ON-THE-FLY AUDITORY MASKING FOR SCALABLE VOIP BRIDGES

ARNAULT NAGLE[1], NICOLAS TSINGOS[2], GUILLAUME LEMAITRE[2] AND AURELIEN SOLLAUD[1]

[1] *France Telecom R&D, 2 avenue Pierre Marzin, 22307 Lannion Cedex, France*
arnault.nagle@orange-ftgroup.com
[2] *INRIA Sophia-Antipolis, 2004 route des lucioles BP 93, F-06902 Sophia-Antipolis, France*
Nicolas.Tsingos@sophia.inria.fr

Endpoints or conference servers of current audio-conferencing solutions use all the audio frames they receive in order to mix them into one final aggregate stream. However, at each time-instant, some of this content may not be audible due to auditory masking. Hence, sending corresponding frames through the network leads to a loss of bandwidth, while decoding them for mixing or spatial audio processing leads to increased processor load. In this paper, we propose a solution based on an efficient on-the-fly auditory masking evaluation. Our technique allows prioritizing audio frames in order to select only those audible for each connected client. We present results of quality tests showing the transparency of the algorithm. We describe its integration in a France Telecom audio conference server. Tests in a 3D game environment with spatialized chat capabilities show a 70% average reduction in required bandwidth, demonstrating the efficiency of our method.

1 INTRODUCTION

A significant number of VoIP systems are available, such as Microsoft Live Messenger (http://get.live.com/messenger/overview), Orange Link (http://orangelink.orange.fr/), Yahoo Messenger (http://fr.messenger.yahoo.com/), Skype (http://www.skype.com/intl/fr/), Teamspeak (http://www.goteamspeak.com/), which enable to create and manage audio chat sessions between remote participants. These systems have seen explosive growth in their usage over the last few years.

Three main configurations are usually used in VoIP audio conferences:

- centralized conferences with a mixing bridge (e.g., Orange Link) that decodes the data, generates a suitable mix for each client and streams the final result,
- loosely coupled conferences which includes multicast or multi-unicast conferences,
- semi centralized conferences with the use of a forwarding bridge. We recall the role of a forwarding bridge in Figure 1. Each client sends one stream to the server and receives from it as many streams as there are remote participants. It must next decode, potentially process (e.g., spatial audio processing) and mix them for final restitution.

In most cases, however, audio conferencing systems do not integrate spatial audio restitution. Spatialized audio gives the listener the feeling of being in a real environment where the voice of each participant is coupled with its location. This location can be arbitrarily set by the software or the user, or e.g. tied to the position of a participant in a game. Although developers can use the capabilities of the 'Xbox Live' (www.xbox.com/live/) for spatialized chat, only Unreal Tournament 2004 (http://www.unrealtournament.com/) seems to use it to date.

With the development of massively multiplayer on-line gaming, chat servers must also face the problem of dealing with an increasingly large number of simultaneous participants. This is particularly true for video game applications where an additional constraint is to send the speech data in separate streams to each participant to allow spatial audio processing prior to restitution.

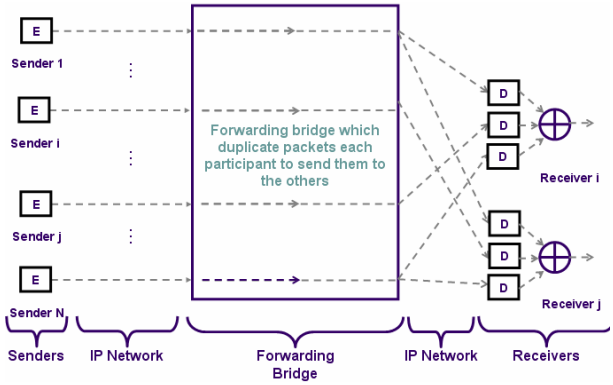

Figure 1: Illustration of the functionality of a forwarding bridge. This bridge receives a frame from a participant, duplicates and sends it towards all the other participants of the audio conference. **E**: Encoding / **D**: Decoding.

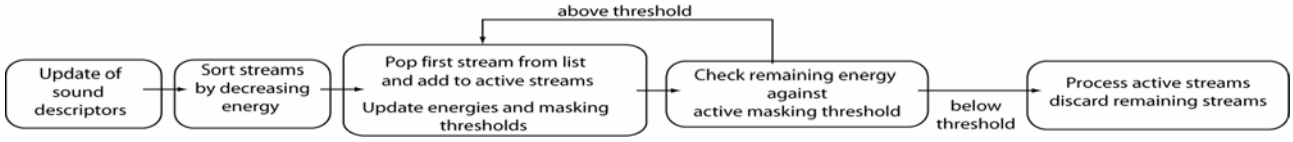

Figure 2 : Overview of our masking algorithm

In this paper, we propose a novel approach in which we stream only the audible audio frames to each participant, depending on an energy importance criteria and a small set of additional audio descriptors computed for each frame. As such, our approach is independent of the coding strategy adopted for streaming the actual audio data.
In the context of a mixing bridge, the approach does not result in any bandwidth gain, except if there is no speaker, but it optimizes the decoding of audio frames prior to mixing. For the same reasons, it is equally useful on client terminals. However, the best use of this algorithm is obviously on the forwarding bridge in order to optimize bandwidth.

This paper presents the details of our masking evaluation technique, its performance and integration into a real-time gaming application, enhanced with spatialized chat capabilities.
In section 2, we present the masking algorithm and the results of an off-line perceptual quality evaluation test. In section 3, we describe its integration in a VoIP bridge. The evaluation of this algorithm in a chat-enabled game is presented in section 4. We finally discuss our approach and outline other possible improvements of our solution before concluding.

2 EFFICIENT MASKING EVALUATION

Our masking algorithm can be decomposed in two steps. First, a set of audio descriptors must be computed for each frame of input audio signal. Typically this will be done on the client and the data will be sent to the server as additional side information together with the coded speech signal. Next, the server performs an on-the-fly masking calculation for each connected client. Since this calculation needs to consider N-1 streams for each of the N connected clients, it must be efficient in order to scale well to large numbers of participants.

2.1 Computing audio descriptors

The first stage of our masking approach computes audio descriptors from which the subsequent real-time operations can be efficiently performed.
For each frame of the input audio signal, we first compute the short-time Fourier Transform (STFT) of the audio data. For our off-line tests scenarios with 44.1 KHz signals, we used 1024 sample Hanning-windowed frames with 50 % overlap, resulting in 512 complex values in the frequency domain. Overlap is not mandatory and can be discarded to avoid additional delays in on-line applications. From the complex STFT, we then compute a number of additional descriptors:

- RMS level, including a spread-of-masking model [1, 2], for a predefined set of *i* frequency bands (e.g., typically 4 to 8 bands on an octave or Bark scale),
- Tonality T calculated as a spectral flatness measure [1]; tonality is a descriptor in *[0,1]* encoding the tonal (when close to 1) or noisy (when close to 0) nature of the signal.

The descriptors can be seen as a compact representation of the signal, typically requiring a few additional kBytes of data per second of audio signals (e.g., 3kBytes/sec. at 44.1 kHz for 1024 sample frames with 50% overlap and 8 frequency bands).
In a client-server setup, this calculation is performed by the client prior to sending the audio data and requires only minimal overhead.

2.2 Masking algorithm

From the descriptors thus obtained we can efficiently evaluate which of the input signals are going to be audible in the final mixture at a given time frame. Signals that have been identified as inaudible can be safely removed from the pipeline reducing both the arithmetic operations to perform and network traffic. Since the calculation must be carried out at each processing frame, it must be very efficient so that it does not result in significant overhead.
The masking algorithm is similar to the one presented in [3, 4] and is illustrated in Figure 2. First, all input frames are sorted according to some importance metric. In [3], a loudness metric was used but some of our recent experiments seem to indicate that the RMS level would perform equally well, if not better on average, due to specificities of some loudness models [5]. If the signals must undergo filtering or equalization operations (e.g., distance attenuation for positional audio rendering), we dynamically weight the RMS level values pre-computed for several frequency-bands to account for the influence of the filtering operations in each band. We can then compute the importance as the sum of all weighted RMS values.
Then, all signals are considered in decreasing importance order for addition to the final mixture according to the following pseudo-code:

$\mathbf{M_{mix}} = 200$
$\mathbf{P_{mix}} = 0$
$\mathbf{T} = 0$
$\mathbf{P_{toGo}} = \sum_k \mathbf{RMS_k}$
while $(dB(\mathbf{P_{toGo}}) > dB(\mathbf{P_{mix}}) - \mathbf{M}_{mix})$
and $(\mathbf{P_{toGo}} > \mathbf{ATH})$ **do**
tag signal S_k as audible
$\mathbf{P_{toGo}} -= \mathbf{RMS_k}$
$\mathbf{P_{mix}} += \mathbf{RMS_k}$
$\mathbf{T} += \mathbf{P_k} * \mathbf{T_k}$
$\mathbf{T_{mix}} = \mathbf{T}/\mathbf{P_{mix}}$
$\mathbf{M_{mix}} = 27 * \mathbf{T_{mix}} + 6 * (1 - \mathbf{T_{mix}})$
k++
end

This process basically adds the level RMS_k of each source to an estimate of the level of the final result in each band P_{mix} (initially set to zero). Accordingly, it subtracts it from an estimate of the remaining level in each band P_{toGo} (initially set to the sum of all RMS levels for all signals). The process stops when the estimated remaining level in each band is below a given threshold M_{mix} from the estimated level of the final result. The process also stops if the remaining level is below the absolute threshold of hearing *ATH* [2]. Threshold M_{mix} is adjusted according to the estimated tonality of the final result T_{mix}, following rules similar to the ones used in perceptual audio coding [1]. In our applications, a constant conservative threshold of -27 dB also gave satisfying results indicating that pre-computing and estimating tonality values might not be mandatory. Note that all operations must be performed for each frequency band, although we simplified the given pseudo-code for the sake of clarity (accordingly, all quantities should be interpreted as vectors whose dimension is the number of used frequency bands and all arithmetic operations as vector arithmetic). In particular, the process stops when the masking threshold is reached in *all* frequency bands.

2.3 Evaluation of the masking procedure

To verify the transparency of our masking evaluation, we conducted off-line quality evaluation tests to assess whether listeners aware of the algorithm principles are able to detect possible artifacts (e.g., over-masking, borderline signals rapidly switching status between masked and audible, etc.).

2.3.1 Experimental Conditions

21 subjects (17 men and 4 women) volunteered as listeners. They were aged from 23 to 40 years old with a median of 29 years old. All reported normal hearing. They all were computer scientists.
Sound reproduction was done over Sennheiser HD600 headsets connected to a laptop computer.

The test stimuli consisted of several mixtures of various sounds. The sounds were chosen in six categories: music (separated tracks of two pieces of pop music, both instrumental and vocal), speech (male and female speakers, speaking English, French, Greek, German, and Polish), environmental noises (transportation noises, animal recordings, usual office furniture noises), mixed (speech, music, noises) and elements of the reverberation of an anechoic recording of percussions computed with the image source method. This latter category is made of several delayed copies of an anechoic recording. This category was chosen because we suspected that such sounds would be difficult cases for our algorithm. In each category, we created three mixtures of sounds, made with different number of sounds. To choose which and how many sounds we mixed, we defined three levels of "masking efficiency" ("low", "medium", and "high") corresponding roughly to cases where respectively 30% to 80% of the signals were found to be masked and could be discarded in the final mixture. Each level was defined by the amount of signals actually removed by our masking algorithm. Hence, eighteen mixtures (six categories, three levels) were created. For each mixture, we created two versions: a reference mix containing all sounds, and a mix resulting from the output of the masking algorithm (i.e., a mixture of the audible sounds only). The mixtures were roughly equalized in loudness in a prior informal session.
We ran a double-blind two-alternative forced-choice (2AFC) with hidden reference procedure: subjects were presented with an interface where three buttons were displayed. The button in the middle (labeled "reference") allowed subjects to listen to the original (unprocessed) sound. The two others (labeled "A" and "B") allowed subjects to listen again to the reference (hidden reference) or to the processed mixture. The mapping between A and B and the processed/unprocessed signals was randomized at each step. Neither the listener nor the experimenter was aware of the mapping (double-blind). Subjects were instructed that one of the two signals (A or B) was different from the reference. The subject had to indicate which one they perceived as different from the reference. They could continuously switch between the three sounds, or restart each at the beginning. They were also able to define portions of the signal for looping playback. Before the test, the algorithm was explained to the subjects and they were familiarized with some clear failures of the algorithm (we had to use an older version of the algorithm to find clear impairments). This procedure was designed to get the subjects trained to identify algorithm failures and to focus on parts of the sounds where the algorithm may provide artifacts. Our hypothesis was that despite these strict conditions, subjects would be unable to correctly identify the processed sounds.

2.3.2 Analysis and discussion

Averaged over all stimuli, the identification rates per subject ranged from 39 % to 78 % with a median of 55%. Due to the binary nature of the test, it was not possible to post-screen the subject consistency hence we used the responses of all subjects as a source of variation. A t-test [6] was not able to reject the null-hypothesis: "over all the sounds, the identification rate is 50 % in the parent population" (df=18, t_{obs}=0.13, $p(t>t_{obs})>0.05$). This indicates that subjects were overall unable to identify the processed sounds better than chance.
To examine each sound individually, we performed 18 Pearson χ^2 tests with one degree of freedom [6] over the identification rates per sound (i.e. averaged over the subjects). The χ^2 hypothesis is "the identification rate is 50 %". As this test is repeated for each sound, we have to use a Bonferroni procedure for multiple comparisons [6], which amounts to decreasing the threshold of significance: $p<0.05/18$. The results of the test are represented in Figure 3.

Identification rates range from 42.8 % to 66.7 % with a median of 52.4 %. Once again, the null hypothesis of the χ^2 test could not be rejected for any of the stimuli. We obtain the same results if we consider only listeners with a musical background. It was not possible to find any relationship between identification rates and type of sound or efficiency of the masking. Thus, we can conclude that signal degradations introduced by applying our masking algorithm to compute mixtures of sounds are statistically unnoticeable.

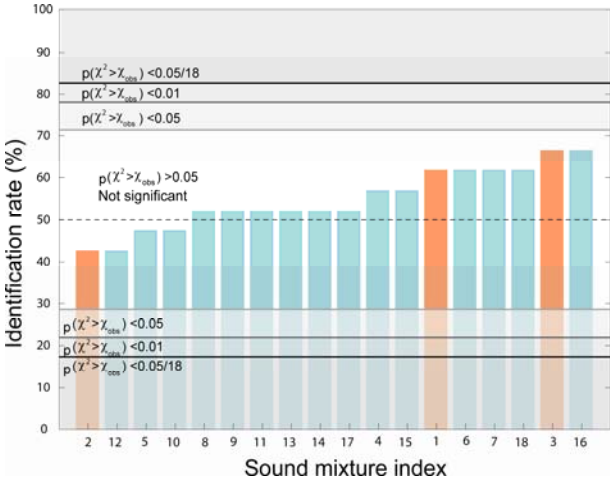

Figure 3 : Rates of identification of the processed mixtures (with masking) and results of the χ^2 test. Highlighted columns correspond to mixtures of speech signals.

However, during post-experimental interviews, several subjects reported to have found differences. Their descriptions were consistent with the expected artifacts (for instance, most of them have reported that reverberations were shorter than in the reference mixture, probably due to over-masking). Furthermore, one subject was able to precisely indicate some part of a mixture where he found a difference. Examination of the sound revealed that the algorithm had actually removed some parts of the mixture at this very location. This shows that some subjects are able to hear the differences introduced by the algorithm in some isolated cases which could not be uncovered by our test procedure. Hence, we can conclude that the masking algorithm is globally transparent, even if we can not exclude that a trained listener, listening carefully, may detect some localized artifacts.

3 INTEGRATION INTO A VOIP BRIDGE

In this section, we describe the integration of the masking algorithm into a VoIP bridge.
Figure 4 illustrates the complete audio treatment chain, from the recording of the voice of one participant P1 to the listening of the spatialized sound of another one P4. Two other participants P2 and P3 are also part of the conference. In our final gaming application, we take into consideration the spatial position of the participants in the game. This is why the Virtools game client appears on the figure. However, it is not directly related to the masking procedure which would work similarly for a non-spatialized audio output.
Each client-listener is processed as P4 and each client-speaker is processed as P1. In the following, the different steps are presented from the client and server perspectives, but temporally occur in order 1 to 5.

3.1 Client Side

3.1.1 Steps 1 and 2

The sound is recorded with the microphone of participant P1, digitized and separated into time- frames (typically 960 samples at 16kHz). For each acquired frame, the client computes necessary audio descriptors (as described in Section 2.1). To avoid additional delay, no overlap was used when analyzing the input frames. Hence, audio descriptors are computed every 60 ms. Next, frames encoded by an audio coder (we used a France Telecom codec at 32 kbits/sec equivalent to standardized G.722.1), together with their side information and the position of the user given by the Virtools client, are multiplexed into a network packet and sent to the server.

3.1.2 Step 5

At the reception end, e.g., for participant P4, audio frames are decoded from audio packets and sent to the Virtools client for final mixing and restitution.

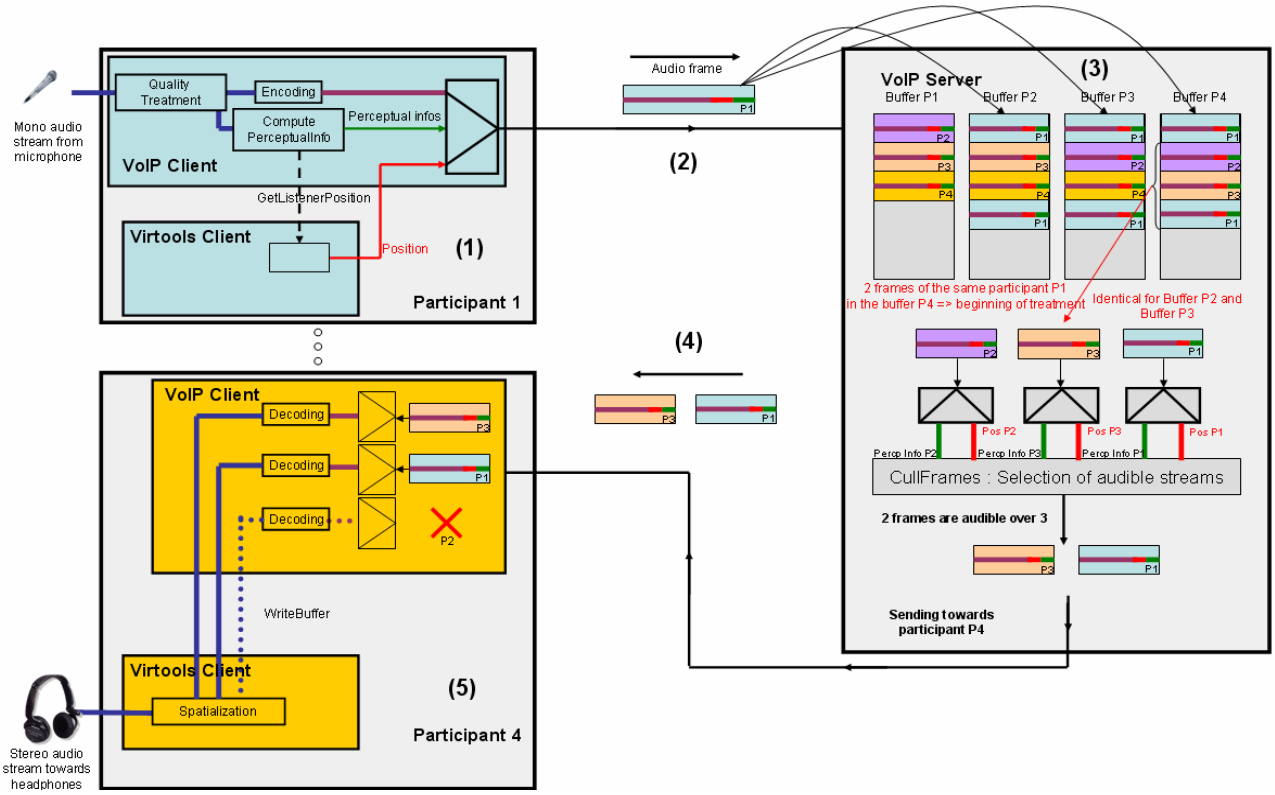

Figure 4: Overview of the VoIP /Virtools processing pipeline for a spatialized chat application.

3.2 Server Side

3.2.1 Step 3

Each audio packet (e.g., originating from P1) arrives at the VoIP server which duplicates it and inserts it in the buffers of all the others participants. On this server, each client has a buffer queue where incoming packets are stored while awaiting transfer to their final destination.

When two packets of the same participant (e.g., P1) are present in a buffer of target participant (e.g., P4), the masking calculation is launched for this buffer.

This treatment extracts perceptual descriptors of each packet available in the buffer except for the last received packet (e.g., the last one of P1). The masking algorithm, enables the selection of audible packets thanks to their associated descriptors and positions in the virtual environment (e.g., to account for distance attenuation). An order of importance is created and allows us to keep the audible packets at the output. In the situation illustrated by Figure 4, only two packets (those of P1 and P3) over three are audible for P4.

In order to avoid possible artifacts due to borderline signals rapidly alternating between audible and masked from one frame to the other, we implemented a smoothing function which remembers the two last results of the masking evaluation and decides to send the packets or not. Initially the algorithm does not send packets and it changes its state only if the current result of the algorithm and the last two are identical and opposed to the current state. A drawback of this approach is that it can erroneously discard the beginning of a sentence.

Note that a packet can be heard by some participants but not by others. Hence, each masking evaluation is performed independently for each participant.

3.2.2 Step 4

Audio packets that have been accepted by the culling function are next sent to the participant (P4 in our case).

3.3 Implementation issues

Some side effects might appear due to the recording quality on the various client platforms. In our case, tests were conducted with average consumer grade audio microphones soundcards and headphones. If no participant speaks at one time instant, the background noise can potentially be considered as a meaningful signal and will be transmitted. In fact, when there was no speaker, all the packets were sent because they all had roughly the same importance. In other cases, only packets with predominant background noise were sent. In those cases, we ideally do not want to transmit the data. However, the masking evaluation procedure does

not treat background noise differently from meaningful speech data.
We first experimented with a fixed threshold to suppress the background noise frames but it was found to depend too much on the hardware.
The implemented solution was to keep setting a very low threshold but to add a quality audio box on each sending terminal. This component located just after recording (e.g., see P1 on Figure 4) reduces noise and increases the audio level when speech is present, in order to help the masking algorithm. This noise reduction component contains:

- A high-pass filter with cut-off frequency set at 50Hz, due to the use of a wideband coder.
- A Noise Reduction block,
- And an Automatic Gain Control block to equalize the level of the audio streams of each participant.

Alternate noise removal or gating strategies could be used to solve this problem [7-10] which is common in VoIP applications and not directly related to the proposed masking evaluation.

4 EVALUATION FOR IN-GAME 3D CHAT

In order to test our algorithm in a more realistic framework, we decided to integrate the France Telecom (FT) VoIP software into the game *Flower Power Shooter* (FPS) created by the company Virtools.
The game FPS is available on "Virtools of Dassault Systèmes" website for test on http://www.virtools.com/applications/games-fps.asp. It is a multi-user game whose goal is to "shoot" the others, with paint-ball guns. Screenshots of in-game action are shown in Figure 5. Demonstration videos are available from:
http://www-sop.inria.fr/reves/OPERA/videos.
The goal was to test our masking-based optimizations for in-game spatialized chat, which is likely to create simultaneous multi-talker dialogs.

Figure 5 : Screenshots of the game Flower Power Shooter

4.1 Integration of the VoIP component in FPS

For audio conferencing applications, we saw that the best use of our masking algorithm is the case of a forwarding bridge.

Both FPS and FT VoIP software have client-server architectures. FT VoIP uses a forwarding bridge and its own signaling protocol to manage audio streams. Hence, we chose, as a preferred solution, to link the clients while keeping the two servers independent.

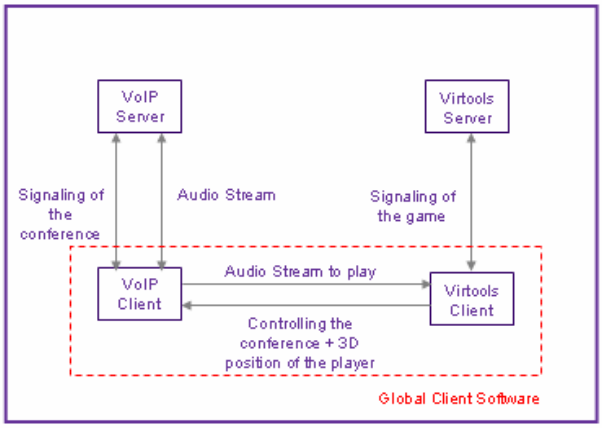

Figure 6 : Integration of the VoIP client in the Virtools client in order to provide some sound from each participant remote participant in the game

This integration leads to the control of the VoIP client by the Virtools Client, as shown in Figure 6. When a player enters the game, the Virtools server informs the other Virtools clients already present the game of his arrival. The Virtools clients signal their VoIP clients to create a new incoming stream for the new participant. The creation of the conference on the VoIP server is done at the first demand from a VoIP client. Moreover, the Virtools client continuously informs its VoIP client of the position of the user in the virtual environment.
The Virtools game client provides spatial audio rendering based on the location of each participant in the 3D game environment. The current implementation uses *openAL* hardware accelerated positional audio rendering (http://www.openal.org). Hence, upon reception, the VoIP client must feed its associated Virtools Client with audio frames received from the others participants.

4.2 Experimental setup

In order to test the algorithm, we decided to play Flower Power Shooter using several configurations. All the participants had stereo headsets with microphones (e.g., Sennheiser HMD 280 pro). One platform was used as game server and another as VoIP forwarding bridge. Pilot tests were done for 3 to 5 simultaneous participants using WIFI and Ethernet connections. For each configuration, we used two test-modes: one where nobody talks and we can observe the effect of the algorithm only in the presence of background noise, and another where players play normally while talking to each other.

We must highlight the fact that soundcards, headphones and microphones were not homogenous.

4.3 Results

Results are summarized in the following tables. “Frames normally sent” are frames which would have to be sent by a regular forwarding bridge, without our algorithm. If N participants are in the conference and one of them sends a frame, N-1 frames should be sent from the bridge to the others. Other ratios are based on this measure. “Frames accepted by the algorithm” correspond to the frames accepted by the masking algorithm but some of them can still be discarded as a result of the smoothing algorithm (see Section 3.3).

Table 1 : Results for 3 participants in the game

	In period of silence	In period of silence / talk / multi-talk
Frames normally sent	31163	169179
Frames really sent	18 ⇔ 0%	27189 ⇔ 16%
Frames accepted by the algorithm	208 ⇔ 0.6 %	37218 ⇔ 21.9%

Table 2 : Results for 4 participants in the game

	In period of silence	In period of silence / talk / multi-talk
Frames normally sent	34528	182177
Frames really sent	915 ⇔ 2.6%	30153 ⇔ 16.5%
Frames accepted by the algorithm	2025 ⇔ 5.8%	41715 ⇔ 22.8%

Table 3 : Results for 5 participants in the game

	In period of silence	In period of silence / talk / multi-talk
Frames normally sent	44161	201386
Frames really sent	1000 ⇔ 2.2%	44628 ⇔ 22.1%
Frames accepted by the algorithm	1679 ⇔ 3.8%	53831 ⇔ 26.73%

First, we can see that during total silence, the algorithm enables a significant reduction of the output bandwidth. Of course, due to the quality of the audio hardware and its associated noise, some frames can still be sent because their energy level is more important. In this case, the bridge operates as a Discontinuous Transmission System.

In periods of silence, talk or multi-talk, the reduction of output bandwidth is again quite important. We can observe the influence of the smoothing algorithm and notice that the percentage of accepted frames increases due to the augmentation of dialog possibilities. However, it does not tend to significantly grow with the number of participants.

These results depend of course of the willingness of the players to chat, of the network, of the quality of the hardware, of the smoothing function, and of the location of the avatars.

5 DISCUSSION

The use of spatialized chat in a real-time environment depends of numerous factors such as the audio hardware, the network bandwidth, the noise rejection technique, the number of results memorized in the smoothing function of the masking procedure, the level of each speaker voice and his localization etc. All of them can affect audio quality but the obtained results are very positive. Due to positional audio rendering, all participants did not have the same restitution level. For instance, the sound of a distant participant is attenuated. As a result, intelligibility can be somewhat compromised but this is a gameplay issue that will probably receive further attention by game designers as spatialized chat capabilities become more widespread. For instance, allowing players to communicate over a walkie-talkie (intelligibility preserved but no spatial aspects) or through more physical positional audio (intelligibility can be compromised by obstacles, attenuation but direction of sound is perceived) could certainly be used to drastically modify gameplay for games where teamwork is required between participants.

On the technical side, further improvements can be added to our system.

We tested a Voice Activity Detection block (VAD) piloting a Discontinuous Transmission (DTX) in the VoIP system and found it to work quite well. However, the integration of the VAD and DTX blocks into the Virtools environment created some problems; hence our results are reported without these blocks. These tools would have helped the VoIP server selecting audible audio frames and reduce the problem highlighted in Section 3.3. Furthermore, a perceptually-based noise reduction model [11] could be used instead of a simple energy-based model.

Perceptual data and positions are currently just added in audio packets without trying to further compress them to reduce the bandwidth. For instance, audio frames of 60 ms in our case are coded with 240 bytes and 3D position with 12 bytes (one 32 bit-float for each dimension). Moreover, in order to reduce the bandwidth from the server towards terminals, we could modify the audio packets with the goal of not transmitting the perceptual data and position.

Assuming a fixed bandwidth from the VoIP server to terminals, the use of variable rate coder driven by the importance factor computed by the masking function would be an interesting extension. In fact, it will allow

the optimization of the available bandwidth in the spirit of [4, 12, 13]. Streams output by a scalable variable rate coder can be cut anywhere and the audio quality depends on the length of the selected audio data. Hence, in this case, no decoding would have to be done to adapt the bandwidth, which remains consistent with the usual concept of a forwarding bridge.

In our application, we do not deal with sound effects coming from the game. Such sounds could mask or be masked by the on-line speech streams. A solution to this problem is to apply again our masking algorithm locally in each client in order to test speech signals against sound effects from the game. Our results using more general sound effects and additional masking tests presented in [3] indicate that the technique would perform equally well in this case.

Currently, the masking decision depends only on energy and tonality criteria. Others ways could certainly be explored to further prioritize the different streams, for instance the use of more perceptually oriented *saliency* metrics, as recently introduced in [14]. Further improvements could be added in the masking algorithm. For instance, we could use a binaural version of this algorithm to better account for spatial audio effects, in a way similar to [3]. However, designing a proper binaural masking model is still a challenging issue.

6 CONCLUSION

In this paper, we presented a novel approach to optimize bandwidth in the context of a voice over IP bridge for spatialized chat application. Our approach is based on an efficient on-the-fly auditory masking evaluation between all the signals generated by each participant. Masking evaluation is performed for all participants in turn so that only the audible audio frames are streamed to each client. We conducted a quality evaluation study showing that our masking algorithm yields transparent output when used to optimize mixing of a number of source signals.

In that context, our masking strategy can be used to typically remove between 20 and 80% of the original content while generating a perceptually-transparent mixture. Moreover, no delay is added by our algorithm, which is very useful in real-time application.

We integrated this algorithm in a forwarding bridge and evaluated its performance in an in-game spatialized chat context. The algorithm was found to perform very well, by discarding almost 70% of the original data in our experiments. Its performance also seems to scale well with the number of participants although we could not conduct massive chat tests during this pilot study.

We believe that such technology can be useful for massive chat applications, especially for future on-line games. Future work includes additional tests with larger numbers of participants and evaluation of a progressive streaming strategy based on the perceptual importance of each source signal.

7 ACKNOWLEDGMENTS

This work was supported by the 2-year RNTL project OPERA, co-funded by the French Ministry of Research and Ministry of Industry. More information can be found at: http://www-sop.inria.fr/reves/OPERA.

8 REFERENCES

1. Painter, J.E.M., Spanias, S., *Perceptual Coding of Digital Audio.* Proceedings of the IEEE, 2000. **88**(4).
2. Zwicker, E., Fastl, H., *Psychoacoustics: Facts and Model*, ed. Springer. 1999.
3. Tsingos, N., Gallo, E., Drettakis, G., *Perceptual Audio Rendering of Complex Virtual Environment {Proceedings of SIGGRAPH'04].* ACM Transactions on Graphics, 2004. **23**(3).
4. Tsingos, N. *Scalable Perceptual Mixing and Filtering of Audio Signals using an Augmented Spectral Representation*. in *8th Int. Conference on Digital Audio Effects (DAFx'05)*. 2005. Madrid, Spain.
5. Gallo, E., Lemaître, G., Tsingos, N. *Prioritizing audio signals for selective processing*. in *International Conference on Audio Displays*. 2005. Limerick, Ireland.
6. Howell, D.C., *Statistical methods for psychology*, ed. PWS-Kent. 1992.
7. Ephraim, Y., Malah, D. Yon, G. Camarillo, *Speech Enhancement Using a- Minimum Mean-Square Error Short-Time Spectral Amplitude Estimator.* IEEE Trans. on Acoustics, Speech and Signal, 1984. **ASSP-21**(6).
8. Mak, B., Junqua, J.-C., Reaves, B. *A robust speech/non-speech detection algorithm using time andfrequency-based features*. in *Acoustics, Speech, and Signal Processing*. 1992. San Francisco, CA, USA.
9. Renevey, P., Drygajlo, A. *Entropy Based Voice Activity Detection in Very Noisy Conditions*. in *EUROSPEECH'01*. 2001.
10. Kang, G., Lidd, M. *Automatic gain control*. in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '84.* 1984.
11. TSOUKALAS, D.E., MOURJOPOULOS, J. N., KOKKINAKIS, G., *Speech enhancement based on audible noise suppression.* IEEE transactions on speech and audio processing (IEEE trans. speech audio process.) ISSN 1063-6676 CODEN IESPEJ, 1997. **vol. 5, no6, pp. 497-514 (32 ref.)**.

12. Kelly, M.C., Tew, A.I. *The continuity illusion in virtual auditory space*. in *112th Audio Engineering Society Convention*. 2002. Munich, Germany.
13. Kelly, M.C., Tew, A.I. *The continuity illusion revisited: coding of multiple concurrent sound sources.* in *1st IEEE Benelux Workshop on Model based Processing and Coding of Audio (MPCA-2002)*. 2002. Leuven, Belgium.
14. Kayser, C., Petkov, C., Lippert, M., Logothetis, N., K., *Mechanisms for Allocating Auditory Attention: An Auditory Saliency Map, Current Biology.* Current Biology, 2005. **15**.