Alex Dalgleish amd96 Robinson College

Part II Project Progress Report

Peer-to-peer collaborative editing

February 3, 2017

Supervised by
Stephan Kollmann

Overseen by
Prof. Marcelo Fiore
Prof. Ian Leslie

DoS
Dr. Alastair Beresford

Work Completed

Firstly, I have written a library for using operation-based CRDTs (Conflict-free Replicated Data Types) to represent an ordered list (a chain of stored symbols) for use in co-operative editing, like Google Docs supports. I have also made a library for an application that might store a CRDT-based list, perform operations on it, and distribute those operations to others on some network, such that everyone eventually will have the same list. This can be in either a client-server model (where all applications connect to one central server) or a peer-to-peer model (where all applications connect to all others). For the case where the application is a client I have written a server which stores the operations it receives and distributes them to everyone else, but does not have a CRDT stored. For the peer-to-peer model, all peers store the operations and distribute their own to everyone else in one hop. Additionally, operations can be done offline, and then synced up with others when you come back online.

To facilitate testing I wrote a basic GUI for this library, such that it may be run as a complete application.

I have also implemented optional encryption for the peer-to-peer mode, where a Diffie-Hellman exchange generates a shared secret between peers, then all future communication is encrypted (and authenticated) with AES-CCM and a key derived from the shared secret.

Finally, I have used a library which can talk to a local Tor process to enable the peers in peer-to-peer mode to advertise Tor Hidden Services for other peers to connect to. The *onion* addresses are globally addressable so can bypass NATs and firewalls. In this way, all traffic is end-to-end encrypted and clients hide their IP addresses, even from other peers.

Difficulties faced

I had an unexpected difficulty writing the peer-to-peer code in resolving the case where two peers try to connect to each other at the same time. Having two connections for one 'channel' is undesirable, so I limited each peer to holding no more than one socket object at the same time. Unfortunately, various interleavings of connection-initiating and connection-accepting threads often caused each end to drop their end of different connections, such that overall both connections failed. This was resolved eventually by introducing asymmetry into the code such that there is a global order on the connections and the same one can be dropped on both sides if necessary.

Work to be completed

As I have mostly completed the extensions I had planned to do, I really just need to do lots of measurements with my library in order to evaluate it. For example:

- Timing how long the operations take to perform locally vs the how many operations have been performed
- Measuring how much data is flowing between all peers vs how many peers there are
- Measuring the time spent in the Tor overlay network for each operation

This puts me roughly on schedule with my original plan.