A Novel Technique for Real-Time Internet Radio Recorder on Non-DSP Embedded System

Lain-Jing Hwang[1], Chien-Chou Shih[2], I-Ting Kuo[1]
[1]*Department of Computer Science and Information Engineering*
Tamkang University, Tamshui, 251, Taipei, Taiwan
micro@mail.tku.edu.tw
[2]*Department of Information and Communication*
Tamkang University, Tamshui, 251, Taipei, Taiwan
ccs@mail.tku.edu.tw

Abstract

The capability of providing real-time multimedia player over the Internet is an important future application for embedded system. However, the main challenge of such an application is the limitation on computation performance and memory size in architecture of embedded system, especially while encoding and decoding. In this paper, a Streaming Packet Tracer (SPT) algorithm is proposed for real-time tracing the streaming packet header. Based on the SPT algorithm, we present a novel software-based technique which enables a non-DSP embedded system to play and record real-time streaming audio simultaneously. To verify the feasibility of the proposed technique, a real-time Internet radio recorder has been implemented on SBC-2410x embedded system successfully. The experimental results show that our achievement has reduced the CPU usage rate from over 100% to about 30%-40% in comparison with other recording method.

Keywords: Embedded system, real-time Internet radio recorder, SBC-2410x.

1. Introduction

Streaming technology [1][2] provides the means of delivering news, entertainment, remote education, documentary, and many other types of communication. With the development of processor design technology for resource constrained SoC (System on Chip) [21], it becomes reality to implement streaming audio decoder in real time on a single RISC core embedded system. However, there are problems of computation performance and storage consumption on a single core embedded system while recording streaming audio in the popular audio format, such as MP3, WMA and WAV etc. In this paper, Mplayer [3] is executed on the ARM9 core embedded system as software-based internet-radio recorder, by which users can connect to the media server to receive and record network streaming audio.

In order to achieve this goal, a Streaming Packet Tracer (SPT) algorithm is proposed. The SPT algorithm synthesizes advantages of the generally recording methods, by which no extra encoding procedure is needed, and no further space to store a WAV audio file is required. On one hand the streaming packet head could be detected by SPT and be recorded in files, and on the other, the associated streaming files could also be used for real-time player by offering frame buffer directly.

The remainder of this paper is organized as follows. Section 2 describes the general methods for audio recording, the proposed SPT method for real-time recording, handling the packet loss problem on network streaming and format-change problem are revealed in section 3. Section 4 is the experimental results and comparisons. Finally, the conclusions and future research are discussed in section 5.

2. Methods for real-time audio recording

In this section, two generally methods for audio recording are discussed, revealing their weaknesses and why these methods are not suitable to implement

978-0-7695-3134-2/08 $25.00 © 2008 IEEE
DOI 10.1109/MUE.2008.36

real-time audio recorder on non-DSP embedded system.

2.1 Recording a streaming audio in WAV format

Recording streaming audio in WAV format is an intuitive approach. As shown in Figure 1, streaming audio datagrams can be decoded in WAV file, which then is copied directly.

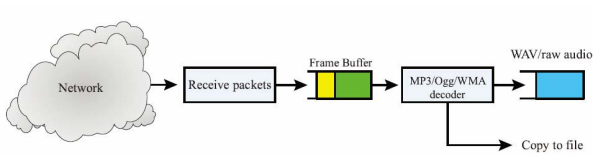


Figure 1. Recording streaming audio in WAV format

Using this method, there are two optional approaches regarding the processing file-copy routine. The first is that to ignore the copy routine and output the WAV [8] format file directly. However, while decoding the downloaded streaming audio, sound cannot be played. The second way is to copy an extra file for recording.

The advantage of recording by this method is that it does not need to perform extra compression and can save the time to handle the decoded file. However, since the WAV file is a lossless audio format, the embedded system must be equipped with extra storage. For instance, a 4 minutes song with 44.1 KHz sampling frequency will need extra storage to save the WAV file as follows:

$$\text{Size} = 44{,}100 * 2 * 2 * 4 * 60 = 42{,}336{,}000 \text{Bytes} = 40.37\text{MB} \quad (1)$$

Since it just copies the raw data, no extra CPU computation for encoding an audio is needed. However, this method is unsuitable to the implement of a real-time audio recorder on a single core embedded system such as SBC-2410x due to its heavy use of storage.

2.2 Recording a streaming audio in compressed format

With the rapid advance of Internet technologies, several common compressed audio formats, such as WMA [8], Ogg [10], MP3 [9], RM, RAM, and RA [11], are provided for the reason of saving the bandwidth. Therefore, almost all the streaming media technologies have employed compression audio format for delivery with enough CPU power and bus bandwidth to support the required data rates.

While recording a streaming audio in compressed format on the embedded system, as shown in Figure 2, the player encodes the streaming audio to a pre-defined format and saves the compressed file at the same time. This method can not only save the storage size, but also reduce the transmission time.

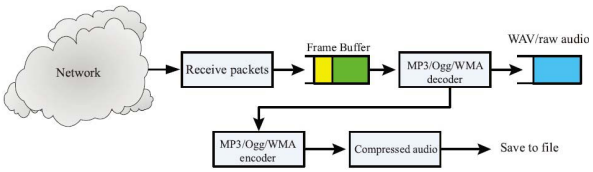


Figure 2. Recording streaming audio in compressed format

Comparatively, recording an audio in compressed format can reduce almost 10 times of storage size. For example, suppose that a digital audio format file is CD acoustic fidelity and the sampling frequency is 44.1 KHz, the bit rate can be calculated as follows:

$$\text{Bit rate} = 44.1 * 1{,}000 * 16 * 2 = 1411.2 \text{ kbps} \quad (2)$$

If the choice bit rate is 128 kbps for standard bit rate quality of MP3 audio, the compressed rate is:

$$\text{Compressed rate} = 1{,}411.2/128 = 11.025 \quad (3)$$

For example, for recording a 4 minute song, the required size to save this audio file is:

$$\text{Size} = 42{,}336{,}000 \text{ Bytes}/11.025 = 3{,}840{,}000 \text{ Bytes} = 3.66 \text{ MB} \quad (4)$$

Thus, this method can reduce 90.93% of storage size in comparison with previous approach. However, the player must decode and encode the streaming audio at the same time. Due to the complex computation of the decoding and encoding processes, limited hardware concerning CPU power for the embedded system is another problem. That is, the method of recording a streaming audio in compressed format can not work on a non-DSP embedded system.

3. New method for real-time recording streaming audio

3.1 Streaming Packet Tracer (SPT) algorithm

The Real-time Transport Protocol (RTP) [5][6][7] defines a standardized packet format for delivering audio and video over the Internet. It provides the common media transport layer, where uncompressed media data (audio data) is captured into a buffer from

which compressed frames are produced. Frames may be encoded in several ways depending on the compression methods. The compressed frames are loaded into RTP packets for transmitting, and these may be fragmented into several RTP packets. A receiver is responsible for collecting the RTP packets from the network streaming, and insert them into a play out buffer, then decode for corresponding audio format and play it.

In this paper, we propose a novel method (SPT) for real-time recording which not only reduce the computing load, but play and record streaming audio at the same time with minimum extra storage (Figure 3). Algorithm 1 shows the tracing procedure for real-time recording. When end-user requests to a media server for recording the streaming audio through the Internet, the record flag will be set to true, then the player will get local time as record-file name and open the record-file for recording (Algorithm 1 line 1-7). Whilst receiving streaming datagrams, if the player fills stream buffer, then the corresponding packet header and payload in the streaming buffer are traced before the streaming media audio is decoded as well as the length of the buffer is also derived. Subsequently, the program will copy the buffer and write it to the record-file for real-time recording (line 10-23 in Algorithm 1). Finally, when user stop recording, then the record flag will be set to false and the record-file is closed (line 26-31 in Algorithm 1). To perform the proposed algorithm, the source code of Mplayer was modified and the position of the streaming data buffer, required to copy streaming data for recording, is found. The above procedure must be done before filling out frame buffer to prevent the audio packet header from being trimmed off.

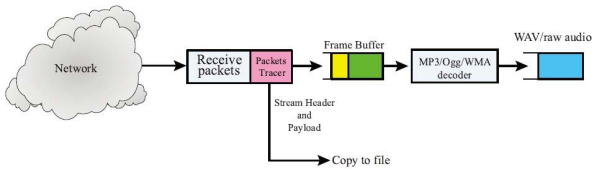


Figure 3. Real-Time recording with SPT algorithm

Algorithm 1: 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.
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**

3.2 Packet loss handling for UDP

Since real-time streaming recorder is a kind of time-sensitive application, which often use UDP [12][13][14] because dropped packets are preferable to delayed packets. Omitting to check whether every packet actually arrived makes UDP faster and more efficient. However, UDP does not guarantee reliability or ordering in the way that TCP does [13][15]. Datagrams may arrive out of order, appear duplicated, or go missing without notice.

In the implementation, there must make some efforts to handle the packet loss problem. When the player receives the streaming data, a pre-defined threshold value is used to decide if the received the packets are accepted, as shown in Figure 4. If the sum of packet losses is greater than the threshold value, it will fail to open the streaming datagram. As shown in Figure 5, a new packet format, which includes a 4-byte sequence number field, can be used for application layer to judge whether it receives the streaming packet in correct sequence or not.

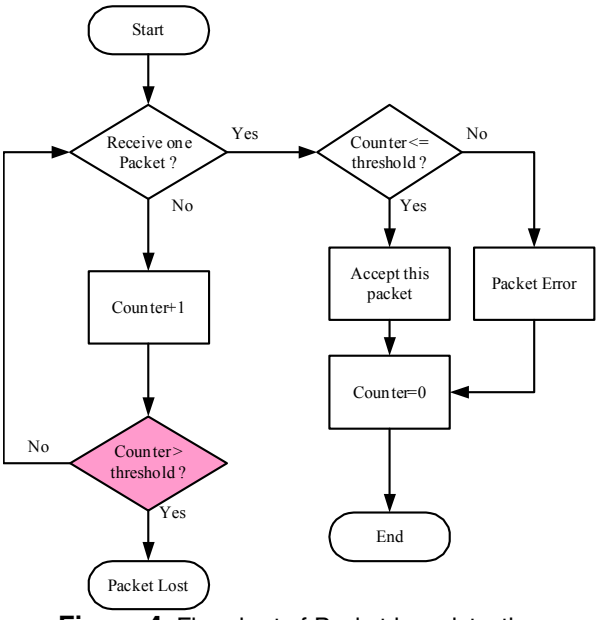


Figure 4. Flowchart of Packet loss detection

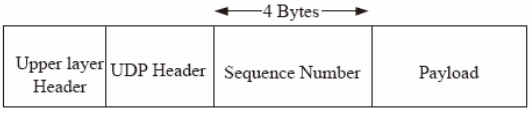


Figure 5. Sequence number section

3.3 Detection of Format-change

When dealing with real-time audio decoding, the audio files with different formats may be downloaded. The corresponding decoder should be taken to perform the decoding processes with respect to the file format. Based on the SPT algorithm, since the streaming data is traced and directly copied, if the file format from the same streaming is different, then the audio file for recording format is also different. This may lead to the problem of format mismatch and be unable to play the recorded audio file, because it has two or more file formats in one file.

In order to solve the format-change problem, the recording procedure must detect that if the identical streaming contains more than one audio file, and the period can transmit header to advice (when changing audio file). Then the record file is closed for previous recording, and the next new record file is opened. We separate the files when changing. In other words, if there are two or more files for streaming, SPT will take the file in the same record number with the same file format.

3.4 Advantages of SPT

Real-time Internet streaming audio player and recorder for different formats such as MP3, WMA and OGG etc. is an important future application for low-cost embedded system. Many such systems thus face severe limitations in size and computation consumption. The proposed SPT algorithm gives consideration to both saving storage and computation overhead. This method makes it practical to playback and recodes internet radio at the same time on the embedded system without powerful calculation capacity and extra storage. Table 1 shows the comparisons of different methods for real-time streaming audio recorder.

Table 1. Comparisons of Real-time streaming audio recording methods

Method	WAV format	Compressed format	SPT
Time for recording	No	High	Low
Extra storage requirement	Huge	small	small
Selected audio format	No	Yes	No
Playing and recording at the same time		No	Yes

4. Experimental results

The Samsung SBC-2410x [4] embedded system was chosen as the target experiment system in this research. SBC-2410x is a low-cost embedded system which bases on Samsung ARM processor (kernel ARM920T). The PCB board of SBC-2410x has six levels. It can execute arm-Linux operating system and Windows CE4.2.net operating system. It is a flexible and ideal control unit for education robot system. The detail about SBC-2410x is shown in Table 2.

In terms of software development, the OS and service components are listed in Table 3.

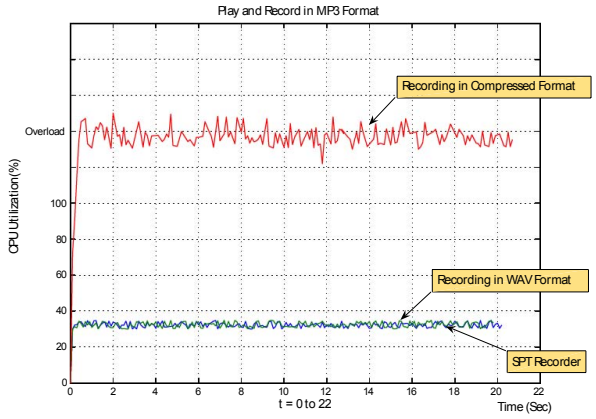
The proposed method was experimented in an average 40 Mbps Internet environment. Figure 6(a) and Figure 6(b) depict the comparisons of CPU usage rate between different methods while recording stream audio in MP3 format and Ogg format respectively. After applying our proposed software implementation to record a 22 seconds Internet audio from the streaming media server, the CPU usage rate can reduce to about 30% - 40% in comparison with the recording method in re-compressed format. Figure 7 shows that our implementation can perform the real-time recording operation successfully and, at the same time, execute Mplayer normally.

Table 2. Hardware of Embedded System

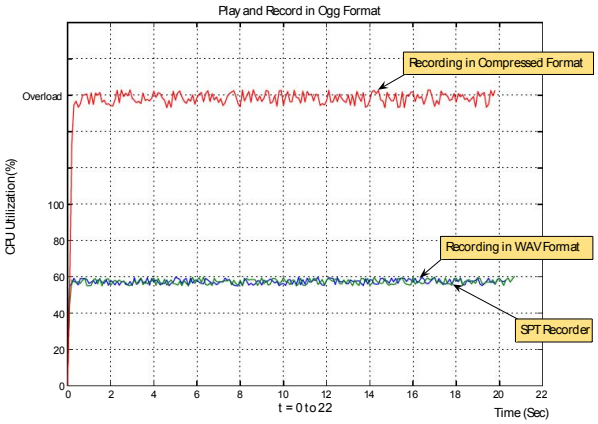
Platform	SBC-2410x
CPU	S3C2410@200MHz 100MIPS ARM 9 RISC32 Core
Memory	64MB SDRAM 64MB NAND Flash R 1MB NOR Flash
LAN	One 10M Ethernet, RJ45 port
SERIAL	One DB9 port
USB	One USB Host Type A One USB Slave Type B
Audio	One Stereo Audio Output One Audio Input
Board size	120mm*90mm

Table 3. Software of Platform

Linux kernel	Kernel 2.4.18 [16]
Basic commands	Busybox 1.2.0 [17]
Telnet login	SSH (Dropbear 0.48.1 [18])
Ftp Server	vsFTPd 2.0.5 [19]
Web Server	Thttpd 2.21b [20]
Open-source MediaPlayer	Mplayer-CBS-20060517 [3]



(a). Play and record a MP3 audio



(b). Play and record an Ogg audio

Figure 6. Comparisons of CPU utility rate

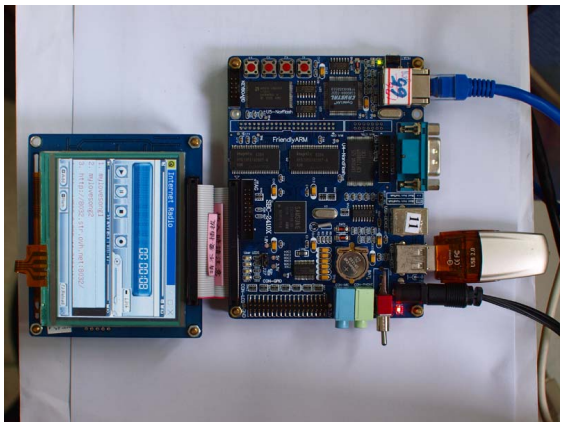


Figure 7. Implementation of Real-time Radio Recorder

5. Conclusions

In this paper, a novel SPT algorithm was proposed to achieve the implementation of real-time recording and playing of streaming audio on a non-DSP embedded system. Based on the proposed techniques, neither mass storage nor extra encoding procedure is spent. As a result, we have successfully developed a real-time Internet-Radio recorder on SBC-2410x embedded system, which can tolerate packet-loss and format-change. Of particular interest is that the system can efficiently play and record streaming audio at the same time. Furthermore, our achievement by software can also make it possible to play and recorder streaming radio on several non-DSP embedded systems such as s3c2440, xscale-255 or xscale-270, etc.

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, Jul. 1997, pp. SS71–SS76.

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

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

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

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

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

[8] "MicrosoftWindow 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. Postel, "UDP: User Datagram ProtocolL." 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," *Proc. IEEE Int. Symp. Wireless Communication Systems*, Sep. 2004, pp. 314–318.

[15] "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.

[21] Nattawut Thepayasuwan, Alex Doboli. "Hardware-software co-design of resource constrained systems on a chip", *Distributed Computing Systems Workshops, Proceedings 24th International Conference*, Mar. 2004, pp. 818-823.