An Internet-Radio Receiver and Real-Time Recorder on SBC-2410x Embedded System

Lain-Jinn Hwang*, Tang-Hsun Tu†, Tzu-Hao Lin‡, I-Ting Kuo§, Chih-Yuan Chen¶
Department of Computer Science and Information Engineering
Tamkang University, Tamshui, 251, Taipei, Taiwan
**E-mail: micro@mail.tku.edu.tw*
†*E-mail: war3_515@hotmail.com*
‡*E-mail: singy000@msn.com*
§*E-mail: yiting2002@hotmail.com*

Abstract

In this paper, we present a software-based internet-radio receiver on SBC-2410x with real-time recording. To achieve this goal, we have developed a "Streaming Packet Tracer (SPT)" algorithm which is a method for real-time trace the streaming packet header. We ported the well known open-source media player (Mplayer) and embedded the SPT algorithm in this player to make SBC-2410x as an internet-radio receiver. The SPT algorithm is shown to be faster than other methods and make SBC-2410x as a player and real-time recorder.

Keyword – Mplayer, internet-radio, SBC-2410x, real-time recording, Streaming Packet Tracer.

1 Introduction

Streaming technology [1], [2] provides the means of delivering news, entertainment, remote education, documentary, and more types of communication by some protocols such as RTP, UDP and so on. RTP (Real-Time Transport protocol) [3], [4], [5] provides the common media transport layer, uncompressed media data (audio data) is captured into a buffer from which compressed frames are produced. Frames may be encoded in several ways upon what compressed format. Compressed frame are loaded into RTP packets for transmitting, they may be fragmented into several RTP packets. A receiver is responsible for collecting RTP packets from the network streaming, and insert them into a playout buffer, then decoding for corresponding audio format and play it.

The hardware resources will be limited for embedded system so that it cannot play and record at the same time. So we design a method which called "Streaming Packet Tracer (SPT)", it does not need to spend the extra space to store file with WAV audio format, it also does not need extra encoder for recording. We find the packet header and payload in buffer for network streaming downloaded and write them in file for recording by modified Mplayer [6], it's just simple as well as fast.

We port Mplayer to SBC-2410x [7] embedded system as software-based internet-radio with Streaming Packet Tracer method, users can penetrate the software to connect the media server to download network streaming and record it.

This paper is organized as follows. Section 2 describes the general method for audio recording, section 3 describes our method for real-time recording (SPT), section 4 discusses the packet loss problem on network streaming, section 5 describes format-change problem and section 6 talks about our experimental results.

2 Methods for Audio Recording

We describe the generally methods for audio recording in this section, the advantages and why it's not suitable on embedded system. We will discuss our method in section 3.

2.1 Recording with WAV Format

First, we describe method which records with WAV [8] format. After audio format decoded, it's audio file (WAV) (Figure 1), no matter what audio format receives from net-

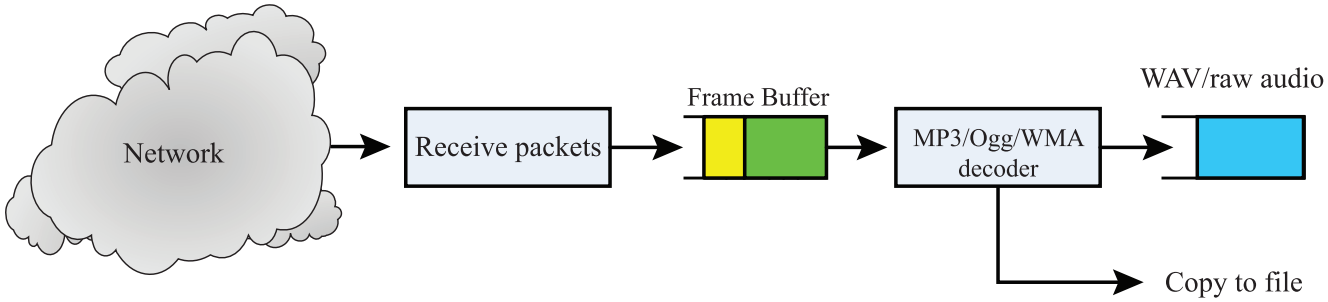


Figure 1: Recording with WAV Format

work, passing through a succession of processing, decoded audio file is the WAV format.

When media player playing, the method directly will output to the WAV format file and copy extra output file for recording, one for output and the other one for recording.

The advantage of recording for this method, certainly does not need to make the extra compression, but uses direct copy to the file. Saving the time to handling to file which decoded, because WAV audio format belongs to the lossless, therefore in capacity, far ratio other compressions audio format (for example: MP3 [9], Ogg [10]) come greatly very many, if does not have the extra storage equipment, then possibly stores up the WAV audio file quite is possibly limited. For a 4 minute song, the 44.1KHz sampling frequency, the file size which WAV occupies as follow:

$$\begin{aligned} \text{Size} &= 44100 * 2 * 2 * 4 * 60 \\ &= 42,336,000\text{Bytes} = 40.37\text{MB} \end{aligned} \quad (1)$$

Because it just copy raw data, it doesn't need extra CPU computing power for encoding compressed format audio. Although it can work on SBC-2410x embedded system, the record file size of this method is too large. So it's not suitable for embedded system.

2.2 Recording with Compressed Format

Common audio compresses format, for example: WMA [8], Ogg, MP3, RM, RAM, RA [11]... etc, on network as a result of the bandwidth question, transmit audio format file which is lossless file, it is consumption cost. Therefore, almost all the streaming media uses the compression audio format for transmit. After receives the compressed audio, the client decode it to WAV format. This way not only saves file size on both side, but also reduce transmission time.

As mentions in section 2.1, recording uncompressed WAV data will have question in the capacity. Therefore, we must compress the audio data. One method for recording compressed audio is shown in Figure 2. After decode the compressed audio, the player encode it to a pre-define format, then save the compressed data to file. Recording audio this way, the file size can differ above 10 times (2). By using Real-time audio recording, the program (player) must decode and encode at the same time, the computing power of the processor needs to be high. In embedded system (SBC-2410x), although this method could save spaces, it will cause situation which CPU is overloaded, or impossible to achieve.

For digital audio format file which is CD acoustic fidelity, as the example, its bit rate (the data for 2-channel audio stream each second bit stream), the sampling frequency is 44.1KHz:

$$\text{Bit rate} = 44.1 * 1000 * 16 * 2 = 1411.2\text{kbps}$$

A supposition choice position bit rate is 128kbps makes the format compression for MP3 audio, its compressed rate:

$$\text{Compressed rate} = 1411.2/128 = 11.025$$

For a 4 minutes song, selects this compression method, its occupies the size is, makes the compression by the (1) calculation size:

$$\begin{aligned} \text{Size} &= 42,336,000\text{Bytes}/11.025 \\ &= 3,840,000\text{Bytes} = 3.66\text{MB} \end{aligned} \quad (2)$$

Compare the size reduced % with record with WAV format by (1), (2):

$$\text{Size reduced\%} = (40.37 - 3.66)\text{MB}/40.37\text{MB} = 90.93\%$$

This method saves the storage for compressed audio format, but its complexity which needs to decode and encode at the same time is too high. Its complexity is too high for SBC-2410x embedded system, so that it can't work on this embedded system and it's not suitable.

3 New Method for Recording

Above the recording methods it will produce different results, if player uses recording with WAV format, it will

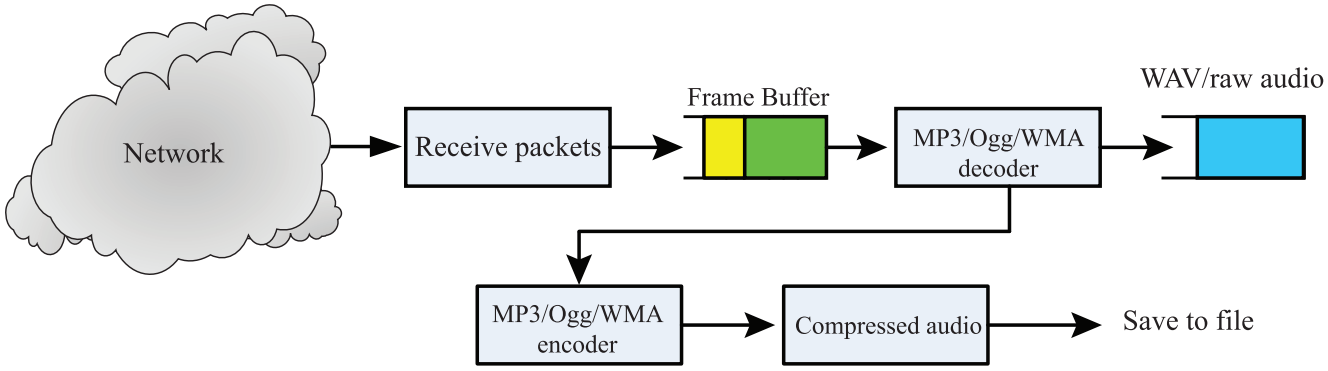


Figure 2: Recording with Compressed Format

consume extra huge storage space. If player uses with compressed format, it will need extra CPU computing power.

Because the hardware resources will be limited for embedded system, these method are not suitable for real-time recording on embedded system.

So we design a method which can play and record at the same time, it doesn't need to spend the extra space to store file with WAV audio format, it also does not need extra encoder for recording.

3.1 Method for Streaming Packet Tracer (SPT)

If we must recording with compressed format on embedded system and play at the same time, we need extra computing power, causing the situation which CPU overloaded and will make the overall system paralysis.

Therefore, we design a method which called "Streaming Packet Tracer" for real-time recording which can reduce the computing power, play and record at then same time. Algorithm 1 gives the tracing procedure for real-time recording, when user wants to record the streaming, the record flag will be set true, then the player will get local time to be the record-file name and open the record-file for recording (Algorithm 1 line 1-7). In receiving network stream, when player fills stream buffer, we find corresponding packet header and payload in the streaming buffer (data) from network stream (Figure 3) before the streaming media audio decodedd and get its length for the buffer. If user wants to record, then it will copy the buffer and write it to record-file for real-time recording(Algorithm 1 line 10-23). Finally, when user stops to record, then the record flag will be false and close the record-file(Algorithm 1 line 26-31).

We modify the source code for Mplayer, finding the position for streaming data (buffer). We copy the data which receives from network streaming and needs to record. This must complete before fill frame buffer, avoiding the packet of audio file its header and payload is cut, causing file incompletely, is unable to play for recording failure.

Algorithm 1 Code for the method "Streaming Packet Tracer" (SPT).

Require: buffer : array[0..MAXBUFFER] of char; {array for stream buffer (data)}
rfp : FILE Pointer; {File pointer for record file.}
len : integer; {Stream Buffer Length.}
rf : bool; {Record flag.}
fn : array[0..40] of char; {Char array for file name.}
Ensure: Find the packet header and payload to record.

```
1:  {Init to record}
2:  if Need to Record then
3:      rf = true;
4:      fn = Get now time for record name;
5:      rfp = open(fn, O_CREAT|O_WRONLY);
6:      .....
7:  end if
8:  .....
9:  {Start to record}
10: while Get network stream do
11:     .....
12:     do Fill stream buffer;
13:     .....
14:     buffer = Get the packet header and payload;
15:     len = length[buffer]; {Get length for stream buffer}
16:     if rf == true then
17:         if write(write(rfp, buffer, len) != len)) then
18:             do Record file for write Error.
19:         end if
20:     end if
21:     .....
22: end while
23: .....
24: {Terminate to record}
25: if Don't want Record then
26:     rf = flase;
27:     close(rfp);
28:     .....
29: end if
```

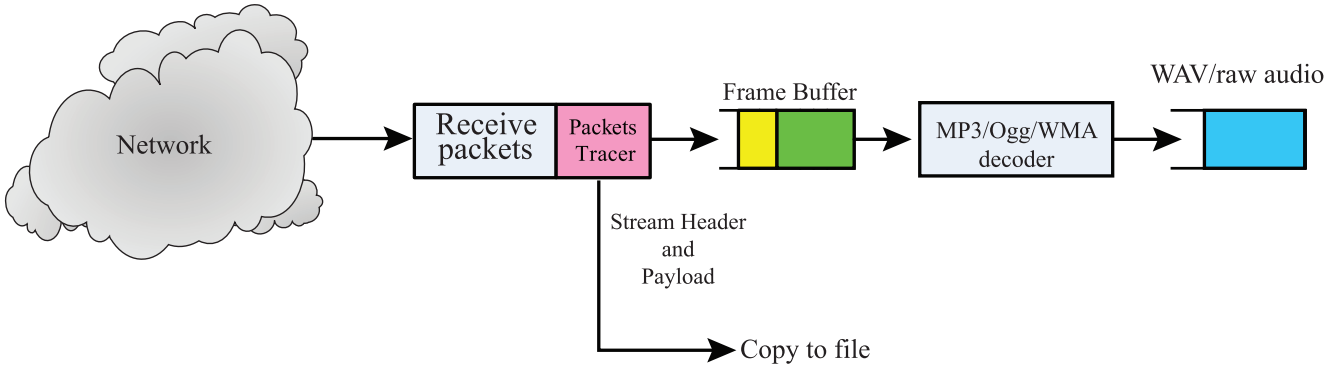


Figure 3: Real-Time Recording with SPT Algorithm

3.2 Advantages for SPT

Compares Real-Time recording(SPT) way which above mentions, certainly does not need to encode for compresses audio format, if the audio data transmitted is compressed, the record file is also compressed format (ex:MP3, OGG), therefore we only need to do copy the streaming buffer(data) to the file(also need find packet header and payload), does not need to spend the extra CPU computation, in the same time, the system can calculate other task more than the method of record with compressed format.

Because from streaming material already for compressed format (if audio data form network streaming is compressed), therefore we make time the above recording method, the record audio file all for compressed format, its size after the compression size for saving the space (Table 1).

Method of Recording	WAV format	compressed format	SPT
Time Comparability	No	High	Low
Storage	Huge	Small	Small
Select Audio Format	No	Yes	No
Encode and Pplay at the same time		No	Yes

Table 1: Compare with Recording Methods

4 Packet Loss Handling for UDP

UDP [12], [13], [14] has Connectionless the service, so it's uses directly transmit data to other side, if other side has receives the data, namely receives, otherwise, it does not receive. This protocol use the method which transmitting for best effort. If this method needs to do have to rely on the upper layer (Application) for handling. Because certainly does not need to confirm weather it receives or not, therefore compares with TCP [13], [15], its transmitted speed is faster than TCP. When transmitting massive file, this characteristic is more obvious, like video media data, using UDP to transmit is better than TCP (if you need speed for transmitting).

When player receives network streaming, if has not received the packets, all may accept in a threshold limit, as shown in Figure 4. When the times which packet loses are larger than threshold value, it will have the failure for opening streaming.

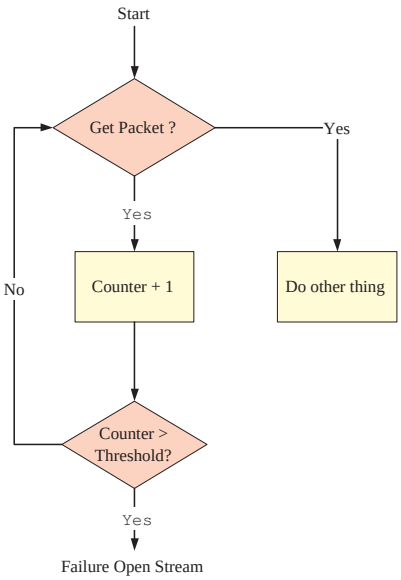


Figure 4: Packet Loss Detection

If UDP must do packet loss handling, the server and client may make a appointment with a new packet format,

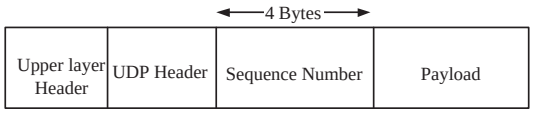


Figure 5: Sequence Number Section

like to append a 4-byte sequence number before UDP payload as shown in Figure 5, then its can judge on Application layer, whether receives the correct sequence of streaming packet, decided whether requests server to transmit again, but really does, must consider the packet section which server must increase to make handling, also needs to judge in the client for handling.

5 Format-Change Detection

It will transmit header to advise client side which file format for transmitting in the streaming process. If the streaming file by file to transmit, it will occur present file format as well as next file format has the possibility to be different. In other words, when player receive packet header, according to its content, it will opens different decoders for corresponding audio format.

If file audio format is changed when streaming transmits, using the method of record with WAV format to record. The complete audio file can transfer WAV format to make the sound recording, it cannot have on format the question. The supposition by the method of record with compressed format sound recording, because it encode the audio file after decoding for corresponding format compression, if uses same compression format, also cannot have on format the question.

By method of SPT, because we are directly find streaming data (Figure 3, Algorithm 1) and copy it, in other words, at present streaming audio format, as well as the audio for copy (record), their audio format will be same. If between two file formats from the same streaming different, then the audio file for recording format is also different. Because it's has two or more file formats in one file, so it will cause the problem which file format damages, perhaps is unable to play.

Our method for solve this problem in SPT is that if identical streaming has many audio file, the period can transmit header the way to advise (when changing audio file). We close the record file for previous recording, and opens next to new record file, take a file as the unit, we separate the files when changing. In other words, if the network streaming has two or more file for streaming, SPT will has the same number record file with the same format for network streaming.

6 Experimental Results

We have implemented our algorithm (SPT) and carried experiments to examine the recording method. The experimetal platform which we built by Table 2, 3.

Platform	SBC-2410x
CPU	S3C2410@200MHz 100MIPS
Memory	64MB SDRAM 64MB NAND Flash 1MB NOR Flash

Table 2: Hardware Platform

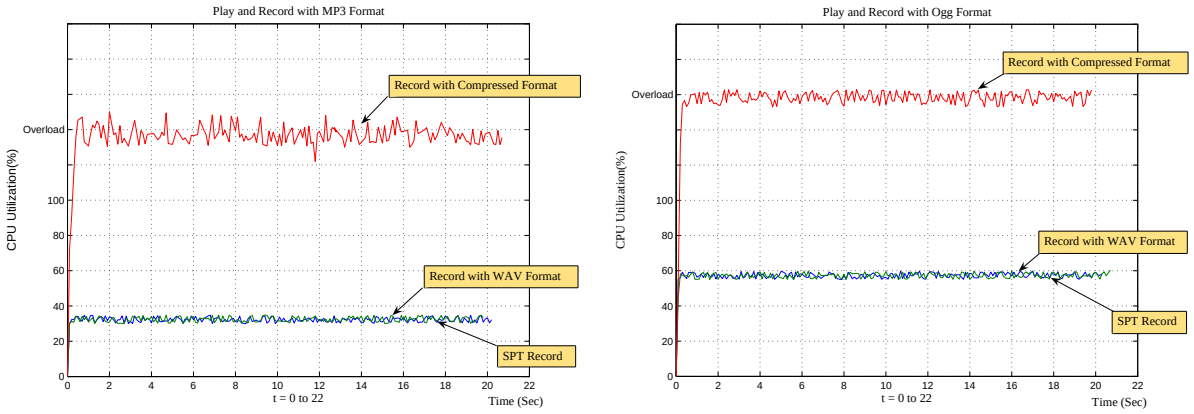
Linux Kernel	Kernel 2.4.18 [16]
Basic Commands	Busybox 1.2.0 [17]
Network Login	SSH (Dropbear 0.48.1 [18])
FTP Server	vsFTPd 2.0.5 [19]
Web Server	thttpd 2.21b [20]
Media Player	Mplayer-CVS-20060517

Table 3: Software Platform

User can use the web browser to control the media player (Mplayer) on SBC-2410x, we see that in Figure 6.

Figure 6: Actual Environment

We choice two network streaming (one is MP3 format (Figure 7(a)), the other is Ogg format (Figure 7(b))) for playing three minutes to test the record methods on SBC-2410x, we show the computing power and record-file size as follow (Figure 7, Table 4):



(a) (b)

Figure 7: CPU Utilization. (a) Record with MP3 Format (b) Record with Ogg Format

Method	Format	Utilization	File Size
Record with WAV Format	MP3	25% ∼ 35%	30.28 MB
	Ogg	55% ∼ 65%	30.28 MB
Record with Compressed Format	MP3	Overload	2.746 MB
	Ogg		2.727 MB
SPT	MP3	25% ∼ 35%	2.746 MB
	Ogg	55% ∼ 65%	2.727 MB

Table 4: Compare with Utilization and File Size

7 Conclusion

In this paper, we proposed an Real-Time Internet-Radio recording approach on SBC-2410x embedded system, which can tolerate packet-loss, format-change. Our approach is based on the SPT method to record, it just find the packet header and payload for copying. It is useful and efficiency on embedded system which has limited hardware resources.

References

[1] A. Ganz, Z. Ganz, and K. Wongthavarawa, *Multimedia wireless networks Technologies, standards, and QoS*. Prentice-Hall, 2004.

[2] K. Jonas, P. Kanzow, and M. Kretchmer, "Audio streaming on the Internet. Experiences with real-time streaming of audio streams," in *Proc. IEEE Int. Symp. Industrial Electronics*, vol. 1, pp. SS71–SS76, Jul 1997.

[3] C. Perkins, *RTP Audio and Video for the internet*. Addison-Wesley, 2003.

[4] H.Schulzrinne, "RTP: A Transport Protocol for Real-Time Application." http://www.faqs.org/rfcs, Jan. 1996. RFC 1889.

[5] V. Hilt, M. Mauve, J. Vogel, and W. Effelsberg, "Recording and playing back interactive media streams," *IEEE Trans. Multimedia*, vol. 7, pp. 960–971, Oct 2005.

[6] "Mplayer: The Movie Player." http://www.mplayerhq.hu.

[7] "Samsung." http://www.samsung.com/tw.

[8] "Microsoft Window Media." http://www.microsoft.com.

[9] "MP3: MPEG-1 Audio Layer3." http://www.thomson.com.

[10] "Xiph Ogg." http://www.xiph.org/ogg.

[11] "RealNetworks." http://www.realnetworks.com.

[12] J. B. Postel, "UDP: User Datagram Protocol." http://www.faqs.org/rfcs, Aug. 1980. RFC 768.

[13] D. E. Comer, *Computer Networks and internets with internet applications*. Prentice-Hall, 4th ed., 2003.

[14] P.P.-K Lam and S.C Liew, "UDP-Liter: an improved UDP protocol for real-time multimedia applications over wireless links," in *Proc. IEEE Int. Symp. Wireless Communication Systems*, pp. 314–318, Sep 2004.

[15] J. B. Postel, "TCP: Transmission Control Protocol." http://www.faqs.org/rfcs, Sep. 1981. RFC 793.

[16] "Linux Kernel." http://www.kernel.org.

[17] "Busybox." http://www.busybox.net.

[18] "Dropbear." http://matt.ucc.asn.au/dropbear/dropbear.html.

[19] "VSftpd." http://vsftpd.beasts.org.

[20] "thttpd." http://www.acme.com/software/thttpd.