Robust Audio Tool (RAT) Supporting Separate Recording and Playback Audio Devices Selection

Cesnet Technical Report No. 10/2009

Tomáš Rebok[1,2], Martin Beneš[1], and Milan Kabát[1]

[1]*Faculty of Informatics, Masaryk University, Botanická 68a, Brno*
[2]*CESNET z.s.p.o., Zikova 4, Prague*

December 11, 2009

Abstract

This technical report describes the modifications of the Robust Audio Tool (RAT) application, that allow its users to select separate recording and playback audio devices. These modifications have been driven especially by the requirement to support professional sound cards providing separate half-duplex recording and playback audio devices only, which the original RAT is not able to make use of.

Keywords: *RAT-HD, RAT, half-duplex devices, separate audio devices, separate recording and playback audio devices.*

1 Introduction

The Robust Audio Tool (RAT)[5] is an open-source cross-platform audio conferencing and streaming application allowing its users to participate in audio conferences over the Internet. It has been originally implemented by Colin Perkins at University College London[1] (UCL) as a part of UCL's toolset supporting remote collaboration[2] and later became maintained mainly by the UCL's SUMOVER project[3]. Currently, it has become maintained by UCL's AVATS project[4], which claims to propose support for its enhanced stability and longevity.

As claimed by Colin Perkins [7], the RAT is one of the earliest voice-over-IP applications, which has pioneered the use of forward error correction in VoIP systems,

[1]`http://www.ucl.ac.uk/`

[2]Among the other tools belong the VIdeoconferencing Tool (VIC)[4] and the shared WhiteBoarD tool (WBD)[1].

[3]`http://mediatools.cs.ucl.ac.uk/nets/mmedia/`

[4]`http://www.cs.ucl.ac.uk/research/avats/`

and furthered development of receiver-based loss concealment algorithms, adaptive playout scheduling, and RTCP-based diagnostics for multicast conferencing. Besides that, it further features a range of different rate and quality codecs, the receiver based loss concealment to mask packet losses, and a sender-based channel coding in the form of redundant audio transmission.

For point-to-point communication, the RAT does not require any special features except for a network connection capable of unicast-based communication and a soundcard. For multi-party conferences the RAT uses the IP multicast and therefore all participants must reside on a multicast capable network. However, to avoid the dependency on the presence of multicast service, which is not universally available and reasonably reliable in many networks yet, software-based tools simulating the multicast functionality in unicast networks can be used—for example, the UDP packet reflector [8], the Active Element [3], the Distributed Active Element [2], etc.

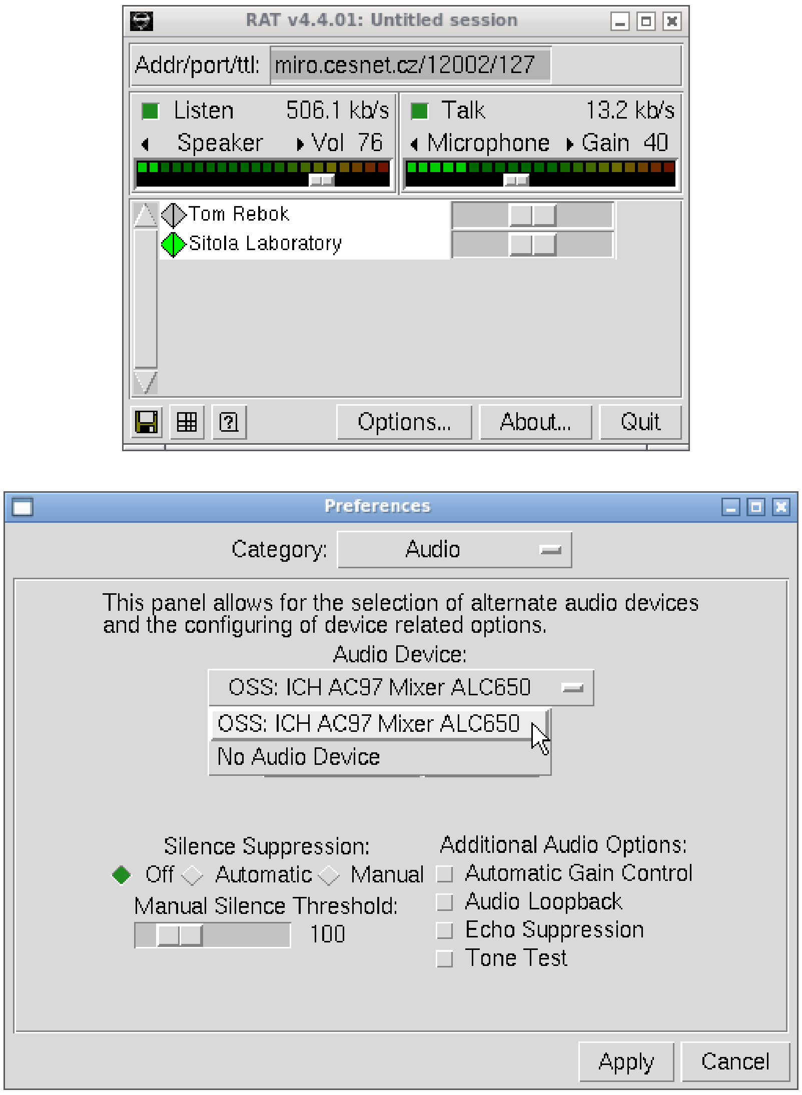
Regarding the RAT's audio subsystem, it supports many various audiosystems including Alsa, OSS, NetBSD audio, MacOS X audio, Win32 audio, etc., which make it a highly platform-independent solution for remote collaboration. Internally, the RAT assumes a single full-duplex audio device, which is used both for voice recording and playback simultaneously (see the Figure 1). Even though this is fully sufficient in common situations, it becomes a limitation in cases when one wants to use it with professional sound cards providing half-duplex devices only. For example, the LynxONE[5] soundcard we use provides four half-duplex devices only—the LynxONE analog output, the LynxONE analog input, the LynxONE digital output, and the LynxONE digital input—which make it unusable with RAT.

Thus, the main goal of our work is to implement the RAT-HD (*RAT with Half-Duplex devices support*)—to modify the original RAT's architecture and GUI in order to make it supporting both full-duplex and half-duplex audio devices, and to enable its users to choose the recording and playback audio devices separately.

2 RAT Architecture

The RAT's architecture comprises three separate processes—the *media-engine*, which is responsible for media transport and rendering, the *user-interface*, which provides a graphical user interface, and the *controller*, which coordinates the previous two. The RAT is invoked by starting the controller, which in turn starts new processes for the media-engine and user-interface. Once started, all the three processes synchronize using the *Mbus mechanism* (see later) and the media-engine in cooperation with the audio subsystem starts to report its capabilities and its status to the user-interface, that uses this information to build appropriate menus for the options that can be configured. Once this setup-phase finishes, the user-interface reacts to user interaction by sending appropriate Mbus messages to the media-engine, and the media-engine reports events to the user-interface (again, using the Mbus messages). [6]

[5]`http://www.lynxstudio.com/product_detail.asp?i=7`


RAT v4.4.01: Untitled session

Addr/port/ttl: miro.cesnet.cz/12002/127

Listen 506.1 kb/s | Talk 13.2 kb/s
Speaker Vol 76 | Microphone Gain 40

Tom Rebok
Sitola Laboratory

Options... About... Quit

Preferences

Category: Audio

This panel allows for the selection of alternate audio devices and the configuring of device related options.

Audio Device: OSS: ICH AC97 Mixer ALC650

OSS: ICH AC97 Mixer ALC650
No Audio Device

Silence Suppression: [✓ Off] Automatic Manual

Manual Silence Threshold: 100

Additional Audio Options:
- Automatic Gain Control
- Audio Loopback
- Echo Suppression
- Tone Test

Apply Cancel

Figure 1: The main screen and audio devices' selection screen of the original RAT.

The mentioned *Mbus mechanism* [6] is a form of inter-process communication proposed by Colin Perkins and Jorg Ott. It is claimed to be a lightweight, message oriented infrastructure for ad-hoc composition of heterogeneous components.

An Mbus message contains a header and a body. The header part defines the source and destination address, while the latter one contains the message, which has to be delivered to the particular process. The messages are in the form of strings and a function maps them into C function calls at the destination.

Concerning the RAT's audio subsystem, the media-engine process communicates with all the audiosystems available on the particular system, and discovers all the provided full-duplex audio devices. Once discovered, the audio device requested by the user is opened and appropriately set; after that, it can be used both for a voice capture and/or playback.

3 RAT Architecture Modifications

To enrich the RAT with desired features we have downloaded its latest version (4.4.01) from the public SVN repository[6] and identified all the necessary modifications. We aimed to minimize the amount of necessary changes, and especially to avoid audio subsystem's/drivers' API changes.

Even though the RAT supports many audiosystems (see the Section 1), for the initial RAT-HD version we have decided to support just the OSS audiosystem (especially because of its direct support for the LynxONE soundcard we have intended to use). Nevertheless, the other audio systems should remain functional, however, they require a full-duplex device to be chosen (identified by the recording audio device).

As depicted in the rest of this section, the modifications, which had to be made in order to reach the desired goal, affected all the three RAT's components:

audio subsystem / OSS driver modifications

- the `audio_desc_t` type, which serves as a unique handle identifying the audio devices, and which is originally of the `int` type, has been modified into the structure having `r` and `w` items (both of which having the `int` type) identifying the read and write devices separately. This has allowed us to identify both the devices without any needs to change the functions' prototypes[7] and to deliver this information into the OSS driver layer, where the functions are able to apply the required operation just to the device they are related to (for example, the `oss_audio_set_igain` relates just to the read audio device, while, e.g., the `oss_audio_open` function relates to both the read and write audio devices). Thus, almost no functions' duplications/device treatments need not to be performed on the higher layers.

- the change of the `audio_desc_t` type has subsequently caused the necessity to change the type of the `descriptor` item of the `audio_device_details_t` structure—its type has been changed from the `audio_desc_t` type into the `int` type, since the structure describes just a single audio device available in the system and thus the `audio_desc_t` type now describing two devices simultaneously has been undesirable.

- according to both the types' changes, all the functions from the `audio.c` file have been subjected to deep study and modified when necessary.

- similarly, all the functions from the `auddev.c` file have been revised and modified in order to cope with the `audio_desc_t` and `audio_device_details_t` types change.

[6]`http://mediatools.cs.ucl.ac.uk/nets/mmedia/wiki/SvnDev#Developerinformation`

[7]In fact, a single function's prototype had to be changed—the argument of the function `audio_if_dev_name`, which serves for discovering the name of the particular audio device, had to be changed from the `audio_desc_t` type (now describing two devices) into the `int` type (describing just a single device, whose name the function should return).

- the most changes related to these layers have been performed in the OSS driver layer (the file `auddev_oss.c`): besides the modifications related to both the types' changes, which had to be performed on all the functions, a few functions had to be modified in order to handle both the read and write devices separately (e.g., the `oss_audio_open` function), and to take the half-duplex devices into account (e.g., the `oss_audio_init` function).

- moreover, since the RAT's OSS driver supposes to have the mixer devices closely related to the audio devices (i.e., to have the mixer device for the `/dev/dspX` audio device accessible through the `/dev/mixerX` file), and since e.g. the LynxONE soundcard provides just a single mixer device for all the four half-duplex audio devices, we have had to revise the OSS driver code in order to cope with this situation. We have found out, that such relation cannot be supposed at all, as claimed in the OSS programmers guide [9]; thus, the OSS driver code has been modified in order to control the mixer device through the audio devices' descriptors (the recommended way to cope with such a situation).

- last, but not least, the function setting the non-blocking access to the audio devices (`oss_audio_non_block`) has been modified to perform no action. Instead, the devices are being opened with the `O_NONBLOCK` flag, which is the recommended way as denoted in the OSS programmers guide [9].

media-engine modifications

- because of the audio subsystem modifications and the mentioned types' changes, all the audio subsystem functions' calls have been revised and appropriately modified,

- the `mbus_engine.c` file has been modified in order to provide the user-interface with read and write audio devices separately (via appropriate Mbus messages),

- the `mbus_engine.c` file has been enriched with the callbacks processing another two Mbus messages (`audio.device.r`, `audio.device.w`) delivering the chosen audio devices from the GUI, and later registering them in the RAT's audio subsystem,

- the `settings.c` file has been modified so that the requested audio read and write devices can be retained throughout the sessions (the devices are saved into the `audioDeviceR` and `audioDeviceW` items).

user-interface modifications

- the RAT-HD's GUI (more precisely, the *Audio* screen from the *Options* window) has been enriched with the possibility to choose the separate read and write audio devices from appropriate list boxes,

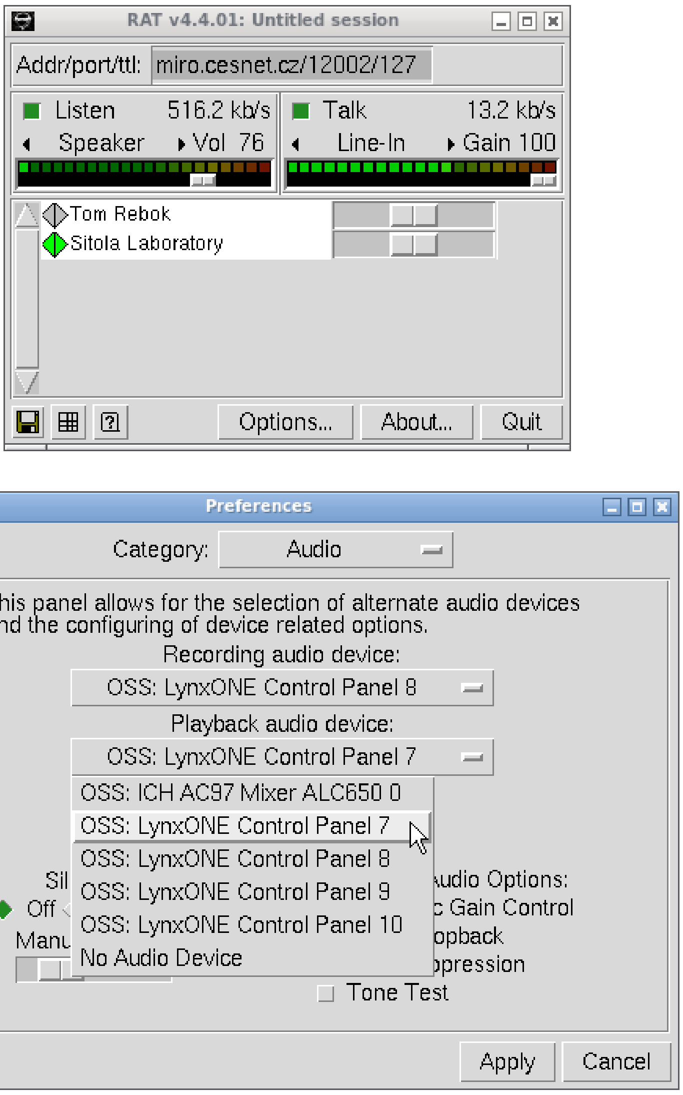
- regarding the communication from the media-engine to the user-interface, the set of original Mbus messages has been enriched with a few messages (`audio.devices.flush.r`, `audio.devices.flush.w`, `audio.devices.add.r`, and `audio.devices.add.w`) providing the available audio devices to the user-interface,
- regarding the communication from the user-interface process to the media-engine process, the Mbus messages have been enriched with two another messages (`audio.device.r`, and `audio.device.w`) providing the chosen audio devices to the media engine,
- all the functions handling the modified/additional Mbus messages have been appropriately modified.

3.1 Restrictions and Further Issues

Nevertheless, the first version of the RAT-HD has a few more or less important restrictions and issues, most of which we plan to solve in the future:

- In spite of the fact that the original RAT supports many audio systems (Alsa, OSS, NetBSD audio, etc.) simultaneously—i.e., it is able to detect and use any of the provided devices at a time—the RAT-HD requires both the chosen read and write devices to be provided by the same audio system. Even though this restriction could be more or less easily overcome, we have decided not to support choosing the read and write audio devices from different audio systems especially because of many problems it may produce.
- Currently, just the OSS audio system is fully supported by the RAT-HD. In the future we would like to add the half-duplex devices support at least to the other most important ones, i.e., Alsa and Win32 audio.
- The RAT-HD enables the users to choose just a single sample rate and a single number of channels for both the read and write audio devices. Again, even though this could be easily overcome, we have decided to keep it especially because of the problems, which the different sample rates for read and write audio devices may produce[8].
- In the current version of RAT-HD, the media-engine obtains just a single list of available audio devices consisting of both read and write devices without any capability indicators (i.e., read-only device, read/write device, write-only device). Thus, the user has to choose the proper recording/playback audio device from the lists, which, however, contain all the recording-only, recording/playback, and playback-only audio devices altogether.

[8]In fact, even two RATs remotely communicating with different sample rates cause many problems and serious audio quality degradations.


RAT v4.4.01: Untitled session

Addr/port/ttl: miro.cesnet.cz/12002/127

Listen 516.2 kb/s
Speaker Vol 76
Talk 13.2 kb/s
Line-In Gain 100

Tom Rebok
Sitola Laboratory

Preferences

Category: Audio

This panel allows for the selection of alternate audio devices and the configuring of device related options.

Recording audio device: OSS: LynxONE Control Panel 8

Playback audio device: OSS: LynxONE Control Panel 7

Options...
About...
Quit
OSS: ICH AC97 Mixer ALC650 0
OSS: LynxONE Control Panel 7
OSS: LynxONE Control Panel 8
OSS: LynxONE Control Panel 9
OSS: LynxONE Control Panel 10
No Audio Device
Sil
Off
Manu
udio Options:
c Gain Control
opback
ppression
Tone Test
Apply
Cancel

Figure 2: The main screen and audio devices' selection screen of the RAT-HD.

In fact, we haven't implemented a devices' separation based on their capabilities into the current version, since it requires changing the API between the RAT's audio subsystem and the audio drivers, which we have decided to avoid. Nevertheless, we have already modified the "*media-engine* $\leftrightarrow$ *GUI*" communication to be prepared for such a separation, and thus just the "*audio subsystem* $\leftrightarrow$ *media-engine*" communication has to be modified in order to provide the user with two lists of just read-capable and write-capable audio devices.

- The audio devices' names detection in the OSS driver is another challenge for the future. The original RAT obtains the devices' names from their dedicated mixer device (in fact, it obtains mixer devices' names), which is not suitable for the soundcards providing several half-duplex devices controlled by a single mixer device (e.g., the LynxONE), because all the audio devices would have the same name, which is, however, used as a unique device identifier identifying the device(s) selected by the user.

In the future we would like to overcome this issue by a detection of the audio devices' real names (not the mixers' names); current RAT-HD implementation uses the mixers' names, which are supplemented by a number indicating the audio device sequence in the system (i.e., the `X` from the `/dev/dspX`).

4 Conclusions

In this technical report we have described the architecture modifications, which we have made in order to allow the Robust Audio Tool (RAT) to support both full-duplex and half-duplex audio devices, and which further enable its users to choose the recording and playback audio devices from the RAT's GUI separately[9]—see the Figure 2. This additional functionality not only increases the RAT-HD's flexibility related to audio devices' selection, but makes it especially able to support professional sound cards, which often provide just half-duplex audio devices, and which the original RAT was unable to make use of.

The RAT-HD has been subjected to a set of tests, which have indicated that it is fully usable in real situations—we were able to choose an arbitrary combination of full-duplex/half-duplex audio devices provided by the used soundcards without any perceptible problems. All the tests were performed on the Intel Pentium 4 machine with 2.00 GHz CPU and 768 MB of RAM; the installed operating system was Ubuntu 8.04.3 LTS with 2.6.24 kernel. There were two soundcards installed in the system—the LynxONE and Intel ICH4 AC'97 Audio Controllers[10]—both of which were controlled by a single audiosystem: the OSS version 4.2.

Concerning the future challenges, we would like to solve the issue of read/write devices detection (see the Section 3.1)—both the capabilities detection and the proper name detection. Moreover, we would like to add the support for half-duplex devices to another RAT's audiosystems drivers, like Alsa and/or Win32 audio.

Acknowledgement
This work has been supported by the research intent *Optical Network of National Research and Its New Applications*, MŠM 6383917201, funded by the Ministry of Education, Youth and Sports of the Czech Republic. Moreover, we would like to express our thanks to Petr Holub, whose initial work in RAT's OSS driver modifications related to the LynxONE soundcard support made integrating its support into the RAT-HD easier.

[9]The source codes of the RAT-HD are available for download at
`https://www.sitola.cz/igrid/index.php/RAT`

[10]The test have been successfully performed with the Sound Blaster Audigy ZS2 soundcard as well.

References

[1] Julian Highfield and Kristian Hasler. Whiteboard (WBD).
`http://www-mice.cs.ucl.ac.uk/multimedia/software/wbd/`, 2009.

[2] Petr Holub and Eva Hladká. Active Elements for High-Definition Video Distribution. In *Network and Parallel Computing (NPC 2006)*, pages 27–36, University of Tokio, Japan, 2006.

[3] Petr Holub, Eva Hladká, Jiří Denemark, and Tomáš Rebok. Active Elements for High-Definition Video Distribution. In *ICT 2006, 13th International Conference on Telecommunications*, pages 1–4, Funchal, Madeira: University of Aveiro, Portugal, March 2006.

[4] Piers O'Hanlon, Kristian Hasler, Doug Kossovic, Socrates Varakliotis, Soo-Hyun, Mark Handley, Isador Kouvelas, Barz Hsu, John Brezak, and Mark S. Petrovic. Videoconferencing Tool (VIC).
`http://mediatools.cs.ucl.ac.uk/nets/mmedia/wiki/VicWiki#VideoconferencingToolVIC`, 2009.

[5] Piers O'Hanlon, Socrates Varakliotis, and Doug Kossovic et al. Robust Audio Tool (RAT).
`http://mediatools.cs.ucl.ac.uk/nets/mmedia/wiki/RatWiki#RobustAudioToolRAT`, 2009.

[6] Jorg Ott, Dirk Kutscher, and Colin Perkins. The Message Bus: A Platform for Component-based Conferencing Applications. In *Proceedings of CBG2000: The CSCW2000 workshop on Component-based Groupware*, 2000.

[7] Colin Perkins. Robust Audio Tool.
`http://csperkins.org/research/rat/index.html`, 2009.

[8] Zdeněk Salvet. Enhanced UDP packet reflector for unfriendly environments. Technical Report 16/2001, CESNET, 2001.

[9] 4Front Technologies. OSS Programmers Guide.
`http://www.opensound.com/pguide/index.html`, 2009.