Multimedia Tools and Applications (2023) 82:17999–18017
https://doi.org/10.1007/s11042-022-14179-y

CoMAC: A cooperation-based multiparty audio conferencing system for mobile users

Sheng-Shih Wang[1] · **Hsueh-Yi Chen**[2] · **Shiann-Tsong Sheu**[2]

Received: 10 August 2021 / Revised: 21 April 2022 / Accepted: 27 October 2022 /
Published online: 21 November 2022

Abstract

The existing architectures used in the multiparty audio conferencing systems are typically categorized as either centralized or decentralized. These architectures expose a trade-off between processing latency and system capacity, namely the number of participants. This paper proposes a multiparty audio conferencing system for mobile users to improve the processing latency and system capacity. Instead of using the pure centralized or decentralized architecture, the proposed system adopts a novel cooperation-based architecture, in which only some participants are selected as the central controllers to deal with the tasks such as acoustic echo cancellation, encoding, decoding, mixing and de-mixing. The proposed system also uses a buffer reordering scheme to solve the problems of network jitters and out-of-order packets. This study analyzed the processing latency of the multiparty audio conferencing systems using the existing and proposed architectures. We also implemented these systems on diverse mobile platforms to compare the processing latency and number of participants. Performance evaluation results confirmed that the proposed cooperation-based architecture can not only reduce the processing latency but also support more participants, compared to the existing architectures.

Keywords Audioconferencing · Buffer reordering · Codec · Cooperation · Multiparty conferencing

Hsueh-Yi Chen and Shiann-Tsong Sheu have contributed equally to this work.

✉ Sheng-Shih Wang
sswang@mail.lhu.edu.tw

Hsueh-Yi Chen
xy.chen@g.ncu.edu.tw

Shiann-Tsong Sheu
stsheu@ce.ncu.edu.tw

[1] Department of Electronic Engineering, Lunghwa University of Science and Technology, Taoyuan, Taiwan

[2] Department of Communication Engineering, National Central University, Taoyuan, Taiwan

1 Introduction

In recent years, the application of voice over Internet Protocol (VoIP) [7] has gradually switched from single to multiparty chatting due to the demand for multi-user communication. Remarkable advances in Internet and multimedia technologies have also caused multiparty audio applications to become popular. In particular, the demand for remote offices and conferences has been increasing, especially during the ongoing global pandemic of coronavirus disease 2019. Multiparty audioconferencing provides a convenient way for a group of people to communicate via the Internet through a variety of methods, such as conference calls, online collaborative learning, and live streaming. Nowadays, many solutions, such as LINE, Skype, Zoom, and Web Real-Time Communication (WebRTC) [8, 15], are widely used for multiparty audioconferencing. Previous studies have also proposed various schemes to improve the quality of experience for system participants as well as the system performance [1, 5].

In general, network architecture and audio processing are two key issues that should be considered when developing a multiparty audio conferencing (MAC) system. The network architecture of MAC systems can be classified into two categories: centralized and decentralized [1, 9]. The MAC systems using centralized architecture consider a server or one of the participants' devices to act as the central controller for relaying the audio or video traffic for participants [9, 10]. The centralized architecture significantly reduces the processing load for participants and supports large numbers of participants. However, it requires a powerful central controller for traffic relaying as the number of participants increases. Thus, the maintenance cost of the central controller significantly increases. Meanwhile, the quality of service probably downgrades due to heavy traffic loads. That is, the central controller acts as a bottleneck in the centralized architecture. In a decentralized architecture-based MAC system, each participant can directly exchange audio traffic with the others due to the lack of a central controller [1, 13]. Therefore, the decentralized architecture has a lower deployment and maintenance cost than the centralized architecture. However, the maximum number of participants in a MAC system based on the decentralized architecture is limited because the bandwidth consumption increases as the number of participants increases.

As for the audio processing, various issues must be solved. They include acoustic echo cancellation, noise suppression, voice activity detection, audio signal compression, and audio mixing and de-mixing. Acoustic echo cancellation and noise suppression aim to provide high-quality audio signals. Voice activity detection, which mainly detects the presence or absence of audio signals, is used to avoid unnecessary signal processing at the participants' devices. Audio signal compression aims to reduce the transmission bandwidth, and audio mixing and de-mixing aim to support multiparty communications with different sound sources. Although numerous existing MAC systems, such as Skype, LINE, Zoom, and WebRTC, consider the centralized architecture because the decentralized architecture is more complicated to implement, the decentralized architecture is a promising and preferable solution due to its lower cost (e.g., for processing, storage, and maintenance), especially for VoIP service providers. Note that the aforementioned audio processing issues need to be dealt with when designing a decentralized architecture-based MAC system. As a result, designing the proper architecture to distribute the audio processing tasks to participants is a challenge in the development of MAC systems. This paper proposes a cooperation-based MAC (CoMAC) system, which adopts a cooperation-based architecture to design a mobile-based decentralized MAC system. The contributions of this paper are summarized as follows:

(1) The design of a cooperation-based multiparty audio conferencing system to support more participants than the MAC systems based on existing architectures. In the proposed system, there is no any central controller, and only some of the participants act as central controllers to accomplish the compression-decompression (codec), acoustic echo cancellation, audio signal compression, and audio mixing and de-mixing.
(2) The design of a buffer reordering scheme to deal with the network delay generated from the inter-arrival jitters and out-of-order frame receptions at receivers.
(3) The performance analysis of the processing latency of MAC systems using different architectures.
(4) The implement of the proposed CoMAC system on mobile devices using open software and open-source libraries, e.g., Opus Codec, WavPack, and WebRTC.

The rest of this paper is organized as follows. Section 2 reviews the key issues and techniques of audio processing. Section 3 discusses the representative network architectures of existing MAC systems. Section 4 presents the proposed CoMAC system and the analysis results. Section 5 presents the system implementation and performance evaluation results. Section 6 provides concluding remarks.

2 Audio processing issues

This section discusses the major audio processing issues mentioned earlier, including acoustic echo cancellation, voice activity detection, codec, and audio mixing/de-mixing. The solutions to these are also mentioned in this section.

2.1 Acoustic echo cancellation

Acoustic echo is a crucial problem for single chatting or MAC applications. When the near-end speaker's microphone starts recording audio, the far-end user may hear echoing audio because the audio recorded by the former speaker's microphone mixes with the playing audio signal of the latter simultaneously. The undesirable acoustic echo distorts the original acoustic signal, so the far-end user is likely to suffer a deteriorating quality of experience. Acoustic echo cancellation (AEC) is a straightforward concept to solve this problem, wherein its reference signal is used to assist in removing the echo.

Typically, either speaker can carry out the cancellation. For cancellation at the near-end speaker, the device places the received far-end audio data into a buffer with timestamps and then plays the audio via the speaker. If the microphone is active for recording the local audio, it records the signal, which includes the local and far-end audio signals, from diverse acoustic echo paths. The filter then eliminates the echoing audio from the mixed audio signal and generates the original local audio. Regarding cancellation at the far-end speaker, when the device receives the far-end audio signal, which mixes the remote audio signal and echoed signal from the remote device, the filter eliminates the echoing audio from the mixed audio signal and generates the original far-end audio data.

Previous studies have proposed many filter based solutions for acoustic echo cancellation. The main ideas of these approaches are twofold: usage of an adaptive filter and control of the adaptation of the filter coefficient. Hamidia and Amrouche [14] evaluate the performances of AEC based on adaptive filtering, where the speech is encoded and decoded by the adaptive multi rate wide band speech codec that is used in the second generation (2G) and third generation (3G) of cellular systems. The scheme proposed by Fukui et al. not only improves the estimation accuracy of the acoustic coupling level, but also reduces the residual echo using an efficient post-filter based on this estimated level [12]. Many algorithms to control the adaptive filter coefficient have been proposed in [6, 16, 35]. The central concepts of these algorithms include the minimum mean square error criterion [6], affine projection [16], and various popular objective functions [35], e.g., least squares, weighted least squares, and mean absolute error. A real-time AEC algorithm based on variable step-size partitioned block frequency-domain adaptive filtering and frequency-domain nonlinear echo processing algorithm is proposed for hands-free voice communication devices [33].

2.2 Voice activity detection

Voice activity detection (VAD) is a widely used technology to detect whether the audio includes voice activity. As VoIP applications are designed to deliver human voices, applying VAD technology can reduce the bandwidth consumption. Kim et al. [21] adopt an attention strategy with the deep neural network for VAD, and propose a training strategy to adaptively deploy the context information according to the noise type and level. A pattern recognition approach has been proposed for voiced–unvoiced–silence classification [2]. The approach is essentially a classical hypothesis-testing procedure based on the statistical decision theory, and classifies a speech signal according to its measurements (e.g., zero-crossing rate and speech energy). Romoli et al. [28] introduced a multi-channel acoustic echo cancellation scheme that mainly considers the estimation of the audio signal's fundamental frequency. The basic idea of the approach is to weaken the linear relation among channels and track the presence of an active human speaker. Both operations are performed in the system's remote and local rooms. Existing studies have also considered various machine learning models, such as support vector machines [17] and deep learning [22], to design VAD techniques. With the prevalence of indoor smart Internet of Things devices, such as security cameras and intelligent speakers, Jung et al. [18] propose an event detection algorithm for unusual user behavior patterns related to vision, audio, and activity. Park et al. [27] propose an event detection technique based on a sound-learning algorithm to construct acoustic models via learning algorithms from collected sound data according to the types of acoustic events.

2.3 Codec

The codec encodes raw audio data to reduce the bandwidth or storage space consumption. Based on the compressed results, there are two methods, i.e., lossless and lossy. Less compressed data are generated by lossy compression than lossless compression, but the original data cannot be completely restored. That is, the audio data compressed and restored by the lossy codec may be distorted. With the characteristics of streaming, even if the decompressed data are incomplete, the audio can play normally. Opus and WavPack are two representative audio codes. Opus is a lossy audio codec designed for interactive Internet applications [32]. It can support a sampling rate varying from 8 (at narrowband) to 48 kHz (at full band), and its bit rates vary from 6 (voice) to 510 kbits/s (stereo music). Compared with other codecs (e.g., Vorbis, G.711, and G.722), Opus achieves a higher quality at most bit rates. Moreover, it outperforms other codecs in terms of latency. The latency of Opus is less than 20 ms, whereas the latency resulting from most other codecs exceeds 20 ms [26]. In addition, the advantages of Opus include good loss robustness, low distortion, and packet loss concealment, making it suitable for VoIP applications.

The open audio codec WavPack provides lossless and high-quality lossy compression and a unique hybrid compression mode [23]. Unlike lossy codecs (e.g., MP3 and Opus), WavPack can recover the original information after audio compression. It determines the different compression ratios according to the source materials. In general, the compression ratio of WavPack ranges from 30% to 70%. The comparison results of the compression ratios and encoding times of various lossless codecs, including the True Audio, Monkey's Audio, and Free Lossless Audio Codec, for six albums are shown in [30]. According to the performance comparison, WavPack has the moderate performance in data compression and process latency.

2.4 Mixing/De-mixing

In addition to an efficient audio codec (typically with a high compression ratio), audio mixing is key to reduce the bandwidth consumed during multiparty conferencing. Using the audio mixing technology, multiple audio streams with the same sampling rate and bit resolution generated from different users are collected by a device to mix these multiple audio streams into a single stream. When a device plays the mixed audio stream, the audio from the different users can be heard simultaneously. On the other hand, the audio de-mixing technology is used to split the mixed stream for transmitting a specific stream to a specific participant in the central controller. The mixed audio played for a specific participant differs from that played for the others. That is, a participant only hears the audio from other participants. Therefore, mixing/de-mixing is regarded as a promising approach to bandwidth reduction.

There are two approaches to accomplish mixing and de-mixing. One is that a server or participant is responsible for performing the mixing and de-mixing tasks to create all mixed audio streams for all participants [9, 10]. The other is that one server or participant performs the mixing task to create a single mixed audio stream that mixes all audio frames from all participants, and the remaining participants perform the de-mixing task to remove their audio frames from the mixed audio stream. The former requires a powerful computing capacity. The latter needs to maintain the audio consistency; otherwise, it generates the echo if the de-mixing task fails to remove the corresponding audio frame.

3 Network architectures of MAC systems

Recall that the network architecture considered in the majority of existing MAC systems is either centralized or decentralized. This section presents the basic concepts of two types of centralized architectures (e.g., server-based [9, 19, 25, 29, 31, 34] and host-based [3, 4, 10, 13]) and a decentralized architecture (mesh-based [1, 20, 24]), as shown in Fig. 1. In this paper, we use the notations listed in Table 1 to represent the operation of these architectures.

3.1 Server-based architecture

In the server-based architecture, shown in Fig. 1(a), a server needs to perform all processes of decoding and encoding the audio frames and then mix all audio frames sent from all participants. When the server receives the audio frames from each client, it mixes them. Then, it de-mixes the mixed frames with the audio frames of the corresponding participants to cancel the echo before sending the mixed audio frames to a participant.

Figure 2 shows an example of the operation of a server-based architecture with three peers. The server must allocate the necessary resources to handle the audio frames from all peers (i.e., P_1, P_2, and P_3). For each peer (i.e., participant), the recorded audio is input into

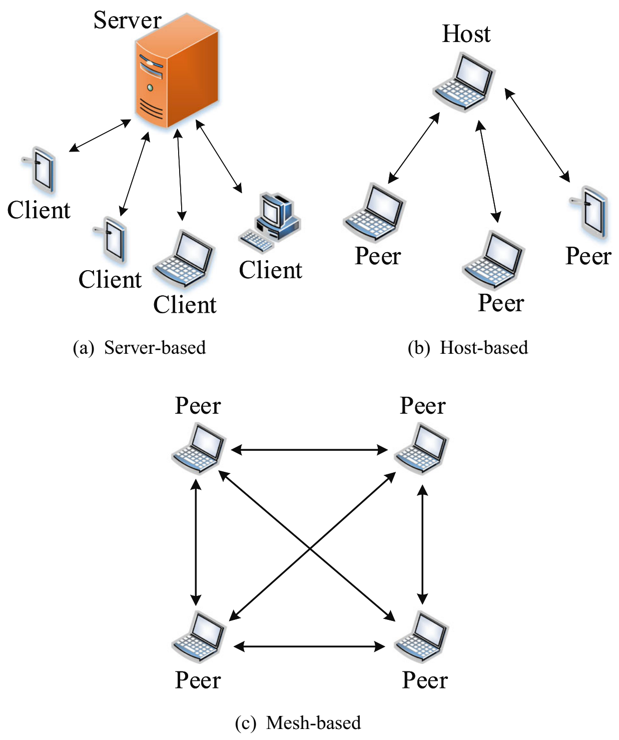

Fig. 1 Typical Network Architectures used in Existing MAC System

AEC for echo cancellation, compressed by LyE for compression, and then sent to the server. The server decodes the audio frames once receiving them. The decoded frame is then input into M to be mixed with the frames of other peers. The mixed frame is also input into the DM of each peer so that the server can de-mix the mixed frame to generate the specific audio frame for the specific peer. That is, the generated audio frame is actually the mixed frame of the other two peers. Then, this frame is put into LyE for compression and sent to

Table 1 Notations used for architectures

Notation	Definition
AEC	Acoustic echo cancellation module
LyE	Lossy encoding module
LyD	Lossy decoding module
LE	Lossless encoding module
LD	Lossless decoding module
DM	De-mixing module
M	Mixing module
Tx	Transmission module
Rx	Reception module
P_i	The i-th peer

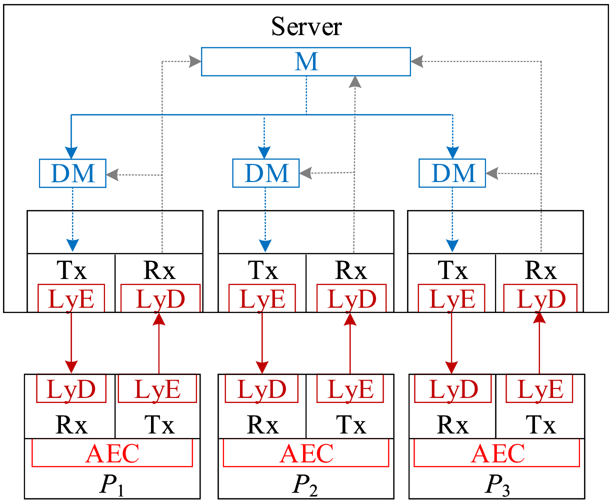

Fig. 2 Operation of the Server-based Architecture with Three Participants

the corresponding peer. Upon receiving the audio signal from the server, the peer decodes the compressed data by LyD, inputs it to AEC, and plays it through the speaker.

3.2 Host-based architecture

As shown in Fig. 1(b), in the host-based architecture, one participant is selected as a central controller (called host) to handle all processes of decoding and encoding the audio frames from the other participants. The host also mixes the received audio frames from all peers and then forwards the mixed frames into AEC to cancel the echo because the mixed frames are also played by the host. Then, the recorded audio frames of the host are mixed with the previously mixed audio frames from the other participants. Before being sent to a participant, the mixed frames must be de-mixed with the audio frames of the corresponding participants for echo cancellation. Compared with a device that plays the server role, a participant has a lower computing capacity and, thus, cannot support too many other participants.

In Fig. 3, we assume that participant P_1 plays the role of the central control node. It allocates additional resources to handle the audio signals from each peer. Each peer periodically records its own audio data, puts it into AEC for echo cancellation, compresses it by LyE, and sends the compressed audio frame to the host. When a peer receives the compressed audio frames sent from the host, it decodes the received data, forwards them to AEC, and then plays the uncompressed audio frame through the speaker.

3.3 Mesh-based architecture

As shown in Fig. 1(c), in the mesh-based architecture, each participant sends its audio frames to the others and receives the others' audio frames as well. Each participant only mixes and plays the audio frames of the other participants. Figure 4 shows the operation of a fully connected mesh-based architecture with four participants (i.e., P_1, P_2, P_3, and P_4). Each participant periodically records the audio data, feeds them into AEC to cancel the echo, compresses the received frames by LyE, and then sends the frames to all peers.

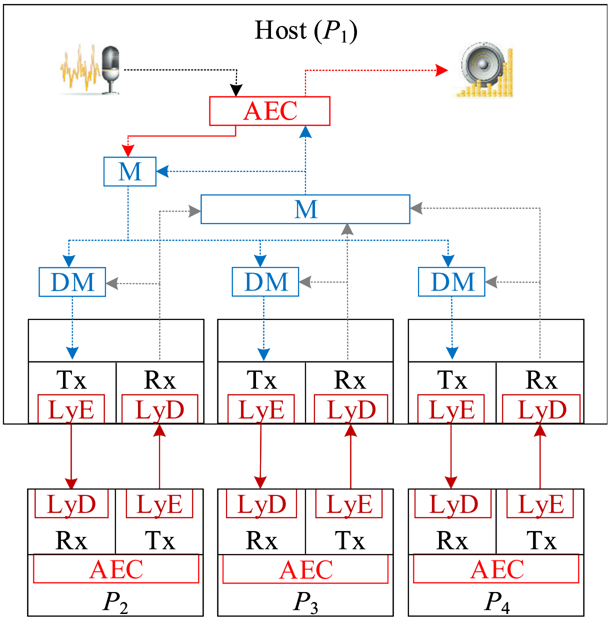

Fig. 3 Operation of the Host-based Architecture with Four Participants

When a participant receives the compressed audio frames of any other participant, it decodes the received frames and mixes them with the other audio frames of the other participants into a single audio frame. The mixed frames are put into AEC and played through the speaker. Compared with the central architectures (including server-based and host-based), the bandwidth usage of the peer is high in this architecture.

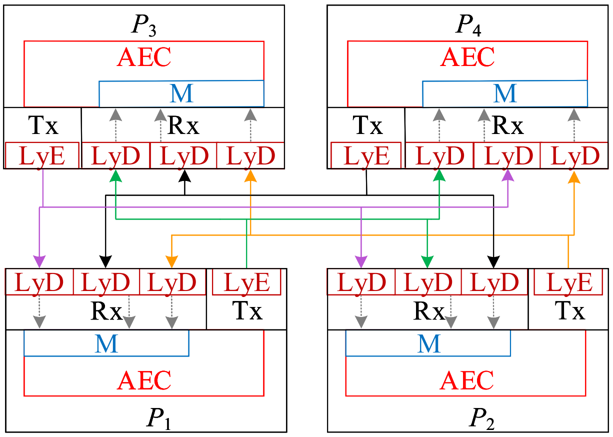

Fig. 4 Operation of the Mesh-based Architecture with Four Participants

4 Proposed CoMAC system

This section describes the proposed CoMAC system, including an efficient buffer reordering scheme and a novel codec scheme. Then, we present the analysis of processing latency of the existing and proposed architectures for multiparty audio conference.

4.1 Buffer reordering scheme

This study introduces a buffer reordering scheme in the proposed CoMAC system to tackle network jitters and out-of-order packet reception at the receiver. The scheme consists of two processes: writing and reading. The writing process is executed when a device receives an audio packet from Internet, and the reading process is executed when a device starts to play audios. The operations of the writing and reading processes are shown in Algorithms 1 and 2 respectively. Table 2 lists the notations used in Algorithms 1 and 2. Assume that the buffer of a device is composed of N slots. The proposed system uses different threads to implement the two processes. For thread safety, the slot must be locked before storing a packet in or retrieving a packet from the slot.

When receiving an incoming packet, $P(i)$, a device first checks the status (i.e., locked or unlocked) of $S(j)$, where $j = i \bmod N$, and then performs Algorithm 1 if $S(j)$ is unlocked. The concept of Algorithm 1 is to store the receiving packet in the specific slot of the buffer. Note that if $S(j)$ is not empty, the $P(i)$ can be stored under specific conditions to prevent the old packet from overwriting the new one.

Denote the packet we intend to read as $P(i)$. At the reading process, a device first checks the status (i.e., locked or unlocked) of $S(j)$, where $j = i \bmod N$, and then performs Algorithm 2 if $S(j)$ is unlocked. In Algorithm 2, if $S(j)$ is not empty and $n_{seq}^{cur}(j) = i$, the packet is read correctly, as shown from line 4 to 6. The operation from line 8 to 10 in Algorithm 2 can avoid reading the old packet. The operation of line 12 to 22 mainly resolves the overrun problem, which occurs when the incoming packets burst. This problem causes that some packets in the buffer are overwritten.

The writing and reading processes of the proposed buffer reordering scheme aim to respectively store and read packets in the correct order event if the network jitters occur. Figure 5 shows an example of operation of the proposed buffer reordering scheme to solve the three problems, i.e., packet loss, overrun, and out-of-order packets. Let k be the sequence number of the receiving packet. Let e and e_n respectively represent the sequence numbers

Table 2 Notations used in Algorithms 1 and 2

Notation	Definition
N	The size of buffer
$S(i)$	The i-th slot in the buffer
$P(i)$	The packet with a sequence number i
$n_{seq}^{cur}(i)$	The sequence number of the packet currently stored in $S(i)$
n_{total}^{cur}	The total number of packets currently stored in the buffer
T_{max}^{tol}	The maximum tolerable time of the system
	(The time is represented by the number of packets.)
n_c	The number of slots cleared in the buffer

```
    Input: $P(i)$ is the incoming packet with sequence number $i$.
 1: Lock $S(j)$
 2: if $S(j)$ is empty then
 3:     Store $P(i)$ in $S(j)$
 4:     $n_{total}^{cur} \leftarrow n_{total}^{cur} + 1$
 5: else
 6:     if $i > n_{seq}^{cur}(j)$ or $i < n_{seq}^{cur}(j) - N$ then
 7:         Store $P(i)$ in $S(j)$
 8:         $n_{total}^{cur} \leftarrow n_{total}^{cur} + 1$
 9:     end if
10: end if
11: Unlock $S(j)$
```

Algorithm 1 Writing process.

of the current reading and next expected packets. At first, the packet with sequence number 0 is received, saved into slot 1, and then n_{total}^{cur} is set as 1. At the reading process, the packet in slot 1 will be read because e_n is 0. After reading, e and e_n are set as 0 and 1 respectively. Assume that the incoming packet with a sequence number 1 is lost. When the three packets (due to $T_{max}^{tol} = 3$) each of whose sequence numbers is not 1 are received, the lost event is detected. Then, the packet with sequence number 1 is skipped, and the packet with sequence number 2 is read. Meanwhile, e and e_n are set as 2 and 3 respectively. For the following two packets with sequence numbers 3 and 4, the reading process works fine. However, when the network jitter occurs (i.e., packets with sequence numbers 5, 6, 7, and 8 burst), the packet with sequence number 3 is overwritten by the packet with sequence number 8. This problem is called the overrun problem. According to the reading process, the packets with sequence number less than 8 are cleared in the buffer, and the packet with sequence number 8 is read. The values of e and e_n are set as 8 and 9 respectively. Then, the problem of out-of-order packets occurs. That is, the packet with sequence number 10 arrives earlier than the packet with sequence number 9. Because $n_{total}^{cur} > T_{max}^{tol}$, the reading processing will read the packets with sequence numbers 9 and 10.

4.2 Lossless codec and lossy codec

Unlike the majority of applications using either lossless codec or lossy codec, the proposed system uses both types of codecs because the lossless codec is less complex than the lossy codec and the lossy codec reduces the bandwidth consumption. In addition, for mixing and de-mixing, the lossless codec is necessary to retrieve the original audio signals from the mixed audio frames. The basic idea of the usage of codecs in the proposed system is shown in Fig. 6, including four participants (i.e., P_1, P_2, P_3, and P_4) and P_1 is assumed to be the mixer. P_2, P_3, or P_4 uses the lossy codec technique to send the compressed stream to the P_1. On the other hand, the mixer (i.e., P_1) uses the lossless codec technique to send the mixed

```
Output: $P(i)$ is the packet we intend to read.
1: Lock $S(j)$
2: if $S(j)$ is not empty then
3:     if $n_{seq}^{cur}(j) = i$ then
4:         Retrieve the packet in $S(j)$
5:         $n_{total}^{cur} \leftarrow n_{total}^{cur} - 1$
6:         $i \leftarrow i + 1$
7:     else
8:         if $n_{seq}^{cur}(j) < i$ then
9:             $n_{total}^{cur} \leftarrow n_{total}^{cur} - 1$
10:            Clear $S(j)$
11:        else
12:            $n_c \leftarrow 0$
13:            for $k \leftarrow 0$ to $N - 1$ do
14:                if $n_{seq}^{cur}(k) < n_{seq}^{cur}(j)$ and $S(k)$ is not empty then
15:                    Clear $S(k)$
16:                    $n_c \leftarrow n_c + 1$
17:                end if
18:            end for
19:            $i \leftarrow n_{seq}^{cur}(j)$
20:            Retrieve the packet in $S(j)$
21:            $n_{total}^{cur} \leftarrow n_{total}^{cur} - 1 - n_c$
22:            $i \leftarrow i + 1$
23:        end if
24:    end if
25: else
26:    if $n_{total}^{cur} > T_{max}^{tol}$ then
27:        $i \leftarrow i + 1$
28:    end if
29: end if
30: Unlock $S(j)$
```

Algorithm 2 Reading process.

data to all participants (i.e., P_2, P_3, and P_4), which perform the decompression task upon receiving the mixed data.

4.3 Cooperation-based architecture

In the proposed cooperation-based architecture, the mixer takes charge of mixing the audio frames from the participants and de-mixing the audio frames from other mixers, while the participant, does not play the role of mixer, executes to de-mix the audio frames. Figure 7 illustrates the operation of the proposed architecture with three peers. Assume that the participant P_1 is selected to be the mixer because it has more computing power and bandwidth than the other participants. The non-mixer peer (i.e., P_2 or P_3) periodically records the audio data, feeds them into AEC for echo cancellation, compresses them by LyE, and then decompresses them by LyD immediately. The peer then sends the compressed audio frames

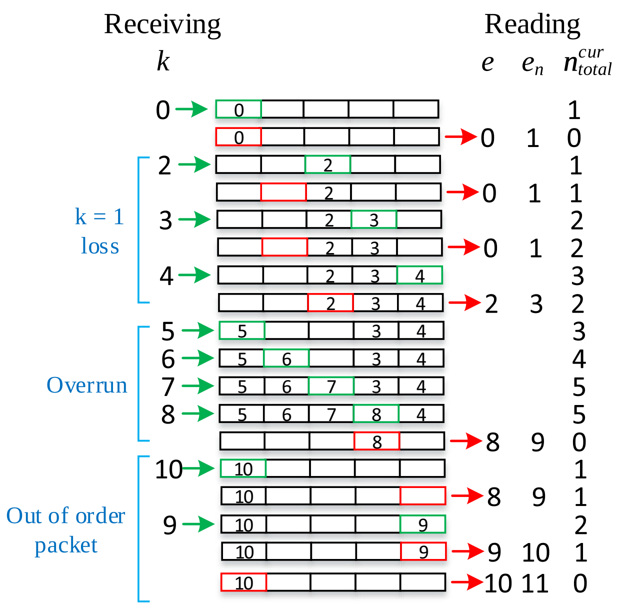

Fig. 5 Example of the Operation of the Proposed Buffer Reordering Scheme with $N = 5$ and $T_{max}^{tol} = 3$

to the mixer. Note that the peer decompresses the compressed data to be sent locally and stores them for further usage to maintain the data consistency.

The mixer in the proposed CoMAC system needs to decode the audio frames by LyD and mix all audio frames received from each participant. The mixed audio frames are then input to AEC, played by the mixer's speaker, and mixed with the recorded audio frames of the mixer. Finally, the mixer uses the LE to compress the mixed audio frames and sends them to all participants. When a participant receives the compressed audio frames sent from the mixer, it decodes the received data by the LD, and then de-mixes it with the previously decompressed audio data by LyD to avoid the echo. Finally, the de-mixed audio data are input to AEC and played by the speaker.

Figure 7 shows the example of one mixer. The proposed CoMAC system supports multiple mixers to extend the number of participants. For a brief description, Fig. 8 shows the cooperation-based topology with six peers, which have two mixers. When there are two or more mixers, each mixer decompresses the audio frames sent from the other mixers by LD and mixes them with the other decompressed audio frames from the associated participants. Note that the mixer, which has a powerful computing ability and large bandwidth, allows the conference to support more users. In Section 5, the performance evaluation will show that despite having a poor computing ability, the mixer can still support more participants than the existing architectures. In this study, we let the initiator of the conference act as the mixer, focusing on the proposed architecture. Unlike the existing server-based architecture, the proposed cooperation-based architecture has no maintenance cost of the server due to the lack of any central controller. Compared with the existing host-based architecture, although the proposed cooperation-based architecture has the mixer role which plays the host role in the host-based architecture, the mixer does not have too many complex tasks. Compared with the existing mesh-based architecture, the bandwidth usage of peers in

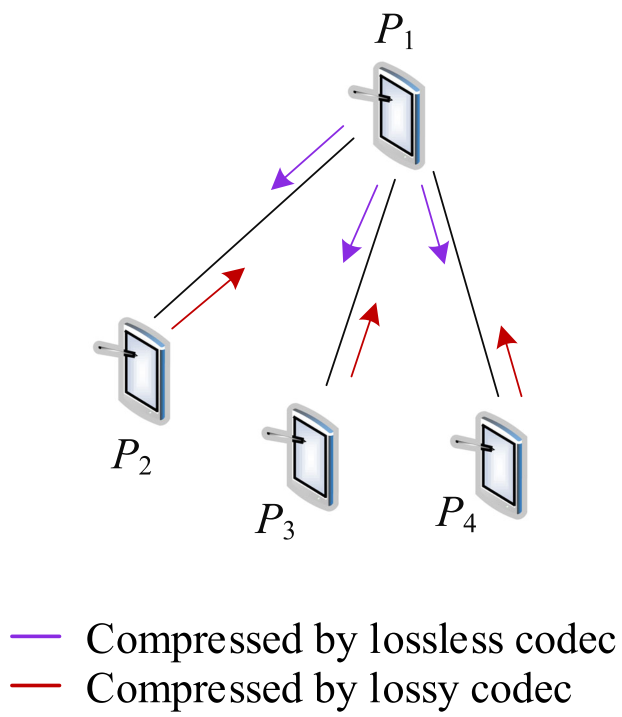

Fig. 6 Basic Concept of Codec Techniques for Communications between the Mixer and Peers

the proposed cooperation-based architecture slightly increases as the number of participants increases. By replacing the lossy codec with the lossless codec in the direction of the mixer to peers, the computing power of the mixer is reduced, and therefore the mixer can process more audio streams from participants.

4.4 Analysis

In this paper, we use the notations listed in Table 3 to analyze the processing latency of the MAC system based on the aforementioned architectures.

In the server-based architecture, for each audio packet, each participant needs to perform echo cancellation, lossy encoding, and lossy decoding. Let T_p^{Server} be the processing time

Table 3 Notations used for analysis

Notation	Definition
T_{AEC}	The processing time for acoustic echo cancellation
T_{LyE}	The processing time for lossy encoding
T_{LyD}	The processing time for lossy decoding
T_{LE}	The processing time for lossless encoding
T_{LD}	The processing time for lossless decoding
T_{DM}	The processing time for de-mixing
T_M	The processing time for mixing

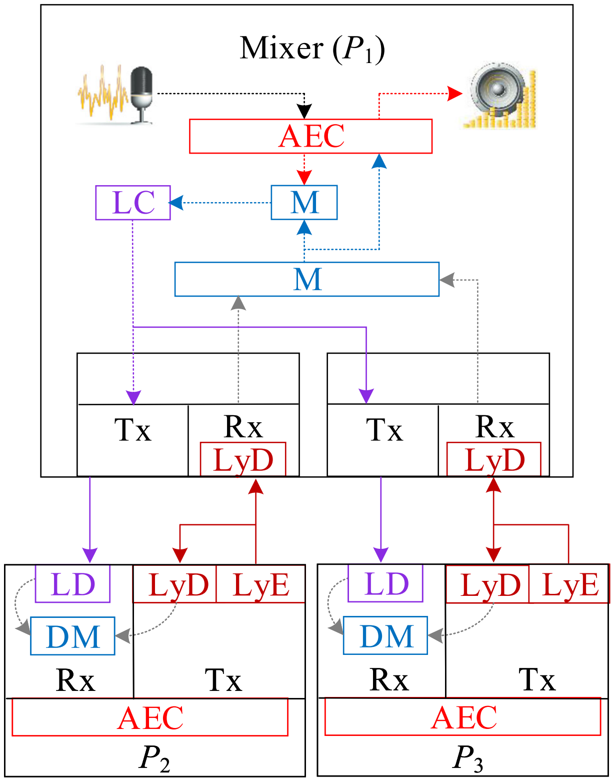

Fig. 7 Operation of the Cooperation-based Architecture with Three Participants

of a participant in the server-based architecture. It can be derived as

$$T_p^{Server} = T_{AEC} + T_{LyE} + T_{LyD}. \tag{1}$$

Let T_s^{Server} be the processing time of the server. We can determine T_s^{Server} as

$$T_s^{Server} = m \cdot (T_{LyD} + T_{LyE} + T_{DM}) + (m - 1) \cdot T_M, \tag{2}$$

where m is the number of active participants.

As for the host-based architecture, since the operation of the participant is identical to that in the server-based architecture, we can derive the processing time of the participant in the host-based architecture, denoted as T_p^{Host} by

$$T_p^{Host} = T_{AEC} + T_{LyE} + T_{LyD}. \tag{3}$$

Let T_h^{Host} be the processing time of the host in the host-based architecture. It can be determined as

$$T_h^{Host} = m \cdot (T_{LyE} + T_{LyD} + T_M + T_{DM}) + T_{AEC}. \tag{4}$$

Recall that there is no central control node in the mesh-based architecture, that is, all participants perform the same procedures. Let T_p^{Mesh} be the processing time of the participant in the mesh-based architecture.

$$T_p^{Mesh} = T_{AEC} + T_{LyE} + (m - 1) \cdot T_{LyD} + (m - 2) \cdot T_M, \tag{5}$$

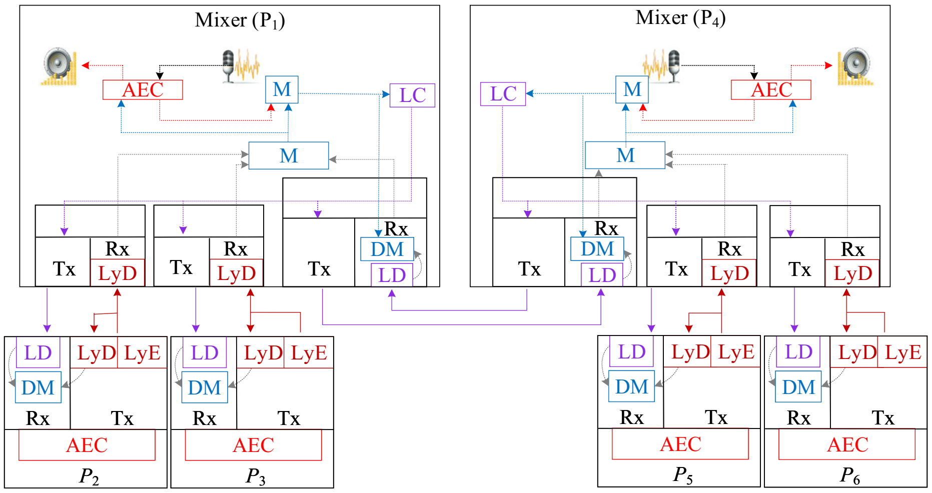

Fig. 8 Operation of the Cooperation-based Architecture with Six Participants. Two Participants are assumed to have been Selected as the Mixers

where m is the number of active participants.

Let T_p^{Coop} be the processing time of the participant in the proposed cooperation-based architecture. Thus, we can derive T_p^{Coop} as

$$T_p^{Coop} = T_{AEC} + T_{LyE} + T_{LyD} + T_{LD} + T_{DM}. \tag{6}$$

Note that the processing time of the mixer in the cooperation-based architecture varies depending on the number of mixers. Let T_m^{Coop} be the processing time of the mixer and N_{mixer} be the number of mixers. We can derive T_m^{Coop} as

$$T_m^{Coop} = (m-1)\cdot(T_{LyD} + T_M) + T_{AEC} + T_{LE} + N_{mixer}\cdot(T_{LD} + T_{DM} + T_M), \tag{7}$$

where m is the number of active participants.

5 Performance evaluation

5.1 Setup

This study considers a simple network without a network address translation [11] problem to evaluate the performance of processing latency. We used the C programming language to implement MAC systems with the different architectures discussed in this paper and ported

Table 4 Mobile devices used in the experiments

Feature	HTC One M8	HTC One X	Samsung E7
CPU	2.5 GHz	1.5 GHz	1.2GHz
Cores	4	4	4
RAM	2 GB	1 GB	2 GB

Table 5 Latency of each module with 10 ms of audio data

Module	HTC One M8	HTC One X	Samsung E7
LyE	1.02 ms	1.38 ms	1.7 ms
LyD	0.07 ms	0.08 ms	0.11 ms
LE	0.18 ms	0.33 ms	0.43 ms
LD	0.06 ms	0.11 ms	0.15 ms
M	0.05 ms	0.07 ms	0.009 ms
DM	0.01 ms	0.015 ms	0.02 ms
AEC	0.85 ms	1.43 ms	1.62 ms

them on different mobile platforms. Table 4 presents the mobile devices used in this study and their corresponding features.

5.2 Results

This study uses the parameters listed in Tables 4 and 5 to derive the preliminary experimental results. Table 5 presents the latency results of the different modules with 10 ms of audio data. The latency was derived from 10,000 rounds on average. According to Table 5, we can find that the processing time of LyE is very long, compared with the one of LE. Therefore, it is necessary to reduce the execution time of LyE to support more participants. That is why the proposed cooperation-based architecture adopts LE instead of LyE. Moreover, the data sizes derived from the lossy and lossless encoders were 26 and 320 bytes, respectively. Although the compressed data size difference is 12 times, the cooperation-based architecture supports more participants. In terms of bandwidth usage, the proposed cooperation-based architecture supports more participants at the expense of consuming more bandwidth. However, the high-speed mobile network enables mobile users to have a large amount of bandwidth, so we don't consider bandwidth limitations.

According to the processing latency of each module, we evaluated the processing latency performance of mesh-based, host-based, and cooperation-based MAC systems with *m* participants for 10 ms of audio data, as presented in Table 6. As shown in Table 6, the processing latency of the mesh-based peer, host-based host, and cooperation-based mixer are proportional to the number of participants. Figure 9 illustrates the relationship among the number of participants, architecture, and device type for the processing latency. No matter which computing power was used, the trends of each architecture were the same. As expected, the processing latency of the host-based architecture significantly increased

Table 6 Results of the processing latency (*m*s) of the three architectures

Architecture	HTC One M8	HTC One X	Samsung E7	
textitmidrule Mesh-based	Peer	1.87+0.07(m-1) +0.005(m-2)	2.81+0.08(m-1) +0.007(m-2)	3.32+0.11(m-1) +0.009(m-2)
Host-based	Host	1.105(m-1)+0.85	1.482(m-1)+1.43	1.839(m-1)+1.62
	Peer	1.94	2.89	3.43
Cooperation-based	Mixer	0.075(m-1)+1.03	0.087(m-1)+1.76	0.119(m-1)+2.05
(one mixer)	Peer	2.01	3.015	3.6

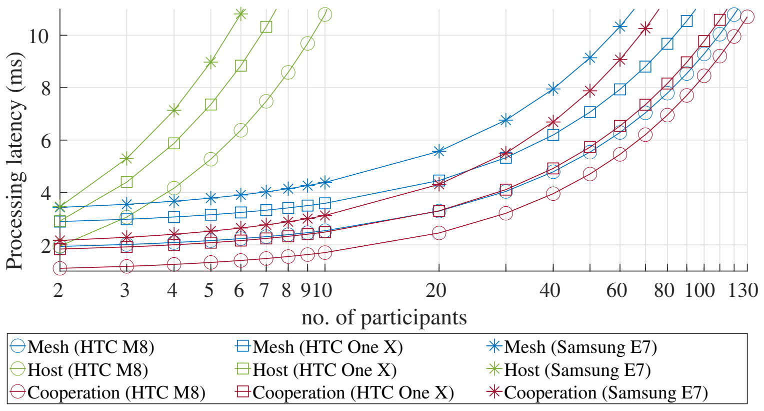

Fig. 9 Processing Latencies of the MAC Systems with Different Architectures

because the host was responsible for too many complex tasks, e.g., generating separate audio frames for each peer. However, the processing latency of the cooperation-based mixer was lower than that of the mesh-based peer because the encoding task was replaced with a low-complexity codec and the mixer only encoded once from the same audio resources for all peers. Therefore, the cooperation-based mixer could process more participants within 10 ms.

In addition to processing latency, system capacity is also a key metric to evaluate the performance of a MAC system. In this study, the capacity was defined as the number of supported participants in the system. Figure 10 shows the capacity of each architecture (i.e., mesh-based [1, 20, 24], host-based [3, 4, 10, 13], and cooperation-based) under 10 ms of audio data per host/mixer. Note that the server-based architecture was not considered in this experiment because the number of supported participants in the system was determined by the capability of the server.

The MAC system considering the proposed cooperation-based architecture significantly outperformed those with the mesh- and host-based architectures. The reason was that with the cooperation-based architecture, the mixer(s) adopted low-complexity codec (i.e., lossless codec) to encode the audio and only encoded once from the same audio resources.

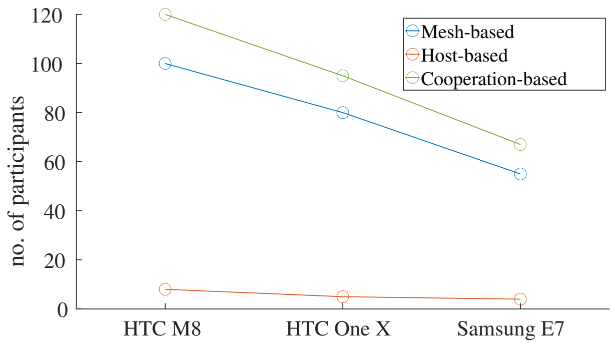

Fig. 10 Capacity of the MAC Systems with Different Architectures

Additionally, the task of de-mixing was distributed to each peer instead of concentrated to one peer (e.g., a host-based host or server-based server).

6 Conclusion

This paper proposed a cooperation-based MAC (CoMAC) system, which adopts a cooperation-based architecture to design a mobile-based decentralized MAC system to support more participants. The proposed architecture operates like the server-based architecture by selecting the participants with powerful capabilities as the mixers to handle the acoustic echo cancellation, encoding, decoding, mixing, and de-mixing. The proposed architecture also uses both lossless and lossy codecs to reduce the processing latency. Although the proposed cooperation-based architecture consumes more bandwidth, it extends the system capacity. Additionally, we introduced a buffer reordering scheme to tackle the problems resulting from network jitters and out-of-order packets reception. We implemented the proposed CoMAC system on mobile devices using open software and open-source libraries for evaluation. The performance evaluation validated that the proposed CoMAC system significantly leads to the low processing latency, compared with MAC systems with the typical architectures. Moreover, the results showed that the proposed CoMAC system can support more participants than other systems.

Declarations

Conflict of Interests The authors have no relevant financial or non-financial interests to disclose.

References

1. Aguirre JV, Alvarez R, Zamora A (2017) Darkcube: a k-hypercube based p2p voip protocol. Peer-to-Peer Netw Appl 10(1):170–183
2. Atal B, Rabiner L (1976) A pattern recognition approach to voiced-unvoiced-silence classification with applications to speech recognition. IEEE Transactions on Acoustics Speech, and Signal Processing 24(3):201–212
3. Caiko J, Patlins A, Nurlan A et al (2020) Video-conference communication platform based on webrtc online meetings. In: 2020 IEEE 61th international scientific conference on power and electrical engineering of riga technical university (RTUCON). IEEE, pp 1–6
4. Chang JC, Liao W (2001) Application-layer conference trees for multimedia multipoint conferences using megaco/h. 248. In: IEEE international conference on multimedia and expo, 2001. ICME 2001. IEEE Computer Society, pp 26–26
5. Chen X, Chen M, Li B et al (2011) Celerity: a low-delay multi-party conferencing solution. In: Proceedings of the 19th ACM international conference on multimedia, pp 493–502
6. Chien YR, Li-You J (2018) Convex combined adaptive filtering algorithm for acoustic echo cancellation in hostile environments. IEEE Access 6:16,138–16,148
7. Dudman J, Backhouse G (2006) Voice over ip: what it is, why people want it, and where it is going. JISC Technology and Standards Watch
8. El-Amine OM, SALL M, BASSE A et al (2021) A webrtc - voip communication platform. In: 2021 10th international conference on internet of everything, microwave engineering (IEMECON), pp 1–4
9. Elleuch W (2013) Models for multimedia conference between browsers based on webrtc. In: 2013 IEEE 9th international conference on wireless and mobile computing, networking and communications (WiMob). IEEE, pp 279–284
10. Elleuch W, Houle AC (2008) Multiparty voice over ip (mvoip) peer-based system for large-scale conference support. In: 2008 IEEE International conference on wireless and mobile computing, Networking and Communications. IEEE, pp 415–416

11. Ford B, Srisuresh P, Kegel D (2005) Peer-to-peer communication across network address translators. In: USENIX Annual technical conference. General Track, pp 179–192
12. Fukui M, Shimauchi S, Hioka Y et al (2014) Double-talk robust acoustic echo cancellation for cd-quality hands-free videoconferencing system. IEEE Trans Consum Electron 60(3):468–475
13. Gu X, Wen Z, Philip SY et al (2008) Peertalk: a peer-to-peer multiparty voice-over-ip system. IEEE Trans Parallel Distrib Syst 19(4):515–528
14. Hamidia M, Amrouche A (2012) Influence of noisy channel on acoustic echo cancellation in mobile communication. In: 2012 24Th international conference on microelectronics (ICM). IEEE, pp 1–4
15. Holmberg C, Hakansson S, Eriksson G (2015) Web real-time communication use cases and requirements. Request for Comments (RFC) 7478
16. Huang X, Li Y, Zakharov YV et al (2021) Affine-projection lorentzian algorithm for vehicle hands-free echo cancellation. IEEE Trans Veh Technol 70:2561–2575
17. Jo QH, Chang JH, Shin J et al (2009) Statistical model-based voice activity detection using support vector machine. IET Signal Proc 3(3):205–210
18. Jung J, Oh R, Lee G et al (2021) Real-time unusual user event detection algorithm fusing vision, audio, activity, and dust patterns. Multimedia Tools and Applications 80:35,773–35,788
19. Kadam P, Kulkarni M, Gaikwad V (2021) Bandwidth management for voip calling through asterisk. In: 2021 2nd Global Conference for Advancement in Technology (GCAT), pp 1–6
20. Khedher DB (2012) A peer-to-peer self-organizing scheme for multiparty session. In: 2012 IEEE International Conference on Communications (ICC). IEEE, pp 6535–6539
21. Kim J, Hahn M (2018) Voice activity detection using an adaptive context attention model. IEEE Signal Process Lett 25(8):1181–1185
22. Krishnakumar H, Williamson DS (2019) A comparison of boosted deep neural networks for voice activity detection. In: 2019 IEEE Global conference on signal and information processing (GlobalSIP). IEEE, pp 1–5
23. Kurtisi Z, Wolf L (2008) Using wavpack for real-time audio coding in interactive applications. In: 2008 IEEE international conference on multimedia and expo. IEEE, pp 1381–1384
24. Li J (2005) Mutualcast: a serverless peer-to-peer multiparty real-time audio conferencing system. In: 2005 IEEE international conference on multimedia and expo. IEEE, pp 602–605
25. Lin CHR, Zhang H, Liu JS et al (2020) Implementation of secure web conferencing, IEEE
26. Opus (2011) Comparison - opus codec. https://opus-codec.org/comparison Accessed 1 July 2021
27. Park JS, Kim SH (2020) Sound learning–based event detection for acoustic surveillance sensors. Multimedia Tools and Applications 79:16,127–16,139
28. Romoli L, Cecchi S, Piazza F (2017) Multichannel acoustic echo cancellation exploiting effective fundamental frequency estimation. Speech Comm 86:97–106
29. Singh K, Nair G, Schulzrinne H (2001) Centralized conferencing using sip. In: Internet Telephony Workshop, pp 57–63
30. Software T (2003) Tta lossless audio codec - codec comparison. http://tausoft.org/wiki/True_Audio_Codec_Comparison Accessed 1 July 2021
31. Su HK, Hsiao WH, Lin YK et al (2009) Design of cluster-based system framework for sip-based multimedia conferencing services. In: 2009 Fifth International Joint Conference on INC, IMS and IDC. IEEE, pp 1381-1387
32. Valin J, Vos K, Terriberry T et al (2012) Rfc 6716: definition of the opus audio codec. Internet engineering task force (IETF) standard
33. Wang Q, Chen X, Liang R et al (2021) A frequency-domain nonlinear echo processing algorithm for high quality hands-free voice communication devices. Multimedia Tools and Applications 80:10,777–10761,
34. Wijayanto A, Adhitama R, Burhanuddin A (2021) Solsr protocol performance analysis for voip application in mesh topology. In: 2021 IEEE International conference on communication, networks and satellite (COMNETSAT). IEEE, pp 348–353
35. Yang X, Zhou Y, Ma Y et al (2020) A new real-time acoustic echo cancellation algorithm using blind source separation and multi-delay filter. In: 2020 IEEE/CIC International Conference on Communications in China (ICCC). IEEE, pp 289–293