

MULTI-PARTY AUDIO CONFERENCING BASED ON A SIMPLER MCU AND CLIENT-SIDE ECHO CANCELLATION

Junlin Li

Georgia Institute of Technology
School of Electrical and Computer Engineering
Atlanta, GA 30332

Li-wei He, Dinei Florêncio

Microsoft Research
One Microsoft Way,
Redmond, WA 98052

ABSTRACT

Traditional multiparty audio conferencing uses a star-shaped topology where all the clients connect to a central MCU (Multipoint Control Unit). The MCU mixes the signals from the speakers, encodes it, and sends back the encoded signal to each client. To prevent the speakers from hearing their own voices, the MCU has to produce and encode a different mixed signal for each speaker. As a result, the CPU load on the MCU increases proportionally to the number of speakers in the conference. In this paper, we introduce a new conferencing architecture, where the MCU produces a single encoded signal sum of all received signals and each client is responsible for removing its own signal if necessary. This architecture can substantially reduce CPU load on the MCU. The major challenge, however, is that the client's original speech is non-linearly distorted by the MCU encoding process. Simply subtracting the original speech from the mixed signal would produce an echo-like distortion. We solve that problem using a novel algorithm which completely removes the echo with minimal artifacts. Mean Opinion Score (MOS) results imply that the proposed algorithm works well, making the proposed multiparty audio conferencing architecture promising.

1. INTRODUCTION

As the Internet evolves into a global communication network, audio conference over the Internet is becoming more pervasive in our everyday life. Many companies now offer audio and video conference services for free. MSN Messenger, for example, hosts over 50 million audio and video conference sessions per month. To these companies, improving communication experience while holding down the operation cost is essential.

The simplest multiparty audio conference architecture is full-mesh [1], where all clients directly communicate to each other. The major advantage of this architecture is its high audio quality because the audio packets are encoded once and require only one hop from the sender to the receiver. And since the packets travel directly between the sender and the receiver, the service provider incurs no additional cost after the connections have been established. However in full-mesh, the bandwidth consumption and computation load on each client increase proportionally to the size of the conference. For a conference with N clients, each client will need $N - 1$ upload and download units of bandwidth and $N - 1$ decode operations. For large conferences, the clients will run into resource problems. Furthermore, direct connections between clients often might not even be possible due to NAT traversal and firewall issues.

To complement the full-mesh architecture (in case of large conferences or NAT/firewall problems), an MCU-based (Multipoint Control Unit) architecture is often used. In such architecture, all clients

Table 1. Distribution of multi-talk in a 4-person conference

# of Talkers	0	1	2	3	4	Avg
Distribution	9%	49%	31%	8%	3%	N/A
# enc. (Old)	0	0	3	4	5	1.40
# enc. (New)	0	0	1	1	1	0.42

connect to a central MCU, which is responsible for mixing the signals from the speakers and sending back the encoded mixed signal to all the clients. It is obvious that the bandwidth and CPU load on each client are greatly reduced since only one unit of upload and download bandwidth and one decode operation are required. However, in order to prevent the speakers from hearing their own voices, the MCU has to produce and encode a different mixed signal for each speaker. Consequently, the CPU load increases proportionally to the number of speakers.

There are other audio conference architectures such as tandem-free [3, 4], peer-to-peer relay and distribution tree, which involve various trade-offs among audio quality and delay, client CPU and bandwidth load, and service operator cost. In this paper, we focus on reducing CPU load on the MCU in the MCU-based architecture by taking the advantage of the CPU power on the client. In our proposed scheme, the MCU produces a single encoded signal sum of all the received signals and each client is responsible for removing its own signal if necessary. This architecture can substantially reduce CPU load on the MCU. Furthermore, since all clients receive the same packet stream, a broadcast medium can be used, when available, to reduce the MCU bandwidth load.

The remainder of this paper is organized as follows. First, we introduce a new MCU-based architecture for multiparty audio conferencing in Section 2. Then the G.722.1 codec used in the system is briefly reviewed in Section 3, and the proposed client-side echo cancellation algorithm is described in Section 4. Section 5 discuss the subjective test results, and Section 6 presents the conclusions.

2. A SCALABLE MCU-BASED SOLUTION

In an MCU-based audio conference architecture, the MCU just needs to forward the packets to other clients when only one person speaks. However, when multiple people speak, a traditional MCU needs to generate and encode $M + 1$ different output packets (where M is the number of speakers). Using the speaker number distribution figures from [2] (see Table 1 for a typical 4-person conference), the traditional MCU requires an average of 1.4 encodes per output period. In our proposed solution, the MCU simply mixes the speeches from all the clients together, encodes it, and sends that *same* stream to all

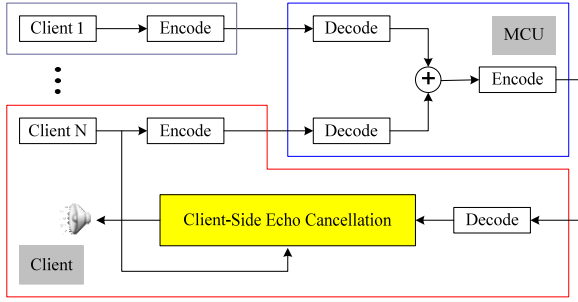


Fig. 1. The proposed MCU-based solution for multi-party audio conferencing.

the clients. Therefore, our proposed solution just needs an average of 0.42 encode per output period – a 70% reduction of the encoding operations. Given that the CPU load on the MCU is dominated by the number of encode operations (e.g. for the specific codec used in this paper ITU-T G.722.1 [5], encoding is 3-4 times more costly than decoding on a typical PC), the overall CPU saving is very significant. Furthermore, since all clients receive the same packet stream, a broadcast medium can be used, when available, to reduce the MCU upload bandwidth cost. For large conferences, the saving in bandwidth can also be substantial.

The major challenge for the proposed solution, however, is that each client needs to remove its own speech (which is non-linearly distorted by the encoding process at the MCU) from the mixed speech before playback. The system diagram of the proposed solution is shown in Fig. 1.

3. OVERVIEW OF G.722.1 ENCODER

Fig. 2 shows a block diagram of the G.722.1 encoder. The Modulated Lapped Transform (MLT) performs a frequency spectrum analysis on audio samples, converting the samples from time domain into a frequency domain representation. The MLT transform coefficients are divided into 16 regions (only 14 regions are used as the bandwidth is 7 KHz), each having 20 transform coefficients, respectively. The region power, defined as the the root-mean-square (rms) value of the MLT coefficients in the region, is determined for each region and quantized with a logarithmic quantizer.

The categorization procedure generates 16 possible categorizations to determine the parameters used to quantize and code the MLT

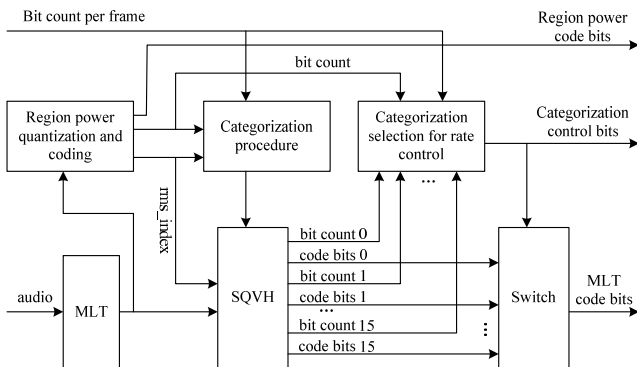


Fig. 2. Block diagram of the G.722.1 encoder

coefficients. Each categorization consists of a set of 14 “category” assignments, one assignment for each of the 14 regions. The category assigned to a region defines the quantization and coding parameters such as quantization step size, dead zone, vector dimension, and variable bit-length code for that region. There are 8 categories: category 0 through category 7. Category 0 has the smallest quantization step size and uses the most bits. Category 7 has only one quantization output value, set to “0”.

MLT coefficients in categories 0 through 6 are normalized, scalar quantized, combined into vectors, and Huffman coded in the Scalar Quantized Vector Huffman Coding (SQVH) module shown in Fig. 2. For each of the 16 possible categorizations, the total number of bits actually required to represent the frame is computed. The categorization with the lowest index is selected from those categorizations which yield bit totals that fit within the bit budget, or the categorization closest is selected if no categorization yields a bit total that fits within the bit budget. It is noted that only the scalar quantization (SQ) in the G.722.1 encoder is lossy.

4. CLIENT-SIDE ECHO CANCELLATION

Each Client receives a signal that includes its own signal and needs to remove that before play back. This resembles other echo cancellation problems, but it is unique in a number of ways. In most other cases, the echo is a linear function of the signal, and is introduced by some natural or uncontrolled phenomenon (e.g., acoustic echo, impedance mismatch, etc.). The task in these applications is exactly to estimate this linear transfer function. In contrast, in our case, the echo was “introduced” at the MCU and is 100% under our control. This means we actually know the gain and delay of the “echo”. Thus, if transcoding at the MCU were lossless, each client could perfectly remove its own signal from the mixed signal. However, non-linear distortions are introduced by the MCU encoder, mainly due to the quantization. This distortion (*i.e.*, quantization noise) is random, but its energy is roughly proportional to the encoded signal, producing a residual signal that sounds like a distorted echo. Our main task is to estimate and remove this non-linear distortion.

Quantization error is generally considered impossible to predict. Nevertheless, ours is a very unique situation: the client has access to both the original signal (which it just sent to the MCU) and to the quantized (mixed) signal. We will try to predict the quantization noise to further reduce the residual echo. To that end, an enforced-quantization based scheme is proposed, as shown in Fig. 3. The original speech will go through two encoding and decoding cycles to simulate the whole process of the echo going through. Since quantization parameters (e.g., quantization step) are input dependent, we force the second encoder in the echo cancellation module to use the same quantization parameters used by the encoder in the MCU. It is worth noting that all the required quantization parameters can be obtained from the bitstream of the mixed signal without extra cost (*i.e.* no additional side information is needed).

We now analyze the behavior of the proposed algorithm for a given client. Let’s call A the signal that the client sends to the MCU, and B the sum of the signals from all the other clients received at the MCU. The MCU will encode the mixed signal $A + B$ into a single common stream and transmit it to all clients. In the following analysis, we denote the samples of signal A and B as a and b , respectively. The signal sample transmitted to the clients from MCU is $Q(a + b) = k_{ab}Q_s$, where $Q(\cdot)$ denotes scalar quantization, Q_s is the quantization step used, and k_{ab} is the appropriate quantization index for $a + b$. Assume a and b are individually quantized to k_a and k_b with quantization error e_a and e_b , respectively, using Q_s , that is,

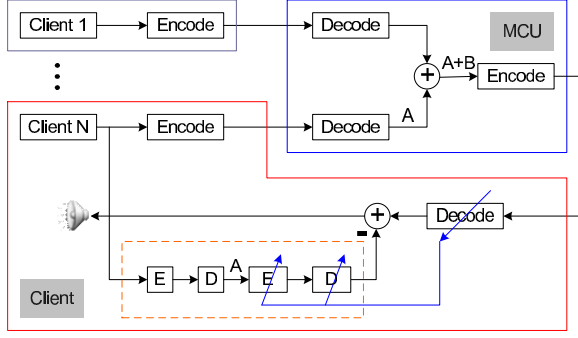


Fig. 3. The proposed client-side echo cancellation algorithm - I: enforced quantization based scheme. In this figure, “E” denotes Encoder, and “D” denotes Decoder.

$a = k_a Q_s + e_a$, and $b = k_b Q_s + e_b$. Then,

$$Q(a + b) = (k_a + k_b)Q_s + Q(e_a + e_b). \quad (1)$$

It is obvious that $|e_a| < \frac{Q_s}{2}$, $|e_b| < \frac{Q_s}{2}$, and $Q(e_a + e_b)$ may be equal to 0, or $\pm Q_s$.

As depicted in Fig. 3, our estimation of the contribution of a to the mixed signal received at the given client is obtained by requantizing a using Q_s . The final signal played at the given client is therefore: $\hat{b} = Q(a + b) - Q(a) = k_b Q_s + Q(e_a + e_b)$. Therefore, the residual distortion after echo cancellation is $b - \hat{b} = e_b - Q(e_a + e_b)$, that is,

$$|b - \hat{b}| = \begin{cases} e_b, & \text{if } Q(e_a + e_b) = 0 \\ Q_s - |e_b|, & \text{if } Q(e_a + e_b) = \pm Q_s \end{cases} \quad (2)$$

In other words, whenever $Q(e_a + e_b) = 0$, the quantization error is the same as if we had never transmitted A . Otherwise, the error may be larger, but still smaller than the quantization step. Listening test shows that this algorithm remove essentially all echo, but it does introduce some perceptible artifacts into the final output speech signal. These have two main origins, which can be easily observed in the Eq. (2). First, the higher quantization step for the mixed signal may quantize too many coefficients to zero. And second, the spurious larger quantization errors may be noticeable in some places.

4.1. Client-Side Echo Cancellation Algorithm - II

We now propose a more elaborate client-side echo cancellation algorithm that alleviates the residual distortion in the previous algorithm. As shown in Fig. 4, this algorithm has three main changes in relation to the basic algorithm described earlier. First, it is performed in the MLT domain instead of time domain. Second, we directly compute the desired signal using a *conditional estimation*, instead of estimating (and subtracting) the echo. Finally, an *adaptive noise-fill* is introduced to further reduce the artifacts.

4.1.1. Conditional Estimation

As shown in the Fig. 4, the original problem can be formulated as follows:

Given a and $Q(a + b)$, find the best estimation \hat{b} of b to minimize the mean-square error $E[(b - \hat{b})^2]$. Here, a and b denote the MLT coefficient samples of the signal A and B , respectively.

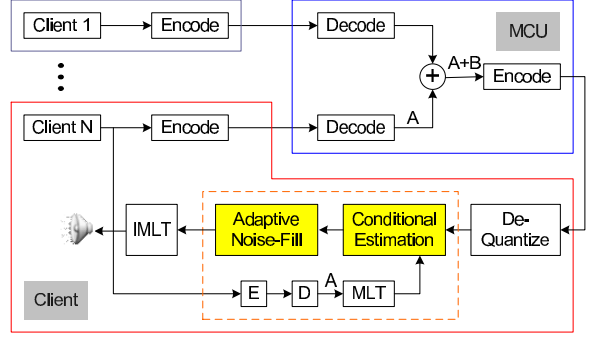


Fig. 4. The proposed client-side echo cancellation algorithm - II. In this figure, “E” denotes Encoder, and “D” denotes Decoder.

It is easy to show that the solution for the above problem is the conditional mean of b , which can be written as:

$$\hat{b} = \frac{\int_l^h x \cdot f(b = x) dx}{\int_l^h f(b = x) dx}, \quad (3)$$

where, $l = Q(a + b) - a - d \cdot Q_s$ and $h = Q(a + b) - a + (1 - d) \cdot Q_s$ are the endpoints of the integration interval, and d is the dead zone of the scalar quantizer defined in ITU-T G.722.1 [5].

To solve Eq. (3), the probability distribution function (PDF) of b , say $f(b)$, is needed, which is unknown. The ITU-T G.722.1 codec groups MLT coefficients in groups of 20, called *regions*. We assume all the 20 MLT coefficients b_i ($i = 1, \dots, 20$) in the same region are independent, identical Gaussian distributed with zero-mean and variance σ^2 . Then $f(b)$ only depends on σ^2 , which can be estimated as follows:

$$\sigma^2 = \frac{1}{20} \sum_{i=1}^{20} \hat{b}_i^2. \quad (4)$$

So the whole problem is to jointly solve the above Eq. (3) and Eq. (4). Note that there are 20 equations included in Eq. (3), one for each of the 20 MLT coefficients in that same region. To solve this system, an iterative procedure is proposed as follows:

1. Initialize \hat{b}_i ($i = 1, \dots, 20$) as:

$$\hat{b}_i = Q(a_i + b_i) - Q(a_i), \quad (5)$$

where, the same quantizer is used for a_i as that for the mixed signal $a_i + b_i$. Actually, the initialized estimation of b_i is just the result of the enforced quantization based algorithm proposed earlier.

2. Iterate the following two steps:

- (a) Estimate σ^2 based on \hat{b}_i ($i = 1, \dots, 20$):

$$\sigma^2 = \frac{1}{20} \sum_{i=1}^{20} \hat{b}_i^2. \quad (6)$$

- (b) Estimate \hat{b}_i ($i = 1, \dots, 20$) based on σ^2 :

$$\hat{b}_i = \begin{cases} 0, & \text{if } Q(a_i + b_i) - Q(a_i) = 0 \\ \frac{\int_l^h x \cdot f(b_i = x) dx}{\int_l^h f(b_i = x) dx}, & \text{otherwise} \end{cases} \quad (7)$$

where, $l = Q(a_i + b_i) - a_i - d \cdot Q_s$ and $h = Q(a_i + b_i) - a_i + (1 - d) \cdot Q_s$, and it is noted that the estimation of \hat{b}_i is set as 0 when b is so small that $Q(a_i + b_i) - Q(a_i) = 0$ to avoid residual echo.

This iterative procedure converges quickly after 2 or 3 iterations.

4.1.2. Adaptive Noise-Fill

In the original ITU-T G.722.1 codec, noise-fill is used for all coefficients in regions assigned category 7 and for coefficients coded as zero in regions assigned category 5 or 6 (since the large quantization step sizes in these categories result in most MLT coefficients being coded as zero). The amplitude of the inserted noise is proportional to the quantization step size, which is related to the region power.

However, the original noise-fill scheme in the ITU-T G.722.1 cannot be directly applied to the proposed audio conferencing system, since here the noise-fill is for the desired speech signal after echo cancellation, but the quantization parameter is determined by the mixed signal. Instead, an adaptive noise-fill scheme is proposed, which addresses the following two questions: (i) where to fill noise, and (ii) how much noise to fill. Noise fill is applied as follows:

- If the desired speech is dominant, compared with the echo speech in this region, (i.e. $E_d \gg E_e$, where E_d and E_e are the quantized energy of the desired speech and the echo speech), the same strategy with the original G.722.1 codec is used, i.e. noise is filled for any zero coefficient in this region if it is assigned category 5, 6 or 7.
- If the desired speech and the echo speech are comparable in this region, and it is assigned category 3, 4, 5, 6 or 7, then noise is filled for the coefficients coded as zero in this region.
- If the echo speech is dominant in this region, i.e. $E_d \ll E_e$, then noise is filled for the coefficients coded as zero in this region regardless of the category assigned.

The amplitude of the filled noise is determined as:

$$N_f = \min \{f_1(Q_s), f_2(E_d)\}, \quad (8)$$

where, Q_s is the quantization step, $f_1(Q_s)$ is a linear function of Q_s , which is same as the original G.722.1 codec, and $f_2(E_d)$ is a linear function of the quantized region power E_d of the desired speech.

The listening test shows that this client-side echo cancellation algorithm not only completely removes the echo, but also greatly reduces the artifacts compared with the basic client-side echo cancellation algorithm proposed earlier.

5. RESULTS AND DISCUSSIONS

To evaluate the effectiveness of our proposed echo-cancellation algorithm (Prop-II) against the traditional MCU-based solution (Trad), we use subjective quality rating from human listeners [6]. In addition to the two algorithms, we added simple signal subtraction algorithm (Naiv) and the proposed solution I (Prop-I) as control conditions. As source, we used 4 clips from a recording of a real 4-person audio conference. Each clip is 5-10 seconds long, and contains multiple speakers. A total of 48 clips were generated by processing the 4 clips with 4 algorithms each coded with G.722.1 codec at three different bit rates (16, 24, and 32 kbps).

In the test, we simulated the effect that the subject is sitting next to one of the 4 conference participants. The test audio clips are generated as stereo files: the left channel contains speech from the participant that the subject is "sitting next to" while the right channel

Table 2. Subjective test results (MOS)

Method	Naiv	Prop-I	Prop-II	Trad.
16 kbps	3.08	3.15	3.81	4.04
24 kbps	3.32	3.30	3.90	3.97
32 kbps	3.46	3.65	3.80	4.15
Overall	3.29	3.36	3.83	4.06

contains the speech of the 3 other participants processed with various echo cancellation algorithms plus a 200 ms delay. By playing the local participant's speech at the same time as the remote speech, the masking effect from the local speech is taken into consideration of the overall echo cancellation quality.

A total of 36 subjects participated in our test, each rating half of the 48 clips. The MOS (min=1, max=5) are shown in Table 2. As expected, the traditional, full-load, MCU solution obtained the highest rating. However, our proposed solution only appears to introduce a small quality degradation while the 2 control algorithms have done considerably worse.

Given the significant amount CPU savings on the MCU and slight quality degradation, we believe that our client-side echo cancellation algorithm offers a viable solution to MCU design.

6. CONCLUSIONS

In this paper, we introduced a new MCU-based architecture for multi-party audio conferencing, which can greatly reduce the computation and bandwidth requirements at both the client side and the MCU at the cost of introducing a non-linearly distorted echo. To remove the echo, a novel client-side echo cancellation algorithm was proposed. The listening test shows that the proposed algorithm not only removes the echo completely, but also minimizes the artifacts, and it outperforms the naive echo cancellation solution greatly. The MOS score also shows that the proposed algorithm only introduces a small quality degradation while gaining a significant CPU saving compared to the traditional MCU-based solution. As future work, we plan to extend the echo cancellation algorithm for different audio codec, such as CELP-based codec, and investigate the impact of the packet loss and delay jitter on the system.

7. REFERENCES

- [1] ITU-T Rec. H.323, "Packet-based multimedia communication systems," Nov. 2000.
- [2] P.J. Smith, P. Kabal, and R. Rabipour, "Speaker selection for tandem-free operation voip conference bridges," in *Proc. IEEE Workshop Speech Coding*, Oct. 2002, pp. 120–122.
- [3] P.J. Smith, P. Kabal, M.L. Blostein, and R. Rabipour, "Tandem-free voip conferencing: A bridge to next-generation networks," *IEEE Communication Magazine*, vol. 41, no. 5, pp. 136–145, May 2003.
- [4] X. Xu, L. He, D. Florencio, and Y. Rui, "Pass: Peer-aware silence suppression for internet voice conferences," in *IEEE International Conference on Multimedia & Expo*, 2006.
- [5] ITU-T Rec. G.722.1, "Coding at 24 and 32 kbit/s for hands-free operation in systems with low frame loss," Sept. 1999.
- [6] ITU-T Rec. P.800, "Methods for subjective determination of transmission quality," Aug. 1996.