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

Alexander Schurman
Yale University CPSC 490, Spring 2014
Advisor: Bryan Ford
Second Reader: Jakub Szefer

1 INTRODUCTION

This project is one of three related projects building on the Dissent system, with the collective goal being to create an iOS application that provides accountable, anonymous, group voice communication in which only a single member of the group is permitted to speak at a time, making the meeting resemble a “town hall” meeting. The Dissent system being developed at Yale provides accountable, anonymous group communication, but it does not support voice, and it allows all participants to send messages at once. For many applications, including orderly voice chat meetings, this is unacceptable, since an anonymous member could flood the meeting with noise and make it impractical for other, well-behaved members to communicate.

This particular project focused on modifying the Dissent system to support permitting only a single node to send messages at a time. This principally involved changes to scheduling, the process by which nodes are assigned bit positions to set as they wish in the overall message. A node’s assigned set of bit positions is called a slot. Previously, any node could open its slot and begin to send messages, so the process of scheduling needed to be modified. This was done in such a way as to ensure fairness. One can imagine a speaker “hogging the mic,” so to speak, by keeping his slot open and thus preventing anyone else from being able to speak. In order to prevent such abuse, there must be mechanisms to ensure that anyone who wishes to speak can speak. This was done through imposing a configurable time limit for speaking and forming a queue of nodes awaiting to speak. Nodes can also optionally be configured to be moderators, allowing them to force the current speaker’s slot to close at any time, thus forcing him to stop speaking. Since this could be a dangerous feature, should an adversarial node be configured as a moderator, it is completely optional.

In addition to providing this particular scheme for scheduling speakers, this project provides a framework by which new scheduling schemes can be more easily added to Dissent.
Most of the changes centered around `Dissent::Anonymity::CSBulkRound`, which handles much of the Dissent protocol described in [1]. Prior to the changes made in this project, it also handled part of the scheduling process. The relevant scheduling-related code was pulled out of `CSBulkRound` and moved to new classes described below.

In order to retain old scheduling behavior without recompiling Dissent, one can select between the new and old scheduling schemes using configuration settings. Additional configuration settings related to this project’s new scheduling scheme were added as well.

In order to interface with the iOS application built upon this modified version of Dissent, the web server functionality of Dissent needed to be modified with two new web services, since the iOS application uses the web server rather than the command line to interface with Dissent.

### 2.1 Scheduler Classes

A new abstract class `Dissent::Anonymity::Scheduler` was added to be a super class of specific scheduler subclasses that would handle the details of opening and closing slots. `CSBulkRound`, which previously contained much of the scheduling code, now creates a subclass of `Scheduler` depending on configuration settings and calls into it to handle the opening and closing of slots.

Existing scheduling-related code was taken from `CSBulkRound` and added to a subclass of `Scheduler` called `AllSpeakScheduler`, since under this scheduling scheme all nodes may open their slots and speak at once. Another subclass of `Scheduler` called `QueueScheduler` contains the new scheduling scheme created for this project.

#### 2.1.1 AllSpeakScheduler

When this class is used for scheduling, the old Dissent behavior is retained. Any node can set its assigned slot-opening bit and begin sending messages in the following round.

#### 2.1.2 QueueScheduler

When this class is used for scheduling, only a single node may have its slot open at once. When a node sets its assigned slot-opening bit, its slot is not necessarily opened for the next round. This bit informs all other nodes to add it to a queue of nodes waiting to speak. This queue is
maintained deterministically by every node, so all nodes should be in agreement about the queue.

At the end of each round, if no one’s slot is open and the queue is not empty, the first node is dequeued and its slot is opened.

Slots are closed in one of three ways under this scheduling scheme. The node whose slot is open can close its slot by setting the same bit used to request a slot. This is always possible in this scheduling scheme regardless of any configuration settings. The second two ways for the speaker’s slot to close are optional depending on configuration settings. If a time limit is configured, each node keeps track of the number of rounds that the speaker’s slot has been opened, and when the time limit is reached, the slot is closed and the next node is dequeued. Finally, if one or more moderator nodes have been configured, a moderator node can close the speaker’s slot by setting its own request slot bit.

2.2 Settings

This project’s modified version of Dissent defaults to the old Dissent behavior in allowing all nodes to open their slots and speak. New configuration settings have been added to access the new behavior.

- **scheduler_type** can be set either to allspeak or queue. The former configures Dissent to behave as it does by default, allowing all nodes to open their slots and speak, while the latter configures Dissent to use the new scheduling scheme created for this project. This configuration setting only makes a difference when **session_type** is set to csbulk.

- **speaker_time_limit** is set to the integer number of rounds during which a node is permitted to have its slot open. After this many number of rounds has passed after a node’s slot is opened, it is forced to close its slot. If this is set to zero, there is no time limit, and slots are only closed manually. The default value is zero. This configuration setting only makes a difference when **scheduler_type** is set to queue.

- **moderator** is a flag that is not set to a value. Nodes that are configured as moderators can close other nodes’ slots. The intent behind adding this option is to provide a means of forcing selfish or annoying users to stop speaking. However, this is an exceedingly dangerous option, since an adversarial moderator has the ability to render the town hall useless by preventing anyone else from speaking. Caution must be taken with this option. This configuration setting only makes a difference when **scheduler_type** is set to queue.

See Dissent::Applications::Settings for the changes made to add the new configuration settings described above. Note also that a moderator flag was added to Dissent::Identity::PublicIdentity.
2.3 **Front-end Changes**

In order to use the new functionality added by this project, some changes needed to be made to the command line and web server interfaces for Dissent.

2.3.1 **Command Line**

The only change needed to be made to the Dissent console is the addition of a new command to close the speaker’s slot. If this command is run without queue scheduling being configured, it will have no effect. If it is run under queue scheduling, it will only have an effect if it is run on a node that has its slot open or on a node configured as a moderator.

2.3.2 **Web Server**

Since the iOS application built upon this modified version of Dissent uses the web server to interact with Dissent, these changes are especially important for the overall application. Two web services were added to access the new functionality added for this project:

- Dissend::Web::CloseSlotService
- Dissend::Web::GetSlotStatusService.

`CloseSlotService` allows client application to close the speaker’s slot. As is the case with the close command added to the Dissend console, this only has an effect under queue scheduling if the node either has its slot opened or is configured as a moderator. One uses this service to close the speaker’s slot by POSTing to [web server root]/session/close_slot.

`GetSlotStatusService` allows client applications to check whether or not the node’s slot is opened. This service works for either scheduling scheme, but it’s obviously more useful under queue scheduling when the client application does not know how long the queue is and when its slot is opened. One uses this service by sending a GET request to [web server root]/session/slot_status. The response is a dictionary with the entry “status” and an integer value of 1 for an open slot and 0 for a closed slot.

3 **Discussions and Future Improvements**

Through this project, Dissend can be configured to allow only a single node’s slot to be opened at a time, enabling client applications that rely on there not being excessive noise, like the town hall voice chat iOS application being developed by another student.

Because this was done through the addition of an abstract class

- Dissend::Anonymity::Scheduler, more scheduling schemes can be more readily added to Dissend by creating new subclasses and adding configuration options to
- Dissend::Applications::Settings to enable them. This extensibility could allow for more varied applications that require new scheduling schemes to be built on Dissend.

It is important to note that these changes were made to an old version of Dissend. During the course of developing this project, Dissend underwent some significant changes, and because of
time constraints and unexpected difficulties in interfacing with the iOS application through web services, this project was not merged with the most recent version of Dissent. The logical next step for this project would be to complete this merge.

4 REFERENCES