A Pattern comparison for Silence Compression in wave file format

Deepak Upadhyay, Banwari, Deepanshu Sharma
Upadhyay.deepak0@gmail.com, banwari.mathur2@gmail.com, sharmadeepanshu159@gmail.com

Abstract— This paper presents a reliable pattern for silence compression in a wave file format. As audio compression reduces the bit rate required to represent an analog audio signal while maintaining the perceive audio quality. As wave file consist of speech and silence, considering noise as a part of speech. A threshold value is calculated for the silence and a reliable pattern for compression in a hexadecimal view. By using this pattern a comparison of compression ratio is done for different silence pattern in different audio wave files. As, by discarding inaudible data reduces the storage, transmission and compute requirements of handling high quality audio file.

.**Index Terms**— Chunk and Sub-chunk, Occurrence count, Silence-pattern, Silence threshold values, Compression ratio, Compression, Decoding.

1 INTRODUCTION

Compression of wide band audio signal to very low bit rates is desirable for a number of applications such as transmission of digital audio, multimedia applications. As audio compression reduces the bit rate required to represent an analog signal , while maintaining the perceived audio quality . Most audio decoders being designed today are called "lossy", they throw away the information that cannot be heard such as silence. As for the transmission purpose, transmission of silence, will increase the traffic overhead in the network.

For the transmission of signal in network, an analog signal is converted into digital signal for the transmission. An analog signal is digitized first as PCM, a sampling rate of 8khz, 8 bit, on mono channel for the transmission. The digital signal consist of speech and silence from which silence is compressed using a following pattern more reliable for compression in the encoding process keeping the audio quality better as without compression. At the receiver end, a delay is produced as for the decoding process, a decompression of that following pattern takes place. A delay is the expansion time of the bits which are compressed or the compressed pattern is decompressed.

2 WAVE FILE FORMAT

The WAVE file format [1] is a subset of Microsoft's RIFF spec, which can include lots of different kinds of data. RIFF is a file format for storing many kinds of data, primarily multimedia data like audio and video. It is based on chunks and sub-chunks. Each chunk has a type, represented by a four-character tag. This chunk type comes first in the file, followed by the size of the chunk, then the contents of the chunk. The entire RIFF file is a big chunk that contains all the other chunks. The first thing in the contents of the RIFF chunk is the "form type," which describes the overall type of the file's contents. So the structure of wave audio file looks like this: a) RIFF Chunk b) Format Chunk c) Data Chunk.

Table 2.1: RIFF chunk

Byte	Description
0-3	"RIFF"(ASCII Character)
4-7	Total Length of Package to follow
8-11	"WAVE"(ASCII Character)

Description of RIFF chunk as follows:

Offset	Length	Contents
0	4 bytes	'RIFF'
4	4 bytes	<file length - 8>
8	4 bytes	'WAVE'

Table 2.2: FORMAT CHUNK

0-3	"fmt_"(ASCII character)
4-7	Length of format chunk(Binary)
8-9	Always 0x01
10-11	Channel nos(0x01=mono,0x02=stereo)
12-15	Sample Rate(Binary,in Hz)
16-19	Bytes Per Second
20-21	Bytes Per Sample 1=8-bit mono,2=8-bit stereo/16-bit mono,4=16-bit stereo
22-23	Bits per sample

Description of FORMAT chunk as follows:

12	4 bytes	'fmt '
16	4 bytes	0x00000010
20	2 bytes	0x0001 // Format tag: 1 = PCM
22	2 bytes	<channels>
24	4 bytes	<sample rate> //
28	4 bytes	<bytes/second>
32	2 bytes	<block align> // channels * bits/sample /
34	2 bytes	<bits/sample> // 8 or 16

Table 3: DATA CHUNK

Byte	Description
0-3	"data"(ASCII character)
4-7	Length of data to follow
8-end	Data(Samples)

Description of DATA chunk as follows:

36	4	bytes	'data'
40	4	bytes	<length of the data block>
44		bytes	<sample data>

3 SILENCE THRESHOLD AND SILENCE PATTERN

Threshold is calculated using 10 samples of audio files as, audio format – PCM , 8000hz , 8 bit , mono channel on hex editor neo tool , by viewing the audio file in hexadecimal view , and clipping speech and silence separately to find out the threshold value. The conclusion after analysis states the threshold value 80H+2 /-2 , for the silence. Converting into decimal gives 126 ,127 , 128 , 129,130 considered as the silence value . In hexadecimal the silence value are 7eH, 7fH, 80H, 81H, 82H . A silence pattern formed to test the higher probability pattern which gives high compression ratio.

These are the following pattern:

1)	80	80	80	80
2)	7f	80	80	80
3)	80	80	7f	7f
4)	80	80	81	81

4 SILENCE COMPRESSION AND DECOMPRESSION ALGORITHM

4.1 Introduction

Silence Compression [4] on sound files is the equivalent of run length encoding on normal data files. In this case, the Runs we encode are sequences of relative silence in a sound file. Here we replace sequences of relative silence with absolute silence. So it is known as Lossy technique.

User Parameters:

1) Threshold Value : It considered as Silence. With 8-bit sample 80H considered as "pure" silence. Any Sample value within a range of plus or minus 2 from 80H considered as silence.

1) Silence_Code: It is code to encode a run of silence pattern . We used value FF to encode silence pattern The Silence_code is followed by a single byte that indicates how many consecutive silence codes there are as for considered pattern.

2) Start_Threshold: It recognize the start of a run of Silence pattern. We would not want to start encoding silence after seeing just a single byte of silence. It does not even become economical until following pattern is formed.

3) Stop_Threshold: It indicates how many consecutive non silence codes need to be seen in the input stream before we declare the silence run to be over.

4) Occurrence_count: It indicate the number of occurrence of the silence pattern in the audio file , so it can be compressed..

4.2 Silence Compression Algorithm (Encoder)

1) Read 8-bit Sample Data From audio file.

2) Checking of Silence pattern means find at least 4 consecutive silence pattern value: 80H or +2 /- 2 from 80H. (Indicate start of silence).

3) If get, Encode with Silence_Code followed by runs. \ (Consecutive Silence values).

4) Stop to Encode when found at least one Non-Silence pattern value.

5) Repeat all above steps until end of file character Found.

6) Print input File size, silence pattern, Occurrence count, Output File Size and Compression Ratio.

This algorithm [4] takes 8-bit wave audio file as input. Here It find starting of silence pattern means check that at least 4 consecutive silence pattern values present or not. Here 80H considered as pure silence and +/- 2 from 80H also consider as silence. If found, then it start encoding process. Here consecutive silence pattern values are encoded by silence_code followed by runs (Consecutive silence pattern values). It stop encoding when it found at least one non silence pattern value. Then it generate compressed file extension of that file is also wav file.

Example of algorithm as follows: Input file consists of sample data like 80 80 80 80 45 Output file consists of compressed data
like FF445.
It display following attributes of input wave audio file.

a. Input file size in terms of bytes
b. Silence pattern
c. Occurrence count
d. Output file size in terms of bytes
e. Compression ratio

4.3 Silence Decompression Algorithm (Decoder)

1) Read 8-bit Sample from Compress file.

2) Check the Silence code means 0xff if it found, Check the next value means runs which Indicate no of silence value.

3) Replace it with 0x80 (silence value) no of runs.

4) Repeat above step until we get end of file Character.

Example of algorithm as follows:

In input file (compressed file (extension of this file

.wav)) we find silence code , if we get then we check next value which indicate no of silence value. Then we replace with silence value no of runs times decided by user. We stop the procedure when we get end of file character.
If we get value 0xff4 in compress file, decode that value by
0x80 0x80 0x80 0x80

5 RESULT/ COMPARISON BETWEEN DIFFERENT SILENCE PATTERN

1. INPUT AUDIO FILE

Name of file: Ramavtarji. Wav
Media length: 2.080 sec
Audio format: PCM, 8000hz, 8-bit, mono
File size: 16,886 bytes

Table 1: Ramavtarji.wav

Input file size(bytes)	Silence Pattern	Occurrence count	Output file size(bytes)	Compression ratio
16,886	80 80 80 80	3,629	9,428	45%
16,886	7f 80 80 80	6	16,674	2%
16,886	80 80 7f 7f	5	16,676	1.25%
16,886	80 80 81 81	6	16,674	2%

2. INPUT AUDIO FILE

Name of file: Minakshi.wav
Media length: 5.520 sec
Audio format: PCM, 8000hz, 8-bit, mono
File size: 44,206 bytes

Table 2: Minakshi.wav

Input file size(bytes)	Silence pattern	Occurrence count	Output file size(bytes)	Compression ratio
44,206	80 80 80 80	10,623	22,960	48%
44,206	7f 80 80 80	44	44,118	1%
44,206	80 80 7f 7f	41	44,124	.28%
44,206	80 80 81 81	36	44,134	.27%

3. INPUT AUDIO FORMAT

Name of file: Kishanji.wav
Media length: 7.120 sec
Audio format: PCM, 8000hz, 8-bit, mono
File size: 57,006 bytes

Table 3: Kishanji.wav

Input file size(bytes)	Silence pattern	Occurrence count	Output file size(bytes)	Compression ratio
57,006	80 80 80 80	11,627	33,752	49%
57,006	7f 80 80 80	161	56,684	1.55%
57,006	80 80 7f 7f	139	56,728	.49%
57,006	80 80 81 81	137	56,732	.48%

4. INPUT AUDIO FORMAT

Name of file: Prem.wav
Media length: 8.800 sec
Audio format: PCM, 8000hz, 8-bit, mono
File size: 70,446 bytes

Table 4: Prem.wav

Input file size(bytes)	Silence pattern	Occurrence count	Output file size(bytes)	Compression ratio
70,446	80 80 80 80	14,493	41,460	42.15%
70,446	7f 80 80 80	114	70,218	.33%
70,446	80 80 7f 7f	85	70,276	.25%
70,446	80 80 81 81	79	70,228	.31%

5. INPUT AUDIO FORMAT

Name of file: Sarita.wav
Media length: 16.080 sec
Audio format: PCM, 8000hz, 8-bit, mono
File size: 1,31,382 bytes

Table 5: Sarita.wav

Input file size(bytes)	Silence pattern	Occurrence count	Output file size(bytes)	Compression ratio
1,31,382	80 80 80 80	20,064	91,254	31.55%
1,31,382	7f 80 80 80	971	1,29,440	1.48%
1,31,382	80 80 7f 7f	985	1,29,412	1.50%
1,31,382	80 80 81 81	990	1,29,402	1.51%

6. INPUT AUDIO FORMAT

Name of file: Anupamji.wav
Media length: 25 sec
Audio format: PCM, 8000hz, 8-bit mono
File size: 2,05,861 bytes

Table 6: Anupamji.wav

Input file size(bytes)	Silence pattern	Occurrence count	Output file size(bytes)	Compression ratio
2,05,861	80 80 80 80	27,412	1,51,037	27.64%
2,05,861	7f 80 80 80	1,486	2,02,889	1.45%
2,05,861	80 80 7f 7f	1,434	2,02,993	1.40%
2,05,861	80 80 81 81	1,442	2,02,977	1.41%

7. INPUT AUDIO FORMAT

Name of file: Deepak.wav
Media length: 36.917 sec
Audio format: PCM, 8000hz, 8-bit, mono
File size: 2,98,077 bytes

Table 7: Deepak.wav

Input file size(bytes)	Silence pattern	Occurrence count	Output file size(bytes)	Compression ratio
2,98,077	80 80 80 80	42,847	2,12,383	29.73%
2,98,077	7f 80 80 80	1,827	2,94,423	1.23%
2,98,077	80 80 7f 7f	1,714	2,94,649	1.16%
2,98,077	80 80 81 81	1,744	2,94,589	1.18%

8. INPUT AUDIO FORMAT
Name of file: Tweety.wav
Media length: 45.022 sec
Audio format: PCM, 8000hz, 8-bit, mono
File size: 3,62,919 bytes

Table 8: Tweety.wav

Input file size(bytes)	Silence pattern	Occurrence count	Output file size(bytes)	Compression ratio
3,62,919	80 80 80 80	51,962	2,58,995	29.64%
3,62,919	7f 80 80 80	2,424	3,58,071	1.34%
3,62,919	80 80 7f 7f	2,208	3,58,503	1.22%
3,62,919	80 80 81 81	2,244	3,58,431	1.24%

9. INPUT AUDIO FORMAT

Name of file: Banwari.wav
Media length: 49.265 sec
Audio format: PCM,8000hz, 8-bit, mono
File size: 3,96,860 bytes

Table 9: Banwari.wav

Input file size(bytes)	Silence pattern	Occurrence count	Output file size(bytes)	Compression ratio
3,96,860	80 80 80 80	55,927	2,85,006	29.19%
3,96,860	7f 80 80 80	2,797	3,91,266	1.39%
3,96,860	80 80 7f 7f	2,435	3,91,990	1.23%
3,96,860	80 80 81 81	2,461	3,91,938	1.25%

10. INPUT AUDIO FORMAT

Name of file: Deepanshu.wav
Media length: 59.216 sec
Audio format: PCM, 8000hz, 8-bit, mono
File size: 4,43,973 bytes

Table 10: Deepanshu.wav

Input file size(bytes)	Silence pattern	Occurrence count	Output file size(bytes)	Compression ratio
4,43,973	80 80 80 80	63,522	3,16,929	29.62%
4,43,973	7f 80 80 80	3,154	4,37,665	1.43%
4,43,973	80 80 7f 7f	2,695	4,38,583	1.28%
4,43,973	80 80 81 81	2,695	4,38,583	1.28%

6 CONCLUSION

We can achieve more compression, using silence pattern compression, without degrading the audio quality as the main factor. It is dealing with pure silence efficiently for the transmission of signal in the network. As if more silence values are there in pattern 80 80 80 80 in audio file , more compression will be there as compared to other pattern , probability of other pattern are less compared to 80 80 80 80. 80H is the pure silence threshold value, and from the tool and the script C ++ language it is calculated that when there is silence ,the pattern 80 80 80 80 is occurred in large number. So using this compression – decompression on this pattern, will provide a high compression ratio without degrading the audio quality. As different pattern of silence have their compression ratio which is less compared to the pure silence pattern (80 80 80 80) which is reliable for audio quality. The compensation is done, as compression ratio is not more than 50% , is due to maintain the quality of audio during transmission. In the network, less bytes are transmitted after compression as compared to without compression of silence, will decrease overhead in the network and bandwidth will be utilized

ACKNOWLEDGMENT

We would like to thank our college Professors and lecturer for providing us with proper support and guidance. They guide us throughout the research we carried out for "A Pattern Comparison for silence compression in wave file format"..

REFERENCES

[1] David pan, " A Tutorial on MPEG/Audio compression", IEEE multimedia, vol 2, No.2
[2] Cliff wootton A practical guide to video and audio compression
[3] Mark Nelson, Data compression , 2nd ed.
[4] David salomon, Data Compression 1995, 2nd ed., The Complete reference
[5] Stephen J.Solari Digital Video and Audio Compression
[6] A moffat, R.Neal, and I. H Witten, " Arithmetic coding revisited", in proc. Data Compression Conf., 1995,pp.202-211
[7] Y.Xie, W.Wolf, and H.Lekatsas, " Code Compression for VLIW processors using variable-to-fixed coding", in proc.Int.Symp.Syst.Synth., 2002, vol 2, no.04, pp.138-143

AUTHORS PROFILE

Deepak Upadhyay received his Bachelor of Technology (B.Tech) degree in Information Technology in 2011 from Govt. Engineering college Ajmer Rajasthan, India. He is currently working towards Master of Technology (M.Tech) degree in Information Technology at School of Information Technology (SOIT) from Center for Development of Advance Computing (C-DAC), Noida, U.P. (India) under the Ministry of Communications and Information.

Banwari received his Bachelor of Technology (B.Tech) degree in Computer Science and engineering in 2010 from Maharaja Agrasen Institute of Technology (MAIT) Delhi, India. He is currently working towards Master of Technology (M.Tech) degree in Information Technology at School of Information Technology (SOIT) from Center for Development of Advance Computing (C-DAC), Noida, U.P. (India) under the Ministry of Communications and Information Technology, Government of India. His area of research includes Software Engineering, Computer Network's Management and Security, Algorithms, Mobile computing and Artificial intelligence.

Deepanshu Sharma received his Bachelor of Technology (B.Tech) degree in Information Technology in 2011 from Mahatma Gandhi Mission's College of Engineering & Technology, Noida, U.P. (India). He is currently working towards Master of Technology (M.Tech) degree in Information Technology at School of Information Technology (SOIT) from Center for Development of Advance Computing (C-DAC), Noida, U.P. (India) under the Ministry of Communications and Information Technology, Government of India. His area of research includes Software Engineering and Computer Network's Management and Security.