PASS: PEER-AWARE SILENCE SUPPRESSION FOR INTERNET VOICE CONFERENCES

Xun Xu [†], *Li-wei He*[‡], *Dinei Florêncio*[‡], and *Yong Rui*[‡]

[†] Beckman Institute,
University of Illinois at Urbana-Champaign,
Urbana, IL 61801
xunxu@ifp.uiuc.edu

[‡] Microsoft Research,
One Microsoft Way,
Redmond, WA 98052-6399
{*lhe, dinei, yongrui*}*@microsoft.com*

ABSTRACT

A novel tandem-free solution for multiparty VoIP conferences called PASS (Peer-Aware Silence Suppression) is presented. Similar to traditional tandem-free solutions, PASS introduces a limit on the number of concurrent speakers in a conference. But in contrast to traditional solutions, PASS silence suppression and speaker selection are completely distributed, running on each client. No speaker selection is performed at the bridge at all. This configuration leads to better scalability, lower bandwidth occupation and jitter buffer delay, and higher compatibility with a wide variety of network topologies. The key component of PASS, distributed silence suppression and speaker selection, is realized through a robust approach proposed in this paper. Based on a voice activity measure derived using machine learning techniques, this approach is able to reliably suppress silence in complex environments, and perform accurate and transparent speaker selection as well.

1. INTRODUCTION

Voice over IP (VoIP) is becoming mainstream. This can be partially attributed to the proliferation of broadband connections, and the availability of low-cost hardware and software. But equally important is the fact that most technological challenges have been addressed. In particular, solutions to minimize delay, delay jitter, and packet loss have all been extensively researched in the last several years. As these technologies go from research to actual products, research effort must concentrate on the next needs of VoIP technology. One of such needs is high-quality multiparty voice conference. When migrating to VoIP, a central bridge-based architecture mimicking PSTN conferencing system seems to be an obvious design choice. However, various new problems arise when this design is applied to VoIP: 1) A best-effort packet network such as the Internet introduces variable delays and packet losses into the transport process, requiring the bridge to absorb variable delay by a jitter buffer and include some loss concealment mechanisms. 2) The central bridge has to decode the clients' voice packets, sum them, compress, and send them back to each client. Because each client requires his own voice to be subtracted from the sum, the packet compression usually has to be done separately for each individual client. 3) Since the voice packets are encoded twice, a problem known as tandem encoding [6] arises, and the voice quality is reduced.

Because of problems 1) and 2), the CPU, memory and bandwidth load on the bridge increase linearly to the number of clients it is connected to. In order to reduce these costs, silence suppression is often used on the clients. By sending out packets only when actual speech is detected from the microphone, the bridge only needs to receive and mix those packets that contain actual voice. Thus the cost on the bridge is reduced substantially.

In practice, however, the effective savings from silence suppression depends highly on external factors such as microphone quality, its position relative to the user's mouth, the gain of the sound card, the level and type of background noise. Since many of those factors are not controllable by the bridge, the bridge is forced to reserve a significant amount of resources to deal with the fluctuation in the number of incoming packets.

Our proposed solution, namely PASS (Peer-Aware Silence Suppression), tries to reduce the amount of packet fluctuation by restricting the number of concurrent speakers to be less than (including) a pre-set number (e.g. 3). This solution is based on the observation that in a natural conservation, it is rare that more than 3 people speak at the same time. And even when that happens, it is not likely that all of them can be understood clearly -- so it is less important to transmit all of them.

There have been several published solutions in which the bridge runs a speaker selection algorithm [2][6]. Our solution has a major difference from those earlier works: the enforcement of such restriction in our solution is distributed. We propose an improved silence suppression algorithm to be run on each client. Unlike the traditional silence suppression, a packet not only has to pass the client's own speech/silence test, the test is also dependent on the level of voice activity of the packets that the client is currently receiving from its peers. Conceptually similar to the Ethernet protocol, when a client does decide to send the packets, its packets can suppress the clients with lower voice activity level from sending their packets if the conference already has more than a pre-set number of speakers.

The distributed architecture of our solution has a number of benefits comparing a bridge-based one: 1) The client sends less packets so bandwidth utilization is more efficient on both the client and the bridge. 2) It offloads some CPU processing from the bridge. 3) Since the client knows the number of concurrent speakers, it can encode the packets at a different bit rate so the total bandwidth from all those speakers is fixed. Accomplishing this using a bridge-based algorithm will require a scalable audio codec. 4) Most importantly, a distributed algorithm can be applied to a variety of network topologies (such as full mesh, bridge-based, or a hybrid of the two) thus allowing the benefits of speaker selections to be applied to more voice conferencing scenarios.

The remainder of this paper is organized as follows. In Section 2 we will give a brief overview of some of the existing works and compare them against our approach. Section 3 describes the basic overview of how the system works. In Section 4 we discuss the voice activity measure and ranking algorithms. Concluding remarks are given in Section 5.

2. RELATED WORKS

As we mentioned, the traditional architecture for VoIP multiparty conference comprising a mixing bridge has quite a few drawbacks, such as the tandem encoding, excess jitter buffer delay, as well as the bridge's heavy demand on the bandwidth and computing power. One solution avoiding these problems is the full-mesh architecture [3], where no bridge (server) exists, and all clients directly communicate among themselves, sending/receiving packets to/from each other. In this architecture, no tandem encoding is involved since every audio packet only goes through

only one encoding-decoding cycle. Jitter buffer delay is also reduced. The major drawback of full-mesh architecture is its high bandwidth consumption because every client is consistently sending/receiving audio packets to/from all the peers. The scalability of this architecture is rather poor because as the number of parties, say *N*, increases, the whole network will get jammed quickly since there are *N(N-1)* streams flowing across the network at anytime. The computing demand for the clients is another potential problem, since each of them needs to decode and mix signals coming from *N-1* peers. Overall, full-mesh is a simple topology that can provide high quality audio for conferencing among a small number of clients (e.g. <5).

A relatively recent proposal to address tandem encoding problem is Tandem Free Operation (TFO) [6]. Unlike a traditional central bridge, the TFO bridge does not mix the packets into a single channel. In order to limit the bandwidth, it forwards packets from at most *M* channels (e.g. 2 or 3, usually *M*<<*N*) at any moment, assuming there are at most *M* concurrent speakers - note that this is a reasonable assumption for conferences in real life, and is also validated by experiments in [6]. In TFO, each client sends some auxiliary bits along with each voice packet. When each voice packet arrives at the bridge, a speaker selection algorithm is run at the bridge to decide if it is going to forward to the other clients or is simply dropped. TFO can be thought of as a bridge-based silence suppression, where channels not selected as the top M channels are treated as if they were in silence. By using this technique, the upper bound of incoming bandwidth to each client is *M* times of a single channel. It is possible to limit the bandwidth further if a scalable audio codec is used and the bridge can do some bit chopping.

However, in the TFO architecture the bandwidth utilization is still not optimized, since all the clients are consistently sending audio packets to the bridge, whereas most of these packets are simply dropped by the bridge through the speaker selection procedure. In other words, these "useless" packets are sent out anyway, resulting in the unnecessary bandwidth occupation in the bridge's downloading channel and the uploading channel of each client as well.

3. SYSTEM OVERVIEW

Our PASS system has a decentralized architecture, the idea distinguishing it from conventional full mesh architecture is that here the clients are more intelligent and able to decide whether to send out packets based on the states of both itself and its peers. Similar to TFO, in PASS the number of concurrent speakers is limited to be *M* (again, usually *M*<<*N*), which implies the existence of a speaker selection mechanism. However in PASS, speaker selection is done separately by each client, unlike TFO where the bridge performs the task.

In the PASS system, each audio packet contains a *Voice Activity Score* (*VAS*), which quantifies the level of voice activity (not simply the *volume*, as we shall see in Section 4) of the audio frame encoded in this packet. Each time a client captures an audio frame from the microphone, it computes the VAS. Subsequently, silence suppression is performed for current audio frame by comparing its VAS with a threshold. If the VAS is below the threshold, the audio frame is considered as silence and discarded immediately. Otherwise, the client further compares its own VAS with those of its peers, which it obtained from the incoming packets. If the local client finds itself ranked among the topmost *M* clients, it encodes the audio frame and sends out the packet (with the VAS embedded in) towards all the other clients. Otherwise, it knows at once that it doesn't have the chance to be heard, thus discarding the audio frame.

The computational load to perform above silence suppression and speaker selection algorithm is quite low. The only relatively computation-intensive routine is the calculation of VAS. However, notice that each client only needs to calculate its own VAS and those of the peers can be obtained through partial decoding of the incoming packets. Of course, in order to replay the audio signals of the peers, the local client also needs to decode the incoming packets and mix the signals. But recall that since the number of concurrent speakers is bounded by *M*, decoding and mixing only need to be done for at most *M* peers. In our experience, on a Pentium 4 3.2G Hz desktop PC, the entire client consumes only about 3% of its CPU time.

It can be seen that PASS generalizes the silence suppression mechanism in full mesh. Now, each client decides whether to suppress its packets not only based on its own voice activity, but also on the voice activities of its peers, hence the name Peer-Aware Silence Suppression. The PASS architecture is also similar to TFO in that both systems try to limit the number of concurrent speakers through some speaker selection procedure. The difference is obvious, as in PASS this is done separately by the clients, therefore the computing and bandwidth load on the TFO bridge is now completely distributed to each client. In this way, the PASS architecture absorbs the merits of both full mesh and TFO, rendering a VoIP conferencing system with better scalability, lower bandwidth occupation and jitter buffer delay.

4. VOICE ACTIVITY SCORE AND ITS RANKING

The core of PASS lies in how to rank the speakers' voice activities thus only allowing the most deserving ones to send their packets. In practice, a good algorithm should possess these merits:

- **Accuracy**: The voice activity should be correctly identified, and silence should be effectively suppressed.
- **Robustness**: Accuracy should be achieved under complex circumstances. For example, users may be using microphones of different quality, meanwhile the environment around them may contain various background noises of different levels.
- **Transparency**: The users should have a natural conferencing experience. This involves several aspects. First, speaker selection should be fair to all the conferees: different users may have different volume level (due to the user's own voice property, the microphone's distance to the user's mouth, or the gain of the sound card), but the one with lower volume level shouldn't have less chance to be selected when speaking. Secondly, some important phenomena that occur in a face-to-face meeting, such as interruption by raising voice, should be allowed. On the other hand, spurious speaker selection, such as quick switching back and forth, should be avoided.

It can be seen in Section 3 that the key of the speaker selection algorithm in PASS is how to compute VAS and how to rank the clients. We are going to show how our speaker selection algorithm tries to satisfy above requirements.

4.1. The Calculation of VAS

In order to satisfy the requirements for a good speaker selection algorithm, the VAS should hold these properties:

a) Effectively discriminating voice and *silence*. We would like to point out that in practice, the definition of silence is in a broad sense, because there are all kinds of background noises captured by the microphone even when the conferee is not speaking. In our experience, an especially interesting type of noise is the breathing of the user if the microphone is close to his/her mouth. The VAS's calculated for a voice frame and a silence (and/or noise) frame should differ as much as possible, so that silence (in broad sense) can be effectively suppressed.
b) Insensitivity to the volume level. The user with lower volume wouldn't be treated unfairly, i.e. receive lower VAS.
c) A sharp volume increase results in higher VAS in short term, allowing a natural way of interruption by raising one's voice.
d) Temporally smooth. Smooth VAS is not only favored for accurate silence suppression and leads to less spurious speaker switching. More importantly, because each client can only compare its current VAS with the delayed version (due to network transmission) of the peers' VAS's, smooth VAS results in less decision discrepancy among clients.

4.1.1. Feature-based VAS (FVAS)

The most natural choice for the quantity measuring the voice activity of an audio frame is its energy. Since frame energy is easy to calculate, it is widely used for silence suppression [5]. Quantifying voice activity with frame energy involves the assumption that background noises have much lower energy level compared to voice. However in our experience, this assumption is not valid. As we mentioned, a user may use a cheap microphone which has low SNR and captures a lot of environmental noises. Furthermore, some noises, e.g. the user's breathing, have high energy level per se. Therefore, many noises cannot be well discriminated from true voice if only frame energy is considered. Instead, we propose a pattern classification based method to calculate a quantity which is able to identify voice frames robustly, even in the existence of various noises with high energy level.

We model silence suppression as a standard two class classification problem. For each audio frame, the MFCC (Mel-Frequency Cepstral Coefficients) [4] and their 1st and 2nd order temporal differences are concatenated, forming a D=39 dimension feature vector. The task is to design a classifier that decides whether an audio frame belongs to voice or noise. To train the classifier, we collected audio signals recorded in meeting rooms and offices, and labeled each audio frame as "voice" or "noise".

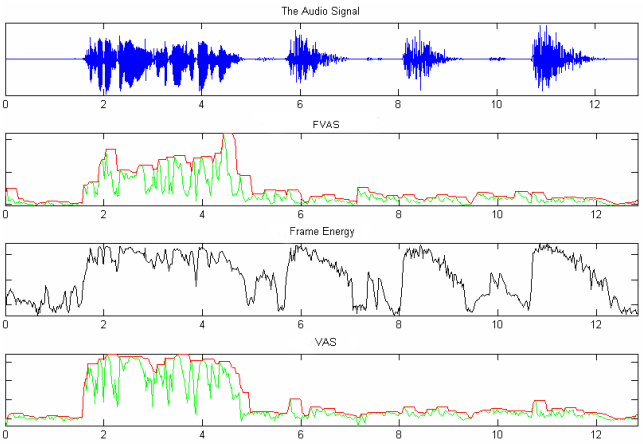

Figure 1: FVAS, VAS and the morphological filtering.

The first step is to seek in the original D-dimensional feature space a lower dimensional subspace, in which the two classes can be well discriminated. By visualizing the training data, it is found that in the feature space the noise samples are surrounded by voice samples and are much more concentrated than the latter. This suggests that silence suppression can be modeled as a "target detection problem" where the target class "noise" should be discriminated from the clutter class "voice". For this type of problem, traditional discriminative method such as Fisher Linear Discriminant is not suitable. Instead, an effective method for seeking a discriminative subspace proposed in [1] is employed. The discriminative projection vector $\mathbf{w}^*$ is obtained by solving:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{\mathbf{w}^T \mathbf{R}_N \mathbf{w}}{\mathbf{w}^T \left[\mathbf{R}_N + \mathbf{R}_V + (\mathbf{m}_N - \mathbf{m}_V)(\mathbf{m}_N - \mathbf{m}_V)^T \right] \mathbf{w}},$$

where $(\mathbf{m}_N, \mathbf{R}_N)$ and $(\mathbf{m}_V, \mathbf{R}_V)$ are the mean-covariance pairs of the noise class and voice class respectively, which are calculated from the training data. This optimization problem can be easily solved through generalized eigenvalue decomposition. It should be pointed out that in [1] just a 1D subspace is sought, by picking the generalized eigenvector with the smallest eigenvalue, whereas in our case d>1 generalized eigenvectors are picked, forming a D-by-d matrix $\mathbf{W}$. The column vectors of $\mathbf{W}$ span a d-dimensional discriminative subspace (d=10 in our experiments).

In the discriminative subspace, we assume Gaussian distribution for the noise class, thus the likelihood that an audio frame belongs to the noise class is given by:

$$p(\mathbf{x} \mid noise) \propto \exp\left(-\tfrac{1}{2}(\mathbf{x}-\mathbf{m}_N)^T \mathbf{K} (\mathbf{x}-\mathbf{m}_N)\right),$$

where $\mathbf{x}$ is the D=39 dimensional feature vector for the audio frame, and $\mathbf{K} = \mathbf{W}\left(\mathbf{W}^T \mathbf{R}_N \mathbf{W}\right)^{-1} \mathbf{W}^T$ is a constant square matrix. We define the audio frame's Feature-based VAS as:

$$FVAS = (\mathbf{x}-\mathbf{m}_N)^T \mathbf{K} (\mathbf{x}-\mathbf{m}_N). \quad (1)$$

Clearly, larger *FVAS* implies that the audio frame is less likely to be noise, in other words, more likely to be voice.

It is worthy noticing that the calculation of *FVAS* (1)can be done fairly efficiently. If we do an orthogonal diagonalization $\mathbf{C} = \left(\mathbf{W}^T \mathbf{R}_N \mathbf{W}\right)^{-1} = \mathbf{U} \Lambda^2 \mathbf{U}^T$, where $\mathbf{U}$ is d-by-d orthogonal matrix and Λ is diagonal - this can always be done since $\mathbf{C}$ is positive semi-definite. Defining $\mathbf{H} = (\mathbf{W}\mathbf{U}\Lambda)^T$, we have:

$$\mathbf{K} = \mathbf{W}\mathbf{U}\Lambda^2\mathbf{U}^T\mathbf{W}^T = \mathbf{H}^T\mathbf{H} \quad (2)$$

And (1) can be written as: $FVAS = \left\| \mathbf{H}\mathbf{x} - \tilde{\mathbf{m}}_N \right\|^2$, (3)

where $\tilde{\mathbf{m}}_N = \mathbf{H}\mathbf{m}_N$ is a d-dimensional constant vector. Equation (3) means that in order to calculate *FVAS*, we only need to project $\mathbf{x}$ to d-dimensional through $\mathbf{H}$, then calculate the SSD (Sum of Squared Differences) between the projected vector and constant vector $\tilde{\mathbf{m}}_N$ in the d-dimensional subspace.

The *FVAS* defined this way is able to effectively discriminate true voice and various noises, including high energy level noises which cannot be suppressed using frame-energy based methods. This is demonstrated in Figure 1. The first row shows the waveforms of an audio clip - among the four speech-like spurts, only the first corresponds to true voice, whereas the other three are actually the breathing noises of the speaker. As shown in the 3rd row, frame-energy could not differentiate the true voice and the

noises. However, the *FVAS*, shown in the 2nd row (green curve), did a good job, correctly assigning very low responses to the high energy level noise frames. The red curve shows the morphologically filtered *FVAS* (see 4.1.3).

4.1.2. Normalized energy and VAS calculation

Although *FVAS* can effectively differentiate voice and noises, it is not suitable for speaker selection since it does not directly reflect the speaker's volume thus does not hold properties b) and c). Therefore, another quantity is introduced, which we call *adaptively normalized frame energy* and denote as $\tilde{E}$. This normalized frame energy is computed as follows:

1. Compute frame energy E.
2. Compute ε, the running average of *voice* energy, i.e. the average energy of most recent (say, in a time window of length T_E=15sec) audio frames which are classified as voice.
3. Obtain $\tilde{E}$ via normalization: $\tilde{E} = E/\varepsilon$.

$\tilde{E}$ holds properties b) and c) that we demanded for VAS. It's clear that $\tilde{E}$ is insensitive to the volume level because it is a normalized quantity. Meanwhile, a sudden increase in E will cause $\tilde{E}$ to increase sharply, but this relatively larger $\tilde{E}$ will last only for a short term till the running average ε follows the increase.

The final VAS is defined to be the linear combination of *FVAS* and $\tilde{E}$:

$$VAS = \alpha \cdot k\tilde{E} + (1-\alpha)\cdot FVAS \quad (4)$$

where the weight $0 \le \alpha \le 1$ is also a function of *FVAS*:

$$\alpha = \left[1 + \exp(b - c \cdot FVAS)\right]^{-1}. \quad (5)$$

Constants *b* and *c* in (5) are chosen so that for noise frames $\alpha \approx 0$ while for voice frames $\alpha \approx 1$. The meaning of (4) is clear: for noise frames, we tend to use FVAS as VAS while for voice frames we favor $k\tilde{E}$. The constant k is chosen to scale $\tilde{E}$ to be comparable with *FVAS*.

4.1.3. Morphological filtering for VAS smoothing

VAS defined in (4) combines the merits of *FVAS* and $\tilde{E}$, thus holding all the properties we listed except for d). Since VAS is calculated independently for each frame, the correlation between neighboring frames hasn't been taken into account so far, resulting in a quickly oscillating sequence. We may smooth this sequence through a (nonlinear) filtering operation:

$$VAS'[n] = \max_{k=0,1,\ldots,K} \left(VAS[n-k]\right), \quad (6)$$

which is actually a unilateral morphological dilation. Note that more complex technique modeling the temporal correlation, such as HMM, could be used instead, but the method we are employing is much faster, easy to implement, and working well for our application. Moreover, this operation also introduces speech hangover which helps avoid audible speech clipping [5].

```
Routine: CompareClient(S1, S2)
If ( PriorityLevel(S1)>PriorityLevel(S2) )
   Return VAS(S1)>=(VAS(S2)-BargeInThres)
If ( PriorityLevel(S1)<PriorityLevel(S2) )
   Return VAS(S1)>( VAS(S2)+BargeInThres )
Else Return VAS(S1)>=VAS(S2)

Subroutine: PriorityLevel(S)
If (S is a current speaker):  Return 1
Else  Return 0
```

Figure 2: The comparison routine for client ranking.

The last row of Figure 1 (green curve) shows the VAS calculated by fusing *FVAS* and frame energy, as well as its morphologically filtered version (red curve). Compared to the FVAS shown in the 2nd row, VAS better reflects the energy properties for the voice frames. Meanwhile, it preserves *FVAS*'s discriminative power on voice and noise frames.

4.2. Ranking VAS for speaker selection

With the well defined VAS, it seems that speaker selection can be done simply by sorting the clients according to their VAS, and selecting the top ranked *M* ones. However, this is not true. Consider, for instance, a simple case where we select *M*=1 speaker from *N*=2 clients, if we simply choose the speaker with higher VAS, spurious switching will occur frequently when the two users are speaking simultaneously. One effective method to prevent this kind of switching is to introduce a "barge-in" mechanism [5]. With this mechanism, the interrupter may suppress the current speaker only if its VAS is higher than the latter's by at least a margin called the *barge-in threshold*.

In the two-client case discussed above, implementing the barge-in mechanism is straightforward. For a general number of clients, we suggest an efficient implementation for barge-in mechanism enabled ranking. The key idea is to design an appropriate comparison routine, so that any standard sorting algorithm (e.g. qsort) can be used immediately. The pseudo code of the comparison routine is given in Figure 2.

5. CONCLUDING REMARKS

Voice over IP is becoming a more and more accepted way of communicating. Compared to the PSTN, the clients of VoIP applications usually have much more powerful hardware available so it becomes easier to run smart algorithms on those clients in order to achieve optimal performance on the whole network. The solution proposed in this paper, PASS (Peer-Aware Silence Suppression), follows this movement. Unlike traditional silence suppression, PASS puts a limit on the number of concurrent speakers in a conference, thus making the number of incoming packets more predictable. Conceptually similar to the Ethernet protocol, our algorithm is distributed to run on each client – making it suitable for a wide variety of network topologies. For example, PASS is compatible with the full mesh architecture and integrating PASS with full mesh clients will considerably improve their scalability.

7. REFERENCES

[1] M. Elad, Y. Hel-Or, and R. Keshet, "*Pattern Detection Using a Maximal Rejection Classifier,*" Pattern Recognition Letters, Vol. 23, No. 12, pp. 1459-1471, Oct 2002.

[2] J. Forgie, C. Feehrer, and P. Weene, "*Voice Conferencing Technology Final Report*," Tech. Rep. DDC AD-A074498, M.I.T. Lincoln Lab., Lexington, MA, Mar. 1979.

[3] ITU-T Recommendation H.323, "*Packet-Based Multimedia Communication Systems*," Nov. 2000.

[4] L. Rabiner and B. Juang, "*Fundamentals of speech recognition*," Prentice-Hall, Inc., 1993.

[5] P.J. Smith, "*Voice conferencing over IP networks*," Master's thesis, McGill University, Montreal, Canada, available online at http://www.tsp.ece.mcgill.ca, Jan. 2002.

[6] P.J. Smith, P. Kabal and R. Rabipour, "*Speaker Selection for Tandem-Free Operation VOIP Conference Bridges*," Proc. IEEE Workshop Speech Coding, pp. 120-122, Oct 2002.