Chapter 1

Introduction

A common use of the Internet is to perform some kind of online editing of documents. Real-time collaborative editors allow multiple clients to edit a document simultaneously, regardless of location. One popular such editor is provided as part of Google Drive[1]. However, by using this service, Google is given access to everything you write, as this extract from the Google Terms Of Service state:

> When you upload, submit, store, send or receive content to or through our Services, you give Google (and those we work with) a worldwide license to use, host, store, reproduce, modify, create derivative works (such as those resulting from translations, adaptations or other changes we make so that your content works better with our Services), communicate, publish, publicly perform, publicly display and distribute such content.

For the privacy conscious, it may be undesirable for Google to be able to *publish* or *distribute* content. Instead of sending all ones writings through large, corporate servers (the client-server architecture) that may inspect the data as they please, my project makes use of a peer-to-peer (P2P) architecture, so that all data about a document is stored only by those editing it.

A P2P real-time collaborative editor is inherently a distributed system, and as such suffers from the usual pitfalls of distributed systems. For an editor we don't necessarily care about *physical* time, only which events happened before others, referred to as *logical* time. However, in some cases events A and B might not be ordered by the happens-before [**?**] relation (they are *concurrent*).

Imagine a simple situation with two clients A and B, who are editing the same document. A types an a, and tells B about this. Before B receives the message, it types b and tells A about this. Thus, naïvely, A would have a final state of ab and B would have ba. A correct editor would somehow ensure that the final state of any

[1] https://www.google.com/drive/

two clients editing the same document is the same. The first part of my project focuses on how this can be done.

So far, I have considered some privacy concerns for storing data at a node in a network (e.g. a server). However, on the Internet, data also travels through many other intermediate nodes, and some inspect the traffic passing through them as they wish[2]. Moreover, they can read the packet headers to find the source and destination of all the data, and so potentially can infer who is collaborating with whom. The Onion Router (Tor) is a project that aims primarily to mitigate the latter concern. Its operation will be described in the Preparation chapter, but using it means packets randomly hop around the world before reaching the destination, such that none of the intermediate nodes know both the sender and recipient of the message. Moreover, the use of its Hidden Services means additionally data is encrypted end to end, so Tor is a convenient way to address both problems presented, and the second part of the project builds up to this.

CRDTs are quite recent (first proposed in 2011 [**?**]), yet there have been huge numbers of them proposed for various applications [**?**]. They have been used for collaborative editing [**?**, **?**, **?**, **?**], and separately there are editors that can be run over the Tor network [**?**]. However, to my knowledge there hasn't been a CRDT-based editor with builtin support for Tor.

[2]https://en.wikipedia.org/wiki/Deep_packet_inspection