Assemble.live
Designing for Schisms in Large Groups in Audio/Video Calls

Dartmouth Computer Science Technical Report TR2017-828

A Thesis

Submitted to the Faculty

in partial fulfillment of the requirements for the

degree of

Bachelor of Arts
in
Computer Science

by

Ben Packer

Department of Computer Science

Dartmouth College

Hanover, New Hampshire

05/31/2017

Advisors:

Tim Tregubov

Seth Frey

Abstract

Although new communication technologies have compressed the space and latency between participants, leading to new forms of computer mediated interaction that scale with the number of participants [Klein, 1999], there still exist no audio/video calling solutions that can accommodate the type of group conversation that takes place in a group of four or more. Groups of this size frequently schism, forming two or more sub-conversations with their own independently operating turn taking systems [Egbert, 1997]. This paper proposes that traditional audio/video calling fails to accommodate schisms because *a)* there is no way to signal intended recipiency, *b)* there exists only one, largely blocking audio channel, and *c)* leaving and joining audio/video calls is too difficult to schism. A solution is developed called *assemble.live* that uses enables users to move throughout a virtual room, and is designed to enable multiple sub-conversations to emerge. From a few recorded sessions of use, it is clear that while this enables multiple conversations to emerge, its affordances for signaling intended recipiency are insufficient.

Acknowledgements

To my advisors, Tim and Seth. To Tim, for providing technical help, discouraging me from large, unnecessary refactors whenever a shiny technology caught my eye and for keeping me on track to complete this thesis on time. To Seth, for showing me examples of how the Internet can be used to study human behavior in ways that make the researcher inside me happy.

To my family and friends, for constant user testing, support, and assurance that my thing with the circles that might be called "assembly" is cool.

Contents

1 Introduction

As information technologies developed and new communication technologies that compressed the space and latency between participants gained popularity, many optimistic theorists of collective decision-making and organization predicted a democratic revolution throughout governments, civil organizations, and businesses, flattening hierarchies and correcting information imbalances [Hartz-Karp and Sullivan, 2014]. Unfortunately, although the exact effect of new communication technologies on decision-making practices is unclear, it is clear that the participatory revolution did not happen as foreseen [Hartz-Karp and Sullivan, 2014].

Although there are many cultural and institutional barriers to the adoption of new methods of decision-making, it is at least partly a problem of design [Towne and Herbsleb, 2012]. Computer platforms used for participatory communication have a unique set of design challenges, including accomodating the flexibility of offline decision-making [for Change, 2013], maintaining coherence, order, and direction as the number of participants increases [Pingree, 2009], creating an equitable, coercion-free discourse environment that facilitates inclusive discussion [Habermas, 1991], and making computer-mediated meetings efficient, fun, and satisfying enough that people choose participatory decision-making instead of adopting hierarchical forms of organization [Graeber, 2013] [Lupia, 2009].

At the center of these hopes was the revolution in communication technologies, which included, in addition to the decrease in latency and cost across distance – the first innovations in **many-to-many** communication in human history [Klein, 1999].

Although innovation in asynchronous text, picture, and recorded video many-to-many communication has occurred with the development of Usenet groups, special purpose association forums, social media, and picture and video sharing websites, that innovation has yet to spread to real-time audio and video calling - computer-mediated-communication's most immersive and life-like form.

Existing solutions for many-to-many audio/video calling are simply derived from applying technical solutions to make applications designed for two-party communication, such as simple telephone calls and video chats, accommodate more participants. While this is sufficient for some use cases, it does not enable multiple concurrent conversations. Thus, unlike asynchronous many-to-many communication text, picture, or recorded video, the quality of the experience degrades as the number of participants increases. The positive relationship between participant count and experience quality is one of the core factors underpinning the revolutionary potential of new communication technologies, so attention and effort deserves to be spent trying to ensure it generalizes to audio/video calling as well.

1.1 Schisms

Conversation analysts have long studied the effects of group size on conversation structure, and determined that a categorical change takes place when the conversation size reaches and increases beyond four. While a two person conversation "exhausts" an encounter and forms a "fully-focused gathering" – each participant is always the speaker or primary addressee - a four or more person encounter has the potential to become a "multifocused gathering" [Goffman, 1963].

The defining characteristic of such a "multifocused gathering" is the simultaneous operation of two independent turn taking systems [Goodwin, 1987], for which four or more are required because two individuals are required to sustain one turn taking system.

1.1.1 Schism Inducing Turns

Of particular important in the creation of multifocused gatherings are **schisms**: events that transform the recipiency structure and turn taking systems being used in an interaction. These either transform a fully-focused gathering into a multifocused gathering, or further rearrange the recipiency structure of an already multifocused gathering [Egbert, 1997]. Research in the conversation analysis paradigm research has shown that schisms are created by **schism inducing turns**, or *SIT*s, in which a first pair part of an adjacency pair (request, question, comment of request-response, question-answer, or comment-acknowledgement) is more specifically directed to solicit recipiency of a subset of the overall participants in the conversation. Like morphemes are the building blocks of words and words the building blocks of sentences, adjacency pairs are the building blocks of conversation, and one pairs is a set of two related units of speech, one performed by each conversation participant.

Egbert [Egbert, 1997] developed a staged approach for SITs:

1. **Before the SIT:** The schism-inducer ascertains the availability and willingness of another's potential recipiency to solicitation through eye contact, bodily orientation, or other nonverbal means.

2. **The SIT:** The schism-inducer launches the SIT by initiating the first pair part.

3. **Schism Acceptance or Rejection:** The intended recipient signals their willingness to join the sub-conversation by completing the second pair part or using bodily orientation and active listening to signal their participation. Alternatively, the intended recipient may ignore the schism induced turn by not completing the adjacency pair, often continuing to pay attention to participants of a different conversation.

4. **Further follow-on:** Individuals other than the one directly targeted may join the branching conversation.

The process involves a high degree of subtle and likely pre-conscious signaling, much of it the native substance of all human discourse.

1.1.2 Schisms in Audio/Video Calls

The inability to develop and sustain multifocused gatherings over audio and video calls is one way that the simple technical generalization of two-party communication to n-party communication fails to achieve effective multiparty telepresence. Although two-party audio/video calls can more closely approximate the interactivity of of face-to-face conversation, the gap between the interactivity of face-to-face communication and audio/video calling grows with the number of participants. This difficulty maintaining engagement comes from the inability for these calls to schism, and is due to a few features of the context.

Problem 1: there is no ability to ascertain the availability of potential SIT recipients in multi-party video calls or conference calls, since there is no eye contact. Multi-party video calls furthermore do not typically employ a WYSISTS model (What You See is What They See), so pointing to other participants or looking at different sections of the screen is not guaranteed to mean attention to the same user. There is, additionally, no bodily orientation that would signal preferred attention.

Problem 2: Although humans are very good at focusing auditory attention on a conversation despite background noise, multiparty video calls have only one audio channel. This causes any simultaneous speaking to exist as overlap instead of a multifocused conversation.

Problem 3: Although it's possible to join a separate video calling sessions for each sub-conversation, the extreme infrequency with which that occurs compared to the ubiquity of schisms face-to-face is evidence that that is not a viable alternative - it is likely too disruptive or not subtle enough. What is more common is for a potential related conversation topic relevant to only a subset of people to be noted of and discussed later.

2 Previous Work

One commonly used, advanced conference calling software is Maestro Conference [mae,], which has a "breakout" function in which the call is split by the event organizers into multiple sub-conversations. The calls can be split according to a series of user attributes, including demographic characteristics, geographical location, or user's answers to polls during the call. However, these mechanisms do not allow users to self-organize and schism in the cooperative process that mirrors multifocused face-to-face gatherings.

On the other hand, virtual reality looms on the horizon as the obvious solution to this problem, and although the social applications of virtual reality have yet to gain widespread adoption, some companies have already started holding meetings over general purpose virtual reality tools such as vTime [vti,] [Sawers, 2016]. These tools allow closer to the full range of nonverbal communication used to signal intended recipiency. However, it's still at least a few years before these devices are widespread enough that it's plausible for a non-technical organization or association of individuals to have a meeting where everyone participates in that manner [Terdiman, 2016].

Finally, concerted attempts to recreate multifocused gatherings on digital platforms can help determine exactly what technical features are required to enable them. This knowledge may help designers of future virtual reality platforms "debug" social interaction if it does not feel up to par.

3 Features

3.1 Core Idea

With this problem – the inability for multifocused gatherings to emerge online – in mind, a new method of online audio/video calling needed to be designed such that

1. **WYSISTS** (What You See Is What They See). This shared visual reference will enable the use of of visual and conversational deictic references (pointing, "above me") that may aid identification of recipients. To accomplish this, I choose for users to interact with the world from a birds eye view.

2. **Audio differentiation**. There must be some mechanism for audio coming from different sources to sound different, and some relationship between that difference and real world state. This is to ease user capacity to isolate different audio sources, required for schisming.

3. **Intuitive influence over audio differentiation.** The user must be able to influence what audio sources have what effects, preferably through interactions that are related to the symbolic representation of that audio source. This allows the translation of user input intending to focus on a particular audio source to reliably effect changes in the audio such that that source becomes easier to differentiate.

To develop these affordances, I conceived of a solution called *assemble.live* which centers around a room in which users can move around, and where the loudness of user A's voice to user B is a function of their virtual avatars proximity in the room.

My hypothesis is that this user controlled movement and distance-based audio attenuation are sufficient to address each of the problems for digital multifocused gathering discussed earlier. Specifically, that user controlled movement and distance-based audio attenuation:

1. can be used to signal intended recipiency,

2. creates the auditory conditions necessary for multiple concurrent channels, each fully intelligible by the participants,

3. is an easy enough method of segregating conversation that users choose to do it.

3.2 Rooms

The core unit of organization in the *room*, which is a plane of connected virtual space. In *assemble.live*, being in the same room as someone means you can potentially see each other's avatars and hear each other's audio if you're close enough. It also means you can join the same *groups* and see each other on the list of users in the room.

The room has boundaries outside of which the user cannot go, but those boundaries expand as the number of users in the room grows. The boundaries are there to prevent a small number of users from getting so far away from each other that they have a hard time getting back.

To ensure easy setup, I use the simple model used by other open source peer-to-peer video conferencing sites such as Chatb [cha,], Appear.in [app,], where any string after the point in

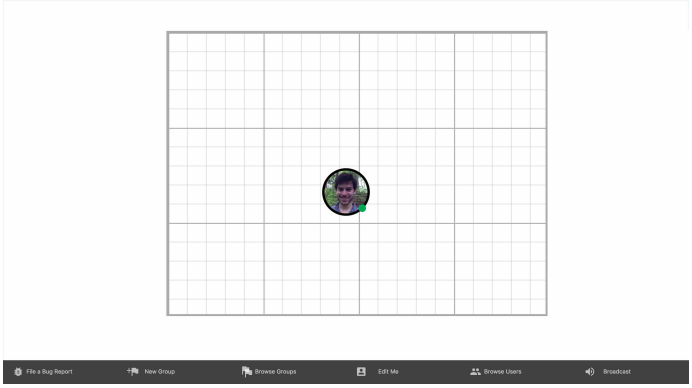

Figure 1: An Assemble Room

the domain's URL is the name of the "room", and any user that visits that room's url joins the room. For example, four users who visit *https://wwww.assemble.live/room/coop-meeting-0526* will be able to jointly move throughout the space, and will not see the users who congregated at *https://www.assemble.live/room/zebra-research-association*. Commercial versions of this approach should include the ability to reserve and password protect rooms.

3.3 Avatar

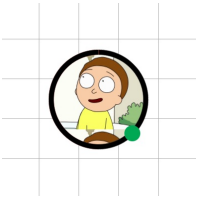

Figure 2: Basic Avatar

The user's avatar is represented as a circle, with an image in the middle. The user can take a picture with their webcam to set this image, or paste an external image url if they do not have a webcam. The image provides a multimedia association with the sound of a users voice that, in the absence of a bodily source, may aid movement decisions of nearby users.

Clicking on the Avatar reveals a set of controls that the user can click to mute themselves, toggle display of their video, and mark themselves as "away", which overlays their avatar with an icon letting other users know that they are away from their computer. When video chatting is enabled, the video shows up inside the circle avatar as well.

3.3.1 Volume

The avatar is also the location of a user's volume indicator, which lets them and other users know a) that their microphone is functioning, by providing that immediate auditory-to-visual feedback, and b) how loudly they are talking, so that other users may determine the source of a particular voice if no users have enabled video sharing.

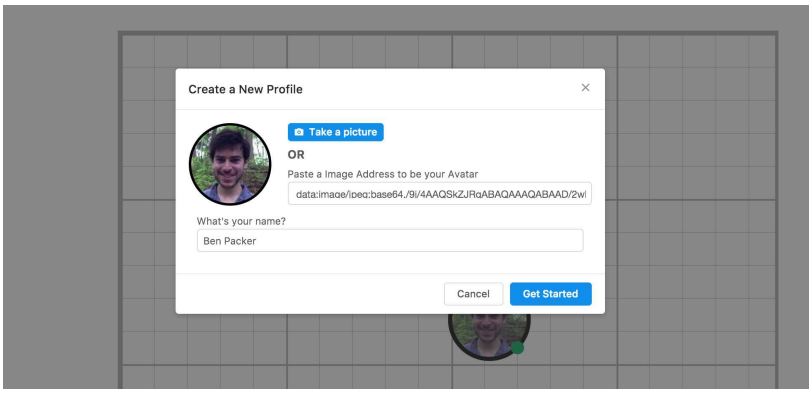

Figure 3: The User's Options to Control Their Own Display

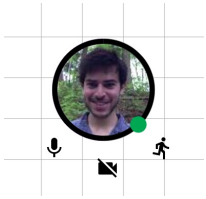

Figure 4: Controls

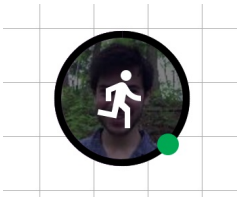

Figure 5: Away User

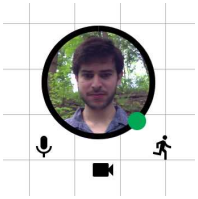

Figure 6: Video On

3.3.2 Connection Status

The avatar is also home to the connection status indicator. When a user looks at their own avatar, the green circle (signifying connected) is present when they are connected to the server. When a user looks at another user's avatar, the connection status takes one of three possible states – connected (green), connecting (blue), and disconnected (grey). User A sees User B as connected if A has established the audio or video or simply data transfer connection desired. User A sees User B as connecting if the signaling process is ongoing, meaning that A has not yet established the desired channel connections. If any error occurs in the signaling process, or the users are not trying to establish a connection to each other, A and B see the status as disconnected, represented by grey.

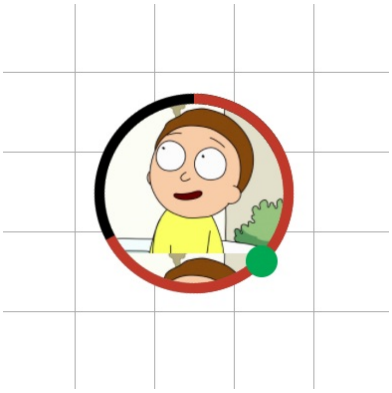

Figure 7: A Currently Speaking User

By concentrating the connection status, volume, and video or identifying image in one place (the user's avatar), I hope to create a concentration of companion visual aides to audio interaction that create a sense of virtual conversational identity.

3.4 Movement

Movement through the room is the central affordance designed to enable schisms. By exploiting the distance controls over volume, users decide what other users they want to be connected to, how loudly they want to hear them, and what groups (to be discussed later) they join.

3.4.1 Methods of Movement

1. **Click and Drag**

   Initially, a user could click and hold the point in the room where they wanted to go, and each fraction of a second that they held the user would travel half of the distance needed to get to that point, leading to a deceleratory feeling. However, initial user testing revealed that user's first intuition was to click and drag themselves, which I believe is intuition carried over from Desktop environments, so that method of movement was implemented. Since its implementation, users have been able to quickly discover how to move throughout the virtual space.

2. **Pan and Double Click**

   Furthermore, a consistent request in user testing was to be able to see other parts of the room without going there so that they could know where to go without leaving the people they were currently talking to. This lead to the development of Google maps style panning throughout the space, where a user can click and drag the background to slide their bird's eye window of view. If they find themselves far from where they started and would like to move into a point in the area they are viewing, they may double click on a space and teleport there.

3. **Direct, to User or to Group Transport**

   Additionally, through the menu bar a user can see all of the users and groups present in the room, and can search through them. If the user clicks on a user or group in the list, they will teleport directly to that other user or group.

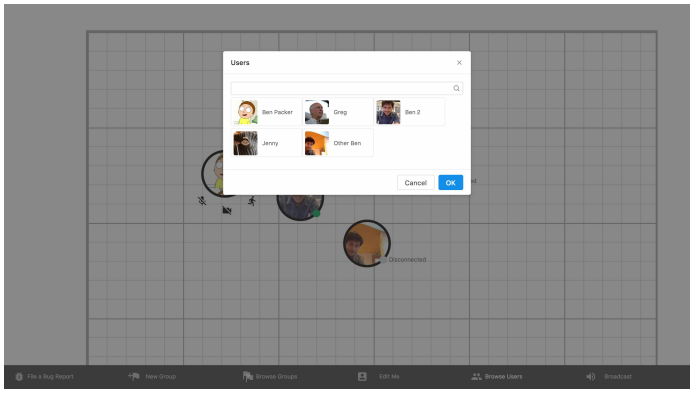

Figure 8: Browsing the Users in the Room

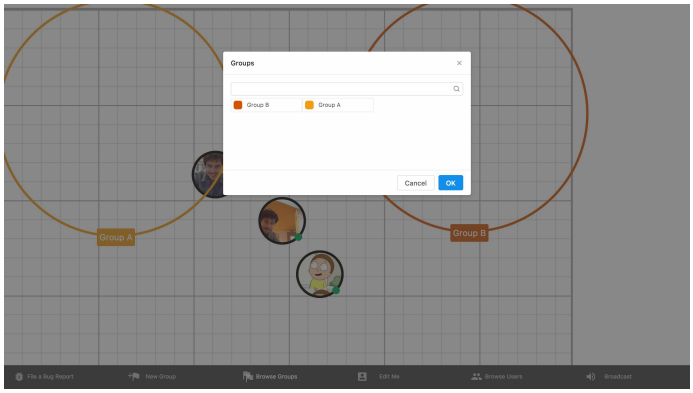

Figure 9: Browsing the Groups in the Room

3.5 Groups and Widgets

The goal of the **groups** feature is to facilitate more organized sub-room interactions, occurring at a size in between the whole room and an individual schism. A group, which just appears as a named circle in the view, is an intentional grouping of people to discuss some related set of issues. Each group can itself be a multifocused gathering, with enough space inside the groups circle to support around three distinct schisms. When a user places their avatar inside of the group's circle, they are a **member**, and they get access to a sidebar of the group that contains its **widgets**. Widgets are self-contained, single purpose utilities that can be used to organize a meeting agenda, take notes, keep track of whose turn it is to speak next, or collaboratively browse the web. Widgets belong to a group and are accessible and editable by any member of the group, although they contain their own internal rules (e.g., a user who is already on the speaking queue cannot add themselves again).

3.6 Broadcast

A consistently expressed feature from user testing was the ability for one individual to speak to everyone in the room at once, which would be used for announcements or instructions about what subgroups should be discussing. Anyone can broadcast themselves, but in order to broadcast oneself there must be no one else broadcasting at the moment.

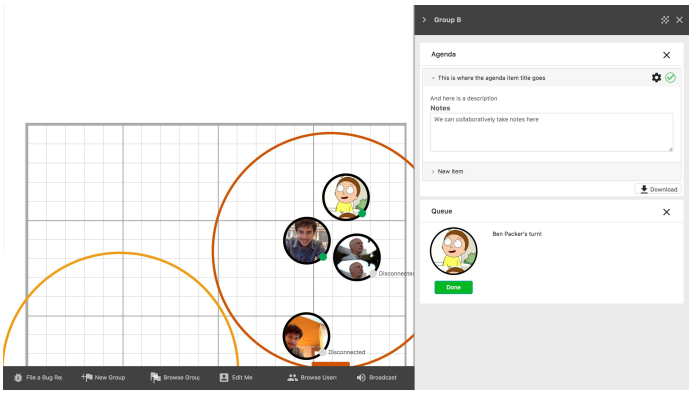

Figure 10: An Example Group with Widgets

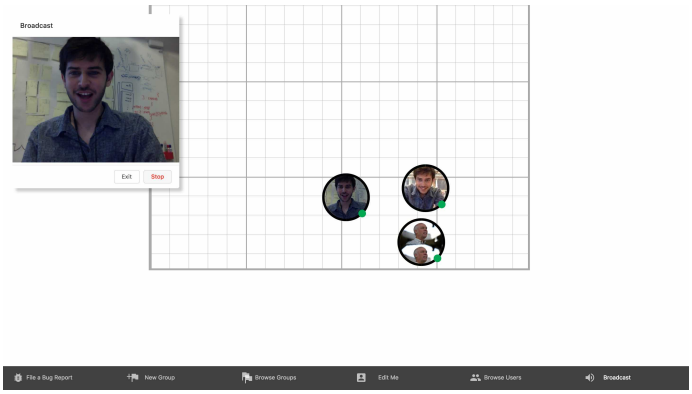

Figure 11: The Broadcaster's View. Other users see a similar popup without the "Exit" and "Start" buttons

4 Implementation

From the desire to have an open source, easily hostable, accessible, multi-party audio and video chat platform come the following design constraints and preferences:

- **Low server costs**. Any organization or association should be able to host the software on third-party machines or their own hardware cheaply.
- **Runs on widely available hardware**. Users should not have to purchase special purpose hardware to run *assemble.live*, but it should run on laptops, desktops, and smartphones already in widespread use.
- **Low installation costs**. Users should not have to download special drivers, applications, or install potentially invasive browser plugins to use *assemble.live*. It should run in their desktop browser or as a downloadable application on their mobile device.

4.1 Code

The full code can be found online on Github at *https://github.com/ben-pr-p/assemble*.

4.2 WebRTC

Thus, *WebRTC*, an emerging web standard for P2P audio and video transfer, was a natural choice. WebRTC is a specification for *a)* how to format signaling data and audio/video metadata, and *b)* a standard, cross-browser Javascript interface for generating and exchanging that metadata. The WebRTC spec defines the capacity to transmit audio, video, and arbitrary data through data channels.

Thus far, it has been implemented in Chrome (and other Chromium based browsers), Firefox, and Edge, [can,] and there exist implementations in a number of other languages for use on other devices, such as iOS and Android [openrtc io, 2017].

Although the transmission of the audio, video, and arbitrary data through WebRTC is peer-to-peer, a third-party is required for giving peer A the signaling data generated by peer B and vice-versa. For this, it is typical to use web sockets, due their low-latency, bi-directional capabilities.

4.3 NodeJS

NodeJS, a port of the Javascript engine that powers Chromium to run in server environments, has grown in popularity. Although the raw, number-crunching performance of Javascript is rather poor because, being dynamically and weakly typed, little type information is known at runtime for optimization, its use of an asynchronous event loop makes it outperform other high-level scripting languages for high I/O operations [Chaniotis et al., 2015].

This performance profile makes it a good choice for the server powering *assemble.live*, which must perform WebRTC signaling and sending updates about the location, volume, and details of other participants. This involves relatively little substantive computation and much more rapid input/output, so NodeJS is a fine choice.

However, to avoid the potential stalling of the main event loop with synchronous computation [Tilkov and Vinoski, 2010], I moved several more computationally intensive operations through separate operating system processes. These include operations such as collision detection, distance computations, and determining optimal relay conditions for broadcasting, which are simple number-crunching computations. NodeJS processes are unable to share memory, so they communicate with the main thread that serves user updates through Redis.

4.4 Redis and Workers

Redis is an in-memory remote key-value store, known for its low latency, smooth handling of simultaneous reads and writes, and scalable publish subscribe system [Da Silva and Tavares, 2015]. The separate worker processes communicate with the main thread through a job queue backed by Redis's pubsub system.

4.5 Inside the Browser

The client side of this project is a rich multimedia web app with real-time interactivity and shared state across users.

4.5.1 React

To manage the relationship between that shared state and the layout a user should see, I use ReactJS, a Javascript framework that makes it easy to map of application state to some output, which in the case of web applications, is the Document Object Model (DOM) that determines what the user sees. ReactDOM, a core companion library, implements an algorithm which, when state is changed, computes the DOM of the next state and makes the minimum number of changes to the rendered HTML document required to achieve the view for the next state. Since modifying the HTML document is a slow, synchronous operation that could stall the client-side event loop, it's important to keep those modifications to a minimum [Rouch, 2015]

4.5.2 Animations

Despite running inside of a browser and thus, being somewhat limited in my capacity to render smooth animations, I was able to achieve relatively performant animations by following best practices that enable such animations to be hardware accelerated, such as avoiding frame by frame computations in Javascript.

4.5.3 P2P State Sharing

To develop standard collaborative capacities to compliment the multimedia meetings, I developed a higher order React component that shares the state of the component it wraps across users through WebRTC data channels. To implement this, I developed and implemented a simple transferable ownership model according to the following rules:

1. Whoever creates the widget is the "owner".
2. Whoever declares themselves the owner becomes the new owner.

3. To update the widget state, send the new state to the owner, who will relay it to all users.

4. If the owner has left or disconnected, whoever wants to update the widget state next must declare themselves the owner first.

5. If a new user joins, the owner is responsible for giving them the current state of the widget.

4.6 Bandwidth Problems and Solutions

In order to have more than two participants in an audio/video call in a peer-to-peer network, participants must open separate WebRTC connections for each peer. The naive implementation involves a full mesh network in which each participant is directly connected to each other participant. This, however, leads a quickly growing $\frac{n(n-1)}{2}$ connections, where n is the number of participants. High numbers of connections only causes bandwidth problems and strain the client's CPU and memory encoding and decoding of those streams. Through casual experimentation, this caused the performance of the main user interaction thread responsible for updating the user location and text data to become very laggy after around 6 simultaneous connections.

A typical conference call or group video chat solves this problem by using a Multipoint Control Unit (MCU) server, which receives all participants audio and video data, mixes them into one stream on that server, and then sends that single stream to all clients. In this model, each client sends only their stream and receives only the composite stream with all audio / video [Ng et al., 2014]. Although this prevents the user's browser from having to do too much work, it poses very high server costs, thus restricting the solution space of group oriented communication to large technical entities with the money and expertise to orchestrate expanding and contracting server resources based on demand. This is not possible in a P2P architecture.

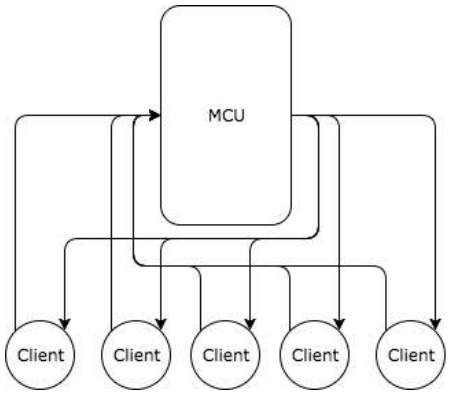

Figure 12: Example MCU Architecture

4.6.1 Disconnecting With Distance

However, the technical question of how to make 10 people all hear and see each other is, I expect, based on a false premise: that it is possible or effective for 10 people to all be engaged in conversation together. That assumptions ignores the reality of multifocused gatherings and schisms in conversations. Thus, the simplest solution to the bandwidth problem is to transparently disconnect users as

they become too far from each other. This allows flexible transitions from fully-focused gathering to multifocused gathering or multifocused gathering to multifocused without bandwidth problems.

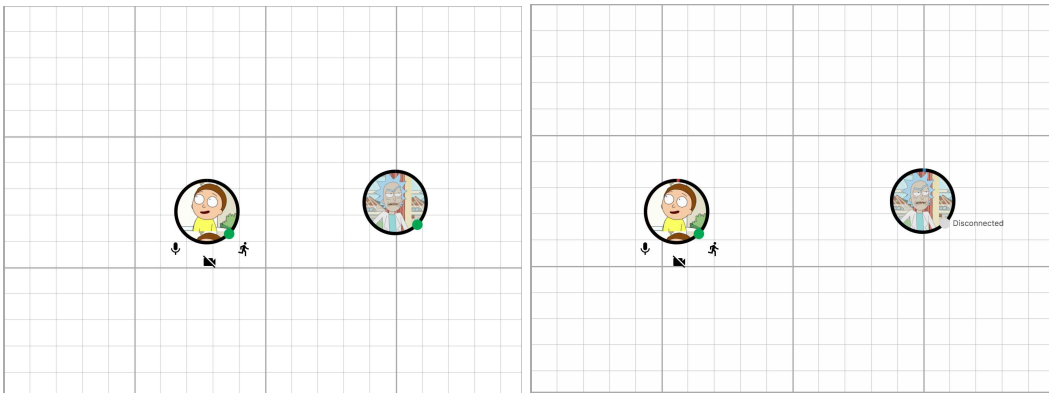

Figure 13: Disconnecting As Users Get Too Far

4.6.2 Broadcast Mode

This solution is only partial – although 10 cannot all talk to each other, one person can and often does talk to 10. So, I created broadcast mode, in which one person is heard by everyone in the room regardless of their location. This is accomplished, without bandwidth difficulties, by employing a "relay" solution in which the broadcaster sends their audio to n = *capacity(broadcaster)* other users, and each user u sends the broadcasters audio to n = *capacity(u)* other users. This is continued until everyone in the room is receiving the audio.

This solution trades bandwidth for an increase in latency, and so in order to minimize that increase in latency, the users with the greater capacity to relay to more peers should exist higher in the relay tree. To achieve this, I place the broadcasting user at the root of the tree. Then, I sort the users by bandwidth, and construct a sorted n-children tree, where the number of children of a particular node depends on their bandwidth.

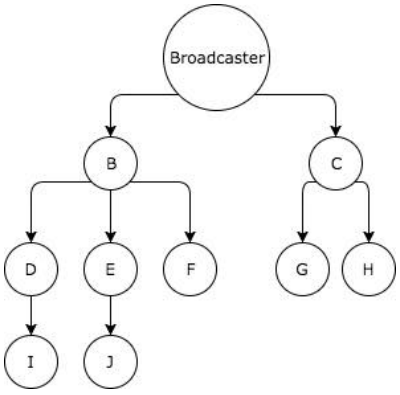

Figure 14: Example Bandwidth Relay Heap

Figure 14 demonstrates a few important things about this solution:

1. The broadcaster is likely not the user with the highest bandwidth. This places an limit on the number of users that can receive the data with a limited number of trips.

2. The entire network may need to be recomputed when a relay node leaves. Since renegotiation and signaling can take a sizable fraction of a second, some words or frames may be dropped.

Although this process of speaking to everyone in the room requires additional user input and intention, the process of clicking a button to decide to speak to everyone is not too distinct from the process of centralizing a multifocused gathering in real life. Although I know of no research on the transition from a multifocused gathering to one centralized on one speaker, personal experience tells me that involves lots of shhing, talking loudly, and maybe hitting a spoon against a glass. Furthermore, research on schisms should be interpreted to suggest that audio dissipation and schisms are the base case, and that fullyfocused large gatherings are the special case. Future iterations can attempt to improve on this solution by implementing a modified P2P-MCU architecture described in [Ng et al., 2014].

4.7 Potential Security Vulnerabilities

4.7.1 Man in the Middle Attacks

The nature of peer-to-peer architectures contains some security threats while eschewing others. Since no audio, video, or text data entered by the participants is stored (or even reaches the server), the security bottleneck is the secure transmission of that data over WebRTC audio, video, and data channels.

Although WebRTC is a relatively secure technology by default, encrypting data in transit and requiring HTTPS in Chrome, there is a possibility for a malevolent client to gain access to the data by staging a man in the middle attack [web, 2015]. If the signaling server is compromised, the attacking client can connect themselves to each client and relay the data, creating the appearance of being connected to each other. The key is to secure the signaling server to ensure that only intended clients receive intended signaling data.

To secure the signaling process, each client is identified by a unique, server-generated id representing their web socket connection. Each time signaling data is sent, a client specifies a socket identifier for that data to be sent to. The server sends the data to the socket corresponding to that id, and attaches to that message a property indicating sever generated unique id it came from. Thus, although any client can send any message to any other client, clients cannot pretend to be another client. This ability to pretend to be another client is required to complete the WebRTC man in the middle attack.

4.7.2 Malevolent Client

However, there are a variety of potential problems that a malevolent client could cause, including transmitting malformed signaling data causing another client to have errors processing it, scripting site interactions in a disruptive manner (move around the room rapidly with annoying noises, etc. Thus, future iterations of the software should investigate MD5 checksums or other software verification solutions to prevent this.

5 Results

5.1 Method

I conducted two user tests, each consisting over several friends having a conversation on the platform lasting at least ten minutes. They were spread out in different areas within one building, so that I could stop by each and check to remedy any connectivity issues, but far enough away that it was at least difficult to hear each other. In both cases, participants were simply instructed to have a conversation and were not explained the rules of the platform.

5.2 Limitations

5.2.1 Connection Difficulties

In each case, one connection pair was not able to be successfully established due to unknown network traversal issues. This let to a few odd instances of "telephone", in which user A had to repeatedly ask B what C said. In addition to being unnatural (I know of no face-to-face analog for only one pair of participants not being to hear each other), this may have caused schisms that otherwise would not have formed.

5.2.2 Novelty

All participants, including those who had briefly been exposed to the platform before, were quite focused on the novelty of the platform and spent their time discovering and discussing its affordances and capabilities. Thus, their interactions with it may have stabilized over time as they collectively developed usage norms. That did not happen in the timespan of the user sessions, however, so a substantial section of the recorded data includes participants discussing what they like about the platform and its interface, how to accomplish various changes to the world, and what they could see themselves using it for.

5.3 Findings

Despite these limitations, there are a couple of findings that can be drawn.

5.3.1 Random Movement

In both tests, users moved about the room, side to side, or to the other side of a speaking user for no apparent reason. It seems that the ability to move meant that users chose to, often out of boredom or simple aimless clicking. This led to a common phenomena of moving conversations, where one user would move to a spot nearby and the other conversation participants would follow.

If it were the case, as was the intention behind the design, that movement would be used and interpreted as discourse cues about who one wanted to talk to, such movements would have been correlated with changes in recipiency structure. Thus, it appears users did not make movement decisions based on who they signaled that they would like to talk to, either because movement was not the appropriate metaphor or because it was simply too fun to click around.

5.3.2 Schisms use Volume Differences but do not Cause Them

In the two sessions, six schisms occurred – two in the four person session and four in the 6 person session. In each case, one speaker initiated an adjacency pair and was only answered by those near them, and shortly after a more distant participant initiated another that overlapped with the response to the first. In each case, the schism was sustained for 3-10 adjacency pairs before recipiency structure reformed, either to a fully-focused gathering or a differently configured multifocused one.

For example, consider the following excerpt from the six person session. Only four people spoke during this short timespan, so only four are represented.

Dre: Wait –
why can I still hear Matthew?
Matthew: Hey –
does Collis have utensils out before late night opens?
Christina: You're probably not far away enough from him.
Matthew: I think they do
right?
Eddie: Yeah yeah.
They leave it out
Matthew: Alright
I'll be right back,
I need something for this rice

In this excerpt, Eddie and Matthew were close to each other and there was a bit of space between them and Christina and Dre, who were close to each other as well. Dre initiated a first pair part, and in the time before Christina's response, Matthew posed another open question. This precipitated a multifocused gathering lasting a few more turns. This type schism was typical, and it is evidence of my expectation for hypothesis parts 2 and 3 was correct. Volume differences were enough to enable multiple concurrent intelligible channels, and movement is not a barrier to forming new conversations like joining new video chats is.

However, each of the first pair parts in this example, and many throughout the sample, were open questions or other first pair parts targeted at the general audience. The fact that schisms using this mechanism were caused by the combinations of relatively random participant locations with open recipient addresses indicates a failure of the movement mechanism to effectively communicate intended recipiency.

In general, first pair parts directed at specific recipients (a typical feature of SITs) occurred rarely, and when it did, was done by directly addressing the participant by name. My expectation is that this failure was to due the depersonalizing effects of the birds eye view and a mistaken design that assumed that any user-initiated gesture could substitute for the eye contact and physical orientation changes that signal intended recipiency in face-to-face interaction.

It is possible, however, that the affordances present in the platform were sufficient for signaling intended recipiency but that conversational norms capable of exploiting those affordances were not given enough time to develop. Even if this is the case, the possibility of other features to

accommodate recipiency should be pursued so that as much practice signaling recipiency from face-to-face interaction is transferable as possible, reducing the time required for new users to effectively communicate on the platform.

6 Conclusion and Future Work

Future iterations of *assemble.live* and other audio/video calling platforms seeking to allow schisms ought to emphasize design affordances that would allow speakers to signal intended recipiency and for recipients to indicate that they are listening *alongside* some version of the user-controlled auditory effects validated here. Although allowing movement and location to control volume effectively enabled a multifocused conversation to emerge, demonstrating its viability as a solution to problems 2 and 3, subtle changes in location were not enough to signal intended recipiency due to a) the capricious nature of movement on the platform and b), the mismatch between the location metaphor and face-to-face recipiency signaling, which is mostly performed by changing physical orientation and eye contact instead of physically moving. Although more precise nonverbal communication of recipiency is an area in which virtual reality may provide substantial benefits, the fact that movement of avatars on a screen that contained little visual resemblance to reality was sufficient to connect disembodied voice to virtual location indicates that full virtual reality solutions may be overkill. Thus, developing methods of signaling recipiency intention without virtual reality remains an area of research capable of generating online audio/video calling platforms that are comfortable and engaging without waiting many years for virtual reality headsets to become widespread.

7 Bibliography

References

[app,] appear.in - one click video call. Accessed: 2017-05-25. Available at https://appear.in.

[can,] Can i use... support tables for html5, css3, etc.

[cha,] Chatb. Accessed: 2017-05-25. Available at https://chatb.org.

[mae,] Live demo signup. Accessed: 2017-05-25. Available at https://maestroconference.com/.

[vti,] vtime: The vr sociable network - out now for gear vr, oculus rift, iphone, google daydream, and google cardboard. Accessed: 2017-05-25. Available at https://vtime.net/.

[web, 2015] (2015). Webrtc and man in the middle attacks.

[Chaniotis et al., 2015] Chaniotis, I. K., Kyriakou, K.-I. D., and Tselikas, N. D. (2015). Is node. js a viable option for building modern web applications? a performance evaluation study. *Computing*, 97(10):1023–1044.

[Da Silva and Tavares, 2015] Da Silva, M. D. and Tavares, H. L. (2015). *Redis Essentials*. Packt Publishing Ltd.

[Egbert, 1997] Egbert, M. M. (1997). Schisming: The collaborative transformation from a single conversation to multiple conversations. *Research on Language and Social Interaction*, 30(1):1–51.

[for Change, 2013] for Change, S. (2013). *A Consensus Handbook: Cooperative decisionmaking for activists, coops and communities*. Seeds for Change Lancaster Cooperative Ltd.

[Goffman, 1963] Goffman, E. (1963). Behavior in public place. *Glencoe: the free press, New York*.

[Goodwin, 1987] Goodwin, C. (1987). Forgetfulness as an interactive resource. *Social psychology quarterly*, pages 115–130.

[Graeber, 2013] Graeber, D. (2013). *The democracy project: A history, a crisis, a movement*. Spiegel & Grau.

[Habermas, 1991] Habermas, J. (1991). *The structural transformation of the public sphere: An inquiry into a category of bourgeois society*. MIT press.

[Hartz-Karp and Sullivan, 2014] Hartz-Karp, J. and Sullivan, B. (2014). The unfulfilled promise of online deliberation. *Journal of Public Deliberation*, 10(1).

[Klein, 1999] Klein, H. K. (1999). Tocqueville in cyberspace: Using the internet for citizen associations. *The Information Society*, 15(4):213–220.

[Lupia, 2009] Lupia, A. (2009). Can online deliberation improve politics? scientific foundations for success.

[Ng et al., 2014] Ng, K.-F., Ching, M.-Y., Liu, Y., Cai, T., Li, L., and Chou, W. (2014). A p2p-mcu approach to multi-party video conference with webrtc. *International Journal of Future Computer and Communication*, 3(5):319.

[openrtc io, 2017] openrtc io (2017). openrtc-io/awesome-webrtc. Accessed: 2017-05-25. Available at https://github.com/openrtc-io/awesome-webrtc.

[Pingree, 2009] Pingree, R. J. (2009). Decision structure: A new approach to three problems in deliberation. *Online deliberations: design, research, and practice*, pages 309–316.

[Rouch, 2015] Rouch, G. (2015). Pure ui.

[Sawers, 2016] Sawers, P. (2016). Group video-calling services have surged this year, here's why.

[Terdiman, 2016] Terdiman, D. (2016). It may take years for virtual reality to go mainstream, says new report.

[Tilkov and Vinoski, 2010] Tilkov, S. and Vinoski, S. (2010). Node. js: Using javascript to build high-performance network programs. *IEEE Internet Computing*, 14(6):80–83.

[Towne and Herbsleb, 2012] Towne, W. B. and Herbsleb, J. D. (2012). Design considerations for online deliberation systems. *Journal of Information Technology & Politics*, 9(1):97–115.