I.J. Engineering and Manufacturing, **2018, 2, 56-65**
Published Online March 2018 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijem.2018.02.06

Available online at http://www.mecs-press.net/ijem

Performance Optimization of Codec in VOIP using Raspberry Pi

Gomathinayagam.P [a], S. Jayanthy [b]

[a] *Student, ME Embedded System Technologies, Sri Ramakrishna Engineering College, Coimbatore-641022, Tamil Nadu.*
[b] *Professor, Department of Electronics and Communication Engineering, Sri Ramakrishna Engineering College, Coimbatore-641022, Tamil Nadu.*

Received: 27 June 2017; Accepted: 11 September 2017; Published: 08 March 2018

Abstract

This paper aims in design and implementation of Voice over Internet Protocol (VoIP) system with a high compression rate. The system uses a Raspberry Pi B+ ARM 11 microcontroller which is faster in processing the audio signal by means of CODEC2 software. An Audio Adapter (CM-108 or 109) is interfaced with the Raspberry Pi, which acts as a bridge in providing the processed voice signal to Raspberry Pi in its required PCM format. The setup is implemented in Local Area Network (LAN) in which the data transmission occurs. During the transmission of the data, the wireshark software was used to analyze the data in the network. A detailed analysis of the comparison results of different CODECs for compression rate and the estimated frame sizes are given. From the results, it is proved that the use of CODEC2 is an efficient system for VoIP. For secure communication, the AES encryption protocol with static private key was implemented. The 3.2Kbps compression technique is suitable for VoIP system as it has high compression rate and less delay compared to the other systems.

Index Terms: CODEC2, VoIP, Raspberry Pi, ALSA, Compression Rate, Delay.

1. Introduction

Transmission of audio signals through Internet has become common technology. Further VoIP services are not only confined to the wired domain but also available across wireless networks (Sheetal Jalendry, Shradha Verma(2015)). Transmission of voice signal has become a common application, but to have an enhanced VOIP performance the latency of transmission have to be reduced. Audio codecs should have low delay and high quality [Valin et al 2010]. The files may be shared as well as information may be changed during transmission. Therefore security techniques has to be implemented in VOIP systems [Simantini and Manish 2015] In this

* Corresponding author. Tel.: 9629432034
E-mail address: gomathinayagam71192@gmail.com, jayanthy.s@srec.ac.in

paper, a low latency and highly secured VOIP system is presented for wired networks. An encoded and compressed signal is transferred over Internet. The system consists of a Raspberry Pi computer with an Audio Adapter (CM-108 or 109). An audio driver based on Linux OS was developed.

Current VOIP systems cannot fulfill the demands in terms of audio quality and latency, making long-range communication sessions less pleasurable. For transmitting audio data over the Internet, the continuous audio stream is split into chunks of data [Website1]. These packets have to find their own way through the Internet, competing with packets from all the other Internet Applications. These packets have to be transmitted efficiently in order to obtain high data rate with low latency.

Normally to achieve a high data rate, large buffers are used that lead to a simulated delay of the packets. Hence, the number of bytes can be reduced by using data compression algorithm. However, most of the audio compression algorithms have a high intrinsic algorithmic delay and thus generate a high latency of audio. For standard VoIP [Carvalho and E. D. S. Mota(2015)] high latency is a moderate problem whereas for some applications exact timing of the audio signal is very important. The core of this embedded system is a Raspberry Pi, a credit-card sized single board computer running Linux. Many hardware and software components complement each other in this system. This includes audio adapter, headphone and mike for the audio hardware, for the voice extraction from the user. The hardware was implemented and the software parts such as a Linux kernel driver for interfacing the audio hardware and the application program for networked VOIP transfer with optimized performance was ported onto the target hardware.

2. Related Works

Xiaohe Wu et.al (2013) have discussed the general operation principle of a typical audio system module; where, a sample implementation that uses a dedicated voice chip is introduced and explained in detail so that it can be used by engineers as a functional reference design. An audio system in the embedded setting may consist of memory module, Compression/Decompression (CODEC) module, Digital-to-Analog Converter (DAC) module, filter module, power amplifier module; and also (if the function of voice recording is needed) the signal conditioning module, Automatic Gain Control (AGC), and Analog-to-Digital Conversion (ADC) module. Depending on the different demands of various application scenarios, different techniques can be used to implement the audio module. If no audio signal processing is required, the approach uses a dedicated, off-the-shelf, voice chip. The limitation of the system is the use of extra components like Voice Chip, ADC, DAC.

Yi Han et.al (2014) has discussed the opus CODEC adaptation for VoIP services. This paper presents an experimental evaluation of the recently standardized Opus CODEC used in a VoIP context. Opus operates in both narrow and wideband modes, similar to Adaptive Multi-Rate (AMR). Through the use of the Wideband Perceptual Evaluation of Speech Quality (WB-PESQ) metric, they have conducted an extensive set of experiments using multiple audio samples encoded at different bit-rates, to investigate the impact of packet loss on resulting speech quality. Using these results, fitting functions for each bit-rate were computed to provide a straightforward manner of evaluating speech quality when given a specified packet loss rate. Using NS-2, a simulation analysis has been conducted to evaluate the effect of background traffic on transmitted Opus streams. The limitation of the system is the use of the NS-2 simulator where the simulation of the data is only done.

Nithya and Manikandan(2014) have discussed the Performance analysis of CODEC's With QoS Constraints in Voice over Internet Protocol where they have used the OPNET for analysis of the network and get the required values for bit rate and frame size. The vital parameters to measure service quality are packet drop, Delay and Mean Opinion Score (MOS). They have implemented VoIP in a Local Area Network with Internet Protocol version 6 (IPv6) capabilities, analyse the popular voice CODECs using QoS parameters and identifying suitable codec techniques for VoIP traffic. OPNET simulator is used to generate real time traffic and simulate IPv6 capable network. The limitation of the paper is the use of the simulator during analysis.

To overcome the drawbacks of the above existing systems, the proposed project employs the use of simple open source CODEC2 that can be ported in embedded hardware. The system has overcome the abovementioned disadvantages of the existing works by the use of dedicated Audio chip for hardware analysis.

3. System Design

The proposed system has used USB voice input device for capturing audio, a USB microphone or sound card could be connected via USB port. A USB adapter was needed for connecting raspberry Pi and remote desktop and network. The SD card is installed with Raspberry Pi's operating system "Raspbian OS" [Website2]. An Ethernet cable is used to connect the Raspberry Pi with the computer for programming purpose. Then the Raspberry Pi's network interface settings are configured according to desired network. For analysis, a secure connection, with the help of SSH protocol using PuTTY software, was established between Raspberry Pi and the laptop. With the help of this connection, voice was recorded, played and streamed to and from the Raspberry Pi to a laptop using a BASH terminal.

3.1. Block Diagram of Implementation

The embedded system consists of the Raspberry Pi board with external SD card as memory along with the audio adapter interfaced with the system. All components needed for transferring an encoded and compressed voice signal over the Internet are integrated on a single Raspberry Pi with an Audio Adapter (CM-108 or 109). The system consists of two main components:

- Raspberry Pi board
- Audio adapter

Raspberry Pi board is the core module for voice capturing, playing and processing the audio signal. The board has ARM Cortex as the main processor, memory, power supply, HDMI Out, Ethernet port, USB ports and many global interfaces.

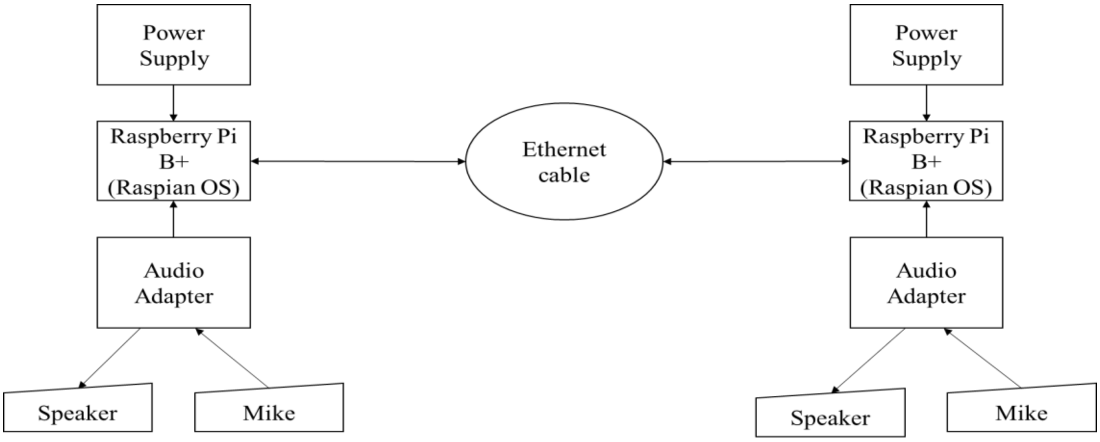


Fig.1. Block Diagram of the Implementation

Fig 1 shows the implementation of the proposed system. The system has the microphone and speaker to perform the input and output operations. Voice processing is done by the ARM Cortex processor present in the Raspberry Pi board.

3.2. Audio Capturing Component Design

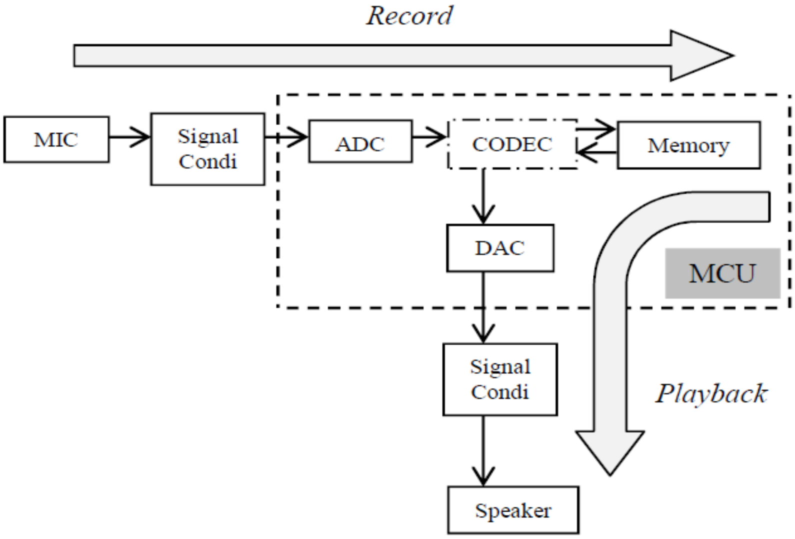


Fig.2. Operating Principle of Audio Module

The block diagram of a typical Audio Module is shown in Fig 2. The two operation modes are record mode and playback mode. The operation of the audio module is explained by breaking the system into these two modes.

3.2.1. Record Mode

In this mode, the voice signal is captured, digitized and coded. The coded signal is stored in the memory module. The air pressure variation caused by the voice signal is converted into electrical signal by the microphone. This electrical voltage thus obtained is filtered and amplified in the Signal Conditioning module. The amplified signal is converted into digital signal using an ADC module [6]. The digital data is then ported in the memory of the ARM Cortex processor. The data stored can be retrieved for further processing or playback by the application code.

3.2.2. Playback Mode

In this Mode, data stored in the Memory module in the record mode is retrieved and the coded data is decoded in the CODEC module and converted back into analog data by passing the data through a DAC module. The analog data is amplified and filtered using a Signal Conditioning unit and sent to a speaker for conversion into audio signal.

4. Software of the Proposed System

The software algorithm shown in Fig 3 describes the analysis of the voice packets in the network using the wireshark software. The voice packet can be analysed for different compression techniques. In the software segment the packet captured had to be analysed in the software following the steps shown below. The packets summary can be got by getting the input/output graph that is generated from the wireshark software from the host system. The capture mechanism has to be done on the host system so that it can help to generate the graph and also solve the respected IP address assigning problems.

STEP 1: Configure a static IP to the raspberry pi boards.
STEP 2: Connect the Boards to a network and check the connection.
STEP 3: Install the CODEC2 & speex in the raspberry pi boards.
STEP 4: Run the application program to check the network connection in the system.
STEP 5: Use the command code to perform the transfer of the data over the network. Add the AES algorithm with a static private key on both the systems.
STEP 6: The output voice can be heard over speakers.
STEP 7: Use Wireshark software to analyze the data packets for compression rate and latency.

Fig.3. Software Algorithm

4.1. Pseudocode

The Fig 4 gives the pseudo code of the process to setup the network connection for transferring packets.

```
start
set
    static IP
if IP is set
    start UDP protocol
if false
    "wrong IP"
set
    start Netcat protocol with assignments
    assign AES encryption with a static private key
    start the 2nd system first
            if TRUE
                then start 1st system
            set
            if TRUE
                    decrypt the data with key
                    connection is established
            else
                    retry
else
end of program
```

Fig.4. Pseudo Code for Network Connection

5. Experimental Results

The proposed VoIP system is interfaced with Raspberry Pi and Audio adapter. Fig 5 shows experimental setup of the system. It consists of the Raspberry Pi Model B+ board which has four USB ports and one Ethernet port. The device can be externally powered using a 5V adapter to a maximum of 2A current [Website3]. The device has an ARM controller which can be used to run Linux and other OS in the device. The USB device is connected with the audio adapter which is connected with the speaker and mike.

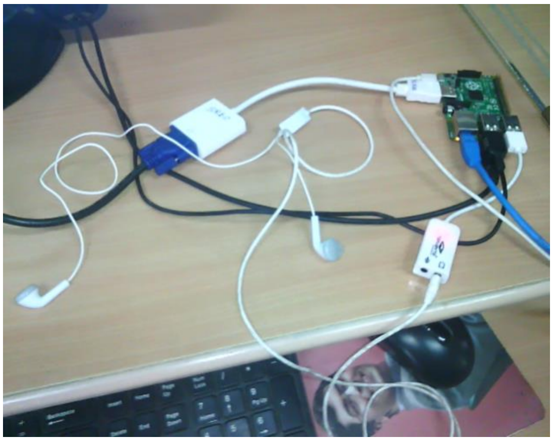
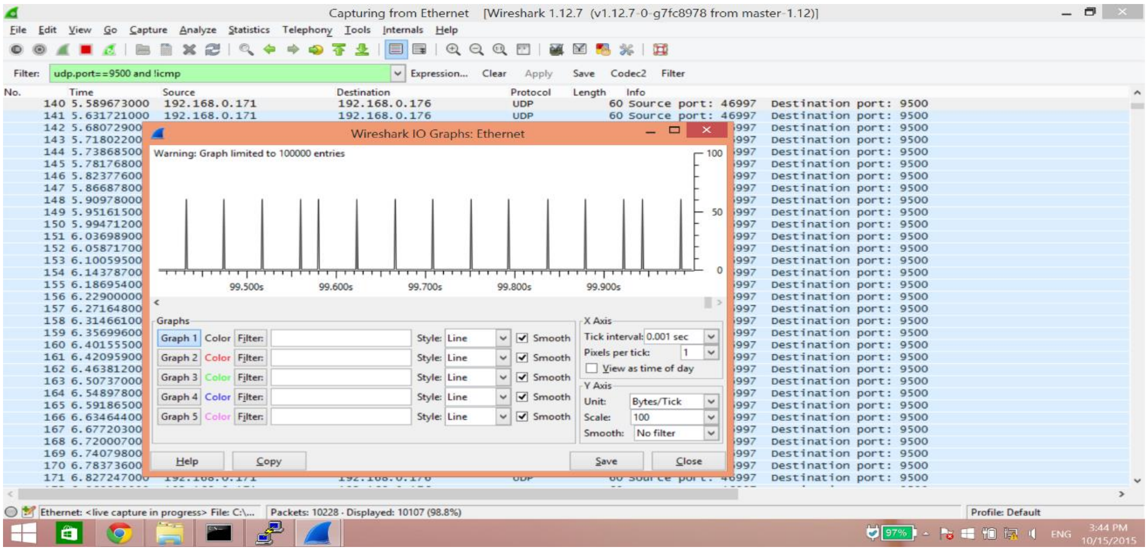


Fig.5. Experimental Setup of the System

5.1. Capturing of Packets


Capturing from Ethernet [Wireshark 1.12.7 (v1.12.7-0-g7fc8978 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save Codec2 Filter
Wireshark IO Graphs: Ethernet
100
50
0
99.500s
99.600s
99.700s
99.800s
99.900s
Help
Copy
Save
Close
ENG
Profile: Default

Filter: udp.port==9500 and !icmp

```
No.      Time          Source           Destination      Protocol  Length  Info
    140 5.589673000   192.168.0.171    192.168.0.176    UDP       60 Source port: 46997  Destination port: 9500
    141 5.631721000   192.168.0.171    192.168.0.176    UDP       60 Source port: 46997  Destination port: 9500
    142 5.68072900                                                            997  Destination port: 9500
    143 5.71802200                                                            997  Destination port: 9500
    144 5.73868500                                                            997  Destination port: 9500
    145 5.78176800                                                            997  Destination port: 9500
    146 5.82377600                                                            997  Destination port: 9500
    147 5.86687800                                                            997  Destination port: 9500
    148 5.90978000                                                            997  Destination port: 9500
    149 5.95161500                                                            997  Destination port: 9500
    150 5.99471200                                                            997  Destination port: 9500
    151 6.03698900                                                            997  Destination port: 9500
    152 6.05871700                                                            997  Destination port: 9500
    153 6.10059500                                                            997  Destination port: 9500
    154 6.14378700                                                            997  Destination port: 9500
    155 6.18695400                                                            997  Destination port: 9500
    156 6.22900000                                                            997  Destination port: 9500
    157 6.27164800                                                            997  Destination port: 9500
    158 6.31466100                                                            997  Destination port: 9500
    159 6.35699600                                                            997  Destination port: 9500
    160 6.40155500                                                            997  Destination port: 9500
    161 6.42095900                                                            997  Destination port: 9500
    162 6.46381200                                                            997  Destination port: 9500
    163 6.50737000                                                            997  Destination port: 9500
    164 6.54897800                                                            997  Destination port: 9500
    165 6.59186500                                                            997  Destination port: 9500
    166 6.63464400                                                            997  Destination port: 9500
    167 6.67720300                                                            997  Destination port: 9500
    168 6.72000700                                                            997  Destination port: 9500
    169 6.74079800                                                            997  Destination port: 9500
    170 6.78373600                                                            997  Destination port: 9500
    171 6.827247000   192.168.0.171    192.168.0.176    UDP       60 Source port: 46997  Destination port: 9500
```

Warning: Graph limited to 100000 entries

Graphs

Graph 1	Color	Filter:		Style: Line	☑ Smooth
Graph 2	Color	Filter:		Style: Line	☑ Smooth
Graph 3	Color	Filter:		Style: Line	☑ Smooth
Graph 4	Color	Filter:		Style: Line	☑ Smooth
Graph 5	Color	Filter:		Style: Line	☑ Smooth

X Axis
Tick interval: 0.001 sec
Pixels per tick: 1
□ View as time of day

Y Axis
Unit: Bytes/Tick
Scale: 100
Smooth: No filter

Ethernet: <live capture in progress> File: C:\...
Packets: 10228 · Displayed: 10107 (98.8%)

97% 3:44 PM 10/15/2015

Fig.6. Co-ordinates Plotted in Wireshark during Execution

Wireshark is an open source network tool. Using the wireshark software the system can capture flow of the packets. In the transmitting side Raspberry Pi is used as a transmitter and at the receiver end the Computer is used to monitor the data packets.

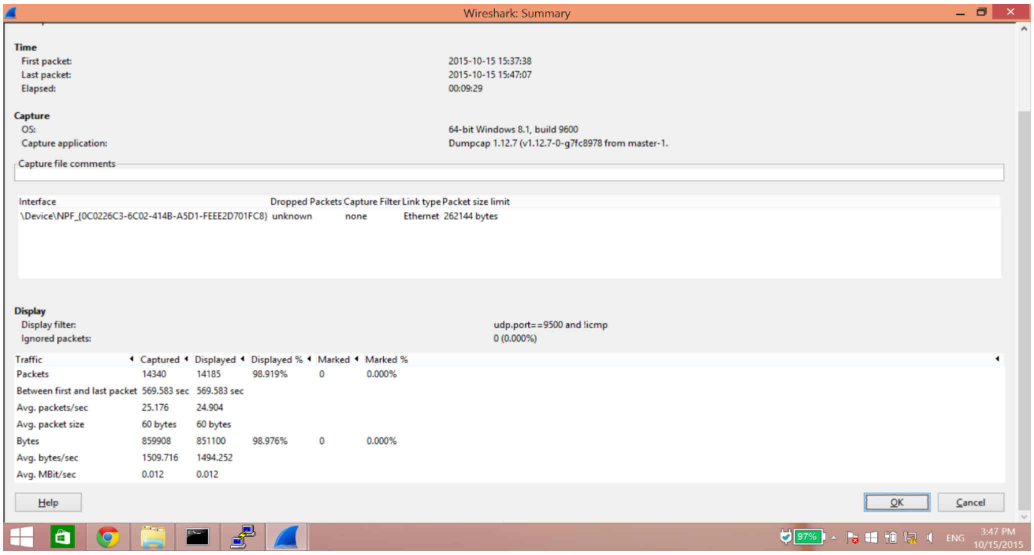
The graph plotted in the wireshark shows the packets transmitted per second over the network using CODEC2 algorithm. The Fig 6 shows the results of the proposed system with a compression rate of 3.2Kbps achieved by use of CODEC2 software. The parameters shown in the image include Tick interval, Scale, Graph style for X and Y axis.

5.2. Summary of the Analysis in Wireshark

Wireshark software can give a complete analysis summary of the data packets movement over the network card, thus can be used to analyse the data flow over the network. The use of the analysis summary is to evaluate and use a proper CODEC for the proper application. The data frame can give the details of the data packets and overhead of the audio packets.

Results are analyzed from the output summary of wireshark software to examine the performance of the network using different CODECs.

In the proposed system, CODEC2 is implemented in the hardware whereas in the related works analysis is done mostly using simulators. Therefore, this is a real time hardware implementation, which can be used for analysis in the real time environment. Wireshark tool can give the detail summary, IO graph and plot of the packets in the network. All the UDP packets in the network are evaluated. The analysis helps to estimate the overhead and the payload of the data in the network.



Time

First packet:	2015-10-15 15:37:38
Last packet:	2015-10-15 15:47:07
Elapsed:	00:09:29

Capture

OS:	64-bit Windows 8.1, build 9600
Capture application:	Dumpcap 1.12.7 (v1.12.7-0-g7fc8978 from master-1.

Capture file comments

Interface	Dropped Packets	Capture Filter	Link type	Packet size limit
\Device\NPF_{0C0226C3-6C02-414B-A5D1-FEEE2D701FC8}	unknown	none	Ethernet	262144 bytes

Display

Display filter:	udp.port==9500 and !icmp
Ignored packets:	0 (0.000%)

Traffic	Captured	Displayed	Displayed %	Marked	Marked %
Packets	14340	14185	98.919%	0	0.000%
Between first and last packet	569.583 sec	569.583 sec			
Avg. packets/sec	25.176	24.904			
Avg. packet size	60 bytes	60 bytes			
Bytes	859908	851100	98.976%	0	0.000%
Avg. bytes/sec	1509.716	1494.252			
Avg. MBit/sec	0.012	0.012			

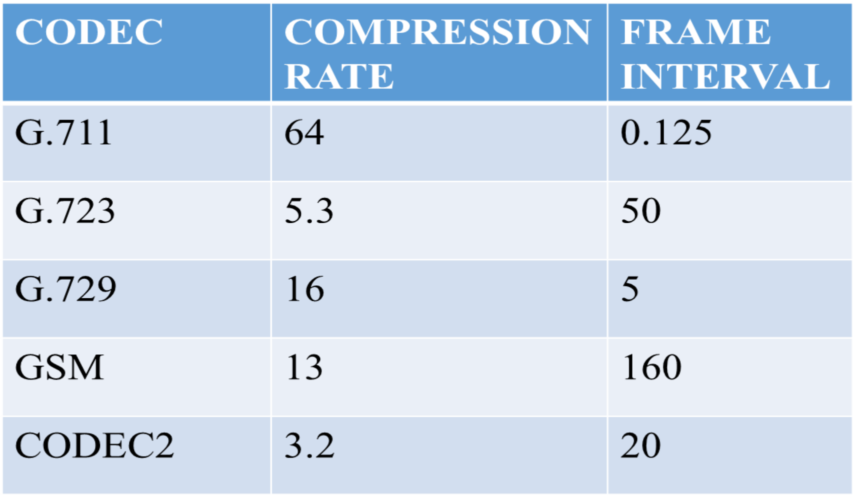
Fig.7. Summary of the Analysis

The GUI of the wireshark is given in the Fig 7. The use of the summary helps to get the details of all the information about the packets in the network which include the bit rate, average packet size, average bit size, bytes of data etc.

5.3. Result Analysis

In the paper [Priyanka Grover, Meenakshi Chawla(2015)], the different CODECs using simulators have been analysed. But in this paper implementation of CODEC2 is done in the hardware and analysis is carried over real time environment. The system implementation gives an upper hand over the existing simulation outputs. Table 1 gives the compression rate and frame interval of various CODECs. The result output shows that the proposed CODEC2 with compression rate of 3.2Kbps has a higher compression rate compared to other systems, which is the main advantage, but with a slightly larger frame interval. The important feature to be noted is that the size of the output data using the proposed CODEC2 is very minimum, so the buffer filling time for the protocol will be high than the other systems.

Table 1. Attributes of CODECS



CODEC	COMPRESSION RATE	FRAME INTERVAL
G.711	64	0.125
G.723	5.3	50
G.729	16	5
GSM	13	160
CODEC2	3.2	20

G.711 is an ITU standard codec with a high bit rate of 64 Kb/s. It uses 8-bit pulse code modulation (PCM) encoding scheme with a sampling rate of 8000 samples per second. There is no need for buffering and hence has the lowest latency (lag) thus increasing the processing power. The drawback is that compared to other CODEC's it consumes more bandwidth.

G.723 is an ITU standard codec. It has a reasonably low compression rate (6.3Kb/s or 5.3Kb/s) and used in H.323 standard. It requires licensing by Sipro Lab Telecom.

Another ITU standard codec is G.729. With a sensibly low bit rate of 8Kbps it offers a quality speech. The disadvantage is that it has increased CPU processing time, which allows only some VoIP phones and adapters to handle only one G.729 channel at a time. The problem with standard is if the user places concurrent calls on both lines of a two-line device call drops can occur.

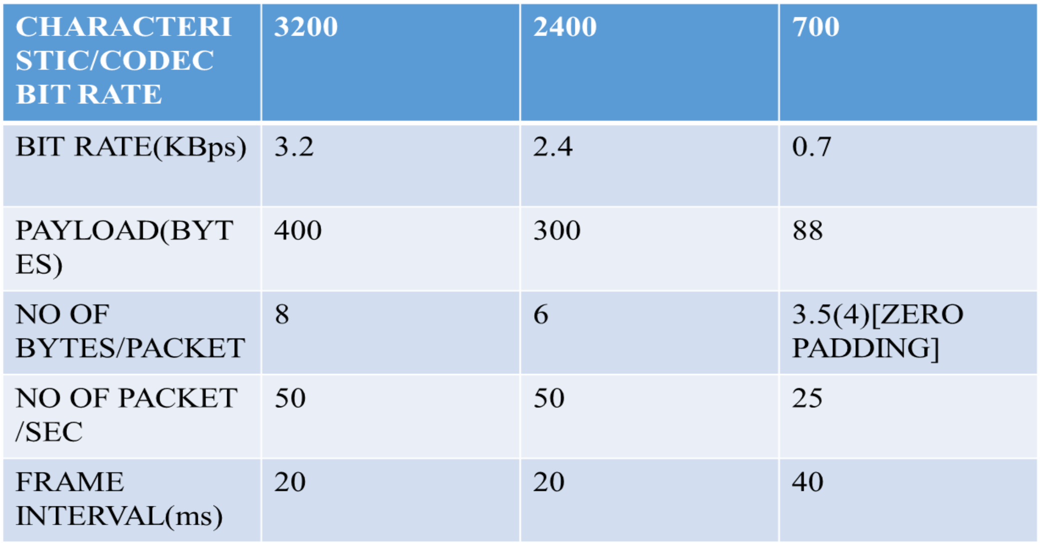
The proposed CODEC2 is an open source CODEC which is free of license and patent problems. Compared to other CODEC types the CODEC2 can produce (3.2kb/s to .7 Kb/s) of data and has minimum latency effects.

The proposed CODEC2 achieves a higher compression rate, but it has a higher frame interval. But by using proper buffer value calculations, the nominal value for latency can be easily calculated. The frame interval is a fixed value because it is the time taken for each frame to be filled with payload and overhead. The use of high compression rate will increase the overhead value in the system but will reduce the payload size.

The 3.2Kbps CODEC2 has a payload of 400 bytes but it has 2600 bytes of overhead data. This simple CODEC2 3.2Kbps can be used in wired base network. Thus, the 3.2Kbps compression rate of CODEC2 is suitable for the VoIP based real time system.

The least compression rate of the CODEC2 is the 700bps method. The data payload of the system is just 88 bytes but the overhead is much larger in value. So this system is not suitable for the VoIP based system on wired network. The 700bps can only be used in a system, which uses less overhead because the payload is just 88 bytes. Thus, this is suitable for the wireless based protocol system.

Table 2. CODEC2 Analysis



CHARACTERISTIC/CODEC BIT RATE	3200	2400	700
BIT RATE(KBps)	3.2	2.4	0.7
PAYLOAD(BYTES)	400	300	88
NO OF BYTES/PACKET	8	6	3.5(4)[ZERO PADDING]
NO OF PACKET /SEC	50	50	25
FRAME INTERVAL(ms)	20	20	40

Table 2 gives a detailed value of the CODEC2 analysis in the network. The output gives a clear proof that the CODEC2 can be used in both wired and wireless systems. The 3.2kbps system can be used in VoIP as it can reduce the data rate and the payload and overhead ratio is nominal, but the 700 bps process has a very less payload with high overhead data for the network. Thus we can use the 700 bps type compression technique for wireless radio application.

6. Conclusion and Future Work

The experimental results confirmed that the transmitter and receiver were working properly and that data could be sent in real time. The assessment of the audio components shows a good overall audio performance. The 3.2kbps compression rate is implemented in the VoIP based system for wired communication. The 700 bps compression technique can be implemented in wireless based systems. Due to widespread use of VOIP security problems arise [Urjashee Shaw and Bobby Sharma, 2016].To provide security feature to the voice, the AES encryption algorithm is implemented.

The future work will include the use of QoS analysis to find the mean value of the system to evaluate it with the present existing systems.

References

[1] Han Yi, Damien Magoniy, Patrick McDonaghz and Liam Murphy. “Determination of Bit-Rate Adaptation Thresholds for the Opus Codec for VoIP Services”.Computers and Communication (ISCC), IEEE Symposium, 2014.

[2] Nithya. S, and Manikandan M.S.K. “Performance Analysis of CODEC's With QoS Constraints in Voice Over Internet Protocol V6”. International Conference on Electronics and Communication System -2014.

[3] Carvalho and E. D. S. Mota, “Survey on application-layer mechanisms for speech quality adaptation in VoIP”. The Eleventh Advanced International Conference on Telecommunications, 2015.

[4] Simantini J. Shivankar and Manish P. Tembhurkar, "Comparative analysis on Security Techniques in VoIP Environment". IEEE SPONSORED 2ND International Conference on Electronics and Communication System, 2015.

[5] Xiaohe Wu, Morrison Obeng, Jing Wang and Daniel Kulas "A Survey of Techniques to Add Audio Module to Embedded Systems". Southeastcon, Proceedings of IEEE, 2012, 978-1-4673-1374-2.

[6] Valin, J.-M. Terriberry, T. Montgomery, C. and G. Maxwell , "A high quality speech and audio codec with less than 10-ms delay," IEEE Transactions on Audio, Speech, and Language Processing ,2010,vol. 18, no. 1,pp. 58–67.

[7] Priyanka Grover, Meenakshi Chawla" Performance Analysis Of VOIP Codec's With QoS Parameters" International Journal of Advance Foundation and Research in Computer (IJAFRC) ,2015,Volume 2, Issue 5.

[8] Sheetal Jalendry, Shradha Verma" Performance Analysis of various Codecs Schemes of VOIP over WiMAX" International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), 2015,Volume 4 Issue 6.

[9] Urjashee Shaw and Bobby Sharma "A Survey Paper on Voice over Internet Protocol (VOIP)" International Journal of Computer Applications (0975 – 8887) April 2016, Volume 139 – No.2.

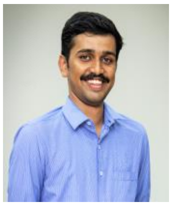
[10] website1:http://blog.marxy.org/2012/12/codec2-and-modem-on-raspberry-pi.html

[11] website2::http://www.klaverstyn.com.au/david/wiki/index.php?title=Asterisk_for_Raspberry_Pi_on_Raspbian

[12] website3::www.raspberrypi.org

[13] website4::http://www.rowetel.com/blog/?p=2255

Authors' Profiles



P.Gomathinayagam has received his Bachelor's Degree in Electronics and Communication Engineering. He has done his Master's degree in Embedded System Technologies at Sri Ramakrishna Engineering College, Coimbatore. He is interested in doing research in embedded field.

Dr. S. Jayanthy is working as a Professor in the Department of Electronics and Communication Engineering at Sri Ramakrishna Engineering College, Coimbatore. She has 20 years of experience in teaching. Her research interests are in VLSI Design & Testing and Embedded systems. She has published many research papers in Journals and Conferences. Guiding research scholars in the area of VLSI and Embedded systems.

How to cite this paper: Gomathinayagam.P, S. Jayanthy,"Performance Optimization of Codec in VOIP using Raspberry Pi", International Journal of Engineering and Manufacturing(IJEM), Vol.8, No.2, pp.56-65, 2018.DOI: 10.5815/ijem.2018.02.06