Anonymous "Town Hall"-Style Meetings: Fair Scheduling of Single Speakers in Dissent

Alexander Schurman
Advisor: Prof. Bryan Ford
Second Reader: Prof. Jakub Szefer

Introduction
The goal of this project is to create a real-time voice chat facility that permits controlled, abuse-resistant, anonymous "town hall"-style meetings. The idea is to create an anonymous virtual setting modeled after real-life town hall meetings, in which a single roving microphone is shared in a large meeting, allowing only a single person to speak at a time to a group of listeners. All meeting participants should have a chance to speak if they wish, and no single individual should be able to "hog the mic" for a long time, preventing others from speaking. This facility would allow large groups to anonymously meet online and organize in an orderly fashion.

My project in particular will be focused on modifying the scheduling mechanism in the Dissent protocol, upon which this meeting facility will be built, to allow anonymous, text-based “town hall”-style meetings. Other related projects for enabling anonymous “town hall”-style meetings that will be completed in parallel by other individuals include modifying Dissent to support voice chat and creating a mobile application for Dissent.

Background
This project will be built using the existing Dissent protocol, which permits provably anonymous communication but currently supports only text messages, not voice, and permits all clients to send messages at once, which would make for a very chaotic town hall meeting.

In Dissent, each client shares a secret with each server, from which pseudorandom strings are generated. Each client submits to the servers a pseudonym key, which the servers then schedule using a verifiable shuffle. This shuffle assigns each client a set of bit positions in a larger message. The clients set their assigned bits arbitrarily, leaving the other bits zero. They then XOR the message with a pseudorandom string for each server generated from the shared secret. The clients send this ciphertext to the servers, which then XOR all the clients' messages, revealing the plaintext from each client, and broadcast the plaintext.
In the scheduling process, each client is assigned both a request bit slot and a message slot, which is initially collapsed to save on overhead from silent clients. If a client wishes to speak, it needs to send a one in its request bit in round $r$ for its message slot to be expanded in round $r+1$.

Since the goal is to allow only a single client to speak at once, this scheduling scheme that allows any client to request a slot will need to be modified. The modified scheduling scheme should only allow a single speaker at a time, and there needs to be a limit on how long a speaker can "hold the mic" if another client requests to speak.

**Possible Solutions**

As this project is still in the very early stages, any thoughts about possible solutions are rather vague and open to significant revision. The main challenges that need to be addressed are (1) how will clients request to speak, (2) how will a speaker inform the group that he is finished, and (3) how will fairness be maintained in the face of abusive clients and/or servers.

**Requesting to Speak**

One possible solution is to use the existing request slot mentioned above to request to become the speaker rather than request to unconditionally open a message slot as it does now. That way, no additional communication protocols will need to be created.

If modifying existing code regarding this request bit proves to be prohibitively complex, another possibility is to use out-of-band communication between clients and servers to request to speak. How exactly this would look is still unclear.

**Ending a Speech**

A possibility here is to use the same request bit again to end a speech if the client is currently speaking. The advantage of this approach would be in minimizing the changes made to the existing message format, but this is prone to abuse. One can imagine abusive users setting the speaker’s request bit to one illegally, forcing the speaker to stop prematurely. This abusive tactic will need to be considered if this approach is taken.

Another approach would be to again use out-of-band communication for the speaker to end speaking. Again, this approach is not clearly defined yet.

**Maintaining Fairness**

If a speaker does not request to end speaking, the servers will need to cut him off at some point to allow other clients to speak, provided other clients have actually requested to speak. One approach is to add requesting clients to a queue maintained by the servers. When the speaker’s time (measured in rounds and recorded by the servers) runs out, the speaker’s message slot closes and the next pseudonym in the queue is given a message slot. If the queue is empty when the speaker’s time runs out, he can continue to speak until someone requests to speak.

If maintaining a queue between the servers proves to be too complex or vulnerable to abuse (one might imagine a rogue server and a client colluding to prevent others from speaking), then another solution might involve randomly selecting the next speaker from the set of clients requesting to speak at the moment of time-out. When the current speaker’s time runs out, the
pseudonym keys of the clients requesting to speak are verifiably shuffled. From the verifiable
shuffle, the next speaker is selected.

**Deliverables**
The deliverables for this project will include source code for the modified Dissent protocol and a
written report describing the system and how it differs from the original Dissent protocol.