Conversational Chat Circles: Being All Here Without Having to Hear It All

Matthew K. Miller[1,2], John C. Tang[1], Gina Venolia[1], Gerard Wilkinson[1,3], Kori Inkpen[1]

[1]Microsoft Research
Redmond, WA USA
{johntang, ginav, kori}
@microsoft.com

[2]Department of Computer Science, University of Saskatchewan
Saskatoon, SK, Canada
matthew.miller@usask.ca

[3]Open Lab, Newcastle University
Newcastle upon Tyne, UK
g.wilkinson@newcastle.ac.uk

ABSTRACT

Live streaming services are a growing form of social media. Most live streaming platforms allow viewers to communicate with each other and the broadcaster via a text chat. However, interaction in a text chat does not work well with too many users. Existing techniques to make text chat work with a larger number of participants often limit who can participate or how much users can participate. In this paper, we describe a new design for a text chat system that allows more people to participate without overwhelming users with too many messages. Our design strategically limits the number of messages a user sees based on the concept of neighborhoods, and emphasizes important messages through upvoting. We present a study comparing our system to a chat system similar to those found in commercial streaming services. Results of the study indicate that the Conversational Circle system is easier to understand and interact with, while supporting community among viewers and highlighting important content for the streamer.

Author Keywords

Text chat; live streaming.

ACM Classification Keywords

H.5.2 User Interfaces

INTRODUCTION

Live streaming services, such as Periscope and Facebook Live, offer users a way to broadcast video and audio in real time while interacting with viewers. The recent popularity of these apps has been documented in the popular press [11,0] and in studies that have looked at early user practices [16]. Although the competing live streaming services each have unique features, at their core they all offer the ability for broadcasters to stream live video – usually from a smartphone – to any number of viewers, from a few friends to large worldwide audiences. A common feature across popular live streaming services is that viewers can send text messages to other viewers and the streamer as a communication backchannel among the participants. Those text messages are integrated with the display of the accompanying video, and are often used to express reactions, add commentary, and even make requests of the broadcasting streamer. Since all the viewers can see these chat messages, they often use it to communicate with each other by building on others' reactions or answering others' questions. Streamers monitor the text chats as well, but typically speak their responses into the live stream rather than type into the chat.

While text chat as a backchannel accompanying live streams works well for smaller crowds, it runs into problems when audiences grow to hundreds or thousands of viewers, which has happened with recent live streams [11,0]. Although most live streaming services have recognized that large-scale text chats become unmanageable and have attempted to avoid the problem, these existing solutions fall short in some way. Most solutions attempt to limit the amount of content generated by restricting the number of messages users can send or restricting which people can send messages. However, artificially limiting users' ability to send messages reduces their ability to participate in the community around the stream. It also functions only as a stopgap solution that will not scale as audiences continue to grow. Live streaming gives users the chance to affect media as they consume it. To enable all users to experience the magic of live streaming, we must allow them to participate without restrictions.

In this paper, we present a new technique for scaling text chats to a high number of participants without limiting the ability for everyone to participate. We designed a new technique, which we call a Conversational Circle, which shows viewers and streamers a manageable amount of content without limiting their ability to participate. Our design goal was to share text messages with enough people that the audience would feel a sense of common ground, and to share the most salient messages with all the viewers and the streamer. After reviewing related work, we describe

CHI 2017, May 06 - 11, 2017, Denver, CO, USA

ACM 978-1-4503-4655-9/17/05…$15.00
DOI: http://dx.doi.org/10.1145/3025453.3025621

our iterative prototyping process, our implemented working prototype, and a study that compares our prototype with conventional text chat mechanisms.

RELATED RESEARCH

Our primary interest in large-scale text interaction is rooted in text chat as a communication backchannel in live streaming services. However, text chat has been integrated into a variety of forms of computer-mediated communication tools, and has been the focus of several studies. Weisz et al. [19] showed that adding a text backchannel when watching video together led to rating the video content higher and liking the other participants better. Since communicating online, especially through text, misses much of the richness of human interaction, studies have explored how to keep users engaged in online communities [1]. Studies of lurking (reading messages without actually posting a contribution) found that among the reasons users lurk is because they feel like they have nothing to offer, relative to others' messages, and that there were too many messages [13]. Managing a large volume of text messages is also a problem various text chat prototypes [18] have explored.

Viewership of highly popular live streams has demonstrated that the text backchannel cannot viably support group interaction at very large scale. A recent live stream of a protester climbing the Trump Tower attracted over 225,000 concurrent viewers [0]. A study of the popular game live streaming service, Twitch, found that users mostly watched streams with over 1000 viewers, and half of the streams they watched exceeded 5000 viewers [7]. They found that with such a large-scale audience, the chat was a source of breakdown in interaction, where viewers could no longer follow the conversation. The text chat becomes more of an ambient roar of the crowd without meaningful one-on-one interaction. This observation is consistent with the modeling done by Nascimento et al. [8] that found that after the audience reaches 1000 people, further increases to the audience size do not increase the volume of chat messages as much. In other words, users joining a high scale text chat are less able or willing to contribute. The Periscope app historically limits text chat participants to about 100 people (others can view and give hearts, but cannot contribute text messages). Facebook Live allows everyone to comment but does not show viewers all incoming messages.

Twitch has several mechanisms to help manage large audiences. Streamers can designate some users as moderators who can remove inappropriate or unwanted messages. Moderators often also help with greeting other viewers and answering viewer questions, easing the burden of interaction on the streamer. Twitch also enables streamers to limit viewers to sending only a certain number of messages per minute. Streamers can also restrict the ability to comment on the video to only paying subscribers to their channel. However, these mechanisms somewhat artificially limit interaction, and rely on the streamer to select moderators and set up interaction limits. Such configuration can be challenging to do in the moment, such as when a live streamer comes upon some breaking news to share.

When large audiences overwhelmed the text backchannel in game live streaming, Hamilton et al. [7] noted that massive streams were still compelling to watch, even though the text chat could not afford meaningfully interacting with others. However, a study of using of live streaming in crisis response [6] found that text requests made by the viewers often were not acted upon, which could be more consequential.

Keeping up with the text chat in large live streams is especially a challenge for the streamer, who is also navigating their actions within the live event. A study of a research prototype that included a text backchannel to the streamer [3] documented the effort of the streamer who is both participating in an event and sharing it with viewers. Reeves et al. [14] also noted the challenge of streaming video, navigating the environment, and interacting with a remote online audience through text in a mixed reality game. In a popular live stream with many text messages, it is difficult for any user to keep up with the chat. For the streamer, who must simultaneously keep the camera appropriately aimed, monitoring the chat is even harder. Based on this prior work and our own experiences in watching live streams that have attracted large viewing audiences, we believe there is a need to design a text communication backchannel to support the needs of both viewers and the streamer in large-scale live streams. Beyond live streaming, this concept could be useful in any large-scale text conversation that unfolds in real time.

One technique that has emerged as a useful way of filtering through large volumes of text contributions is an upvoting mechanism: enabling the community to vote up or down on text messages to highlight the most salient messages and filter out ones that do not need more attention. Upvoting has been studied in asynchronous social sites, such as Q&A sites [17] and Yik Yak [15], where it was found to be a successful way of community filtering to highlight important contributions. Upvoting is more challenging in real time events, where there may be a limited time to collect user input and act on it before the opportunity has passed. Tools like Sli.do (https://www.sli.do/home), are used to collect questions from an audience of a live event to select which ones to ask the presenter. Faieta et. al.'s group discussion system [4] uses upvoting in a structured proposal reviewing system to find consensus among large groups. However, neither are designed to support general text chat discussion in addition to their primary function. This work led us to explore how we could integrate upvoting in a more flexible and unintrusive way to help filter large volumes of text messages associated with large-scale live streams.

PROTOTYPING

To explore this design space, we tried some quick prototypes to investigate the role of upvoting and subgrouping.

Upvoting

Our initial approach to managing text chat volume at high scale was to highlight messages with greater saliency to the viewers and streamer. We chose to do this by allowing users to upvote chat messages by clicking on them. In this way, the crowd could indicate which messages were important to them. To test this concept, we used a pre-existing tool, Sli.do, which has similar affordances. Sli.do is designed for attendees of conferences or other speaking events and allows them to submit questions and upvote questions they want answered. Speakers or moderators at events can see the most popular questions sorted by number of upvotes.

Design

We used Sli.do as a pseudo text chat to test the concept of upvoting messages during a live stream. To accomplish this, we created a webpage with an embedded Facebook Live stream and an embedded Sli.do window. Users could submit any message they wanted into the Sli.do system, as if it were a text chat system accompanying the video live stream, and to click the upvote button on messages which they wanted the streamer to see. The interface is shown in Figure 1.

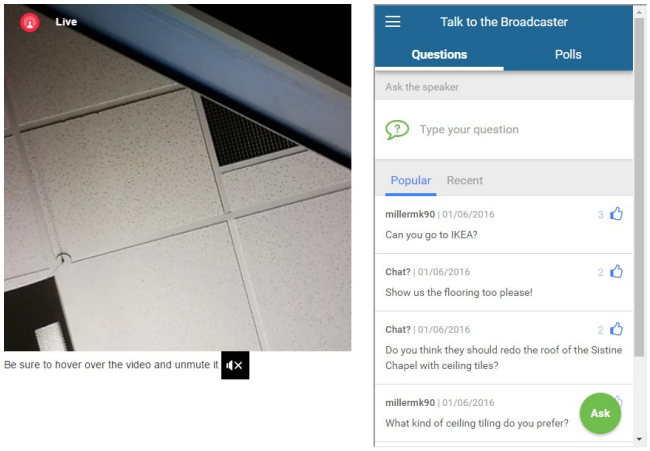

Figure 1 The viewer interface of the initial upvoting prototype with the stream on the left and the Sli.do window on the right

The streamer carried two phones, one running the standard Facebook Android app used to broadcast the live stream, and the other opened to the Sli.do administrator page to show a list of messages sorted by number of upvotes.

Testing

To test this interface, we sent a streamer to a local street carnival event in the Seattle area. The streamer shared a power tool racing event, where participants constructed and raced small cars powered by power tools. We then recruited 151 viewers using Amazon's Mechanical Turk platform to watch the live stream and participate in the text chat. After viewing the stream for 20 minutes, participants completed a questionnaire about their experience.

Results

Unfortunately, this testing suffered from a few technical issues, which did not allow for a smooth viewer experience. Due to poor cellular signal at the carnival location the stream was very laggy and frequently cut out entirely. Furthermore, the load on the Sli.do system was too high. Although Sli.do is normally able to accommodate many more than 150 users, our system created an unusual load for the system because it was used as a text chat, resulting in much more interaction than Sli.do's intended use case of submitting questions to a speaker. Because of this increased load, the Sli.do window was often unresponsive and laggy.

Despite technical issues, feedback from participants still allowed us to learn from this prototype. We asked users to suggest one thing they would improve from the experience. Beyond the technical glitches in the video stream cited by most users (117), 18 responded that the chat scrolled by too quickly. The other 15 users gave a variety of other suggestions. The most consistent feedback apart from the technical issues was that the volume of incoming messages – generated by 151 concurrent viewers – was high. Too many messages are not only difficult to read, but especially difficult to interact with (e.g., upvote) because the user must click on a moving button attached to the message. This issue was exacerbated by the fact that messages on screen jumped downward to make space for new messages at the top. These movements, which featured no animation, were impossible to anticipate.

Most systems that use upvoting allow users to browse content at their own pace, rather than upvote as it scrolls by. Therefore, we were interested to know if upvoting would be useful in discerning which messages were more important than others. We asked if users felt that the upvoting mechanism was helpful in highlighting important content, on a 5 point Likert scale from strongly disagree (1) to strongly agree (5). The mean response was 3.21, slightly above neutral. We also asked what kinds of comments they upvoted. Most indicated that they upvoted messages about the technical issues. However, 22 users indicated that they upvoted humorous messages. We coded the chat log by tagging each message with at least one message type. We found that 30% (339/1143) of all messages got at least one upvote, while 71% (10/14) of messages we tagged as humorous got upvoted, and 27% (126/470) of messages tagged as technical issue related got upvoted. We did not do any formal statistical analysis of the prototype data because it was highly confounded by the video stream issues, but did feel that the results warranted further investigation into real-time upvoting of comments.

Grouped Upvoting

Although there was some initial promise for the idea of upvoting, the first prototype made it clear upvoting alone

would not enable text chat to scale to hundreds of users. It was necessary to lower the overall volume of messages so that users could read and interact with them. To do this, we decided to partition users into separate groups, each group having its own text chat system.

Design

To test the concept of grouping viewers, we used the same interface as the previous prototype, but rather than embedding the same Sli.do instance on every user's page, we assigned one of several separate Sli.do instances to each incoming user, thus partitioning them into groups. By limiting the number of users chatting in each instance, we avoided the scalability issues of the first prototype.

Testing the first prototype showed that it was awkward for the streamer to carry around two separate devices. We did not want to exacerbate that problem with this prototype, since there were multiple separate chat instances to monitor. To solve this problem, we used human moderators who monitored the text chats and copied the most upvoted messages from Sli.do into the comments section of the Facebook live stream. The viewers saw Facebook's embedded live video player, with comments disabled to prompt them to use the accompanying Sli.do instance as a text chat with upvoting. However, the streamer saw only the messages copied by the human operators into the Facebook Live app they used for streaming.

Testing

To test this interface, we sent a streamer to broadcast a virtual tour of exhibits at the Museum of Flight, a local museum of aviation artifacts. We recruited 142 viewers using Amazon's Mechanical Turk platform and evenly divided them into five subgroups. They all watched the same live stream but each subgroup had its own Sli.do text chat instance. Two human moderators monitored the upvoted messages in each Sli.do instance to forward to the streamer on Facebook Live. After viewing the stream for 20 minutes, participants completed a questionnaire about their experience.

Results

Based on this deployment we found that users were better able to read and upvote messages due to the grouping feature, which significantly reduced the volume of incoming messages within each Sli.do instance. In response to the same question about what one thing from the experience they would like improved, none of the participants said the volume of incoming messages was too high. The vast majority suggested that the streamer keep the camera steadier or show different things in the museum. While we did not statistically compare the open-ended feedback from the two studies, we believe that the lower number of messages made reading and interacting easier.

We also saw evidence that the upvoting system was working as intended. We again asked if users felt that the upvoting mechanism was helpful in highlighting important content (again on a 5-point Likert scale from strongly disagree to strongly agree). For this prototype, the mean response was 4.03, indicating user agreement that upvoting was helpful in highlighting content. Again, we looked through the chat data for objective evidence that upvoting was helpful. Since there were far fewer messages that were humorous or about technical issues, the coding of the messages changed such that a direct comparison with the previous prototype was not possible. Instead, we tagged all messages in the chat log that asked the streamer to do or show something as requests. Requests are a common type of message that are only effective when seen by the streamer. We found that 69% (83/120) of all requests got at least one upvote while 48% (528/1107) of all other messages got at least one upvote. A Pearson Chi-Square test showed this difference is significant ($X^2(1, N = 1227) = 19.963$, $p \approx .000$). Thus, upvoting was meaningful in highlighting requests, one of the most common types of message intended for the streamer.

However, there were some issues remaining with the design. The most significant issue is that statically sized groups are not practical for real-world use. In order to accommodate users joining and leaving a stream in a real-world application, group sizes would have to adjust to allow evenly splitting viewers into groups. However, changing group sizes would involve moving users from one group to another, which may be confusing to users (as they would suddenly be chatting with different people). Alternately, one or more groups could simply have a different size than the rest, but this means that smaller groups may not have a healthy level of interaction, and larger groups may have a harder time upvoting messages.

Another issue is that viewers did not share enough context to understand the streamer's responses to text comments. For example, if a viewer in group A asked the streamer a question which the streamer verbally responded to in the stream, viewers in groups other than A had no way of knowing the context for the streamer's response, as they would have never seen the message posing the question.

CONVERSATIONAL CIRCLES

Prototyping showed that the ideas of grouping viewers and allowing messages to be upvoted complement each other. Grouping viewers creates a lower, more manageable number of messages that allows them to be read and acted upon by upvoting. But grouping results in several disjoint conversations, making it harder for the streamer to manage and understand the ongoing conversation. Voting requires a manageable number of messages, as users need to be able to interact with messages as they appear on screen. It also provides a way to filter through a large number of messages, by selecting messages with a higher number of upvotes.

Flexible Grouping with Neighborhoods

A Conversational Circle is a concept designed to overcome the issues of rigid grouping of participants, while still trying

to foster community by allowing users to interact with a relatively stable group of peers. In a Conversational Circle, not all the users see each message; rather, each message is seen by a subset of people who have a chance to upvote it. As more people upvote the message, it is shown to more people. If it crosses a threshold, it is shown to everyone including the streamer. The algorithm used to implement this concept places all the participants in a hypothetical circle and calculates distances to determine who sees what message.

Each user is assigned a fixed position, p on a hypothetical circle, represented by a value from [0,1). This means that for a pair of users, U_1 and U_2, with positions p_1 and p_2, a distance can be calculated between them as the shorter of the two arcs connecting them:

$$\boldsymbol{dist}(\boldsymbol{p_1}, \boldsymbol{p_2}) = \boldsymbol{min}(|\boldsymbol{p_1} - \boldsymbol{p_2}|, \boldsymbol{1} - |\boldsymbol{p_1} - \boldsymbol{p_2}|)$$

Rather than fixed groups of users, each user on the Conversational Circle has a neighborhood, N, defined by a distance threshold. For a user U_x, all other users whose distances to the user are less than the threshold, T_x are part of the user's neighborhood:

$$\boldsymbol{N_x} = \{\boldsymbol{U_y} | \boldsymbol{dist}(\boldsymbol{p_x}, \boldsymbol{p_y}) < \boldsymbol{T_x}\}$$

Every user sees all messages from users in their neighborhood. The neighborhood relationship is asymmetric; that is, user U_x belonging to the neighborhood of U_y does not guarantee that user U_y belongs to the neighborhood of U_x. In practice, it will be symmetric for the vast majority of pairs of users. The neighborhood of one user is depicted in Figure 2.

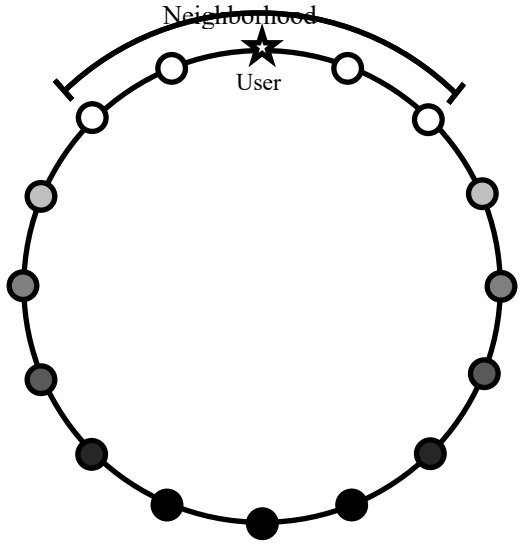

Figure 2 An illustration of a Conversational Circle. The user (star) sees all messages from other users in their neighborhood (unfilled circles), some popular messages from users closer to their neighborhood (lighter circles) and only the most popular messages from users further away (darker circles).

Shared Context with Upvoting

In the previous prototype, we used human moderators to pass the most popular messages on to the streamer. However, the staffing required by human moderation is not realistic in many cases. A Conversational Circle uses upvoting to identify which messages are most popular, in effect spreading the moderation work across all the participants in the chat. Each message that is sent has an initial audience, which is all users whose neighborhoods include the sender of the message. Viewers can upvote a message at any time by tapping the message. As messages float up a user's screen, messages that get no or few upvotes fade out of view, leaving visual gaps in the flow of messages on screen. The system fills these gaps with popular messages that the user would not otherwise have seen. These messages come from outside the user's current neighborhood, and are close in age to the message which was removed from the gap. This means that more popular messages are shown to a larger audience.

Because minimizing perceptual and cognitive burden on the streamer is important, we chose to only show them up to three messages at any time. These are the messages with the most upvotes in the system, which we call the top three messages. All viewers also see the same top three messages, which sit stationary at the top of the chat window. That way, viewers share common ground with each other and the streamer. While messages shown in the rest of the chat window may differ for each user, the top messages are the same for all viewers and the streamer. This shared context addresses the problem of not knowing what message a streamer is referring to when viewers do not share a view with the streamer.

Managing Content Volume with Dynamic Adjustments

Feedback from our initial prototypes indicated that users did not want to be overwhelmed by too many messages so that reading and interacting with them was difficult. To avoid this problem, the Conversational Circle makes dynamic adjustments that control the number of messages a user sees based on moving averages of activity in the system. The system bases these adjustments on target values for how much content users should see.

- **Neighborhood Threshold**: The neighborhood threshold increases if there are too few messages sent by others in a user's neighborhood and decreases if there are too many.
- **Message Removal Aggressiveness**: The system becomes more or less aggressive in removing messages from users' screens, attempting to remove a target fraction of all messages before they float off the top of the screen.
- **Gap Filling Aggressiveness**: The system becomes more or less aggressive in adding messages into visual gaps on the screen, attempting to add messages at approximately a target rate.

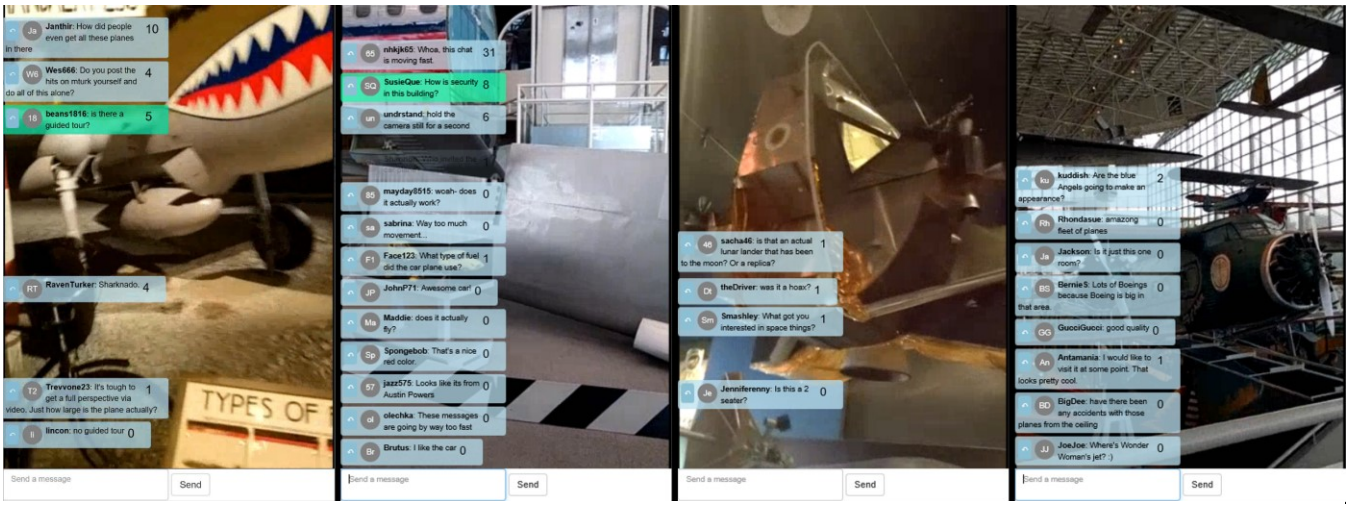

Figure 3 (l-r) The chat system as it appeared in Condition A (throttling on, top three on), Condition B (throttling off, top three on, Condition C (throttling on, top three off), and Condition D (throttling off, top three off)

All dynamic adjustments are based on rolling averages, which means that some natural variance in message volume will occur. For example, a brief burst in incoming messages from a user's neighborhood will result in an increased number of incoming messages. However, if the increase is sustained, the user's neighborhood will shrink to maintain a desirable volume of messages.

SYSTEM DEVELOPMENT

To evaluate the Conversational Circles concept, we implemented a web based live stream viewing platform. For the live video stream we used a Wowza Media server (https://www.wowza.com). We embedded the video stream on a web page using video.js (http://www.videojs.com). We built a text chat that implements the Conversational Circles concept as well as a standard text chat mode for comparison. The text chat backend was a Firebase Database (https://firebase.google.com), which allowed for easy real-time synchronization of data among clients, and required no server-side programming. We built the text chat client using plain HTML and JavaScript.

We also built an Android streaming application to enable us to test the system. The streaming application was similar to commercial applications, featuring a full-screen video preview with incoming chat messages from users overlaid at the bottom. It has two modes, one that shows all incoming user messages, and one that shows only the three most popular messages in the system. Both allow the streamer to "accept" a message, which highlights it in a green color on the streaming and viewing clients. The streamer could not send text messages; they communicated verbally.

To investigate a more fine-grained understanding of how the system compared with traditional text chat systems we implemented settings to control the behavior of the system:

- **Throttling**: When on, users see only messages from their neighborhood, plus popular ones from elsewhere on the Conversational Circle. When off, users see all messages.
- **Top Three**: When on, users see a section at the top of their screens showing the three messages with the most upvotes currently in the system, and the streamer sees only these three messages. When off, users do not see the top three section, and the streamer sees all incoming messages.

USER STUDY

We conducted a study of the Conversational Circles system to understand whether it made it easier for users to process and interact with the text chat, and whether it enabled the most salient messages to be shown to the streamer. We experimented with the two system settings (throttling and top three) as factors, each having two levels (on and off). This resulted in a 2x2 design with the following four conditions:

- **Condition A**: throttling on, top three on, the full realization of the Conversational Circles concept.
- **Condition B**: throttling off, top three on.
- **Condition C**: throttling on, top three off.
- **Condition D**: throttling off, top three off, similar to how current services such as Periscope operate.

Figure 3 shows the viewing experience for each of the conditions, and the video figure provides a sample of each condition.

We chose a between-subjects design because we asked the streamer to show similar content for each condition, meaning that a within-subjects design would be susceptible to ordering effects. Given that we needed large numbers of viewers in each condition, it would have been challenging to counter the ordering effects by running trials in different condition sequences. We recruited about 180 viewers via Mechanical Turk for each condition, who were not allowed to participate in more than one condition. We returned to the same local aviation museum with the same streamer.

Measures

We asked users to answer a series of questions about their experience after the live streaming experience. Most were Likert questions, but a few were asked as a balance between two competing factors. For the balance questions, the *midpoint* of the scale is a good balance. We asked participants questions in four areas:

- **Volume of incoming messages**: ability to read all messages, balance between too many and too few messages, balance between messages and video.
- **Ease of use**: ability to like messages, reply to messages, understand which messages the streamer was responding to
- **Effectiveness of upvoting**: of the text messages which got many upvotes, how many were important and how many were unimportant
- **Connectedness among viewers**: ability to understand what other viewers liked, sense of community

To test users' perception of throttling we asked them to estimate how many people were viewing and communicating with them. We also asked participants for open ended feedback, and collected logs of all system interaction.

Participants

We used Amazon's Mechanical Turk to recruit 200 participants for each condition to view the live stream. Mechanical Turk has previously been used for gathering feedback on prototypes [9]. While Mechanical Turk does allow less control than a lab study, it enabled us to recruit hundreds of participants and test at a scale which would not have been possible in a lab study. We instructed the users as follows: "You will be viewing a stream for at least 15 minutes and sharing your requests and comments along with other viewers also watching the live stream. You can communicate with each other via a text chat tool. Feel free to ask questions, express your reactions, request what you'd like to see, or interact with the other viewers. After 15 minutes, you can stop watching the live stream and complete the HIT by filling out a survey about your experience." The questionnaire was short, requiring no more than 10 minutes to fill out, resulting in a total task time of about 25 minutes. We paid the users $5 for participating in the experiment, which is consistent with a fair minimum wage of $10/hour.

Since some participants who signed up for the experiment ended up not doing the task after all, and we could not replace them in time to view the stream concurrently with everyone else, we ended up with slightly fewer participants in each condition. The actual number of participants for each condition was 185 for Condition A, 177 for Condition B, 179 for Condition C, and 176 for Condition D.

RESULTS

Questionnaire Data

Figure 4 shows the mean values for the survey questions across conditions. We performed a two-factor multivariate ANOVA on the Likert scale survey questions, with *throttling* and *top three* as factors. Each factor had two levels (on/off). Although the group sizes differ slightly, ANOVAs are robust to slightly uneven group sizes for main effects [5].

We found several effects of throttling. Throttling significantly increased the ability to read messages ($F_{1,1}$=198.904, p≈.000, η_p^2=.239), appropriateness of the number of messages ($F_{1,1}$=381.916, p≈.000, η_p^2=.376), and balance between messages and video ($F_{1,1}$=11.834, p=.001, η_p^2=.018). It also increased ease of use as measured by increased ability to understand what messages the streamer was responding to ($F_{1,1}$=27.865, p≈.000, η_p^2=.042), ease of upvoting ($F_{1,1}$=85.908, p≈.000, η_p^2=.119), ability to reply to other viewers ($F_{1,1}$=16.204, p≈.000, η_p^2=.025), and sense of what was popular with other viewers ($F_{1,1}$=5.721, p=.017, η_p^2=.009).

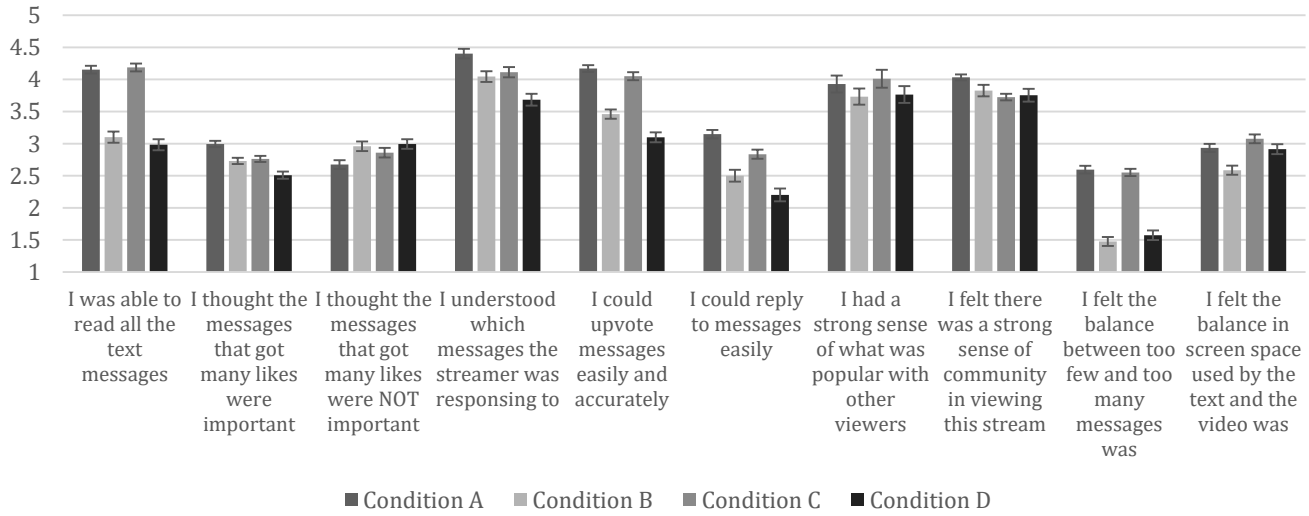

Figure 4 Means (±SE) for questionnaire questions. All questions used a Likert scale from Strongly Disagree (1) to Strongly Agree (5), except the last two questions regarding balance, where the midpoint of the scale (3) represents a good balance

Mean estimates for number of people users were communicating with were 46 in Condition A, 100 in Condition B, 42 in Condition C, and 80 in Condition D. The estimates were approximately twice as high on average when throttling was turned off. These differences are not statistically significant, as variance in the guesses is high. Interestingly, users underestimated the actual number of viewers (the number of simultaneous viewers was between 150 and 175 for the majority of each stream).

We also found correlations between showing the top three and efficacy of the upvoting system. Specifically, adding the top three section increased agreement that messages which got more likes were more important ($F_{1,1}$=9.833, p=.002, η_p^2=.015), ability to understand what messages the streamer was responding to ($F_{1,1}$=23.446, p≈.000, η_p^2=.015), and feelings of community ($F_{1,1}$=3.963, p=.031, η_p^2=.007).

Throttling and top three each contributed to a more usable experience that also had a sense of community. All other effects of condition on the questionnaire responses were not significant at p=.05. No interaction effects were significant at p=.05.

Chat Data

We performed several analyses on the logged chat data to get an objective sense of user behavior. Prior to analysis of the logged chat messages, we filtered the dataset to only messages from the most active 10 minutes of each streaming session, as defined by number of simultaneous viewers. We did this to focus on the most challenging level of simultaneous interaction in each condition. The start and end of each session do not represent a typical scenario for a chat system because many users are greeting each other or saying goodbye during these portions.

We wanted to explore if upvoting helped to highlight important content. The concept of important messages is difficult, if not impossible, to define, especially because the importance of messages may vary widely with the content of the stream. However, for our study, one type of message that is important is requests, such as "Show us the space shuttle" or "Hold the camera closer to the display". These requests are important because they are how viewers influence the stream. Unlike some other messages, they need the streamer – not other viewers – to see them to accomplish their goal. Because requests are a good example of what we believe is important content, we categorized all messages from the chat logs as requests or other content. Because throttling affects the number of people that see a message, comparing amounts of upvotes across conditions is not a fair measure. Therefore, we classified all messages as having received no upvotes, or having received at least one upvote. For each condition, we performed a Chi-Square test to see if the fraction of request type messages with at least one upvote was significantly different from the fraction of other types of messages with at least one upvote. Table 1 shows the results of these tests. In conditions A and B (which both show the top three), a significantly higher fraction of requests got upvoted than other messages. In conditions C and D, the difference was not statistically significant. We conclude that the top three section encourages some kinds of content to be upvoted more than other kinds, meaning that upvoting can be a helpful mechanism for highlighting what is important.

We also analyzed user participation across conditions. One measure of user participation in social media is lurking. We counted the number of lurkers, defined as users who sent zero text messages, in each condition. We used Pearson Chi-Square test to check for significant differences. When the top three section was off, 20.6% of users were lurkers, versus 20.3% when it was on. This difference was not statistically significant (X^2(1, N = 836) = .016, p = .901). When throttling was off 25.6% of users were lurkers, compared to only 14.8% when throttling was on. This difference is statistically significant (X^2(1, N = 836) = 14.801, p ≈ .000). The data reflect that throttling increased participation by decreasing the number of lurkers.

DISCUSSION

Increasing Usability

As expected, throttling the number of messages that users see increased the usability of the system in a number of ways. First, users were able to understand the chat better when throttling was on. Participants reported that they were able to read a higher portion of the messages appearing on screen, and have a better sense of what was popular with other viewers in the throttling conditions. Participant B156's feedback from the unthrottled condition explained the problem with a traditional text chat at high scale: "there should be a better way to communicate with others. The messages, at certain points, were moving way to [sic] fast to be able to read thoroughly and it makes it hard to reply or see replies from others."

Second, throttling allowed users to more easily interact with the content on screen. When throttling was turned off,

Table 1 Portion of requests receiving upvotes compared to other messages

Condition	Fraction of requests with at least one upvote	Fraction of other messages with at least one upvote	Pearson Chi-Square Comparison
A	35% (18/51)	23% (168/740)	X^2(1, N = 791) = 4.206, p = .040
B	74% (29/39)	41% (299/724)	X^2(1, N = 763) = 16.504, p ≈ .000
C	32.% (11/34)	28% (169/614)	X^2(1, N = 648) = 0.374, p = .541
D	39% (26/66)	42% (330/787)	X^2(1, N = 853) = 0.161, p = .688

participant D132 noted that, "it was very, very difficult to read most of the messages, let alone click on a specific one." When throttling was turned on, users reported increased ease of reading and upvoting. In the throttled condition, messages move up the screen at a steady pace. This is possible because the rate of incoming messages is controlled. When throttling is off, messages cannot move at a steady pace because variance in the rate of incoming messages means they may need to move out of the way quickly to make room for incoming messages. Because throttling allows messages to move in a predictable way on the screen, it is easier for users to click on them to upvote. The increased ability to interact with content when throttling is on is likely a result of the fact that there is less content and that it moves more predictably on screen. Throttling also reduced the portion of lurkers. This may be because too many messages intimidate some would-be contributors, or because they feel that when there is a high volume of messages their message will not matter anyway.

Finally, users indicated directly that throttling results in a more manageable number of messages. When throttling was on users indicated that the balance between text messages and video was better, and that the number of messages shown was more appropriate. Participant A80 said in the feedback that "the text stream wasn't super fast but not slow either." It is important to note that while our results indicate that throttling was beneficial, the chat is not artificially made too slow by throttling. We describe this balance further in the Potential Optimizations section below.

Supporting Community

The use of throttling and a top three section also resulted in some positive signals around community among viewers and with the streamer. Throttling resulted in fewer lurkers, suggesting a more inclusive and engaged environment. Interfaces that encourage more people to participate give more users a chance to feel like a meaningful part of a community. The presence of the top three section increased the sense of community among viewers according to questionnaire responses. This may be because the top three section is the same for all viewers, and represents the result of their collective interactions, giving users a tangible outcome of the group's messaging and voting behaviors. As mentioned previously, throttling increased the ability to reply to messages while the top three section increased the ability to understand what messages the streamer was referring to. Both of these differences suggest that throttling and the top three section aid in forming community with other viewers and the streamer.

Highlighting Salient Messages

The Conversational Circle concept intentionally reduces the number of messages shown to the streamer, thus lowering the amount of information the streamer has to process and their cognitive effort. However, it relies on the assumption that upvoting is useful in promoting the messages which are important for the streamer to see. Questionnaire responses indicate that adding the top three section increased user agreement that the messages that got many likes were important. Participant A98's feedback elaborated on the effect of voting: "the experience is tailored to those who are watching. By this I mean that if the majority want to view a specific exhibit, then that text will be voted on, and acted upon by the streamer." Results from coding the chat messages indicate that when the top three section is present, requests (a common type of message that is important for the streamer) get upvotes more often than other kinds of messages, another indicator that voting is useful in differentiating among different types of messages.

The top three section may increase the effectiveness of voting in highlighting salient content for several reasons. It provides a clear visual indicator of the end result of voting. As users upvote content, they can see the most popular messages move up into the top three section. By contrast, without a top three section, the only indication in the interface of the impact of upvoting is the incremented vote counter shown while the message is displayed. Further, the top three section allows the streamer to easily see which content is receiving the most upvotes. Without the top three section, the streamer must attempt to pick out popular messages from a large stream of incoming messages, in addition to managing their camera and responding to viewers. Beyond simply helping the streamer, this may increase user desire to upvote because the results of upvoting are more easily actionable by the streamer.

Because the streamer was a member of the research team familiar with the goals of the study, we cannot make a formal comparison among conditions from the streamer's perspective. Informally, the streamer found that the top three section was useful in reducing the effort needed to monitor the chat, allowing him to concentrate on navigating the museum and framing the camera. Thus, he felt it gave him a better sense of the chat because it is difficult to pick out important messages from a fast-moving text chat while controlling the camera and walking around at the same time.

Potential Optimizations

The Conversational Circle system exposes several parameters to tune its behavior. These include the target rate of incoming messages for a user, how many unpopular messages to fade out of view, and how many popular messages to add in the gaps. For our study, we set these parameters based on estimates from our prototype studies about how much content is appropriate for users to process. We also built a chat simulator that created fake messages and upvotes which allowed us to experiment with the system during development and tune the parameter values to increase the usability of the system.

One indicator of how well the parameters are set is participants' response to the questionnaire questions regarding the balance of text messages. On the scale for this question, 1 represents too many messages, 5 represents too few messages, and 3 means the number of messages was about right. In the throttled conditions, users responded a bit below the middle of the scale. The mean response was 2.59 for Condition A and 2.55 for Condition C, indicating the volume of messages was

close to ideal but still a bit high. This is an improvement than the non-throttled Conditions B and D where the means were 1.47 and 1.57 respectively. Though the responses were better when throttling, they were still below the ideal value, suggesting room for optimization.

Tuning the balance between the number of messages from a user's neighborhood and the number of popular messages which get added into the gaps is another potential optimization. While we did not conduct a formal comparison of different values for these settings, we expect they may change the efficacy of the upvoting system and outcomes of community.

LIMITATIONS

The goal of our system is to provide a useful and manageable way for a large number of users to interact while viewing a live stream. To study the system, we used an audience size that is at or above what streamers have reported as difficult to manage. However, live streams today may experience audiences in the hundreds of thousands, and it is challenging to recruit study participants to watch a live stream simultaneously at that scale. Yet, we do believe that our studies simulate interactions in a viewing audience that is larger than the number of study participants.

We used crowd sourcing to recruit our study participants, who have different motivations and affordances than regular viewers. Because they are being paid to participate in the study they may feel obligated to interact with the stream, thus sending more messages than they might otherwise. The relatively low level of lurking (14.8%-25.6%) is another indicator of increased interaction over a typical chat forum. The participants also completed the task in a web browser on their computers, using a keyboard and mouse to interact. In contrast, many streaming services focus on mobile devices, where users must type messages on a small touch keyboard. Because of potential feelings of obligation to participate, and the high-speed input afforded by physical keyboards, our participants may have generated more content than a typical, real world, mobile viewer.

Our study considered only one kind of subject material for a live stream. Because we streamed from a museum, many messages were about the airplanes in the museum. However, content in live streams varies widely. For example, in a stream from a public figure or celebrity, the chats may have many questions that the viewers want answered. On the other hand, a stream may show a live event unfolding such as a rock concert, where most of the chat messages may just be among viewers reacting to and discussing the performance. Because the type of messages may be different with other types of streams, users may use an upvoting system differently, and it would be important to confirm that it still highlights the most salient messages.

Finally, our study focused on the viewer experience using the Conversational Circles system. Although our streamer indicated a preference for the top three conditions, where the most popular messages are shown to the streamer, we cannot make any firm conclusions about how other streamers would react to the Conversational Circles concept. Doing so would require replicating the current study with a number of different streamers, each requiring hundreds of viewers, which we leave for future work.

FUTURE WORK

The current study provides an initial evaluation for the Conversational Circle concept, but more work is needed to understand how the Conversational Circles concept translates to other types of content, and how it scales to even larger audience sizes.

The study also exposed potential improvements to increase the utility of Conversational Circle concept. First, users often asked questions that had already been answered by the streamer. This may be because they missed the first time the question was answered, perhaps because they joined the stream afterwards. One example of this in the current study was users asking where the museum was located. Users continued to ask this question even after it was answered multiple times. In the case of this and other common questions, providing metadata such as location may help users understand the content of a stream better and reduce repeated questions. Alternatively, a way of tracking questions that were already answered and relaying the response to users who repeat a previous question would provide a more general solution to the problem.

Finally, throttling messages can be accomplished in several ways, such as randomly selecting a subset of all incoming messages, selecting every *n*th message, or creating neighborhoods which randomly add or remove users. Tracking people who like or comment at similar times or importing pre-existing social media connections could also be used to form neighborhoods. We designed our neighborhood system on a circle because as the neighborhood size changes, the core neighbors (closest to each other on the circle) remain. We did not evaluate our system in comparison to different methods of forming neighborhoods for throttling. A future study in which more users join, leave, and even rejoin the stream during the broadcast would provide better validity for evaluating this component of the Conversational Circle system.

CONCLUSION

We have designed a new type of text chat, Conversational Circles, which offers better usability than a standard text chat with many concurrent users, without sacrificing the ability for all users to participate. Our testing confirms that the Conversational Circle system offers advantages over a standard text chat. One of the key aspects of live streaming that differentiates it from other forms of social media like video sharing over YouTube or vlogging is that viewers can influence the content of the stream as it happens. While some other approaches to text chat in large-scale live streams focus on limiting interaction, Conversational Circles gracefully scales to larger numbers of viewers while giving all users the chance to interact with media as they view it.

ACKNOWLEDGEMENTS

We thank the hundreds of anonymous Amazon Mechanical Turk workers who gave us feedback on our prototypes.

REFERENCES

1. Jonathan Bishop. 2007. Increasing participation in online communities: A framework for human–computer interaction. *Computers in Human Behavior.* 23 (2007), 1881–1893.
2. Susan E. Brennan. 1998. The Grounding Problem in Conversations With and Through Computers. In *Social and cognitive psychological approaches to interpersonal communication.* S. R. Fussell and R. J. Kreuz (Eds.) Lawrence Erlbaum, Hillsdale, NJ, pp. 201-225.
3. Arvid Engström, Mark Perry, and Oskar Juhlin. 2012. Amateur vision and recreational orientation: Creating live video together. In *Proceedings of the SIGCHI conference on Computer Supported Cooperative Work (CSCW 2012)*, 651-660. http://doi.acm.org/10.1145/2145204.2145304
4. Baldo Faieta, Bernardo Huberman, and Paul Verhaeghe. 2006. Scalable online discussions as listening technology. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*, 15c-15c. http://doi.org/10.1109/HICSS.2006.427
5. Andy Field. 2013. *Discovering statistics using IBM SPSS statistics*. Sage.
6. Elodie Fichet, John Robinson, Dharma Daily, and Kate Starbird. 2016. Eyes on the Ground: Emerging Practices in Periscope Use during Crisis Events. In *Proceedings of the 13th Annual Conference for Information Systems for Crisis Response and Management (ISCRAM 2016)*
7. William A. Hamilton, Oliver Garretson, and Andruid Kerne. 2014. Streaming on Twitch: Fostering participatory communities of play within live mixed media. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI 2014)*, 1315-1324. http://doi.acm.org/10.1145/2556288.2557048
8. William A. Hamilton, John C. Tang, Gina Venolia, Kori Inkpen, Jakob Zillner, and Derek Huang. 2016. Rivulet: Exploring Participation in Live Events through Multi-Stream Experiences. In *Proceedings of the ACM International Conference on Interactive Experiences for TV and Online Video (TVX 2016)*, 31-42. http://doi.acm.org/10.1145/2932206.2932211
9. Brian McInnis and Gilly Leshed 2016. Running user studies with crowd workers. *interactions*. 23(5), 50-53. http://dx.doi.org/10.1145/2968077
10. Gustavo Nascimento, Manoel Ribeiro, Loic Cerf, Natalia Cesario, Mehdi Kaytoue, Chedy Raissi, Thiago Vasconcelos, Wagner Meira. 2014. Modeling and Analyzing the Video Game Live-Streaming Community. In *Proceedings of the 9th Latin American Web Congress (LA-WEB 2014)*, 1-9. http://doi.ieeecomputersociety.org/10.1109/LAWeb.2014.9
11. Tasneem Nashrulla. 2016. We Blew Up A Watermelon And Everyone Lost Their Freaking Minds. (8 April 2016). Retrieved September 8, 2016 from https://www.buzzfeed.com/tasneemnashrulla/we-blew-up-a-watermelon-and-everyone-lost-their-freaking-min
12. Frank Palotta. 2016. Trump Tower climber keeps media networks hanging on live. (10 August 2016). Retrieved August 17, 2016 from http://www.crossroadstoday.com/story/32732450/trump-tower-climber-keeps-media-networks-hanging-on-live
13. Jenny Preece, Blair Nonnecke, and Dorine Andrews. 2004. The top five reasons for lurking: Improving community experiences for everyone. *Computers in Human Behavior.* 20 (2004), 201–223.
14. Stuart Reeves, Christian Greiffenhagen, Martin Flintham, Steve Benford, Matt Adams, Ju Row-Farr, Nick Tandavanitj. 2015. I'd Hide You: Performing Live Broadcasting in Public. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI 2015)*, 2573-2582. http://doi.acm.org/10.1145/2702123.2702257
15. Martin Saveski, Sophie Chou, and Deb Roy. 2016. Tracking the Yak: An Empirical Study of Yik Yak. In *Proceedings of the Tenth International AAAI Conference on Web and Social Media (ICWSM 2016),* 671-674.
16. John C. Tang, Gina Venolia, and Kori Inkpen. 2016. Meerkat and Periscope: I Stream, You Stream, Apps Stream for Live Streams. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI 2016)*, 4770-4780. http://doi.acm.org/10.1145/2858036.2858374
17. Bogdan Vasilescu, Alexander Serebrenik, Prem Devanbu, Vladimir Filkov. 2014. How social Q&A sites are changing knowledge sharing in open source software communities. In *Proceedings of the ACM conference on Computer Supported Cooperative Work (CSCW 2014),* 342-354. http://doi.acm.org/10.1145/2531602.2531659
18. David Vronay, Marc Smith, and Steven Drucker. 1999. Alternative Interfaces for Chat. In *Proceedings of the 12th annual ACM symposium on User interface software and technology (UIST 1999)*, 19-26. http://doi.acm.org/10.1145/320719.322579
19. Justin D. Weisz, Sara Kiesler, Hui Zhang, Yuqing Ren, Robert E. Kraut, and Joseph A. Konstan. 2007. Watching together: Integrating text chat with video. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI 2007)*, 877-886. http://doi.acm.org/10.1145/1240624.1240756