Reprinted from the

Proceedings of the Linux Symposium

July 23rd–26th, 2008
Ottawa, Ontario
Canada

Audio streaming over Bluetooth

Marcel Holtmann
BlueZ Project
marcel@holtmann.org

Abstract

During the last year the Linux Bluetooth community worked hard to establish a solution for streaming audio using the Bluetooth wireless technology. This solution is based on multiple standards for headset (mono quality) and headphones (stereo/high quality) that have been published by the Bluetooth SIG. It also includes support for remote control and meta data information like song titles, etc.

This includes work on the open source implementation of the Subband-codec which has been highly improved and can now measure up against any commercial implementation of this codec.

Introduction

The initial Bluetooth specification [1] came with support for the Headset profile (HSP) which allowed connecting mono headsets to cellphones and enabled voice playback and capture support. Later specifications introduced the Handsfree profile (HFP) which added support for caller identification and additional status indication to allow better integration with cellular networks. In addition to the mono headset support in HSP, the Bluetooth SIG created the Advanced Audio Distribution Profile (A2DP) to support high quality audio streaming.

During the last three years various attempts have been made to add proper support for all three profiles to the Linux Bluetooth stack. The initial attempt was with the Bluetooth ASLA project [2] and produced a kernel module called *btsco*. It only supported the Headset profile. The second attempt was the PlugZ project which derived from the Bluetooth ALSA project. It collected various attempts for Headset profile (*headsetd*) and A2DP (*a2dpd*). Both daemons came with plugins for the Advanced Linux Sound Architecture (ALSA) [3]. Figure 1 shows the planned architecture

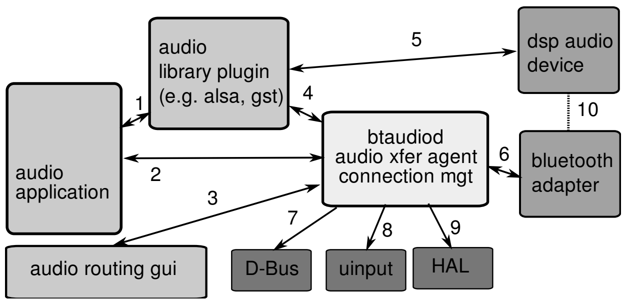


Figure 1: PlugZ architecture

of PlugZ. The support for GStreamer [4] and D-Bus [5] was never finished.

The Bluetooth ALSA project and PlugZ project had various deficiencies. The main problem was that both were too tightly coupled with ALSA. For PlugZ the support for alternate media frameworks was planned, but never integrated since the design was not flexible enough. The other issue was that neither of them were real zero-copy designs. The audio data was always copied two times. This killed the latency and increased the CPU load. The Bluetooth audio service was created to solve the issues of Bluetooth ALSA and PlugZ and fully replace them.

Technical background

The Headset profile and Handsfree profile use an RFCOMM channel as control channel. The protocol of this channel is based on standard AT commands with Bluetooth specific extensions. An example of these extensions is the volume control or the buttons to accept or reject calls. For the transport of the audio stream, the Bluetooth Synchronous Connection Oriented link (SCO) channel is used. The SCO channel is a designated channel within the Bluetooth piconet that allows the transport of 8 kHz Pulse Coded Modulation (PCM) audio over the air that will be encoded using Continuous Variable Slope Delta (CVSD) modulation.

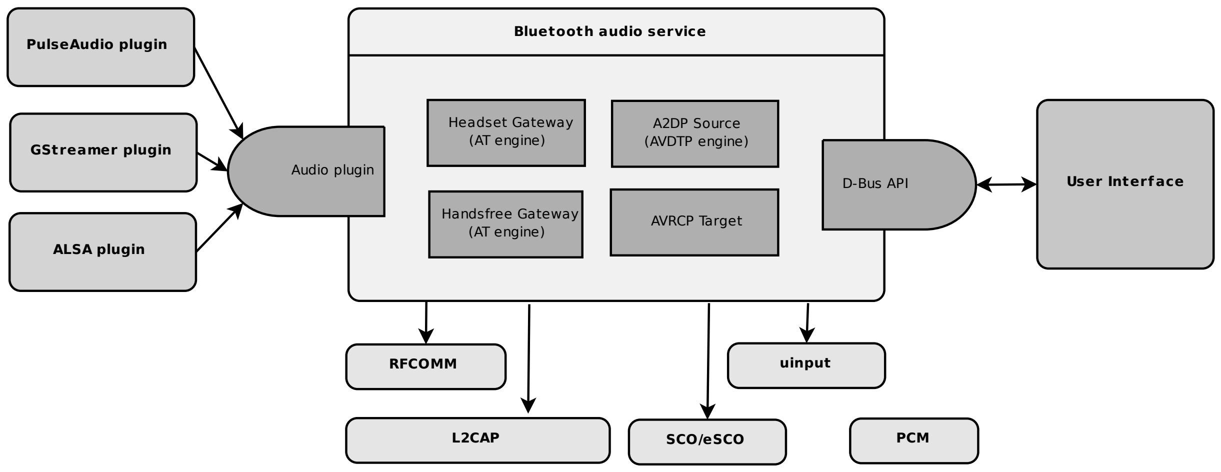


Figure 2: Bluetooth audio architecture

The audio data transferred over the SCO channel can be provided via the normal Host Controller Interface (HCI) hardware driver or via a PCM back-channel. In case of a desktop computer, the HCI will be used. In case of an embedded device (for example a mobile phone), the SCO channel will be directly connected via a PCM interface to the main audio codec.

The Advanced Audio Distribution Profile uses the Logical Link Controller and Adaptation Protocl (L2CAP) and the Audio/Video Distribution Transport Protocol (AVDTP) for control and streaming. This protocol defines a binary control protocol and a binary streaming protocol based on RTP [6]. To compress the audio stream a royalty free Subband-codec (SBC) has been introduced together with the A2DP specification. In addition to the mandatory SBC it is possible for Bluetooth devices to support MP3, ACC or other vendor codes.

Bluetooth audio service

The focus during the Bluetooth audio service development was to fix all the limitations of Bluetooth ALSA and PlugZ and present a flexible infrastructure that could be used for all Bluetooth audio related profiles. The following requirements were identified during the design:

- Treat mono and high quality profiles as equal

  With the Service Discovery Protocol (SDP) it is possible to retrieve the supported profile list from any remote Bluetooth device. Together with the information about the audio stream it is possible to select the correct profile automatically and do the needed conversation transparent for the user.

- Integrate with all multimedia frameworks

  Choosing ALSA as the basic multimedia framework is not the best choice. Actually ALSA is pretty bad when it comes to virtual soundcards and that is what Bluetooth audio is. There is no audio hardware directly attached to the system. All headsets, headphones or speakers are connected via an invisible radio link.

  The frameworks GStreamer and PulseAudio [7] are much better when it comes to handling virtual audio devices. So there is no need to treat them as second class citizens.

- Low-latency and high performance

  In cases where the host has to handle all audio data processing, it should be done the most efficient way and data copying should be avoided at all costs. This increases the performance and at the same time results in good latency. In addition this will reduce the power consumption.

- Full integration with D-Bus

  Provide a full D-Bus interface for control and notifications. It should allow creating, configuring and controlling audio connections.

Integrate with the Audio/Video Remote Control Profile (AVRCP) for handling keys and displays on remote devices.

The current release of BlueZ contains the audio service with support for HSP, HFP, A2DP and AVRCP. It comes with plugins for ALSA and GStreamer. Figure 2 shows the high-level architecture of the audio service.

Subband-codec implementation

During the work on PlugZ and later the Bluetooth audio service, the open source community produced an LGPL licensed implementation of SBC. This implementation has been highly optimized for performance and quality by the Instituto Nokia de Tecnologia (INdT) in Brazil [8].

The source code of the SBC implementation can be found at its Sourceforge project [9] or as part of the BlueZ source code [10].

ALSA support

The ALSA plugin has been created to allow legacy applications to use Bluetooth audio streaming. To make use of a remote headset, headphone or speaker the device has to be configured first. The file *.asoundrc* in the home directory needs to be extended with the lines from Figure 3.

```
pcm.bluetooth {
  type bluetooth
  device 00:11:22:33:44:55
}
```

Figure 3: ALSA configuration entry

This creates a virtual PCM device *bluetooth* which can now be used as if it were a normal soundcard, for example with *aplay -D bluetooth example.wav*.

GStreamer support

With the usage of GStreamer the possibilities become more flexible. The GStreamer framework allows a lot of configuration since everything can be abstracted into elements or containers and then a pipe can be constructed out of them.

The GStreamer plugin that provides access to the Bluetooth audio services consists of multiple elements that can be combined in various ways. Figure 4 shows the details of these elements.

```
# gst-inspect bluetooth
Plugin Details:
  Name:           bluetooth
  Description:    Bluetooth plugin library
  Filename:       libgstbluetooth.so
  Version:        3.30
  License:        LGPL
  Source module:  bluez-utils
  Binary package: BlueZ
  Origin URL:     http://www.bluez.org/

  rtpsbcpay: RTP packet payloader
  a2dpsink:  Bluetooth A2DP sink
  avdtpsink: Bluetooth AVDTP sink
  sbcparse:  Bluetooth SBC parser
  sbcdec:    Bluetooth SBC decoder
  sbcenc:    Bluetooth SBC encoder
bluetooth: sbc: sbc

  7 features:
  +-- 6 elements
  +-- 1 types
```

Figure 4: GStreamer plugin

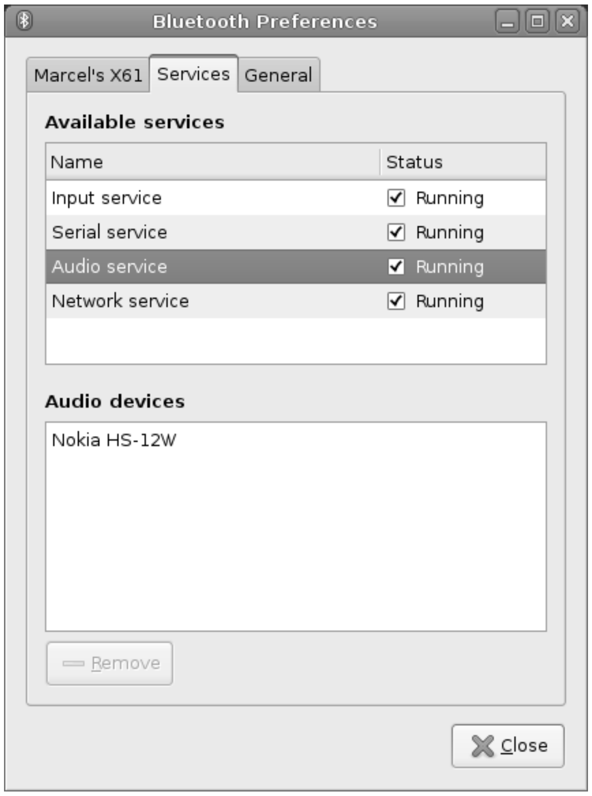
Besides the elements the plugin also contains the definition for the SBC data type. This allows GStreamer enabled applications to load or store files with and SBC encoded audio stream.

The *sbcparse*, *sbcdec* and *sbcenc* elements contain the SBC implementation and besides using them for Bluetooth device, the architecture of GStreamer would allow them to be used within other multimedia applications.

The *a2dpsink* and *avdtpsink* provide an easy abstraction of the inner workings of A2DP and its underlying protocol AVDTP. The *rtpsbcpay* element provides the RTP payload with SBC encoded data. An alternative would be to use an MP3 payloader from a different GStreamer plugin. This however only works if the headset also supports an MP3 encoded stream which is not a mandatory feature.

GNOME integration

The audio service provides an extensive D-Bus API to control the audio devices and to allow user applications easy access. The current integration into the Bluetooth GNOME application has just started. Figure 5 shows the initial integration.



Available services

Name	Status
Input service	✓ Running
Serial service	✓ Running
Audio service	✓ Running
Network service	✓ Running

Audio devices

Nokia HS-12W

Figure 5: GNOME integration

The missing piece is integration with the Bluetooth wizard to provide an easy setup of Bluetooth audio devices.

Future work

Currently work is undergoing to develop the PulseAudio plugin to integrate directly with the third major multimedia framework.

The Bluetooth SIG has released updates of the HFP, HSP, A2DP and AVRCP specifications. These new specifications include updates for Simple Pairing support and meta-data transfer. The meta-data transfer allows to transfer ID3 information like song titles and artist information to the headset. In the future headsets or speakers with a display could use the meta-data transfer to display them. The support of meta-data transfer within the GStreamer plugin is work in progress.

Conclusion

The design and implementation of the current architecture has been widely accepted and used in products like the Nokia N810 [11].

The quality and performance of the Subband-codec can easily measure up against any commercial implementation. The architecture is flexible enough to support embedded hardware and also advanced systems with DSP or codec offload of the audio processing.

References

[1] Bluetooth Special Interest Group:
Bluetooth Core Specification Version 2.0 + EDR, November 2004

[2] Bluetooth ALSA Project:
`http://bluetooth-alsa.sf.net/`

[3] Advanced Linux Sound Architecture (ALSA):
`http://www.alsa-project.org/`

[4] GStreamer Multimedia Framework:
`http://www.gstreamer.net/`

[5] freedesktop.org:
D-BUS Specification Version 0.12
`http://dbus.freedesktop.org/doc/dbus-specification.html`

[6] Request for Comments:
RTP: A Transport Protocol for Real-Time Applications (RFC3550)

[7] PulseAudio Sound Server:
`http://www.pulseaudio.org/`

[8] Instituto Nokia de Tecnologia (INdT):
`http://www.indt.org.br/`

[9] Subband-codec (SBC) implementation:
`http://sbc.sf.net/`

[10] BlueZ Project:
`http://www.bluez.org/`

[11] Nokia N810 Internet Tablet:
`http://www.nokiausa.com/A4626058`