ABSTRACT:

Real time multi-user communication through Voice over IP grants a unique opportunity: the ability to conduct conference calls. Available technologies, however, present challenges such as being proprietary in nature, or limited in features. Especially true is the ubiquitous problem of in-conference person identification. Although solved in video conferencing, the bandwidth demands make this approach to person identification highly limited. This paper presents available front-end and back-en­­d changes to the voice conference status quo with the goal of offering a more intuitive and immersive experience. On the back end, implementation of 2-dimensional sound spatialization gives users depth and richness of voice present in a live conference. Front-end modifications to the traditional phoneinterface focus on the creation of private side chats, with hopes of providing a more natural interaction environment. We attempt to implement these changes on an Android application.

Introduction:

Incepted in 2005 by Cornell Professor Graeme Bailey, OpenComm is a , Human-Computer Interaction(HCI) and Security (HCISec) research group tackling the central question: How do we make mobile conferencing as intuitive and immersive as possible? The short answer is we emulate and build upon the natural conferencing experience. Emulation of what is natural is refreshingly divergent from currently established technologies. This paper serves as a primer to OpenComm's development and implementation of a Voice over IP(VoIP) conferencing platform featuring sound spatialization and private side chats.

Real time voicecommunication, the backbone of mobile conferencing, dates back to the late 19th century. Yet advancements in voice communication have been sluggish. The Public Switched Telephone Network is still the infrastructure backbone of this country's landline system - a should be obsolete technology invented in 1889. Recent developments in cellular technology have led to a surge in the number of mobile devices. Now numbering over five billion devices across the world, cellphones - and by extension - digital communication is becoming an intrinsic part of our future.

The future is now. Google and Microsoft, two industry behemoths, offer users mobile conferencing via VoIP Free open source alternatives exist, using protcols such as the Session Initiation Protocol (SIP) . The usability of VoIP software match and in many cases exceed the traditional landline system. But unchanged is how voice is inputted and outputted. We seem dead-set on faithfully reproducing the sampled voice data.

OpenComm is about modifying the output voice stream through the introduction of sound spatialization. To provide for this innovative idea, OpenComm is also about creating the best graphical user interface. As such, OpemComm is also an innovator in the realm of Human-Computer Interfacing, primairly through the creation of voice specialized subchats.User Experience

OpenComm provides the user with an environment that mimics and builds upon the traditional conference room. It is not always possible to physically accommodate all participants of a conference, especially with the globalization of companies and operations. Thus, OpenCommis important in that it alleivates frustrations users have with the mobile conferencing status quo.. In this section we specifically see how an intuitive application of private subchats give conferences a much more dynamic nature

The user needs only an Android smartphone and a set of headphones. From her phone, the user—Alice, for example—may open up the OpenComm application to get started. Then, the creation of a virtual conference is as easy as inviting people from Alice's contact list.

Within Alice's conference,the three curved lines at the bottom of the conversation area indicate Alice's relative position to other users, whom are represented each by a colored border icon.. These icons are draggable on the screen.. This distance from her to each person represents the arrangement in the conference space. This arrangement is reflected in the audio feedback through the headphones via a method called sound spatialization. More details on sound spatialization later.

If Alice places Bob’s icon to her right, she will primarily hear Bob’s voice coming through her right headphone. As she drags Bob’s icon to her left, the sound will transition seamlessly until Bob appears to be speaking to Alice from the left. As Bob’s icon is dragged farther away from Alice’s, Bob's volume will decrease, and as he is brought closer, Alice will perceive Bob to be louder. Note that each user can have a unique configuration of the icons, and as such each user is granted full customization powers, as per the user's preferences.

in a conference scenario, Alice may want to convey some information to Bob in private. In the real world, we are limited by the methods we can use to achieve this. Alice may simply whisper to Bob, but only if he is sitting next to her. Or with the advancement of mobile technology, Alice may discretely send Bob a text message as an alternative. Both of these options are fairly obvious to their peers and are hardly foolproof. In the virtual world, we are able to guarantee secure communciation channels through the notion of side chats. Below the conference space is a bar containing several squares. The anchored leftmost square, always links back to the main conference. The remaining squares are a scrollable list of the user’s side chats. Alice can create a new side chat by clicking on the plus button and inviting members of the conference. The side chat is similar to the main conference in functionality, including sound spatialization. However, if Alice is in one of her side chats, only the people in that chat can hear her. She can still hear all that is going on in main conference at a lower volume than the voices of the people in the chat. Alice is given free reign to say whatever she wants to Bob without the threat that everyone else might hear her. Also key is the lack of notice in the main conference menu with regards to being involved in a side chat. This grants a further layer of anonymity to protect the conversation.

Each chat, including the main conference, has a moderator; the moderator is the person who created the chat. The moderator is the only person who has the ability to add and remove users. Other users may send a request to the moderator to invite someone from their contact list. If a user leaves a chat for which he is the moderator, he is given the options to either close the chat or give moderator access to another person.

With a simple, intuitive user interface, OpenComm gives the deserved attention to an oft-overlooked aspect of conferencing - the ability to hold subchats.

Back End:

Granting users the immersive experience of a real world conference is a goal of OpenComm. As such, the backbone of the conference - voice - has to be engaging. We propose a fundamentally different approach to how voice is delivered in VoIP clients to meet this goal.

The current procedure in VoIP is to reproduce a digitally sampled voice without modification. While straightforward to implement, this procedure suffer from several drawbacks. First, how do we tell similar voices apart? And second, how do we effectively communicate in a conference environment when multiple voices come ontop of each other? Furthermore, with available VoIP software such as Google Voice and Skype, mono-channel output is simply duplicated. An easy way to test this is to listen to a chat with headphones, switch which ear goes on which side of the headphone, and then listen again. There is no difference because the left and right channels are identical.

Thus far, all varieties of VoIP implementations only reproduce the input voice stream. We propose to manipulate the voice stream digitally via an algorithm. The output is voice localized on a 2-dimensional plane for the listener. To understand and implement the sound spatialization algorithm, we are interested in psychoacoustics, the scientific study of sound perception.

Psychoacoustics dissects the listening experience into areas. We are currently not focusing on the neurological and psychological aspects. They make for intriguing future explorations . What we are interested in is the human response to sound as a mechanical wave. Human response is best characterized by the head-related transfer function(HRTF), a complex response function describing an ear's perception of sound. The ideal scenario for OpenComm would be to calculate the HRTF of a user's two ears, and using this, parameterize our algorithm accordingly. However, measuring HRTF is an extremely impractical and time-consuming procedure currently. Our solution is to approximate HRTF using two factors: interaural time difference and volume difference.

There are some limitations with the spatialization algorithm. First and foremost, humans' natural localization cues are 3-D in nature, taking into account azimuth, zenith and distance. Our 2-D representation is notfaithful in this regard. Second, because sound sources in the physical world are in 3D, frequency responses vary according to the shape and size of the ear. These factors are similarly unaccounted for.

The first sound spatialization factor is interaural time difference, meaning there is a difference in when voice reaches the left and right ear, and thus, a difference in when voice is percieved by our brain. We can emulate this natural process with our algorithm.

As written on our M.Eng Project Report: " When the angle is defined from the center of the face to the right, it becomes positive to the right side of the nose, and negative to the left side of the nose. For example, when the angle completely lies on a half line from the center of the head to the left ear, becomes -90 degree."



The interaural time delay between two ears follows the formula:

interaural time delay =

Negative delay means that the left ear first hears the sound. Here, is in radians.

By calculating the distance to both ears and the speed of sound, our algorithm computes the difference in time for any source on the 2-D plane to reach the left and right ears giving a virtual soundstage

Volume difference is the second factor used to approximate HRTF. Volume, or sound pressure, is inversely proportional to the distance. In plain terms, sound is less loud the farther away it is. The relationship between volume differences and sound localization called the interaural level difference. Our calculation is dependent on the distance of each ear from the sound source.

Consulting our M.Eng Project Report: " The difference in distance between two ears is calculated from the interaural time delay. Since the speed of sound was assumed to be constant, from the time delay the difference of distance can be obtained, and from it the actual distances of each ear from sound source can be calculated."

From here, we set the appropriate volume for each ear individually according to our algorithm

Actual Implementation:

Our Android application combines OpenComm's front-end and back-end changes with an established VoIP framework. For OpenComm, the VoIP framework is a system of XMPP, Jingle and RPT protocols on the Jabber server. The end result allows conference calls to be established. Built on top of this framework are the sound spatialization algorithm and the graphical user interface, featuringprivate side chats.

The server backbone of OpenComm is Jabber. Jabber is the original name of the current XMPP server project, and is now one of the biggest nodes on the open XMPP network. The key service Jabber provides is a secure server client supporting the XMPP protocol, the basis of a conference chatroom.

XMPP - extensible messaging and presence protocol- handles session negotiations and connections management. We utilize Smack, an open source client library for instant messaging and presence that is an extension of XMPP. The Smack library is ported for the Android platform via the *asmack* package. Available from the library are classes for connection management, authentication mechanisms, and the creation of multi-user-chat(MUC) rooms. For example, logging into a server with authentication involves the command: connection.login(userJID,password);.

The Jingle protocol is an extension to XMPP that allows Jabber clients to set up, manage and tear down multimedia, and in our case, VoIP sessions. This involves the sending, receiving and parsing of Jingle packets. Sending and receiving packets is taken care of via asmack.However, the parasing of packets present a more challenging problem. The existing Jingle libraries are unfeasable for OpenComm, so an original library has been developed.. This Jingle library accomplishes three major things: construction of Jingle packet, parsing of packets and determining of events and appropriate transitions based on packets.

(This planned might be more technical in nature than needed?) (Paragraph regarding the brand spankin new Jingle protocol)

The combination of XMPP, Jabber and Jingle protocols allow for session connection and management. Now comes the issue of actually sending and receiving the voice packets. OpenComm recieved its inspiration in the form of Sipdroid, a free SIP/VoIP client for Android. We utilize Sipdroid's implementation, encoding the packets using the G.711 codec at 64 Kbits/second. To transport the encoded packets, OpenComm uses Real-Time Transport Protocol (RTP), which is built on top User Datagram Protocol(UDP). UDP is a defined internet transport protocol (IP), hence Voice over **IP**. However, complicating our implementation is the fact that Android does not support multicast addressing (having multiple recipients for a single source). This problem is solved with multiple streaming and receiving threads.

(NAT traversal?)

OpenComm's two norm-defying changes , sound spatialization and private subchats, are implemented above this robust VoIP framework. On the back-end, we use the AudioTrack class to do the heavy lifting. Two AudioTrack objects handle the left and right sides separately. Interaural time difference is taken into account by having a time difference between the two streams. The existing setStereoVolume() method in AudioTrack succinctly takes care of the volume differences. Combining the volume and time difference allows OpenComm to spatialize a user icon on the interface to a sound source in 2-D space.

The front-end implementation of our application involves Human-Computer Interaction(HCI) Design and GUI development on the Android platform. The keystone of our HCI Design is the creation of private side chats that operate and behave congruently with the main conference chat. Alongside with the chat screens, our application features a minimalistic theme. The GUI itself is designed using vector graphics with an emphasis on consistency and cleanness. As previously discussed, our front-end implementation emphasizes the user experience.

<screenshots of the week5-6 cycle>

Conclusion:

Reiteration of our innovations, focus less on implementation, mention future inquiry/possibilities. Also look at big picture potential.

Other stuff/To-Do: (For after 1st draft)

-Add Bailey's inspiration - which was a music search service and once successful, how to find the actual song in a quick manner - answer: present songs in a spatialized plane.

-Audience awareness: cut and add as needed.

-Talk to group leaders/members regarding implementation and interview for quotes perhaps?

-Broad intelligent appeal: attempt to meet this goal

Referenced works:

M.Eng Project Report - OpenComm Design Documentation & Developer Handover

http://xmpp.org/about-xmpp/technology-overview/jingle/

http://www.jabber.org/faq/

M.Eng Project Report - Implementation of Simple Sound Spatialization Algorithm in Android

http://www.music.miami.edu/programs/mue/Research/mescobar/thesis/web/Psychoacoustics.htm

http://www.meridian-audio.com/ara/multips3.pdf

The Power of Sound, published by Healing Arts Press. (c) 2001 Joshua Leeds.

http://dsplab.eng.fiu.edu/DSP/Research/Research\_007.html

http://www.noisebetweenstations.com/personal/essays/audio\_on\_the\_internet/Spatialization.html

ACOUSTIC LOCALIZATION BY INTERAURAL LEVEL DIFFERENCE (PDF)