Evaluating CRDTs for Real-time Document Editing

Mehdi Ahmed-Nacer, Claudia-Lavinia Ignat, Gérald Oster, Hyun-Gul Roh, Pascal Urso

▶ **To cite this version:**

Mehdi Ahmed-Nacer, Claudia-Lavinia Ignat, Gérald Oster, Hyun-Gul Roh, Pascal Urso. Evaluating CRDTs for Real-time Document Editing. 11th ACM Symposium on Document Engineering, Sep 2011, Mountain View, California, United States. pp.103–112, 10.1145/2034691.2034717 . inria-00629503

HAL Id: inria-00629503
https://inria.hal.science/inria-00629503v1

Submitted on 6 Oct 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

HAL Authorization

Evaluating CRDTs for Real-time Document Editing*

Mehdi Ahmed-Nacer
Université de Lorraine
LORIA
54506 Vandœuvre-lès-Nancy, France
mahmedna@loria.fr

Claudia-Lavinia Ignat
INRIA Nancy - Grand Est
LORIA
54600 Villers-lès-Nancy, France
ignatcla@loria.fr

Gérald Oster
Université de Lorraine
LORIA
54506 Vandœuvre-lès-Nancy, France
oster@loria.fr

Hyun-Gul Roh
INRIA Nancy - Grand Est
LORIA
54600 Villers-lès-Nancy, France
roh@loria.fr

Pascal Urso
Université de Lorraine
LORIA
54506 Vandœuvre-lès-Nancy, France
urso@loria.fr

ABSTRACT

Nowadays, real-time editing systems are catching on. Tools such as Etherpad or Google Docs enable multiple authors at dispersed locations to collaboratively write shared documents. In such systems, a replication mechanism is required to ensure consistency when merging concurrent changes performed on the same document. Current editing systems make use of operational transformation (OT), a traditional replication mechanism for concurrent document editing.

Recently, Commutative Replicated Data Types (CRDTs) were introduced as a new class of replication mechanisms whose concurrent operations are designed to be natively commutative. CRDTs, such as WOOT, Logoot, Treedoc, and RGAs, are expected to be substitutes of replication mechanisms in collaborative editing systems.

This paper demonstrates the suitability of CRDTs for real-time collaborative editing. To reflect the tendency of decentralised collaboration, which can resist censorship, tolerate failures, and let users have control over documents, we collected editing logs from real-time peer-to-peer collaborations. We present our experiment results obtained by replaying those editing logs on various CRDTs and an OT algorithm implemented in the same environment.

Categories and Subject Descriptors

I.7.1 [**Document and Text Processing**]: Document and Text Editing; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*

General Terms

Algorithms, Experimentation, Performance

Keywords

Real-time Editing, Collaboration, Benchmark, Commutative Replicated Data Types

1. INTRODUCTION

Collaboration is a very important aspect of any team activity and hence of importance to any organisation such as business, science, education, administration, political or social institutes. Central to collaboration is editing of shared documents. Collaborative editing has to take into account the geographical distribution of team members, possibly across a wide range of time zones, together with the mobility of individuals. Collaborative systems require that users be able to edit shared documents as easily as one edits a single-author document. The major benefits of collaborative editing include reducing task completion time, reducing errors, getting different viewpoints and skills, and obtaining an accurate text [13, 27].

The nature of collaboration varies extensively in terms of strategies and proximity of writing groups [27]. For instance, there exist various writing strategies; users can jointly write a document by working closely together; or they can work separately, and afterwards their works are subject to review by other group members. According to proximity, all members of some groups can work in the same location and on the same time schedule, while other groups work on different schedules and may be located thousands of miles apart.

Depending on such strategies and proximities, the collaboration between users can be synchronous or asynchronous. In asynchronous collaboration, members of the group modify the copies of the documents in isolation, synchronising afterwards their copies to re-establish a common view of the data.

*This work is partially funded by the french national research programs ConcoRDanT (ANR-10-BLAN-0208) and STREAMS (ANR-10-SEGI-010).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DocEng'11, September 19–22, 2011, Mountain View, California, USA.
Copyright 2011 ACM 978-1-4503-0863-2/11/09 ...$10.00.

Synchronous or real-time collaboration means that modifications by a member are immediately seen by other members of the group. Such real-time and synchronous features could be very helpful; for example, when a paper deadline is imminent, authors of a paper can avoid the time consumed for synchronisation of the copied documents and can edit their parts, being aware of the changes made by the other members.

Recently, real-time collaborative editing gained much attention as Google Docs and Google Wave support such features. It is also a main subject of investigation by recent user studies [20] that focus on user behaviour during real-time collaboration. This paper focuses on consistency maintenance algorithms for real-time collaborative editing.

Data replication is necessary in collaborative editing to achieve high responsiveness. Many mechanisms dealing with data replication have been introduced. In the domains of replicated databases, pessimistic approaches are usually used, which gives users the illusion of a single copy [1]. High availability can be obtained by allowing read operations on any replica, while update operations have to be atomically applied on all replicas. Pessimistic approaches cannot ensure high responsiveness for real-time collaboration because the initiator of an update should acquire an exclusive access, and an update cannot be applied immediately before resolving conflicts.

Optimistic replication [19] is a family of approaches suitable for real-time collaboration. Compared to pessimistic approaches, optimistic replication needs no atomic update. Though replicas are allowed to diverge temporarily, they are expected to converge eventually.

Optimistic synchronisation algorithms are classified into state-based and operation-based. State-based approaches use only information about the different states of the documents and no information about the evolution of one state into another. Examples of state-based approaches are three-way merges adopted by version control systems such as Subversion [3] and differential synchronisation approaches [5]. On the other hand, operation-based merging approaches keep the information about the evolution of one state of the document into another in a buffer. The merging is done by executing the operations performed on a copy of the document onto the other copy of the document to be merged. State-based approaches are generally not suitable to be used in the real-time communication as the difference between versions has to be computed each time an operation is performed resulting in an increased time complexity. In this paper we study operation-based optimistic synchronisation approaches.

Operational transformation (OT) approach has been identified as an appropriate operation-based synchronisation mechanism for real-time collaboration [4, 17, 25, 12, 24, 29, 26]. Modifications are represented by means of operations such as insertion and deletion of a character in the case of textual documents. OT changes the index of an operation based on the history of operations in order to take into account the effects of concurrent operations and to eventually achieve consistency. A few commercial document editing systems such as Google Wave and Google Docs adopt centralised OT algorithms, such as Jupiter [12]. However, the centralised approaches may store not only shared documents and all changes, but also some personal information, which could be a privacy threat, in the hands of a single large corporation.

In order to overcome the disadvantages of central authority, i.e., to let users have control over their documents, to resist censorship, and to tolerate failures, a tendency is to move towards decentralised peer-to-peer collaboration [2]. This solution is suitable for user communities that rely on sharing their infrastructures and administration costs. An example of this tendency in the domain of real-time collaborative editing is Microsoft SharePoint Workspace 2010 [22] (previously known as Groove). This software allows documents of desktop office applications to be synchronised through a peer-to-peer network of SharePoint servers while managing security of local copy. However, most decentralised OT algorithms use version vectors [24, 26] or central timestamps [29] to detect concurrent operations which do not scale well in cloud and peer-to-peer environments with dynamic groups.

Performance is a key factor of the success of real-time collaborative applications. If these applications cannot quickly respond to user actions, users may get frustrated and will quit the application. Studies in [23, 8] found that people can comfortably notice changes which respond to local and remote actions within $50\,ms$.

Very recently, a new class of algorithms called CRDT (commutative replicated data types) [15, 31, 16, 18] that ensures consistency of highly dynamic contents on peer-to-peer networks emerged. Unlike OT algorithms, CRDTs require no history of operations, and no detection of concurrency in order to ensure consistency. Instead, they are designed for concurrent operations to be natively commutative by actively using the characteristics of abstract data types such as lists or ordered trees. However, CRDT algorithms have not yet been applied for real-time collaborative editing.

In this paper, we investigate the suitability of CRDT algorithms by performing evaluations of CRDT algorithms against real editing logs of real-time peer-to-peer collaboration. The results we obtained show that CRDT algorithms are suitable for real-time collaboration and that they outperform a representative class of OT approaches.

The paper is structured as follows. In the next section, we start by giving an overview of main CRDTs and OT algorithms, and study their theoretical complexity. Then, we describe how we designed real-time peer-to-peer collaborations from which the editing logs are collected, and provide some details on the logs. We then provide evaluation results of the analysed CRDTs and OT algorithms by replaying the collected logs. We also compare our work with existing experimental results of related work. In the last section we provide concluding remarks and directions of future work.

2. STUDIED REAL-TIME DOCUMENT EDITING ALGORITHMS

In the nineties, various OT algorithms have been proposed [4, 17, 25, 12, 24, 29, 26, 10]. They constitute the groundwork of CRDT approaches that emerged in the last decade: WOOT [15], Treedoc[16], Logoot[31], RGA [18], partial persistent data structure (PPDS) [33] and causal tree model [6] were presented.

In the following subsections, we briefly describe some representative algorithms belonging to the families of CRDT

and OT approaches. We also provide their theoretical complexities.

2.1 WOOT Algorithm

WOOT [15] is the first CRDT algorithm which was proposed. Operations used by this algorithm are insertion and deletion of elements in a linear structure. Elements are uniquely identified. An insertion is defined by specifying the new element identifier, the element content and the identifiers of the preceding and following elements. Concurrent operations determine partial orders between elements. The merging mechanism can be seen as a linearisation of the partial order to obtain a total order. To obtain convergence, the total order has to be the same at all peers. As operations are integrated in any order at a site, merging has to be computed incrementally and independently of the order of arrival. WOOTO [30] is an optimisation of WOOT that uses element degree to compare unordered elements.

The advantage of these algorithms is their suitability for open user groups where users often join and leave the network. Moreover they do not require a causal delivery of operations. A disadvantage is that the algorithms use tombstones, i.e. elements are not physically deleted but only marked as deleted. Since tombstones cannot be removed without compromising consistency, performance degrades during time.

We designed a new version of WOOTO, called WOOTH, inspired by RGA approach. A hash table and a linked list are used for optimising retrieval, update and insertion of an element.

2.2 Logoot Algorithm

Logoot [31] is another CRDT approach that ensures consistency of linear structures. Logoot associates to the list of elements of the structure, an ordered list of identifiers. Identifiers are composed by a list of positions. Positions are 3-tuples formed with a digit in specific numeric base, a unique site identifier and a clock value. When inserting an element, Logoot generates a new identifier. Identifiers have unbounded lengths and are totally ordered by a lexicographic order. So a new identifier can always be generated between two consecutive elements. Different strategies can be adopted to produce the new identifier [32], all of them using randomness to prevent different replicas to produce concurrently close identifiers.

The advantage of Logoot is the absence of tombstones and an improvement of the algorithmic complexity compared to WOOT/WOOTO/WOOTH. A disadvantage is the size of the identifier that can grow unbounded. However, experiments made on collaboration traces produced on Wikipedia shown that the size of identifiers stays low even for largest collaborative contributions [32].

2.3 Replicated Growable Array (RGA)

Replicated growable array [18] is a CRDT that supports not only insertion and deletion but also update operations which replace the content of an element without changing the size of the document. In this paper, we evaluate only insertion and deletion operations. RGAs maintain a linked list of elements, via which local operations find their target elements with integer indexes. Meanwhile, a remote operation retrieves its target element via a hash table with a unique index of the target.

As a unique index, the paper [18] introduces s4vector consisting of four integers. A unique s4vector is issued with every operation, and the oldness or newness of multiple s4vectors can be determined transitively, respecting causality. Using the properties of uniqueness and transitivity, the s4vector associated with the insertion that creates an element is used as a unique index of the element, which is also used to resolve conflicts between concurrent insertions at the same position. An insertion compares its s4vector with the s4vector identifiers of elements next to its target element, and adds its new element in front of the first encountered element that has an older s4vector. That is, a newer insertion inserts an element closely to its target position with higher precedence than relatively order concurrent insertions. Such precedence transitivity, realised with s4vectors, ensures consistency of concurrent insertions. As some other CRDTs, RGAs also uses tombstones for deleted elements. Tombstones should be preserved as long as they can be accessed by other remote operations.

2.4 Operational Transformation Algorithms

Most representative OT algorithms that do not make any assumption on using a central server for a total order broadcast of operations are SOCT2 [24] and GOTO [25] algorithms. The principle of this class of algorithms is that when a causally ready operation is integrated at a site, the whole log of operations is traversed and reordered. After reordering, causally preceding operations come before concurrent ones in the history buffer. Finally, the remote operation has to be transformed according to the sequence of concurrent operations. In order to reduce the complexity of the integration mechanism of a remote operation, history buffer is pruned by using a garbage collection mechanism as proposed in [9, 26]. Replicas use this mechanism to remove operations they know to be received by all other replicas. However, the garbage collection mechanism has no effect in open peer-to-peer networks were users often join and leave.

These algorithms require transformation functions satisfying conditions C_1 and C_2 [17, 24]. Satisfying C_1 allows executing in any order two concurrent operations defined on the same document state while ensuring convergence of the document. C_1 is sufficient with only two sites or in client-server applications. C_2 expresses the equality between an operation transformed against two equivalent sequences of operations. These two conditions ensure that transforming any operation with any two sequences of the same set of concurrent operations in different execution orders always yields the same result. C_1 and C_2 are sufficient for ensuring convergence in a peer-to-peer architecture. In [7], it was shown that many proposed transformation functions fail to satisfy these conditions. The only existing transformation functions that satisfy conditions C_1 and C_2 are the ones proposed by the TTF (Tombstone Transformation Functions) approach [14]. In the TTF approach when a deletion of a character is performed, the character is not physically removed from the document, but just marked for deletion, i.e. deleted characters are replaced by tombstones.

2.5 Theoretical Evaluation

The worst case complexity for each of the above described algorithms is presented in Table 1. We consider the time complexity of generation of a local user operation (single character insert or delete) and for the execution of a remote

operation. We denote by R the number of replicas and by H the number of operations that had affected the document. We consider constant time for accessing an element in a hash table. In the worst case scenario for the approaches that use tombstones, the document size including tombstones equals H. For the approaches that use state vectors we took into account the complexity of state vector creation, i.e. $O(R)$, associated with the operation at the moment of its generation.

Algorithm	local		remote	
	ins	del	ins	del
WOOT	$O(H^3)$	$O(H)$	$O(H^3)$	$O(H)$
WOOTO	$O(H^2)$	$O(H)$	$O(H^2)$	$O(H)$
WOOTH	$O(H^2)$	$O(H)$	$O(H^2)$	$O(log(H))$
Logoot	$O(H)$	$O(1)$	$O(H.log(H))$	$O(H.log(H))$
RGA	$O(H)$	$O(H)$	$O(H)$	$O(log(H))$
SOCT2/TTF	$O(H+R)$	$O(H+R)$	$O(H^2)$	$O(H^2)$

Table 1: Worst-case time-complexity analysis

The average complexity of each of the above described algorithms is presented in Table 2. We denote by:

- c the average number of operations concurrent to a given one,
- n the size of the document (non deleted characters),
- N the total number of inserted characters (including the ones that were deleted called tombstones),
- k the average size of Logoot identifier[1].
- $t = N - n$ the number of tombstones,
- $d = \lceil (t+c)/n \rceil$ the average number of elements (tombstones and concurrently inserted elements) found between to successive document elements

Algorithms using tombstones (WOOTs, RGA and TTF) have a complexity depending on N for retrieving an element or a document position in their model. WOOT algorithms have a complexity proportional to d^2 since they call a recursive algorithm to place a newly inserted element between these $O(d)$ elements. RGA algorithm compares a remote inserted element only with the elements inserted concurrently at the same position (c/n in average). The SOCT2 algorithm reorders each operation of the log against $O(c)$ concurrent ones. With an efficient garbage collection mechanism, there are $O(c)$ operations in the SOCT2 log.

Algorithm	avg. local		avg. remote	
	ins	del	ins	del
WOOT	$O(N.d^2)$	$O(N)$	$O(N.d^2)$	$O(N)$
WOOTO	$O(N.d^2)$	$O(N)$	$O(N+d^2)$	$O(N)$
WOOTH	$O(N+d^2)$	$O(N)$	$O(d^2)$	$O(1)$
Logoot	$O(k)$	$O(1)$	$O(k.log(n))$	$O(k.log(n))$
RGA	$O(N)$	$O(N)$	$O(1+c/n)$	$O(1)$
SOCT2/TTF	$O(N+R)$	$O(N+R)$	$O(H.c)$	$O(H.c)$
with g.c.	$O(N+R)$	$O(N+R)$	$O(c^2)$	$O(c^2)$

Table 2: Average time-complexity analysis

The space complexity of meta-data used by each replica is presented in Table 3. In average, algorithms using tombstones need to store N elements in their model. Logoot stores n identifiers with an average size of $O(k)$. SOCT2 additionally stores a log of operations, each one containing a version vector with size of $O(R)$.

Algorithm	space complexity	
	worst	average
WOOT-WOOTO-WOOTH	$O(H)$	$O(N)$
Logoot	$O(H^2)$	$O(k.n)$
RGA	$O(H)$	$O(N)$
SOCT2/TTF	$O(H.R)$	$O(H.R)$
SOCT2/TTF with g.c.	$O(H.R)$	$O(N+c.R)$

Table 3: Space complexity analysis of meta-data

3. OBTAINING LOGS FROM REAL-TIME P2P COLLABORATION

Currently, some commercial real-time collaboration systems such as Google Docs are on service, but their logs are not complete and freely available. For example, the revision log provided by Google server is a serialisation of user operations transformed by the Jupiter algorithm [12]. Therefore, the revision logs open to the public do not include the information needed for replaying the real-time peer-to-peer collaboration, such as version vectors.

Due to unavailability of logs of the real-time peer-to-peer collaboration, we set up an experiment where we asked students to collaboratively write documents by using a collaborative editor and logged a number of operations generated during this experiment.

3.1 Design of Real-time P2P Collaborations

We designed real-time peer-to-peer collaborations in order to obtain their logs. In this section, we describe the collaboration design and some features of the obtained operation logs such as their lengths and operation types. TeamEdit [28] is the real-time collaborative editor used in our collaborations. We modified it in order to log user operations performed during the collaboration. TeamEdit software uses a central server, but only to establish communication between the different sites. It does not serialise the operations, nor uses the server as a merge mechanism for concurrent operations. Students could use the undo/redo feature we added to TeamEdit, but, however, these operations were transformed into the corresponding insert/delete operations.

We performed two collaborations with groups of students: report and series. The first collaboration (report) was performed with 13 master students divided into three groups: two groups composed of 4 students and one group composed of 5 students. Each group was asked to collaboratively write a report of its semester project in the Software Engineering lecture. Each student in a group worked on the report from a private computer, and was not allowed to use any other communication tools except TeamEdit. The collaboration lasted for one and a half hours. Collaboration was encouraged by noticing that each student will be evaluated according not only to the content of the report but also to the size of his/her contribution. Students were allowed to copy-paste text blocks from some other documents.

In our second collaboration (series) that lasted for about one and a half hours, we asked 18 students to watch an

[1]Theoretically, the size of a Logoot identifier is only bounded by H, but due to stochastic nature of Logoot identifier generation, it has only an infinitesimal chance to be proportional to H.

episode of the series "The Big Bang Theory" and to produce a transcription of the episode while watching it. The transcription of the episode was edited into a shared document with TeamEdit editor. Students were divided into nine groups of two, and each group was asked to make a transcription of a certain hero or to describe the environment and actions that happened during the episode. During this collaboration, students were allowed to communicate mutually. Each student had his/her own computer for editing the shared document. The same episode was played twice, and we assigned different groups to each task in order to obtain more operations from the collaboration.

To minimise internal threats to validity, the whole experiment was conducted in the same period (during one morning), with the same working stations, and with students not aware of our research and non experts in real-time collaborative editing. Of course, this experiment only captures a subset of all the behaviors observable when users collaborate. This selection bias can only be reduced with access to the internal logs of a public widely used real-time editor. Another threat to validity is that the experiment was conducted on a local area network. Thus, propagation time between user desktops was shorter than in a wide area network, leading to a slightly lower concurrency degree between operations. Thus, one can expect sightly lower performance for all studied algorithms in wide area networks.

3.2 Description of Collaboration Logs

TeamEdit was modified to log the following user operations: insertions of a text block and deletions of a range of characters. Text blocks and ranges have a size of one character when a user types on the keyboard. They have a larger size when a user copy-pastes a text block or deletes a selected block. When an operation is generated it has associated a version vector that indicates the number of operations received by the generating site from each of the other sites. In order to apply the studied algorithms on the generated logs, user operations must be transformed into character operations. In the Table 4, the total numbers of user operations and character operations are specified for all the collaborations, the report and the series.

	Report			Series	
	group 1	group 2	group 3	doc 1	doc 2
No. user operations	11 211	11 066	13 702	9 042	9 828
No. char. operations	26 956	47 992	42 443	29 882	10 268
% of del	12	12	12	9	5

Table 4: Total number of user/character operations

The proportion of delete operations is smaller in the series experiments due to the difficulty for non-specialists to type a transcript as quickly as actors talk. So the students had less time to make corrections on their document.

We also observed that, without instructions from us, some students disconnected and then reconnected to the real-time collaboration session during almost all collaborations.

4. EXPERIMENTAL EVALUATION

To evaluate algorithms performance, we designed a framework called ReplicationBenchmark in Java, and reveal the source on GitHub platform[2] under the terms of the GPL license.

The framework provides base classes for common elements of real-time document editing algorithms, such as document, operation, generation and integration algorithms, and version vector, so that each algorithm can be implemented by inheriting them. The framework lets replicas of every algorithm generate character operations in its own formats for the given user operations in the logs; we first measured this generation and local execution time of user operations. The framework also provides a dispatcher that enables each replica to receive generated character operations in the same order as that in the logs. A replica, therefore, can execute character operations, enabling measurement of the net execution time of character operations in each algorithm. The framework uses java.lang.System.nanoTime() for the measurement of execution time of each user operation and each character operation.

To obtain the presented results, we ran each algorithm on each log twenty times on the same JVM execution. The heap size was 1GB. We compute the average execution time per replica for each user operation and for each character operation. From the obtained results, we remove the aberrant values due to java garbage collection, i.e. values more than twice the average for a given operation.

We ran our experiments on a dual processor machine with Intel(R) Xeon(R) 5160 dual-core processor (4Mb Cache, 3.00 GHz, 1333 MHz FSB), that has installed GNU/Linux 2.6.9-5. During the experiment, only one processor was used. We present no results for the WOOT algorithm since it is obviously outperformed by its optimised versions WOOTO and WOOTH algorithms.

4.1 Execution times

4.1.1 User operations

The execution times, in microseconds, of user operations including generation of the corresponding character operations are presented in Table 5. We present the average and the maximum response time, and the standard deviation σ[3].

During execution of an operation the user interface of a document editor is frozen and the user is prevented from typing. If this value is greater than the $50\,ms$, users will notice the bad performance of the collaborative application. The average values measured and presented in Table 5 give the impression that all algorithms performed very well since no average value is greater than $0.2\,ms$. However, if we consider $50ms$ as a limit for the maximum response time, the algorithms belonging to the WOOT family cannot be used safely to build a real-time collaborative applications since the maximum value, confirmed by a high standard deviation, is often greater than $50\,ms$.

The maximum execution time, for every experiment and algorithm, is almost always due to an operation inserting or suppressing a block of hundreds, or thousands, of characters. The only experiment where all algorithms perform very well is the second series since there is no such user operation.

4.1.2 Character operations

The average execution time of character operations are

[2] http://github.com/PascalUrso/ReplicationBenchmark

[3] σ computed using the classical formula $\sqrt{\frac{\sum_{i=1}^{N}(x_i - x_{av})^2}{N}}$.

		Report			Series	
		group 1	group 2	group 3	doc 1	doc 2
Logoot	Avg	6	7	7	5	5
	Max	751	901	2 322	2 267	77
	σ	10	18	26	30	3
WootH	Avg	26	43	49	46	16
	Max	3 623	40 042	**156 407**	**164 934**	453
	σ	44	464	1 396	1 735	16
WootO	Avg	43	112	96	110	23
	Max	13 489	**208 985**	**340 068**	**494 030**	162
	σ	207	2 948	3 331	5 388	16
SOCT2	Avg	21	40	30	27	19
	Max	5 753	24 741	15 312	8 912	147
	σ	85	389	190	119	16
RGA	Avg	27	32	32	20	17
	Max	998	2 082	1 971	2 671	550
	σ	32	46	38	52	19

Table 5: User operation execution times (in μs)

presented in Table 6. This value represents the computation time needed to integrate a remote incoming character operation into the current document.

		Report			Series	
		group 1	group 2	group 3	doc 1	doc 2
Logoot	Avg	5	6	5	3	4
	Max	91	127	110	80	36
	σ	2	3	3	2	3
WootH	Avg	2	4	7	8	2
	Max	694	44 190	4 330	567	200
	σ	8	211	24	17	3
WootO	Avg	54	97	99	84	25
	Max	2 027	**72 937**	8 903	1 334	133
	σ	51	352	108	71	18
SOCT2	Avg	80	573	130	**1 383**	305
	Max	5 286	13 278	5 832	20 727	2 848
	σ	133	1 087	175	1 974	397
RGA	Avg	2	2	2	1	2
	Max	750	1 295	1 002	403	252
	σ	5	7	7	3	3

Table 6: Character operation execution times (in μs)

Since one user operation corresponds to one or more character operations (up to 5000 characters for the biggest copy-paste), we expect that each algorithm performs better for character operations. This is actually the case for almost all maximum execution times. However, the WOOTO algorithm still has a maximum execution time higher than $50\,ms$.

The average character operation execution times are much better than user ones for WOOTH and RGA algorithms (due to hash table usage) and similar for Logoot and WOOTO. The case of SOCT2 is different. SOCT2 has a low maximum execution time but an average execution time that exceeds $1\,ms$. Let us consider a user that copy-pastes a block of 5000 characters, as seen in our experiments. This user operation is translated into 5000 character operations that will be executed one by one on the document replicas of other users. It means that the other users will see the characters of the inserted block appearing one by one in a total duration of five seconds. This is not acceptable from a user's point of view. This performance issues are mainly due to the transformation algorithm coupled with the inefficiency of the garbage collection mechanism as seen in the behavior study described in the next subsection.

Finally, concerning character operations execution times, Logoot algorithm has the best maximum execution times (MAX values in Table 6) and RGA has the best average for every experiment.

4.1.3 *Consistency with theoretical evaluation*

The average results obtained for user and character operations are consistent with the average time-complexities presented in table 2. For local user operations, Logoot has the best results with $O(k)/O(1)$, WOOTO has the worst results with $O(N.d^2)/O(N)$ while RGA, WOOTH and SOCT2 have medium results with complexities around $O(N)$. For remote character operations, RGA and WOOTH have the best complexities and best results, followed by Logoot.

The worst case complexities are also consistent with the results obtained. WOOTO and WOOTH have the worst maximum result for user operations with $O(H^2)/O(H)$. SOCT2 and WOOTO have the worst maximum results for character operations with $O(H^2)$. These correlations validate our implementation of the algorithms for average and worst cases.

4.2 Behaviour

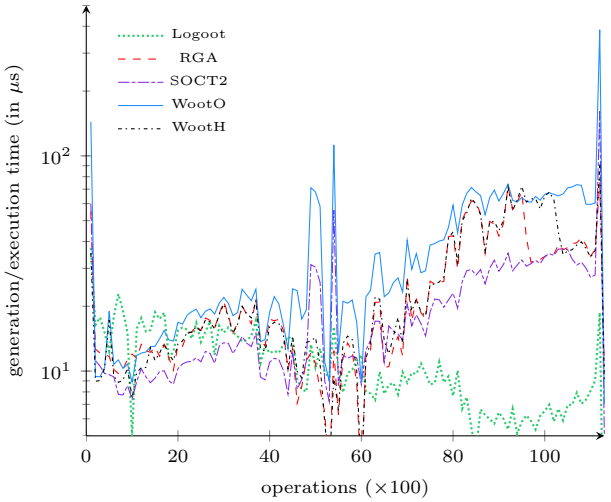

Figure 1: User operation execution times - 1st group report

Since performance of all studied algorithms may degrade over time due to tombstones or growing identifiers, we present in this subsection performance behaviour over time.

The observed behaviour was approximately the same within the two collaboration categories (report and series). We therefore present here only a selection of the comparisons we obtained for certain collaboration logs.

In order to obtain meaningful representations of algorithms behaviour we computed an average of the local generation and respectively execution times for every hundred user operations and every three hundred character operations. The horizontal axis uses a linear scale representing the number

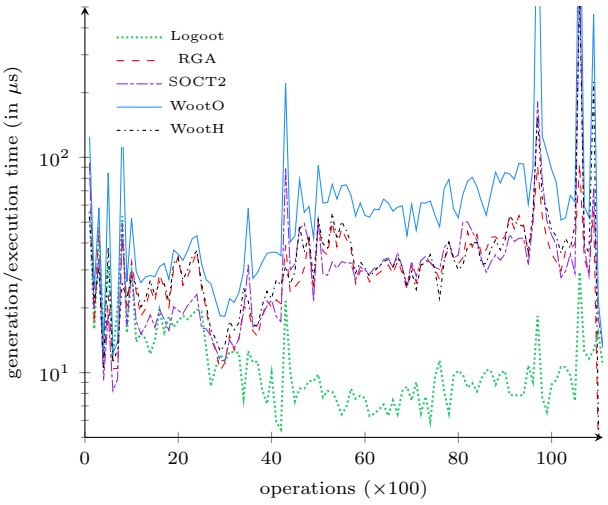

Figure 2: User operation execution times - 2nd group report

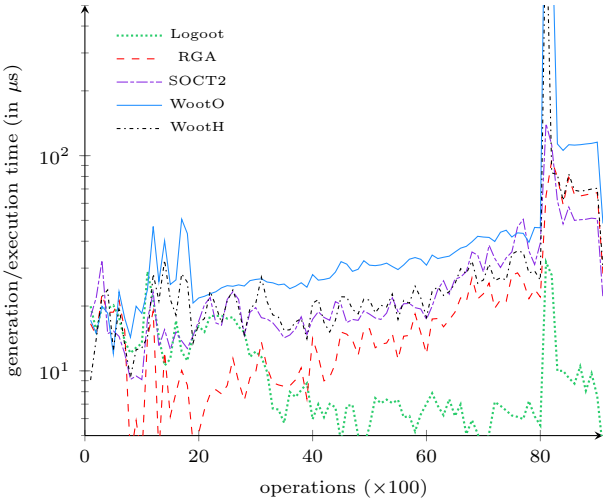

Figure 3: User operation execution times - 1st series

of elapsed operations. The vertical axis uses a logarithmic scale and represents the average time, in microseconds, required to execute operations.

4.2.1 User operations

In Figure 1, 2 and 3 we present algorithms' behaviour for execution times of user operation for the first and second project report and respectively for the first series.

Accordingly to their average time-complexity, the performances of the algorithms using tombstones (WOOTs, RGA and SOCT2) eventually degrade over time. Indeed, all these algorithms have to count the number of tombstones and characters present before an inserted or suppressed string. Temporary improvement of the performance of these algorithms, for instance in first report between the 40th and 60th block of operations, are due to a period where users mostly edit at the beginning of the document. This is also the case at the very end of the three report experiments due to instructions given to students to sign the document by typing their names at the very beginning of the document. The performance of Logoot remains good during all experiments. Indeed, the average size of Logoot identifier stays very low even for large documents as demonstrated in [32].

For every experiment, peaks of low performance common to all algorithms exist, for instance in second report for the 43th block of operations. Such peaks are all due to operation inserting a block of hundreds, or thousands, of characters.

Finally, the global performance behaviors of RGA, WOOTH and SOCT2 algorithms are quite similar (especially for 2nd report), even if they are very different algorithms. This similarity is less obvious in table 5. Such an observation, leads to state the conjecture that any algorithm counting tombstones will have, at best, similar performances.

4.2.2 Character operations

In Figure 4, 5 and 6 we present execution time behaviours for character operations for the second and third project report and respectively for the second series.

During these experiments, their is no peak of low performance common to all the algorithm as for user operation. So, their is no character operation that represent the worst case for all the algorithm. Only WOOTH and WOOTO have such common peaks, for instance in the second report for the 126th block of operation, due to the similar nature of these algorithms.

We can notice that the performance remains stable for Logoot. The performance of RGA and WOOTH are, in average, better than Logoot but have a more erratic behavior for the reports. The behavior of RGA and WOOTH is composed of a base line at 1 μs and some lower performance periods due to more frequent concurrent editing. RGA overperform WOOTH in case of concurrent delete operation.

The performances of WOOTO and SOCT2 degrade over time since they are around ten times slower at the end of the experiments than at the beginning. SOCT2 have the most erratic behavior and the worst average performance.

The behavior of SOCT2 performance is mainly due to its garbage collection mechanism. In our experiments, some users had a period of inactivity between performing two successive modifications. Their inactivity implies that other users do not receive any information regarding the progress of the document state of this inactive user and therefore the garbage mechanism cannot purge the history log. The same situation happens when users left the editing session without notification. It is well-known that pruning history in peer-to-peer networks by using a garbage collection is impossible in these situations. One the over hand, when a user inactive since long produces an operation the performance of SOCT2 temporally improves, for instance in the third report for the 87th block of operation.

The experiments on the series have a slightly different behavior since they contain few deletes and a lot of concurrent operations. Indeed, the students don't have time to type as fast as actor talks.

5. RELATED WORK

Although OT algorithms have been studied since 1989, the first performance report was published in 2006 on the analysis of SDT and ABT algorithms [10]. Performance of the improved versions of these algorithms was published in [11] [21]. In [18] an evaluation of RGA approach was provided. However, these algorithms were not compared with

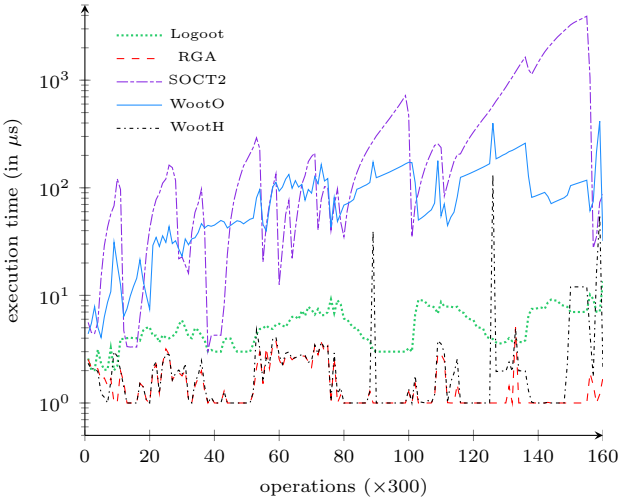

Figure 4: Character operation execution times - 2nd group report

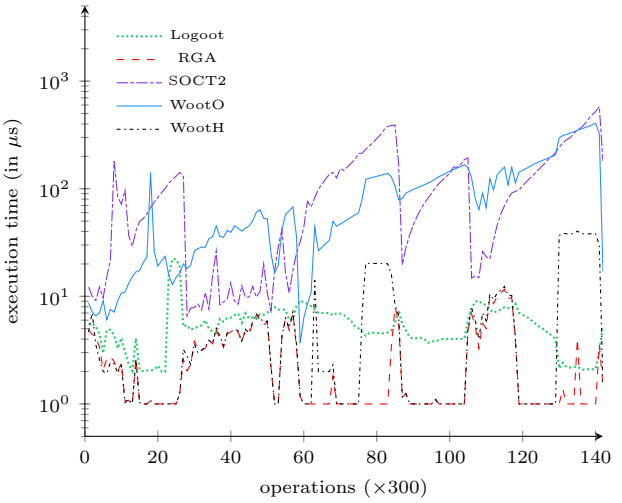

Figure 5: Character operation execution times - 3rd group report

other existing ones. Moreover, performance was measured by using simulated data and not real collaboration traces.

CRDTs such as Treedoc [16] and Logoot[31, 32] presented experimental results, but they were focused not on performance, but on the overhead incurred by tombstones or meta data. PPDS approach [33] presented performance evaluation on their own algorithm without comparing it with other algorithms. All the above mentioned CRDT approaches [16, 31, 32, 33] were evaluated by using Wikipedia and/or Subversion collaboration traces. However, Wikipedia and Subversion traces represent a serialisation of user operations where conflicts between concurrent changes were already resolved by the users.

In this regard, this is the first paper that presents the comparison of real-time document editing algorithms written in the same language and environment. Moreover, this is the first work that evaluates algorithms by using collaboration

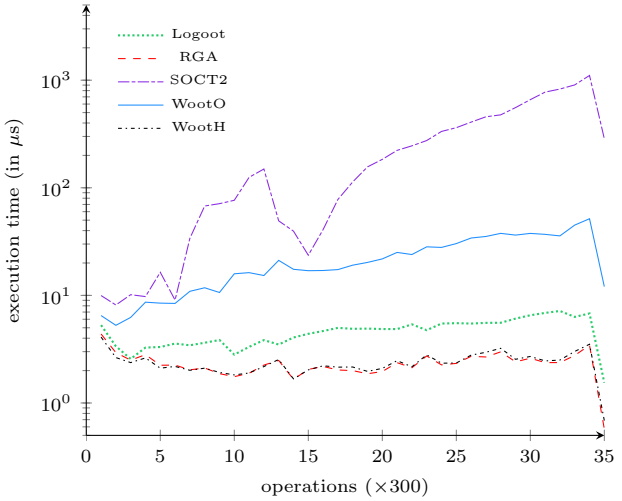

Figure 6: Character operation execution times - 2nd series

traces including concurrency and generated during real-time collaborative editing.

6. CONCLUSION AND FUTURE WORK

In this paper, we evaluated representative consistency maintenance algorithms for real-time collaboration. We provided a theoretical evaluation as well as an experimental one against traces of real-time collaborative editing. We found out that CRDT algorithms initially designed for peer-to-peer asynchronous collaboration are suitable for real-time collaboration. Moreover, they outperform some representative operational transformation approaches that were well established for real-time collaboration in terms of local generation time and remote integration time. As an example, in average case, Logoot and RGA algorithms outperform between 25 and 1000 times faster than SOCT2 OT algorithm. We can also notice that the results we obtained are conforming to the worst case and average theoretical complexities. Best overall performances are obtained by RGA and Logoot algorithms.

One of our directions for future work is to extend our study to other operational transformation and CRDT algorithms and to study other evaluation criteria such as memory occupation, communication complexity and convergence latency. We also plan to obtain larger size traces of real-time collaborations and to generate automatically traces that have the same characteristics as real traces. In this paper we considered and compared decentralised algorithms for real-time collaborative editing. We plan to extend our study to centralised real-time collaborative editing and to analyse suitability of CRDT approaches for this kind of collaboration.

7. REFERENCES

[1] P. A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, Boston, MA, USA, 1987.

[2] S. Buchegger, D. Schiöberg, L. H. Vu, and A. Datta. PeerSoN: P2P Social Networking - Early Experiences and Insights. In *Proceedings of the Second ACM*

EuroSys Workshop on Social Network Systems - SNS 2009, pages 46–52, Nürnberg, Germany, March 2009. ACM Press.

[3] B. Collins-Sussman, B. W. Fitzpatrick, and C. M. Pilato. *Version control with Subversion*. O'Reilly & Associates, Inc., 2004.

[4] C. A. Ellis and S. J. Gibbs. Concurrency Control in Groupware Systems. *SIGMOD Record : Proceedings of the ACM SIGMOD Conference on the Management of Data - SIGMOD '89*, 18(2):399–407, May 1989.

[5] N. Fraser. Differential Synchronization. In *Proceedings of the 9th ACM Symposium on Document engineering - DocEng 2009*, pages 13–20, Munich, Germany, September 2009. ACM Press.

[6] V. Grishchenko. Deep Hypertext with Embedded Revision Control Implemented in Regular Expressions. In *Proceedings of the 6th International Symposium on Wikis and Open Collaboration - WikiSym 2010*, pages 1–10, Gdańsk, Poland, July 2010. ACM Press.

[7] A. Imine, P. Molli, G. Oster, and M. Rusinowitch. Proving Correctness of Transformation Functions in Real-Time Groupware. In *Proceedings of the European Conference on Computer-Supported Cooperative Work - ECSCW 2003*, pages 277–293, Helsinki, Finland, September 2003. Kluwer Academic Publishers.

[8] C. Jay, M. Glencross, and R. Hubbold. Modeling the Effects of Delayed Haptic and Visual Feedback in a Collaborative Virtual Environment. *ACM Transactions on Computer-Human Interaction*, 14(2), August 2007.

[9] P. R. Johnson and R. H. Thomas. Maintenance of Duplicate Databases. RFC 677, Internet Engineering Task Force, January 1975. `http://www.ietf.org/rfc/rfc677.txt`.

[10] D. Li and R. Li. A Performance Study of Group Editing Algorithms. In *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS 2006*, pages 300–307, Minneapolis, MN, USA, July 2006. IEEE Computer Society.

[11] D. Li and R. Li. An Operational Transformation Algorithm and Performance Evaluation. *Computer Supported Cooperative Work*, 17(5-6):469–508, December 2008.

[12] D. A. Nichols, P. Curtis, M. Dixon, and J. Lamping. High-latency, Low-bandwidth Windowing in the Jupiter Collaboration System. In *Proceedings of the 8th Annual ACM Symposium on User interface and Software Technology - UIST '95*, pages 111–120, Pittsburgh, PA, USA, November 1995. ACM Press.

[13] S. Noël and J.-M. Robert. Empirical Study on Collaborative Writing: What Do Co-authors Do, Use, and Like? *Computer Supported Cooperative Work*, 13(1):63–89, 2004.

[14] G. Oster, P. Molli, P. Urso, and A. Imine. Tombstone Transformation Functions for Ensuring Consistency in Collaborative Editing Systems. In *Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing - CollaborateCom 2006*, pages 1–10, Atlanta, GA, USA, November 2006. IEEE Computer Society.

[15] G. Oster, P. Urso, P. Molli, and A. Imine. Data Consistency for P2P Collaborative Editing. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work - CSCW 2006*, pages 259–267, Banff, AB, Canada, November 2006. ACM Press.

[16] N. Preguiça, J. M. Marquès, M. Shapiro, and M. Letia. A Commutative Replicated Data Type for Cooperative Editing. In *Proceedings of the 29th International Conference on Distributed Computing Systems - ICDCS 2009*, pages 395–403, Montreal, QC, Canada, June 2009. IEEE Computer Society.

[17] M. Ressel, D. Nitsche-Ruhland, and R. Gunzenhäuser. An Integrating, Transformation-Oriented Approach to Concurrency Control and Undo in Group Editors. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work - CSCW '96*, pages 288–297, Boston, MA, USA, November 1996. ACM Press.

[18] H.-G. Roh, M. Jeon, J. Kim, and J. Lee. Replicated Abstract Data Types: Building Blocks for Collaborative Applications. *Journal of Parallel and Distributed Computing*, 71(3):354–368, 2011.

[19] Y. Saito and M. Shapiro. Optimistic Replication. *ACM Computing Surveys*, 37(1):42–81, March 2005.

[20] L. Scissors, N. S. Shami, T. Ishihara, S. Rohall, and S. Saito. Real-Time Collaborative Editing Behavior in USA and Japanese Distributed Teams. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems - CHI 2011*, pages 1119–1128, Vancouver, BC, Canada, May 2011. ACM Press.

[21] B. Shao, D. Li, and N. Gu. A Fast Operational Transformation Algorithm for Mobile and Asynchronous Collaboration. *IEEE Transactions on Parallel and Distributed Systems*, 21(12):1707–1720, December 2010.

[22] Microsoft sharepoint workspace 2010. `http://office.microsoft.com/en-us/sharepoint-workspace/`.

[23] B. Shneiderman. Response Time and Display Rate in Human Performance with Computers. *ACM Computing Surveys*, 16(3):265–285, September 1984.

[24] M. Suleiman, M. Cart, and J. Ferrié. Serialization of Concurrent Operations in a Distributed Collaborative Environment. In *Proceedings of the ACM SIGGROUP Conference on Supporting Group Work - GROUP '97*, pages 435–445, Phoenix, AZ, USA, November 1997. ACM Press.

[25] C. Sun and C. Ellis. Operational Transformation in Real-Time Group Editors: Issues, Algorithms and Achievements. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work - CSCW '98*, pages 59–68, Seattle, WA, USA, November 1998. ACM Press.

[26] C. Sun, X. Jia, Y. Zhang, Y. Yang, and D. Chen. Achieving Convergence, Causality Preservation, and Intention Preservation in Real-Time Cooperative Editing Systems. *ACM Transactions on Computer-Human Interaction*, 5(1):63–108, March 1998.

[27] S. G. Tammaro, J. N. Mosier, N. C. Goodwin, and G. Spitz. Collaborative Writing Is Hard to Support: A Field Study of Collaborative Writing.

Computer-Supported Cooperative Work, 6(1):19–51, 1997.

[28] TeamEdit. *A collaborative text editor*. `http://teamedit.sourceforge.net/`.

[29] N. Vidot, M. Cart, J. Ferrié, and M. Suleiman. Copies Convergence in a Distributed Real-Time Collaborative Environment. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work - CSCW 2000*, pages 171–180, Philadelphia, PA, USA, December 2000. ACM Press.

[30] S. Weiss, P. Urso, and P. Molli. Wooki: A P2P Wiki-based Collaborative Writing Tool. In *Proceedings of the International Conference on Web Information Systems Engineering - WISE 2007*, pages 503–512, Nancy, France, December 2007. Springer-Verlag.

[31] S. Weiss, P. Urso, and P. Molli. Logoot : A Scalable Optimistic Replication Algorithm for Collaborative Editing on P2P Networks. In *Proceedings of the 29th International Conference on Distributed Computing Systems - ICDCS 2009*, pages 404–412, Montreal, QC, Canada, June 2009. IEEE Computer Society.

[32] S. Weiss, P. Urso, and P. Molli. Logoot-Undo: Distributed Collaborative Editing System on P2P Networks. *IEEE Transactions on Parallel and Distributed Systems*, 21(8):1162–1174, August 2010.

[33] Q. Wu, C. Pu, and J. E. Ferreira. A Partial Persistent Data Structure to Support Consistency in Real-Time Collaborative Editing. In *Proceedings of the 26th IEEE International Conference on Data Engineering - ICDE 2010*, pages 776–779, Long Beach, CA, USA, March 2010. IEEE Computer Society.