 -s -w - "
FADE=" fade $FT $LEN $FT"

/src/sox -t nul -c 2 -r 44100 -s -w - testcd_02_23_hz.wav synth 30 sine 23.1 fade
0.05 30 0.05
```

USA's Emergency Broadcast System tone

```
sox -t nul -c 2 -r 44100 -s -w - 960.wav synth 1 sawtooth create 960
sox -t nul -c 2 -r 44100 -s -w - 853.wav synth 1 sawtooth create 853
soxmix 853.wav 960.wav ebs.wav
play ebs.wav
```

Festival notes

Festival was used to generate voices used in testing the system.

```
root@nebula:/usr/local/src/festival# ls festival/lib/voices/english/
don_diphone/  kal_diphone/  rab_diphone/  us2_mbrola/
en1_mbrola/   ked_diphone/  us1_mbrola/   us3_mbrola/

root@nebula:/usr/local/src/festival# ls festival/lib/voices/us/

guy
(voice_cmu_us_awb_arctic_hts)
low
(voice_cmu_us_jmk_arctic_hts)
wargames
(voice_cmu_us_bdl_arctic_hts)

chick
(voice_cmu_us_slt_arctic_hts)


  (voice_rab_diphone) Select voice (Britsh Male)
  (voice_ked_diphone) Select voice (American Male)

(voice_cmu_us_bdl_arctic_hts)

(SayText "one two three Four Five Six Seven Eight Nine Ten")

(SayText "mark 10.")

(voice_cmu_us_bdl_arctic_hts)
(SayText "1.")

two three Four Five Six Seven Eight Nine Ten")

(+ NUM1 NUM2 ...)
   Returns the sum of NUM1 and NUM2 ... An error is given is any argument is not a number.

(getenv VARNAME)
   Returns value of UNIX environment variable VARNAME, or nil if VARNAME is unset.

(setenv VARNAME VALUE)
   Set the UNIX environment variable VARNAME to VALUE.

(voice_reset)
This resets all variables back to acceptable values that may affect voice generation. This
function should always be called at the start of any function defining a voice. In addition
to reseting standard variables the function current_voice_reset will be called. This should
always be set by the voice definition function (even if it does nothing). This allows voice
specific changes to be reset when a new voice is selection. Unfortunately I can't force this
to be used.
```