Experimental Evaluation Platform for Voice Transmission Over Internet of Things (Voiots)

[1]Qutaiba Ibrahim Ali, [2]Ina'am Fathi, [3]Jassim M. Abdul-Jabbar

[123]Computer Engineering Mosul university Mosul/Iraq

qut1974@gmail.com, inamfth@gmail.com, drjssm@gmail.com

Abstract

With the revolutionary development of Wireless Sensor Network (WSN) technologies, researchers started to take benefits of integration embedded low-cost, low-power WSN technology in a various IoTs applications. Real-time voice transmission over IoTs is one interesting application that began to be explored by many researchers for a wide range of emergency scenarios. Thus, this paper presents a performance study for transmission of voice over WSN (VoWSN) with and without presence of Internet. A framework using a Raspberry Pi3 (RPi3) and open source FFmpeg technology for processing, compressing and streaming voice to a remote computer is proposed, implemented and evaluated. The performance of the proposed framework is evaluated by studying its behaviour utilizing three audio encoding algorithms: AC3, MP3 and OPUS with different sampling rates. Furthermore, a set of evaluation metrics procedures such as: One-way delay, jitter, Bandwidth (B.W), CPU usage and packet losses are proposed and implemented effectively.

Keywords: IoTs, VoWSN, Raspberry Pi, Audio encoding, Real time voice streaming, Embedded systems.

Subject Classification: E.g., Mathematics Subject Classification; Library of Congress Classification

Type (Method/Approach): Provide examples of relevant research types, methods, and approaches for this field: E.g., Historical Inquiry; Quasi-Experimental; Literary Analysis; Survey/Interview

1. Introduction

Wireless voice transmission represents an expeditious communications mechanism and therefore unsurprising that this mechanism is always wanted to be exploited it in a wide range of emergency scenarios, where the time is a dominant factor. For many years the high cost and high-power technologies such as interphone, mobile telephone, microwave, and satellite have been utilized to accomplish wireless voice communication system. An alternative approach can be contributed by using new advent embedded low-cost, low-power WSN technology [1]. The vast upgrades in low-power computing, networking and microphone technologies have provided WSN devices more capabilities for real-time voice transmission. Consequently, the researchers have started experiencing the advantages of real time transmission of voice over WSN (VoWSN) in various fields such as military applications, surveillance and emergency scenarios.

Generally, IoT can be characterized as a type of a network that associate anything from anywhere and anytime with the Internet based on predefined protocols through sensing devices. Embedded systems are now assuming an essential part in the advancement of the IoTs. Recent expectations guarantee that there will be more than 50 billion associated devices at 2020[2]- [3]. These devices are expected to be typical IoTs devices, such as small-size, low-cost and low-power networked embedded devices. These networked embedded devices may be characterized as a WSN. Thus, it may be thinking about the WSN as a subset of IoTs that connecting things to the Internet through a gateway as shown in Fig. 1 [3]- [4].

While most of the traffic generated by IoTs devices are management data or measurements data (temperature, humidity,), it is expected that by 2019 the multimedia traffic will account for 80% of all Internet Protocol (IP) traffic [2]. The integration of VoWSN with the IoTs enables the audio data can be accessed from anywhere in the world in any time. In emergency scenarios, VoWSN with IoTs can play significant role in saving lives. Moreover, it might offer a flexible technique to give human interaction. Additionally, voice can give a more adaptable user experience involvement in a more conservative manner than customary techniques like touch screens or data input [5].

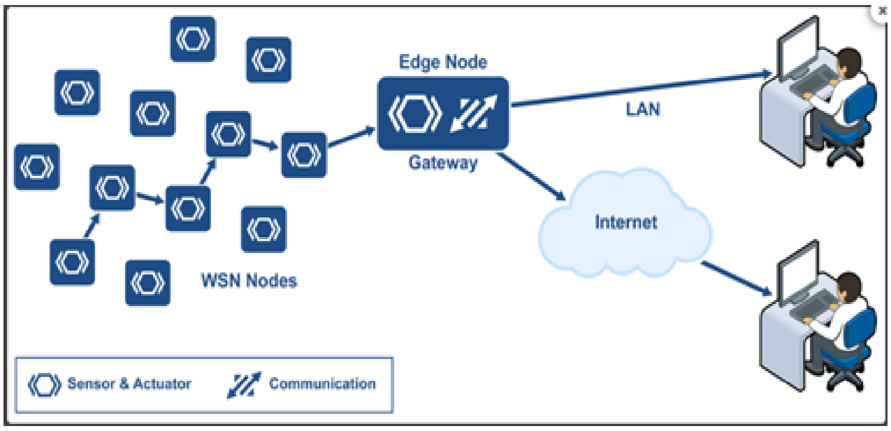


Fig.1. WSN with IoT system Architecture.

So, in this paper we explore the feasibility of streaming voice over IoTs in real-time constrains where jitter<5ms and one-way delay<150 ms utilizing three audio encoding algorithms. The main contribution of this project is proposing and implementing a procedure to calculate jitter and one-way delay factors utilizing Real-time Transport Protocol (RTP), Real Time Control Protocol (RTCP) headers. These factors are important factors for evaluating the real-time performance of such systems. The procedure is based on the synchronization between the WSN node and remote PC which achieved by Network Time Protocol (NTP).

This paper is organized as follows: section I presents an introduction of VoWSN and Voice over IoTs (VoIoTs) systems. Section II includes a brief literature review related to VoWSN and Voice VoIoTs subjects. Section III gives an overview of streaming protocols and MPEG-2 TS container. Our proposed framework with hardware and software implementation methodology is given in section IV and V respectively. Section VI presents testing methodology while section VII reviews testing results and discussion of VoWSN and VoIoTs system. Section VIII gives an outdoor VoWSN case study. Finally, Section IX summarizes conclusions and research findings.

I. Literature Review

In fact, VoWSN issue has taken a great deal of researcher's interest for many years. Generally, in the beginning the researcher's efforts were mainly focused on feasibility of implementing and identify the problems to be solved related to VoWSN subject. However, in research [1] the authors I. Fathi et. al. aimed to give a brief study of the previous works concerned with the subject of VoWSN. The research showed that the early attempt to stream VoWSN was implemented using Fire-Fly nodes by R. Mangharam et. al. [6]. The system operated in a global Time division Multiple Access (TDMA) schedule and used Adaptive Differential Pulse Code Modulation (ADPCM) as a software compression technique. While in [7], D. Brunelli et. al. investigated Zigbee networks for voice streaming and analysed performance using metrics such as throughput, packet loss and jitter and compressed audio data using an external ADPCM processor. Also, in [8] the authors, H. Rong-lin et. al. presented an architecture of the VoWSN node and performed ADPCM encoding by DSP processor. In [9], the authors designed a smart helmet as a mobile sensor node and used a ZigBee WSN for voice transmission with a G.726 voice codec algorithm. While in [10], the authors developed a voice compression algorithm comparable to ADPCM, to fit in low memory profile 8-bit microcontroller. However, all the above listed previous researches have been studied the transmission of VoWSN without Internet.

The area of audio streaming in the IoTs using resource constrained hardware is however less explored. S. Haritha et. al. presented efficient audio streaming using Raspberry pi as a broadcaster which encoded audio into mp3 file format with bit rate at 96kbps and 64kbps. The audio could be obtained to the listener at a particular URL with a unique mount point [11]. Additionally, an audio streaming application that could stream data to a set of smart speaker nodes over wireless links based on IEEE 802.11n was implemented on a proposed platform using the Raspberry Pi2 in [12].

II. Streaming Protocols and Containers

The most popular multimedia streaming protocols are Real-time Transport Protocol (RTP), Real Time Control Protocol (RTCP) [13]. RTCP is a companion protocol to RTP carries a feedback from the receiver to the sender concerning quality of service (QoS) statistics. In particular, RTP provides a set of services concerning of real time transportation of audio and video data. These services are identified in the RTP header and include:

- Payload type identification: To indicate the encoded format of the payload.
- Sequence numbering: For packet de-sequencing.
- Time stamping: For jitter calculation.

RTP payload format is identified by a 7-bit numeric identifier. Codes below 96 are assigned statically, while codes in the range 96-127 are assigned dynamically by means outside of RTP profile or protocol specification (SDP).

FFmpeg technology can encode to a wide variety of lossy audio formats such as AC3, MP3, OPUS and so on. These techniques are interpreted as dynamic payload in the RTP header and assigned to code 96. As a consequence, FFplay (media player tool at the receiver side) cannot play the stream without an SDP file associated with it. Also, for correctly calculation of jitter and one-way delay, the sampling rate must be known. But with dynamic payload types this is hard as the codec type and its sampling rate need to be known and this requires that the session information must be in the trace and the program need to know the utilized codec. To overcome this situation, the encoded audio data can be wrapped into a standard container.

MPEG-2 Transport Stream (MPEG-2 TS) is a popular container formats specified in MPEG-2 Part 1 Systems and used for less reliable transmission as well as storage of audio, video and metadata [14]. Generally, Transport Stream characterized by several important features:

- Support error checking/correction mechanism.
- Stream synchronization capability.
- Multiplexing: It can transfer several 'compressed media formats' in a single container.
- Independency of the media format (encoded data).
- Simple and easy to implement, process and handle MPEG2-TS in the wireless network.

These features give the transport stream robustness in streaming applications. This robustness of MPEG-TS is one reason why it is widely used in environments where errors are likely occurring i.e., broadcast systems [15]. However, only certain audio CODECS (a software that used to compress and decompress) can be fit into MPEG2-TS.

III. VoIoTs System Hardware Implementation Methodology

The main VoIoTs system hardware components are shown in Fig. 2. In this system RPi3 model B is chosen as a WSN node. Unfortunately, the RPi3 doesn't come with a built-in audio input and setting one up is not straight forward. So, RPi3 is equipped with an external USB soundcard. This audio data is transferred through Wi-Fi network to a remote PC (laptop) which connected to the AP to receive the audio stream and reconstruct the original audio information. Although different systems for voice transmission using technologies like radio waves and Bluetooth, but compared to IEEE 802.11 WLAN, these technologies have several limitations in range and security issues [16]. So, the main motivation for utilizing Wi-Fi networks for VoWSN is the potential infrastructure deployment cost saving, ease of installation, extendibility to the internet, portability, and data rates which is in the order of 11 Mbps [17]- [18].

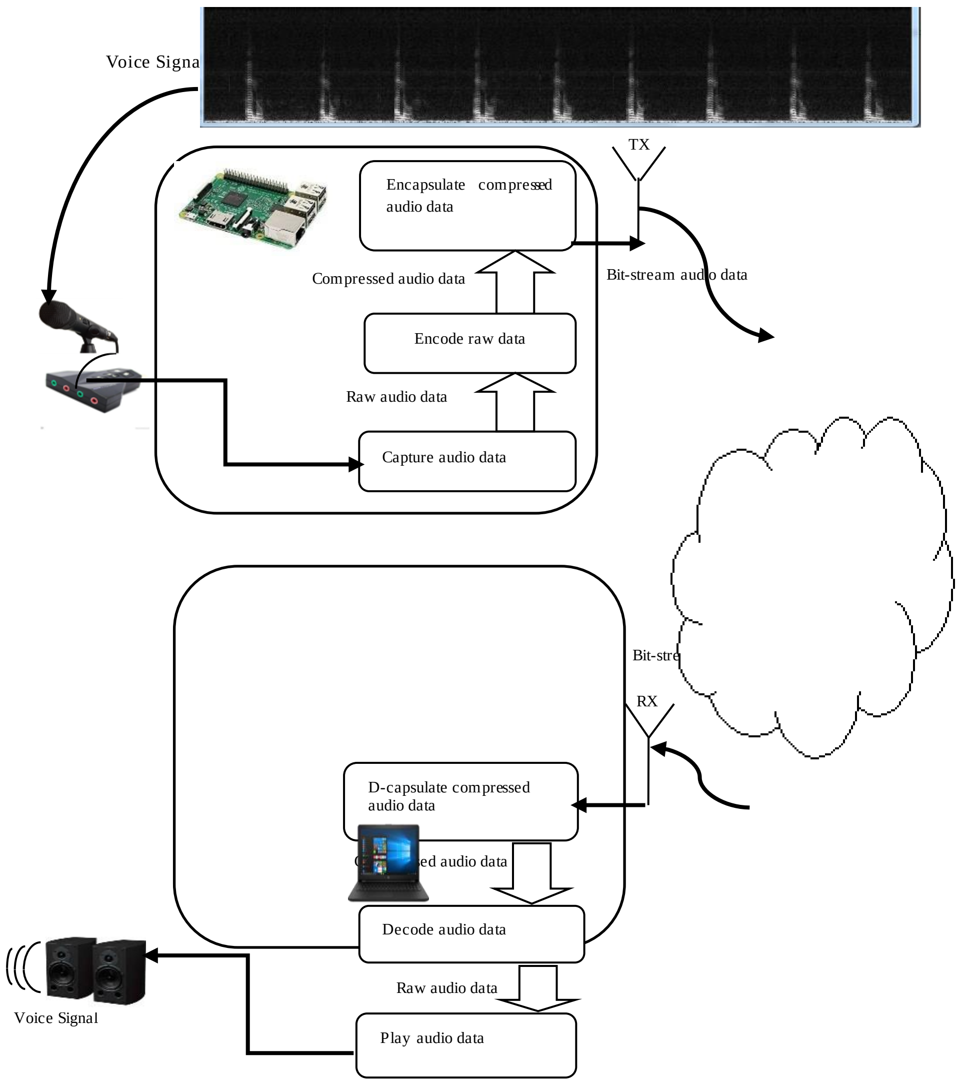


Fig. 2. VoWSN System Hardware Implementation.

IV. VoIoTs System Software Implementation Methodology

Software implementation methodology of VoIoTs system composes of two phases: System Software Initialization Phase and System Software Operation Phase. Generally, system software initialization includes the installation and configuration of the required software at the sender and receiver devices. The sequence of operations related to system initialization is shown in Fig. 3.

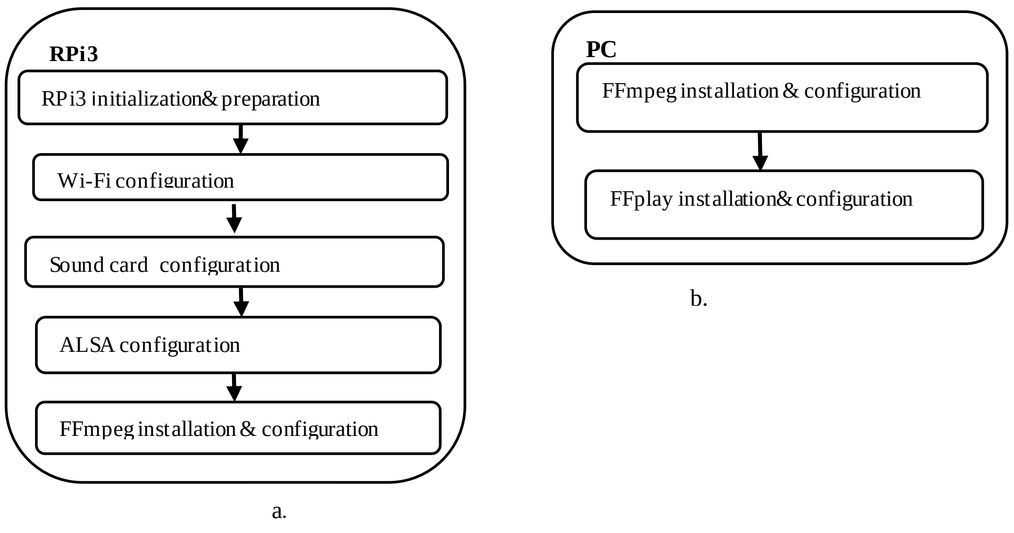

RPi3
RPi3 initialization& preparation
Wi-Fi configuration
Sound card configuration
ALSA configuration
FFmpeg installation & configuration
a.
PC
FFmpeg installation & configuration
FFplay installation& configuration
b.

Fig.3. System Initialization Phase: a. RPi3 b. PC.

After completing system initialization, the system can be operated to accomplish what we designed it for; i.e. ready to go into System Software Operation Phase. At the sender side (i.e. RPi3 where the source of audio is placing close to it), the operations are carried out in accordance with the sequence shown in Fig. 4. For voice streaming, this paper presents unreliable transport mechanisms involving combinations of MPEG-2 TS container, RTP and User Datagram Protocol (UDP). To keep the cost low, the operating system used within this research is the Raspbian and the device driver is Advance Linux Sound Architecture (ALSA). Consequently, on the receiving side (i.e. remote PC) the opposite sequence is followed by FFplay software media player to get the original audio to be playing back by the speakers.

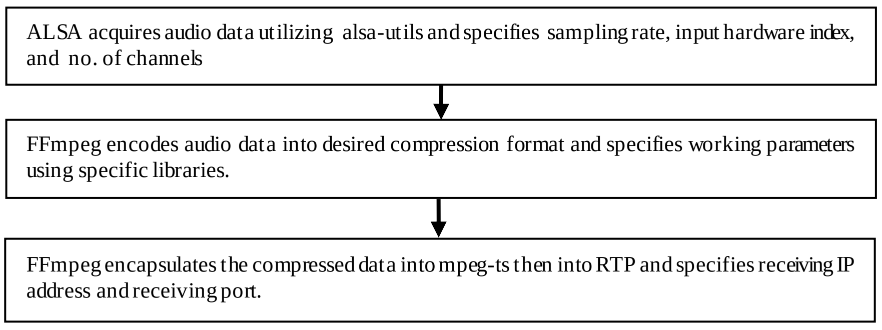

ALSA acquires audio data utilizing alsa-utils and specifies sampling rate, input hardware index, and no. of channels
FFmpeg encodes audio data into desired compression format and specifies working parameters using specific libraries.
FFmpeg encapsulates the compressed data into mpeg-ts then into RTP and specifies receiving IP address and receiving port.

Fig. 4. System Initialization Phase at RPi3.

V. VoIoTs System Testing Methodology

The work methodology employed by studying and evaluating the effect of sampling frequency, in mono and stereo states, on the behaviour of each encoding technique. The most important metrics utilizing for evaluation are:

CPU usage: describes how much the processor is working. Typically, CPU usage is not constant and varies depending on the type of operations that are being performed. CPU usage of RPi3 is measured using one of the following methods:

1. Using proc/stat file.
2. Using top tools.

3. Using htop tools.

The first method is a file created and modified during the operation of the CPU. This file can be verified during the streaming operation to acquire the CPU utilization. While the second and the third methods are command-line tools. These tools show the CPU utilization of the four cores as a percentage as well as show the memory usage.

One-way delay: One-way-delay is the delay introduced by the transmission of packets from source to destination. As illustrated in Fig. 5.

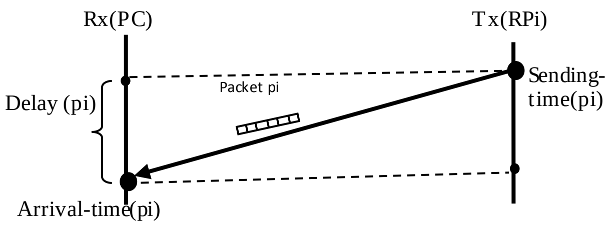
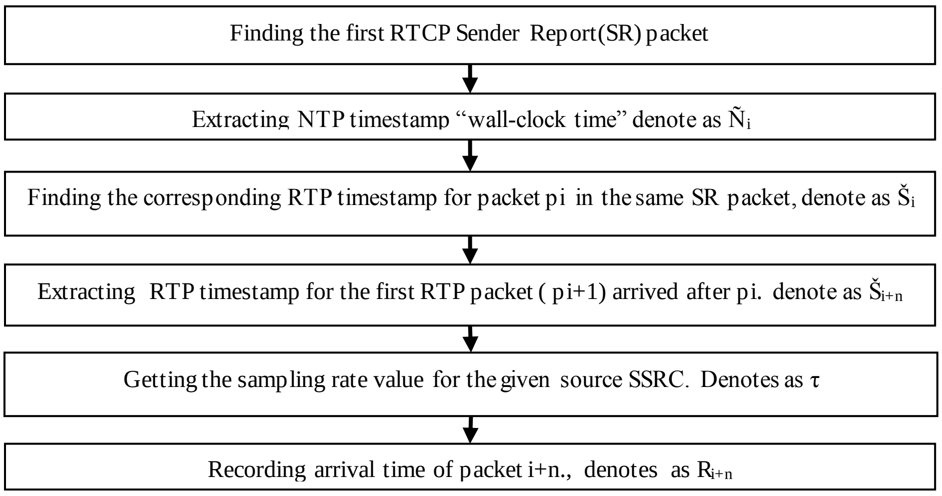


Fig. 5. Packet transmission delay.

So, the delay for a packet pi represents the difference between the time this packet arrived to the PC and the time this packet was sent from RPi3 as expressed in (1):

Delay(pi) = arrival-time(pi) – sending-time(pi) (1)

Using Wireshark program, arrival-time(pi) in time units can be extracted from a packet stream. While sending-time(pi) need to be calculated as it is not determined. To calculate sending-time(pi), RTP timestamps of RTP packets is utilized. RTP timestamps are randomly-initiated incremental units [19]. However, we cannot directly use the RTP timestamps and we need to convert sending-time(pi) to time units (wall-clock time). To do so, a reference wall-clock time is needed. Given a stream of RTP/RTCP packets from a source synchronization (SSRC), the sequence of conversion is shown in Fig. 6.



Finding the first RTCP Sender Report(SR) packet

Extracting NTP timestamp "wall-clock time" denote as $\tilde{N}_i$

Finding the corresponding RTP timestamp for packet pi in the same SR packet, denote as $\check{S}_i$

Extracting RTP timestamp for the first RTP packet (pi+1) arrived after pi. denote as $\check{S}_{i+n}$

Getting the sampling rate value for the given source SSRC. Denotes as τ

Recording arrival time of packet i+n., denotes as R_{i+n}

Fig. 6. Sequence of Sending-Time(pi) Conversion.

Then the sending time(pi) denotes as Si+n can be computed using (2):

$$S_{i+n} = \check{N}_i + \frac{\check{S}_{i+n}-\check{S}_i}{\tau} \quad [S] \quad n = (1,2,3 \ldots \qquad (2)$$

And the one-way delay of the packet i+n is computed using (3):

$$\text{one} - \text{way delay}(P_{i+n}) = R_{i+n} - S_{i+n} \ [S] \qquad (3)$$

As an example:

from first RTCP packet, we extract:

$\check{S}_i = 3989055281$, $\tilde{N}i = 46.43499$

and timestamps for the first packet after RTCP packet:

$\check{S}_{i+1} = 3989054077$, $R_{i+1} = 46.439904$.

So, sending-times(pi) (in time units) equal to:

$S_{i+1} = 46.43499 + (3989054077 - 3989055281)/90000 = 46.421622$

one-way delay(pi) = 46.439904-46.421622= 18.28 ms

As shown that we use τ=90kHz, this is because MPEG audio uses 90KHz media clock for compatibility with another MPEG content [20].

The mean one-way delay is defined in (4):

$$\text{mean} = \frac{\text{one}-\text{way delay1}+\text{one}-\text{way delay2}+\cdots+\text{one}-\text{way delayN}}{N} \ [S] \qquad (4)$$

For accurate measuring of one-way-delay parameter it is important to assure the synchronization between the RPi3 and the remote PC. This Synchronization can be accomplished using Network Time Protocol (NTP). We need to set up a RPi3 to act as an NTP server to which the remote PC will synchronize it's time to as shown in Fig. 7.

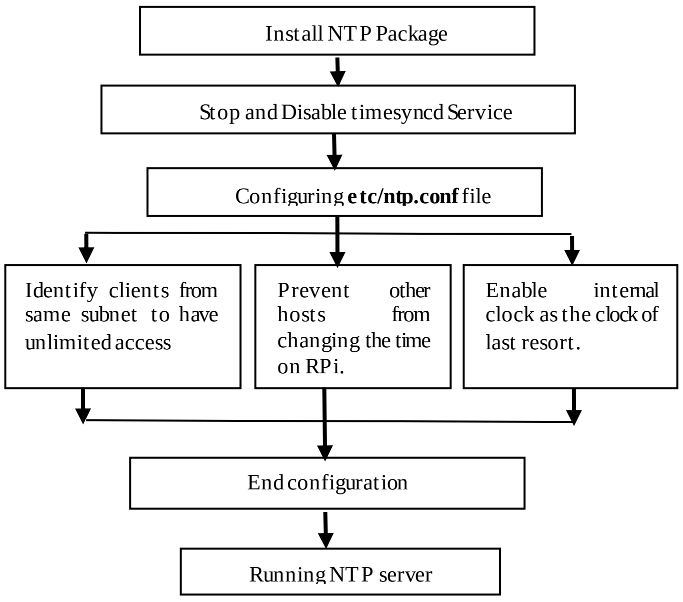

Install NTP Package
Stop and Disable timesyncd Service
Configuring etc/ntp.conf file
Identify clients from same subnet to have unlimited access
Prevent other hosts from changing the time on RPi.
Enable internal clock as the clock of last resort.
End configuration
Running NTP server

Fig. 7. Setting up RPi3 as NTP Server.

Consequently, on the other side the remote PC must be configured as Client NTP. By default, it only updates intermittently on the time.windows.com server. So, its clock must be point to the IP of PRi3, which represents NTP server.

Jitter: Jitter is the variation delay of packets and it is a very important QoS parameter for real time streaming. The jitter can be calculated as follows:

If T_i is the RTP timestamp from packet i, R_i is arrival time of packet i in time units, then for two consecutive packets i and j the difference D (i, j) may be expressed using (5):

$$D(i,j) = (R_j - R_i) - (T_j - T_i) \qquad (5)$$

Using this difference D for that packet i and the previous packet i-1 in order of arrival, the inter coming jitter can be calculated according to (6):

$$J(i) = J(i-1) + \frac{(|D(i-1,i)| - J(i-1))}{16} \; [S] \qquad (6)$$

Continuo with the above trace:

T_i= 3989009077, R_j = 45.938498

T_j = 3989017920, R_j =46.040784

J(i-1) =0

D (i, j) = (46.040784-45.938498) -(3989017920-3989009077)

Jitter J(i)= 0+ (|0.00403|-0)/16=0.2519 ms

The mean jitter is defined using (7):

$$\text{mean jitter} = \frac{|\text{jitter}_1| + |\text{jitter}_2| + \cdots + |\text{jitter}_N|}{N} \qquad (7)$$

Bandwidth: Bandwidth representing the amount of data that can be transmitted between two points in a set period of time. The calculation of bandwidth is expressed as follows [21]:

- total packet size (bits) = [(Ethernet header(bytes)) +(IP/UDP/RTP header (bytes)) + (Mpeg-its header(bytes)) + (voice payload size(bytes))] * 8
- Bandwidth = total packet size (bits) * [(codec bit rate) / Total payload size (bits)] = B.W bps (8)

Where Total payload size (bits) represents the actual audio data without stuffing.

Number of lost packets: the packets when transmitting from their source to destination can suffer from losing or eliminating by a router. The elimination of packets depends just on the present conditions of the network, and it cannot be anticipated.

VI. Framework Testing Results and Discussion

This section describes the effect of Sampling Rate (SR) on different compression techniques behaviour during 30-second of voice streaming over the suggested platform. The audio encoding algorithms are chosen based on two considerations: first they supported by FFmpeg technology and second, they fit into MPEG2-TS container.

Generally, AC3 and MP3 CODECs are tested with 8k, 16k, 32k and 48k Hz, while OPUS codec is tested with 8k, 16k, 24k and 48k Hz. First of all, the behaviour of the network is recorded for 24 hours. The behaviour of the network without connecting to the Internet is characterized by ping time which is found to be equal to 20 ms. While the behaviour of the network when connecting to the Internet is characterized by download, upload and ping times as shown in the Fig. 8 and Fig. 9 respectively.

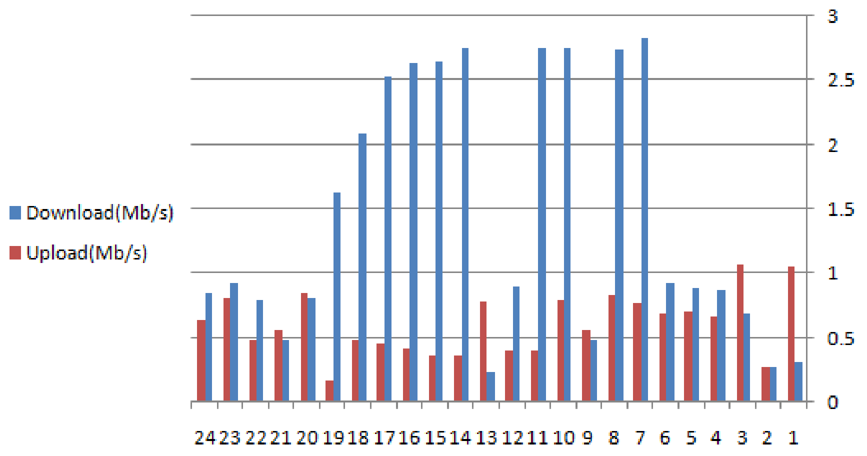


Fig. 8. Internet Download and Upload Behaviour.

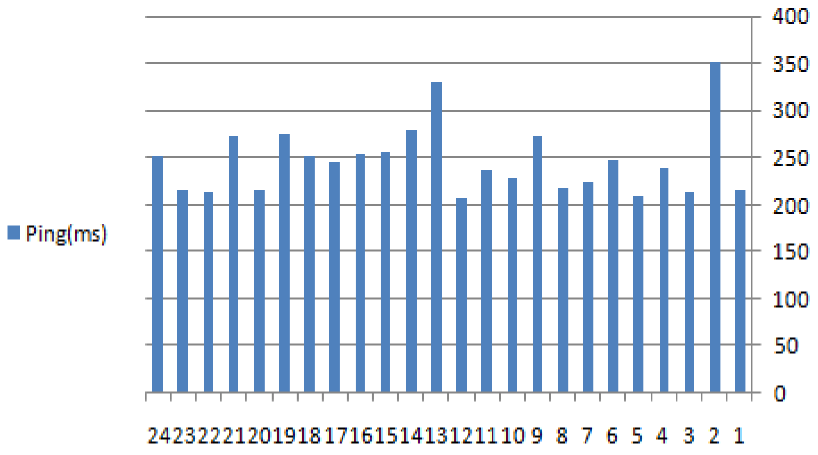


Fig. 9. Internet Ping Time Behaviour.

For all tests, it is found that the packet losses are zero and other metrics measurements results are listed in tables below:

Table 1. Mono-CPU/Mem.Usage for AC3, MP3 &OPUS.

	AC3		MP3		OPUS	
SR	CPU(%)	Mem.(%)	CPU(%)	Mem.(%)	CPU(%)	Mem.(%)
8k	3.6	3	6.6	3	8.6	3.1
16k	6.3	3	11.9	3	12.6	3
24k	NA	NA	NA	NA	19.8	3
32k	11.6	3	20.6	3	NA	NA
48k	15.6	2.9	31.7	2.9	22.6	2.9

Table 2. Stereo-CPU/Mem.Usage for AC3, MP3&OPUS (local & IoTs).

	AC3		MP3		OPUS	
SR	CPU(%)	Mem.(%)	CPU(%)	Mem.(%)	CPU(%)	Mem.(%)
8k	4.9	3	7.6	3	13.6	3
16k	8	3	13.2	3	19.9	3

The measurements in Tables (1) and (2) represent the CPU/memory usage in mono and stereo states respectively for VoWSN (local connecting without internet) and VoIoTs (connecting to the internet). It can be noticed that increasing SR increases CPU usage for the three techniques in mono and stereo states but the Memory usage for all tests almost constant and ranging between 2.9 to 3.1 (%).

Table 3. Mono-BW(kbps) for AC3, MP3 & OPUS.

SR	AC3	MP3	OPUS
8k	107	11	99
16k	107	26	99.6
24k	NA	NA	99.4
32k	105	54	NA
48k	107	71	79.3

Table 4. Stereo-BW(kbps) for AC3, MP3 & OPUS.

SR	AC3	MP3	OPUS
8k	175	26	141
16k	215	54	123

Moreover, Table (3) shows that in mono state the SR almost has no effect on AC3 bandwidth but increases MP3 bandwidth, while OPUS bandwidth is nearly constant with the increment of SR but reduced when SR= 48k. In stereo state, the SR increases the bandwidth of the three techniques as shown in Table (4). Also, these measurements represent bandwidth measurements for local and IoTs.

Table 5. Mono-Jitter times (ms)for AC3, MP3 & OPUS

	AC3		MP3		OPUS	
SR	Local	IoTs	Local	IoTs	Local	IoTs
8k	4.2	3.4	9.4	9.9	4.6	4.8
16k	2.7	2.3	6.6	4.6	4.6	4.7
24k	NA		NA		3.7	3.8
32k	2.6	2.6	3.3	3.1	NA	

48k	1.7	2.4	2.1	2.1	1.5	2.9

Table 6. Stereo-Jitter times(ms)for AC3, MP3 & OPUS.

SR	AC3		MP3		OPUS	
	Local	IoTs	Local	IoTs	Local	IoTs
8k	1.9	1.9	2.9	4.7	4.9	4
16k	1.7	1.9	2.1	2.2	1.7	1.6

Tables (5) & (6) show that increasing SR reduces the jitter time in mono and stereo states for the three techniques. It can be noticed that the jitter times for AC3 and OPUs are remaining in the range of real-time constrains (<5 ms).

For calculation of one-way delay, we calculate the delay for the RTP data packets between each two consecutive RTCP packets which are varied from test to test. Then the mean delay is computed according to (4). Tables (7) & (8), show that increasing the SR reduces the one-way delay for AC3 & MP3 for local and IoTs in mono and stereo states.

Table 7. Mono-Delay(ms) for AC3, MP3 & OPUS.

SR	AC3		MP3		OPUS	
	Local	IoTs	Local	IoTs	Local	IoTs
8k	182	669	392.8	616	15	441
16k	83	569	144.25	465	17	436
24k	NA		NA		14	408
32k	39	501	121.07	462	NA	NA
48k	21	484	66.7	359	16	400

Table 8. Stereo-Delay(ms)for AC3, MP3 & OPUS

SR	AC3		MP3		OPUS	
	Local	IoTs	Local	IoTs	Local	IoTs
8k	177	436	360	445	17	220
16k	78	333	199.6	297	16	213

While in OPUS situation the delay almost constant with SR increment in mono and stereo states. This is because the RTP timestamp clock frequency for OPUS represents the highest supported sampling rate, i.e. 48 kHz, for all modes and sampling rates [22]. So, to determine the RTP timestamp for SR < 48 kHz the number of samples has to be multiplied by a multiplier according to Table (9).

Table 9. OPUS sampling Rate Multiplier.

SR	Multiplier
8k	6
12k	4
16k	3
24k	2
48k	1

The measurements show that the best one-way delay time is obtained using OPUS technique in mono and stereo states.

VII. Outdoor Case Study

One application in which VoWSN system can play an important role is disasters. Disasters pose challenges in ensuring reliable communications with a reasonable cost. As a case study, we applied the proposed VoWSNs system in a disaster area, which is a two-story building, where the rescuer moves through the building to the roof. The total distance between the rescuer and the rescue centre is about 20m as shown in Fig. 10. OPUS encoding is used in this scenario (for its best behaviour) and the results of jitter and one-way delay times are shown in Tables (10) & (11).

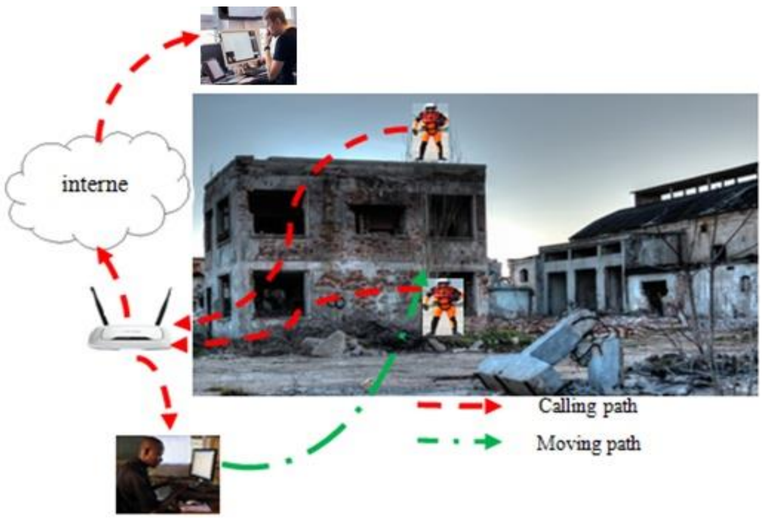


Fig. 10. Outdoor Case Study Scenario.

As shown from Tables (10) & (11) the jitter times remain in real-time constrains in mono and stereo states for local and IoTs, while one-way delay times are in real-time range for local only.

Table 10. Mono-OPUS measurements

SR	Jitter(ms)		Delay(ms)	
	Local	IoTs	Local	IoTs
8k	5.6	4.8	38	441
16k	4.8	4.7	27	436
24k	3.6	3.8	28	408
48k	2.7	2.9	48	400

Table11. Stereo-OPUS measurements

SR	Jitter(ms)		Delay(ms)	
	Local	IoTs	Local	IoTs
8k	5.1	220	38	220
16k	3.3	213	39	213

VIII Conclusion

The area of voice streaming in the IoTs using resource constrained hardware is began to be explored by many researchers for its advantages in opening new scenarios in WSN applications. In this paper, IoTs based low-cost low-power WSN for real-time voice streaming system has been designed and implemented effectively. The performance of the proposed platform firstly was examined in local network without connecting to the internet (i.e. VoWSN) and secondly by connecting to the internet (i.e. VoIoTs). Moreover, for performance evaluation, a set of procedures for CPU/Memory usage, one-way delay, jitter bandwidth calculation is proposed and implemented. The main features of the proposed VoIoTs system can be characterized from two different perspectives: architecture overview and performance analysis overview.

From architecture overview, open source Linux architecture of the proposed system make it easily accessed by users and supposed to be safe from many attacks. Furthermore, the system is proved to be cost effective since it is exploiting a low-cost RPi3 with a built in Wi-Fi connection as voice transmission system so there is no need for internet and SIM card. Also, there is no need for external CODECs because RPi3 can implements different compression algorithms in software.

From performance analysis overview, the proposed system could ensure real-time voice transmission for VoWSN system without internet in several tests since jitter < 5 ms and delay < 150 ms even in outdoor case (with the presence of obstacles). Also, the system could ensure jitter for voice transmission in real-time constrains (< 5 ms) for VoIoTs system and a reasonable one-way delay times for VoIoTs system compared to the max ping time of our internet network which is 350 ms as shown in Fig. (8).

References

1. I. Fathi, Q. I. Ali, J. M. Abdul-Jabbar, "Voice over Wireless Sensor Network (VoWSN) System: A Literature Survey ", International Journal of Information Engineering and Applications Vol.1, No.1, Publication Date: Mar. 10, 2018, Page: 23-36.

2. Stefan F., Federico V. F., "Low Delay Video Streaming on the Internet of Things Using Raspberry Pi", Electronics 5.3 (2016): 60.

3. Shu Y.,"Internet of Things: Wireless Sensor Networks", White paper, IEC, Geneva, Switzerland 2014.

4. Khalil N., Abid M. R., Benhaddou D., & Gerndt M., "Wireless sensors networks for Internet of Things." arXiv preprint arXiv:1606.08407 (2016).

5. The Role of Voice in the Internet of Things [https://www.strategyanalytics.com/strategy-analytics/blogs/ iot/ 2016/02/19/the-role-of-voice-in-the-internet-of-thing.

6. R. Mangharam, A. Rowe, R. Rajkumar &R. Suzuki "Voice over sensor networks", Real-Time Systems Symposium, 2006. RTSS'06. 27th IEEE International, Rio de Janeiro, Brazil, IEEE,2006.

7. D. Brunelli, M. Maggiorotti, L. Benini & F. L. Bellifemine, "Analysis of audio streaming capability of zigbee networks", EWSN'08 Proceedings of the 5th European conference on Wireless sensor networks, Springer, Berlin, Heidelberg, 2008. 189-204.

8. H. Rong-lin, Y. Jian-rong, G. Xia-jun, G. Xiang-ping& C. Li-Qing, "the research and design on TDD voice WSN", Multimedia Technology (ICMT), 2010 International Conference on, Ningbo, China, IEEE, 2010.

9. A. Geetha, "Intelligent Helmet for Coal Miners with Voice over ZigBee and environmental Monitoring", World Applied Sciences Journal 29.8 (2014): 1031-1034.

10. Kh. Sahal & R. Sharma, "Voice Communication Over Zigbee Protocol: A Literature Review", International Journal of Engineering Research and General Science Volume 3, Issue 3 May-June, 2015 ISSN 2091-2730.

11. S. HARITHA, R. ANIRUDH REDDY, "Design and Implementation of Efficient Audio Streaming System Using Raspberry Pi", International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE), ISSN: 0976-1353 Volume 12 Issue 1– DECEMBER 2014.

12. Johan Uttermalm,"Audio streaming on top of 802.11n in an IoT context", Faculty of Health, Science and Technology 2016-06-08.

13. H. Schulzrinne, RTP: A Transport Protocol for Real-Time Applications, Network Working Group, RFC 3550.

14. A. D. Shah, S. Agrawal & K. Sharma, "Transport Stream Playout System for MPEG-TS using Program Clock Reference." International Journal of Computer Applications, 2015: 117.16.

15. Z. Yuan, “Quality of Service Differentiation for Multimedia Delivery in Wireless LANs”, PhD thesis, Dublin City University, 2012.

16. R. L. Dash, A. R. Bevi, "Real-time Transmission of Voice over 802.11 Wireless Networks Using Raspberry Pi", International Journal of Engineering Development and Research (IJEDR), ISSN:2321-9939, Vol.2, Issue 1, pp.793-800, March 2014.

17. J. K. Saha, Md. M. A. Ghuri, Md. M. R. Mamur, M. J. Hossain, T. A. Chowdhury, B. Paul, "A Novel Design and Implementation of a Real-time Wireless Video and Audio Transmission Device", WSEAS TRANSACTIONS on COMPUTER RESEARC, E-ISSN: 2415-1521, Volume 4, 2016.

18. J. M. Montanino, E. M. Respaut, "AWESUM: AUDIO WI-FI EMBEDDED STREAMING USING MICROCONTROLLERS", design project report, Cornell University, 2012.

19. H. Schulzrinne, RTP: A Transport Protocol for Real-Time Applications, Network Working Group, RFC 3550.

20. Colin Perkins, "RTP: Audio and Video for the internet", Addison-Wesley Professional, 2003.

21. https://www.cisco.com/c/en/us/support/docs/voice/voice-quality / 7934 -bwidth-consume.html#anc3.

22. J. Spittka, K. Vos, JM. Valin " RTP Payload Format and File Storage Format for Opus Speech and Audio Codec", Network Working Group, Internet-Draft, July 2011.