CRATE: Writing Stories Together with our Browsers
Brice Nédelec, Pascal Molli, Achour Mostefaoui

To cite this version:
CRATE: Writing Stories Together with our Browsers

Brice Nédelec
Université de Nantes, LINA
2 rue de la Houssinière
Nantes, France

Pascal Molli
Université de Nantes, LINA
2 rue de la Houssinière
Nantes, France

Achour Mostefaoui
Université de Nantes, LINA
2 rue de la Houssinière
Nantes, France
first.last@univ-nantes.fr

ABSTRACT

Real-time collaborative editors are common tools for distributing work across space, time, and organizations. Unfortunately, mainstream editors such as Google Docs rely on central servers and raise privacy and scalability issues. CRATE is a real-time decentralized collaborative editor that runs directly in web browsers thanks to WebRTC. Compared to state-of-the-art, CRATE is the first real-time editor that only requires browsers in order to support collaborative editing and to transparently handle from small to large groups of users. Consequently, CRATE can also be used in massive online lectures, TV shows or large conferences to allow users to share their notes. CRATE’s properties rely on two scientific results: (i) a replicated sequence structure with sub-linear upper bound on space complexity; this prevents the editor from running costly distributed garbage collectors, (ii) an adaptive peer sampling protocol; this prevent the editor from oversizing routing tables, hence from letting small networks pay the price of large networks. This paper describes CRATE, its properties and its usage.

Keywords

Collaborative editor; decentralized; real-time

1. INTRODUCTION

Real-time collaborative editors allow authors to simultaneously write shared documents. Trending editors such as Google Docs rely on central servers. They raise privacy issues since service providers take advantage of their mediating position to observe all users and documents. They also raise scalability issues, for they require a server to support an editing session, so handling many editing sessions requires many resources. Managing large editing sessions puts a lot of pressure on the server which may become overloaded.

CRATE is a distributed and decentralized Collaborative Editor providing scalable real-time editing capabilities directly within web browsers. Compared to state-of-the-art, CRATE is the first real-time editor that only requires browsers in order to support collaborative editing and to transparently handle from small to large groups of users.

Suppose massive online lecture platforms allow students to share their notes. Many lectures run in parallel involving various number of students, i.e., from few to thousands. Also, even during the editing session the audience is subject to significant changes in size, going from thousands to hundreds. Finally, the resulting document size is hardly predictable, for it depends of the number and zealous of involved students.

CRATE handles these situations by combining different algorithms to achieve an acceptable trade-off between space, time and communication complexities. In laboratory, we observed this trade-off on configurations involving up till 600 browsers. In this paper, we describe the usage, the architecture and the properties of CRATE. Also we propose a live experiment to WWW2016 participants aiming to confirm laboratory experiments about CRATE.

Section 2 depicts the usage of CRATE. Section 3 details the overall architecture of CRATE along with its basic functioning. Section 4 shows the way to build decentralized networks directly in browsers. Section 5 describes the distributed data structure that represents the document. Section 6 shows our laboratory result and describes the live experiment setup. Section 7 concludes.

2. USAGE

CRATE’s video, source code and online demo are freely available at \(\text{https://github.com/Chat-Wane/CRATE}\).

In the online demo\(^1\) a user creates a document by clicking the(BigInteger) icon at the top of the screen (see Figure 1). At the time, the document is only local and no one can read nor modify it apart from its creator. When the user is ready to share, she clicks on the chain icon and CRATE is crafting an editing session URL that can be sent by mail, published on Twitter, or advertised by any other mean.

Once the collaborators open the link, it automatically connects them to the editing session of the creator. It retrieves the shared document and they can start the real-time collaborative editing. In turns, they are able to share the access to the document too.

Figure 1 shows a screenshot of the graphical user interface of CRATE. In this scenario, two editors are running in a same instance of the web application. The editing session comprises two members nonetheless. The leftmost author

\(^1\text{http://chat-wane.github.io/CRATE/}\) the browser must support WebRTC.
writes the **readme** file of the Github repository about **Crate**. The green earth indicates that the editing session is live. The blue (spinning) circle indicates that she opens the access to her document. Therefore, anyone can join it with the proper URL. The rightmost editor renders the document written in markdown language. It has a green earth indicating a connected state but contrarily to the leftmost editor, it is not sharing the access to the document.

URLs are the means for accessing to editing sessions, hence, to documents. Internally, when a browser opens the link it retrieves the web application. The latter queries a mediator server with the parameters contained within the link (see Figure 3). The mediator server tries to establish the very first contact between the joining browser and an editing session sharer. The mediator ensures solely a minimal service consisting in the editing session accessibility.

Yet, an URL does not always grant the access to the live editing session, for there must be at least one sharer among the editors. Also, there may exist multiple URLs leading to a same editing session using different joining parameters. As we observe in Figure 1, **Crate** enables hyperlinks. By the same, it allows surfing between editing sessions like any user would do on Wikipedia, enhanced with real-time capabilities.

### 3. ARCHITECTURE

To provide availability and responsiveness of documents, **Crate** follows the optimistic replication scheme. Each editor replicates the document locally and directly performs operations on it. Next, the editor spreads its changes to all other participants. The system is correct iff editors integrating a same set of changes have replicas converging to an equivalent state, i.e., users read the same document. This property is called strong eventual consistency [4].

**Crate** uses a Conflict-free Replicated Data Type (CRDT) for sequences to ensure such property [4]. CRDTs ensure consistency at the price of a unique identifier attached to each element of the sequence. Recently, **LSeq** [2] proposed a strategy to bound the space complexity of these identifiers to \(O(\log(d)^2)\) where \(d\) is the document size. As such, **Crate** does not require costly distributed garbage-collection-like mechanisms to maintain its efficiency.

Strong eventual consistency requires the eventual delivery of all operations to all editing session members. **Crate** builds an editing session using **Spray** [3], a random peer sampling protocol [1] (RPS) built on top of WebRTC. A peer sampling protocol provides each network member with a partial view of the network significantly smaller compared to this latter. Maintaining randomness among partial views ensures the connectivity of the network. Unlike prior RPS, **Spray** allows adapting the partial view sizes to the editing session size without measuring its number of participants. Since the propagation protocol of messages extensively uses these tables, the network traffic inherits this scalability. **Crate** adapts its operation to the needs of the editing session.

The decentralized collaborative editor **Crate** is developed in web languages only: HTML, CSS, and mostly JavaScript. As shown by Figure 2, **Crate** comprises four layers:

- **The graphical user interface** that renders the document to the users (cf. Figure 1). Each local update is immediately applied to the document view for real-time sake. Then, the update is applied to the replicated data type;
does the same. As a result, they roughly have an identical and the oldest neighbor. In response, the oldest neighbor fling with the oldest of its neighbors. The initiator serves as the global number of connections is $N$. Since the number of neighbors connects to the joining peer. When they try to initiate a shuffle with it. Since such detections presumably happen $\log(N)$ times, they all duplicate one of their neighbors except for one; the probability being processed using their own neighborhood table size.

The network layer that (i) builds a network of browsers for each editing session and (ii) uses it to propagate the updates to all collaborators (cf. §4).

4. NETWORK OF BROWSERS

Crate uses Spray to build a network of browsers thanks to WebRTC. The latter is a recent technology that allows establishing browser-to-browser channels of communication. Its connection establishment protocol imposes a round-trip of messages. Thus, if an editor $e_1$ wants to connect to another editor $e_2$, it needs to send a message to $e_2$ and acknowledge its response. While the first of such connection must use a mediator (e.g. mail, or dedicated server), once established, the network can assume the position of mediator and start working autonomously.

Random peer sampling protocols provide each editor with a local neighborhood table. These tables allow communicating to a subset of editors. Messages travel from neighbor to neighbor to reach any editor in a scalable manner.

Spray’s lifecycle at each editor comprises three steps that aim to provide and maintain logarithmically scaling neighbor tables compared to the editing session size:

- **The joining:** the editor wants to join the editing session. It contacts an editor inside the network and waits for its answer to establish the very first WebRTC connection between them. Using this connection, the joining editor asks to its contact editor to operate as mediator to advertise its presence to the contact’s neighbors. Each of these neighbors connects to the joining peer. Since the number of neighbors is supposed logarithmic, the number of connections in the network grows by $\log(N) + 1$ where $\log(N)$ is the natural logarithm of the network size $N$. Following such growth, the global number of connections is $N \log(N)$.

- **The shuffling:** each editor regularly initiates a table shuffling with the oldest of its neighbors. The initiator serves as mediator between half of its neighborhood chosen at random and the oldest neighbor. In response, the oldest neighbor does the same. As a result, they roughly have an identical neighborhood size which tackles the load-balancing problem that may rise from the joining part of the protocol. It is worth noting that the number of connection remains constant. In addition, the references in neighborhood tables are randomly distributed. The network converges in exponential time toward a topology exposing properties similar to those of random graphs (cf. Figure 3).

- **The leaving:** the editor can leave the network without giving notice, i.e., equivalent to a crash. Nevertheless, its removal from the system must lead to a decrease of $\log(N) + 1$ connections while currently it represents $\log(N)$ connections from its neighborhood tables, and $\log(N)$ connections from other editors’ neighborhood tables pointing to it. The adjusting falls under the responsibility of these other editors. For this purpose, they detect the departure of a neighbor by using their own neighborhood table size.

Random peer sampling protocol constitutes the ground of various protocols such as topology managements. Crate uses it to disseminate all operations of the replicated sequence to all collaborators in a scalable manner. Referred as epidemic dissemination, rumor mongering, or gossiping, the protocol is a two-step operation: (i) Each operation performed on the local document replica leads to the creation of a message. The latter is sent to all the neighborhood provided by Spray. (ii) Each editor receiving such broadcast message transmits it again to its neighborhood. Thus, messages transitively reach all collaborators very efficiently. This dissemination protocol inherits from Spray’s adaptiveness. Hence, traffic follows the editing session size.

Prior approaches generally oversize the partial views to handle largest groups of users. Unfortunately, smaller groups suffer from such a priori dimensioning. Since Spray automatically adjusts neighborhood table sizes, it equally and transparently handles from smallest to largest groups.

5. REPLICATED DOCUMENT

To provide eventually consistent replicas, Crate uses a conflict-free replicated data type for sequences. Such sequence type avoids the difficult and error-prone task of solving conflicts by the mean of commutative operations. Thus, the sequence type provides two basic operations: the insertion and removal of an element. If two users perform solving conflicts by the mean of commutative operations. Thus, the sequence type provides two basic operations: the insertion and removal of an element. If two users perform solving conflicts by the mean of commutative operations. Thus, the sequence type provides two basic operations: the

A lexicographic total order amongst these identifiers allows retrieving the sequence:

$$\begin{align*}
\ell_i < \ell_j & \iff (p_i < p_j) \lor (p_i = p_j) \land (s_i < s_j) \lor (p_i = p_j) \land (s_i = s_j) \land (c_i < c_j) \\
\ell_i = \ell_j & \iff \neg (\ell_i < \ell_j) \land \neg (\ell_j < \ell_i)
\end{align*}$$
6. EXPERIMENT AND DEMONSTRATION

In laboratory, we tested CRATE on the Grid’5000 testbed with configurations involving up till 600 browsers. Figure 4 shows the traffic generated at each member by intensive experiments involving up till 600 browsers. Figure 4 shows the combined effects of SPRAY and LSEQ. Indeed, we observe that the traffic logarithmically scales to the editing session size thanks to SPRAY. The members of the smallest editing session (101 browsers) are less traffic intensive than ones of the largest group (601 browsers). But the messages transiting the network are important too. The growth of each plot corresponds to the size of updates. Keeping the size under such a sub-linear bound avoids the need of an unaffordable distributed garbage-collecting-like mechanism.

7. CONCLUSION

In this paper we described CRATE, a distributed and decentralized real-time collaborative editor running in web browsers. Compared to state-of-the-art, CRATE allows real-time editing anytime, anywhere, whatever the number of participants. It can be seen as an alternative way to provide a collaborative editing service without Cloud support. Using a scalable replicated data structure for sequences, and an adaptive peer sampling protocol, CRATE is able to alleviate scalability issues and privacy issues without compromising ease-of-access. CRATE’s video, source code and online demo are available at \text{https://github.com/Chat-Wane/CRATE}

8. ACKNOWLEDGMENTS

This work was partially funded by the French ANR project SocioPlug (ANR-13-INFR-0003), and by the DeSeNt project granted by the Labex CominLabs excellence laboratory (ANR-10-LABX-07-01). Experiments presented in this paper were carried out using the Grid’5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see \text{https://www.grid5000.fr}).

9. REFERENCES


\text{https://en.wikipedia.org/wiki/Exquisite_corps}