Concurrency Effects Over Variable-size Identifiers in Distributed Collaborative Editing

Brice Nédelec, Pascal Molli, Achour Mostefaoui, Emmanuel Desmontils
LINA, 2 rue de la Houssinière
BP92208, 44322 Nantes Cedex 03
first.last@univ-nantes.fr

ABSTRACT

Distributed collaborative editors such as Google Docs or Etherpad allow to distribute the work across time, space and organizations. In this paper, we focus on distributed collaborative editors based on the Conflict-free Replicated Data Type approach (CRDT). CRDTs encompass a set of well-known data types such as sets, graphs, sequences, etc. CRDTs for sequences model a document as a set of elements (character, line, paragraph, etc.) with unique identifiers, providing two commutative update operations: insert and delete. The identifiers of elements can be either of fixed-size or variable-size. Recently, a strategy for assigning variable-size identifiers called LSEQ has been proposed for CRDTs for sequences. LSEQ lowers the space complexity of variable-size identifiers CRDTs from linear to sub-linear. While experiments show that it works locally, it fails to provide this bound with multiple users and latency. In this paper, we propose h-LSEQ, an improvement of LSEQ that preserves its space complexity among multiple collaborators, regardless of the latency. Ultimately, this improvement allows to safely build distributed collaborative editors based on CRDTs. We validate our approach with simulations involving latency and multiple users.

Categories and Subject Descriptors

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

Keywords

Distributed Documents; Document Authoring Tools and Systems; Distributed Collaborative Editing; Real-time Editing; Conflict-free Replicated Data Types

This work is licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported License (CC BY-SA 3.0). To view a copy of the license, visit http://creativecommons.org/licenses/by-sa/3.0/.

DChanges 2013, September 10th, 2013, Florence, Italy.
ceur-ws.org Volume 1008, http://ceur-ws.org/Vol-1008/paper5.pdf.

1. INTRODUCTION

Distributed collaborative editors allow to distribute the work across space, time and organizations. Some trending editors such as Google Docs [5] use the Operational Transform (OT) approach [14, 15]. However, an alternative based on Conflict-free Replicated Data Types (CRDTs) [12, 13] exists. Compared to OT, CRDTs are more decentralized and scale better.

The CRDTs belong to the optimistic replication [10, 11] approach. Therefore, replicas involved in the collaboration are guaranteed eventual convergence to an identical state. To provide convergence in a replicated sequence, CRDTs use unique identifiers to link elements. When the sequence is a document, the elements can be characters, lines, paragraphs, etc.

We distinguish two types of sequence CRDTs: (i) The tombstone CRDTs [1, 3, 7, 8, 9, 16, 18, 19] use fixed-size identifiers. However, the delete operations only mark and hide elements to the users. Consequently, these deleted elements permanently affect performances. (ii) The variable-size identifiers CRDTs [8, 17] directly encode the order relation in the identifiers. However, their identifiers can grow linearly depending on the editing behaviour. Consequently, they remain unsafe in the distributed collaborative editing context.

To overcome their respective limitations, these approaches need an additional protocol [4, 9] related to garbage collection mechanisms. Nevertheless, these protocols require global knowledge over participants. In the general case, such knowledge remains prohibitively expensive in the context of distributed networks subject to churn.

Recently, LSEQ [6] allowed variable-size identifiers CRDTs to get rid of this costly additional protocol by lowering their upper bound on space complexity from linear to sub-linear. Yet, LSEQ does not guarantee a safe allocation. Indeed, it uses multiple antagonist strategies which, without any coordination between collaborators, leads to an quadratic growth of identifiers.

The contributions of this paper are: (i) experiments that highlight the negative effect of multiple users edition on the size of LSEQ identifiers (ii) experiments that highlight that an increasing latency does not have a negative impact on the size of LSEQ identifiers. (iii) an improvement of LSEQ called h-LSEQ that extends the single user sub-linear upper bound on space complexity to any number of users, regardless of the latency, and without any additional cost.

Ultimately, h-LSEQ allows distributed collaborative editors to safely use variable-size identifiers CRDTs.

The rest of the paper is organized as follow: Section 2 details variable-size identifiers CRDTs for sequences, the allocation strategy LSEQ, and highlights the motivation of this paper. Section 3 presents h-LSEQ a combination of LSEQ and a hash-based choice strategy to overcome the LSEQ limitations. Section 4 shows the results of experiments over LSEQ and h-LSEQ considering latency and multiple users. Finally, Section 5 reviews the related works.

2. BACKGROUND

Conflict-free Replicated Data Types belong to the optimistic replication approach [10, 11]. In the context of distributed collaborative editing, the replicated data is a document, where: (1) each insert/delete operation is prepared locally and broadcast, (2) each remote replica receives and integrates the changes, (3) all involved replicas eventually converge to an identical state.

To provide the convergence property, CRDTs for sequences model a document as a set of couples $\langle elt, id\rangle$ where $id \in (\mathcal{I}, <_{id})$, $<_{id}$ being a strict and dense total order over $\mathcal{I}$, and elt any element, e.g., a character, a line. Two commutative operations allow to update the document:

- $insert(p \in \mathcal{I}, elt, q \in \mathcal{I})$ that generates an identifier id_{elt} with $p < id_{elt} < q$ and adds $\langle elt, id_{elt}\rangle$ to the document.
- $delete(id_{elt})$ that removes $\langle elt, id_{elt}\rangle$ from the document.

The variable-size identifiers CRDTs [8, 17] define identifiers as a series of numbers that can designate paths in a tree. An allocation strategy is in charge of choosing these sequences of numbers. It aims to keep these identifiers as small as possible for the sake of performance.

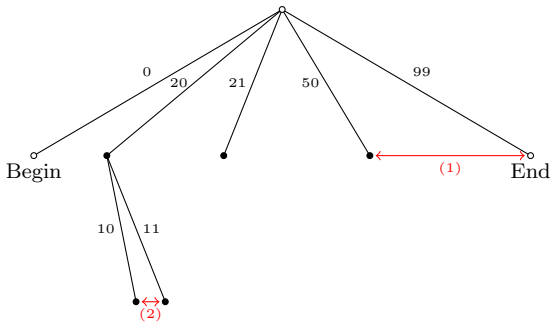

Figure 1: Underlying 100-ary tree model of variable-size identifiers CRDTs. It contains 5 identifiers. 3 identifiers at depth-1: [20], [21], [50] and 2 identifiers at depth-2: [20.10], [20.11]. For the sake of simplicity the elements linked to identifiers are not displayed.

Figure 1 illustrates the underlying model of variable-size CRDTs. If an insertion operation is performed at (1), the allocation strategy chooses an identifier [X] with $50 < X < 99$. When the insert operation is performed at (2), there is no room for another identifier, therefore, the new identifier will be a sequence of three numbers: [20.10.X], where $0 < X < 100$.

These identifiers can grow linearly depending on the position of insert operations.

Recently, LSEQ [6] lowered the space complexity of these CRDTs from linear to sub-linear without favouring any editing behaviour. This improvement aims to get rid of the previously mandatory garbage collecting protocol. Three aspects define LSEQ:

- base doubling: regarding the underlying tree model, each node can have twice more children than its parent. It follows the properties of exponential trees [2].
- two antagonist allocation strategies: *boundary+* and *boundary–* that are designed for end-edition and front-edition respectively.
- random strategy choice: it randomly assigns an allocation strategy to each depth of the tree. It follows the intuition: *since we have no prior knowledge of the editing behaviour, the strategy choice should not favor any editing behaviour. Consequently, the frequencies of appearance of each allocation strategy have to be equal.*

Despite the fact that numerous experiments have been performed using LSEQ including real documents extracted from Wikipedia, they did not include any concurrency. Although, Wikipedia's revisions already present a serialization of the document without explicit concurrency, experiments with this type of documents are of great importance [1].

	#insert operations				
	10	**100**	**200**	**500**	**1000**
1 user (bit/id)	6.5	26.8	32.7	56.0	64.2
10 users (bit/id)	9.5	125.8	377.0	1962.1	5468.0

Table 1: Average bit-length of LSEQ identifiers for single and multiple user(s) and the generation of documents of 10, 100, 200, 500, and 1000 lines.

Table 1 shows the average bit-length of the identifiers assigned by LSEQ on different size synthetic documents created either by a single user or by a group of 10 collaborators. Both documents were edited at the end. We observe that while the identifiers resulting from a single author are sub-linearly upper-bounded, the bit-length of identifiers generated by 10 users are quadratically increasing.

Definition 1 (Problem statement). *Let $\mathcal{D}$ be a document on which n insert operations have been performed. Let $\mathcal{I}(\mathcal{D}) = \{id | (\_, id) \in \mathcal{D}\}$. The function alloc($id_p$,$id_q$) should provide identifiers such as:*

$$\sum_{id \in \mathcal{I}} \frac{log_2(id)}{n} < O(n)$$

Definition 1 from [6] states the *alloc* function property: a sub-linear upper-bound on the average bit-length of identifiers. By excluding any reference to the two phases of optimistic replication operations, it encompasses both the local generation of identifiers and the integration of remote operations. According to Table 1, LSEQ only partially answers the problem statement.

The identified problem concerns the preservation of the space complexity of LSEQ from single user to multiple users edition. Using multiple antagonist strategies without any coordination between collaborators leads to a quadratic growth of identifiers. Consequently, providing a mechanism of agreement in the choice of strategies would greatly improve LSEQ as well as any composition of allocation strategies. Furthermore, such improvement would make variable-size identifiers CRDTs actually usable in the distributed collaborative editing context.

3. H-LSEQ

The h-LSEQ allocation strategy is mainly based on LSEQ, they differ in the strategy choice component. Using LSEQ, each replica involved in the collaboration makes independent random choices. However, Section 2 showed that such strategy fails to provide sub-linearly upper-bounded identifiers in collaboration involving multiple users.

The solution is to reach a global agreement over replicas on which is the allocation strategy used at each depth of the underlying tree model. Since LSEQ aims to get rid of protocols related to garbage collecting mechanisms, any solution that requires additional communication is inconceivable.

We propose to use a hash strategy choice component which have the same local behaviour than the original random strategy choice, but it also provides the required implicit global agreement. Indeed, instead of synchronizing the set of users, we provide an *a priori* agreement by the mean of a hash function. This agreement avoids the possibility of antagonist choices which would have led to a bad global allocation of identifiers. Furthermore, it does not introduce any additional cost to LSEQ.

Algorithm 1 details the h-LSEQ allocation strategy. The changes over LSEQ are simply highlighted. This section explains this algorithm. Extracted from [6], the *prefix* function return a copy of the identifier *id*, i.e., a series of numbers truncated at *depth* or increased until *depth*. The function carefully encodes each number of this copy in a base depending on its depth in order to directly apply the arithmetic operations of line 26 and line 31. For instance, assuming that the departure base is 2^5, a call to $prefix([13.42.37], 2)$ encodes [13.42] using $5 + 6$ bits.

The functions *boundary+* and *boundary−* are two allocation strategies, note that they do not strictly respect the *alloc* function signature in order to factorize the computation of *depth* and *step*.

3.1 Locally

The strategy choice function is surjective and its signature is: $h(depth \in \mathbb{N}^*) : \mathbb{N}$. The returned value corresponds to an allocation strategy unique identifier. The algorithm of the allocation strategy h-LSEQ follows 3 steps:

1. lines 8-11: computes the depth of the future identifiers to allocate
2. line 13: calls the strategy choice function h using the depth
3. line 15: calls the allocation strategy using the identifier returned by h

The strategy choice function returns strategy identifiers following a uniform law. Thus, frequencies of appearance of each strategy are equal and does not favor any editing behaviour. Furthermore, the unpredictability due to randomness preserves the model from intentional and malicious attacks.

The hash function fulfills the requirement of the strategy choice function h. However, contrarily to the random strategy choice component, it requires an initialization. Indeed, the creator of the document must share a hidden seed (cf. line 2) within the document that each collaborator will use to generate the hash function specific to this document. In Algorithm 1 at line 21, we simply use the common random function initialized with the seed and the depth.

Algorithm 1 h-LSEQ allocation function

```
1: let boundary := 10                          ▷ Any constant
2: let seed := 123456789                       ▷ Init with document
3: let S := {⟨0, boundary+⟩, ⟨1, boundary−⟩};
4:     ▷ map<id, allocation strategy>
5:
6: function ALLOC(p, q ∈ I)
7:     let depth := 0; interval := 0;
8:     while (interval < 1) do                 ▷ Not enough for 1 insert
9:         depth + +;
10:        interval := prefix(q, depth) − prefix(p, depth) − 1;
11:    end while
12:    let step := min(boundary, interval);    ▷ Process the
       maximum step to stay between p and q
13:    let idStrat := h(depth);
14:        ▷ Call the hash function
15:    let id := S.get(idStrat).invoke(p, q, depth, step);
16:        ▷ Call the allocation strategy according to hash
17:    return id;
18: end function
19:
20: function H( depth ∈ N* )
21:    return Random(seed ∗ depth).nextInt(0, 1);
22: end function
23:
24: function BOUNDARY+(p, q ∈ I, depth, step ∈ N*)
25:    let addVal := RandInt(0, step) + 1;
26:    return prefix(p, depth) + addVal;
27: end function
28:
29: function BOUNDARY−(p, q ∈ I, depth, step ∈ N*)
30:    let subVal := RandInt(0, step) + 1;
31:    return prefix(q, depth) − subVal;
32: end function
33:
34: function PREFIX(id ∈ I, depth ∈ N*)
35:    let idCopy := [ ];
36:    for (cpt := 1 to depth) do
37:        if (cpt < id.size) then             ▷ Copy the value
38:            idCopy := idCopy.append(id.at(cpt));
39:        else      ▷ Add 0 encoded in the right base
40:            idCopy := idCopy.append(0_base(cpt));
41:        end if
42:    end for
43:    return idCopy;
44: end function
```

3.2 Remotely

Each collaborator generates the same hash function thanks to the shared seed. Also, all collaborators use the same mapping $id \rightarrow allocation\ strategy$ (cf. line 3). Consequently, all hash functions map the same depth to the same allocation strategies. The idea is to reach a consensus on which strategies to employ in order to avoid the waste of identifiers' space due to the choice of antagonist strategies.

Figure 2 highlights the differences between the original random strategy choice and our hash strategy choice. Both cases present two collaborators inserting 2 elements one after the other. The expected result is the sequence "abcd". Hence, Collaborator 1 generates the "a" and "c" and Collaborator 2 generates the "b" and "d". In both cases they draw the same numbers. However, in the case of random strategy choice, one collaborator randomly chooses *boundary+* twice while the other randomly chooses *boundary−* twice. When the editing behaviour is monotonic, the size of identifiers

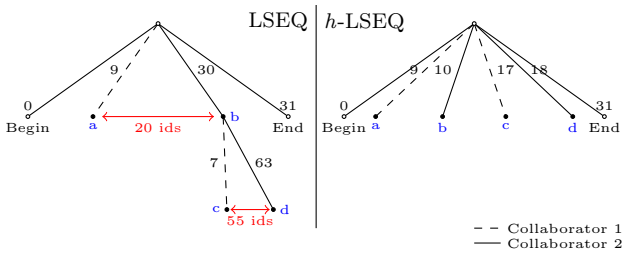

Figure 2: Representation of the underlying exponential tree model of LSEQ and h-LSEQ. Two collaborators insert two elements at the end of the document in order to get the sequence "abcd". On the random side, the collaborators employ antagonist strategies. On the hash side, the collaborators use the same allocation strategy. Both sides draw the same random numbers.

quickly grows. On the opposite, the collaborators using h-LSEQ implicitly agree on using *boundary+* at depth-1, consequently the allocation follows the properties of the one user edition presented in [6].

The next section aims to corroborate our assumptions by performing experiments on different setups with varying the number of collaborators and the latency of the network.

4. EXPERIMENTS

This section is composed of two parts. First, the experimentation focuses on the influence of the number of collaborators on the size of LSEQ and h-LSEQ identifiers. A set of synthetic collaborators generates a document by successively performing insert operations at the end. This experiment aims to show the behaviour of the two strategy choices, and compare the results with the expected sub-linear space complexity.

The second part of experiments consists in highlighting the effect of latency on a document edited by multiple users. Once again, it aims to compare LSEQ and h-LSEQ and to show the impact of concurrency on the average size of identifiers in the document.

The experiments focus on the digit bit-length of generated identifiers. Indeed, all variable-size CRDTs rely on source and clock to guarantee the unicity of identifiers, nevertheless, the space complexity mainly depends on the digit choice made by the allocation strategy.

To perform these experiments, we implemented a simulation framework called *HumbleSimulator*. The sources are available on the Github platform under the terms of the GPL licence [1].

4.1 Multiple users experiment

Objective: show that the random strategy choice without taking into account the strategies employed by other collaborators leads to a quick growth in the size of identifiers. On the opposite, when all collaborators uses a common allocation strategy at a given depth, the space complexity remains sub-linearly upper-bounded.

Description: we evaluate LSEQ and h-LSEQ strategies on a group of 10 collaborators. Thus, the setups are (i) a random strategy choice (**rand**) and (ii) a hash strategy choice

1. `https://github.com/Chat-Wane/HumbleSimulator`

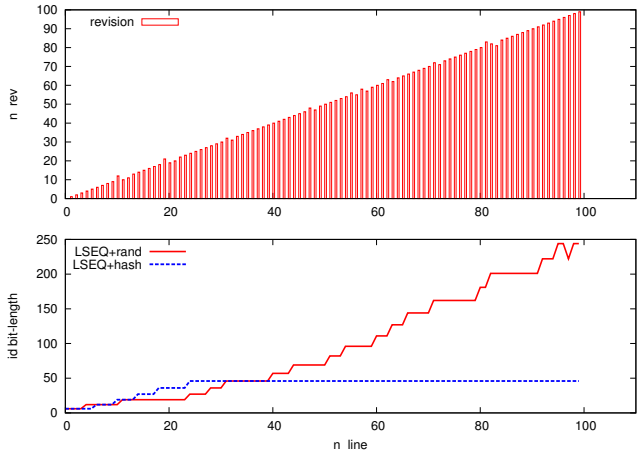

Figure 3: Experiments on a synthetic document of 100 lines edited at the end by 10 collaborators with 10 operations per user. The top figure shows the revision number of the operation. The bottom figure shows the bit-length of each identifier allocated to each line. On average, the bit-length of LSEQ and h-LSEQ identifiers are 94.7 and 39.1 bit/id respectively.

(**hash**). Both of these setups use *boundary+* and *boundary–* and have the same variable values $boundary = 10$ and $base = 2^{4+depth}$. A set of 10 collaborators produces a document of 100 lines by performing 10 insert operations each.

Results: Figure 3 shows on the top part the spectrum of the synthetic document, i.e., the revision date of each line. Since the bars are getting taller when they are closer of the end of the document, it indicates that the users edited at the end monotonically. On the second part of the Figure 3, it shows the identifier bit-length associated to each line. We observe that the identifiers of **rand** setup quickly increase while the **hash** identifiers remain similar to the ones in the experiment made for one user in [6]. Consequently, the **hash** setup is the most suitable setup in the context of distributed collaborative editing.

Reasons: both setups use *boundary+* and *boundary–* allocation strategies. However, each collaborator in the **rand** setup makes independent choices of allocation strategies when required. Thus, if a collaborator chooses a particular strategy, and another user chooses the antagonist strategy, then a large number of identifiers is wasted when their operations are delivered to each other. On the other hand, when the same hash function spread over all collaborators generates the same strategy choices, it keeps the random behaviour locally and also makes an implicit agreement on which strategy to employ at any given depth.

4.2 Latency experiment

Objective: show that increasing the latency, i.e., the time between the generation and the delivery of an operation, does not imply a growth in the size of identifiers. On the contrary, when the latency increases, the average size of allocated identifiers decreases. This behaviour is expected for LSEQ and h-LSEQ strategies.

Description: we simulate the production of a 100 lines document by a set of 10 collaborators. Each one of these collaborators performs 10 insert operations at the end of the

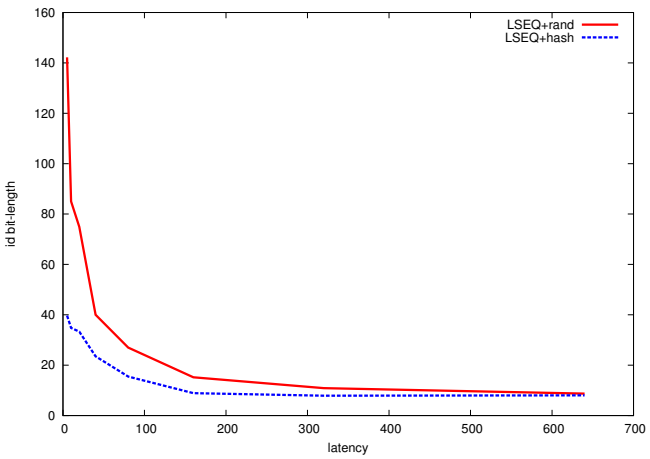

Figure 4: Experiments of latency effects over the average bit-length of identifiers. The x-axis shows the latency variation in number of rounds, i.e., the average time between the send and the receive of an operation. The y-axis shows the average bit-length of identifiers. A group of 10 collaborators generates a synthetic document of 100 lines by performing 10 operations each. Two configurations of LSEQ: the original random strategy choice and the hash strategy choice (h-LSEQ).

document. The users generate operations following the Poisson distribution of parameter $\lambda = 100$, i.e., users more likely generate an operation after 100 rounds (arbitrary unit) from their previous operation. Measurements are done at a latency of 5, 10, 20, 40, 80, 160, 320, 640 rounds. They concern the average bit-length of identifiers in the document. We evaluate two setups: the random strategy choice (**rand**) and the hash strategy choice (**hash**). Both setups use the allocation strategies $boundary+$ and $boundary-$ with $boundary = 10$ and departure $base = 2^{4+depth}$.

Results: Figure 4 shows the results of the latency experiments. As expected, the two setups allocate identifiers with an average bit-length that quickly decreases when the latency increases. Therefore, the latency does not badly affect LSEQ and h-LSEQ allocation strategies. Also, it confirms observations made in the previous experiment: **hash** performs better than **rand**, especially under low latency.

Reasons: increasing the latency delays the arrival of operations to the remote collaborators. Thus, each collaborator works locally until operations generated by other users are delivered. Considering the extreme case with the maximum latency, each collaborator generates his 10 insertions, and then merges the others' 90 operations. Consequently, the 100 operations share the same space, i.e., the digit part of identifiers can be the same for multiple elements, and the total order is maintained by the source and clock.

4.3 Synthesis

The experiments evaluated the effect of concurrency on the average bit-length of identifiers by varying the number of collaborators and the latency. They showed that an implicit and *a priori* agreement on allocation strategy choice is required to maintain the sub-linear upper-bound on space complexity. Employing the same hash function to choose the allocation strategy permits to reach this agreement. Consequently, the improvement of LSEQ called h-LSEQ can be safely used in distributed collaborative editing. The latency experiment shows that increasing the latency parameter only results in sharing the identifier space, and consequently lowers the size of identifiers. As a corollary, considering an immediate delivery of operations constitutes an upper-bound on the behaviour of the overall system for any latency.

5. RELATED WORK

Current distributed collaborative editors use optimistic replication [10, 11] to provide availability of data at a consistency cost. Optimistic replication designs operations in two phases. First a replica prepares the result of an operation and then broadcasts it to other collaborators. Second, remote replicas integrate the newly received data.

Operational Transform approach (OT) and Conflict-free Replicated Data Type approach (CRDT) belong to the optimistic class of replication. While CRDTs share the computational cost of operations between the local and remote parts of the replication protocol, the OT approaches prefer to offer free local generation and pay an additional cost on the integration of remote operations. Since the number of remote operations quadratically grows when the number of collaborators increases, the CRDTs scale better than OT approaches.

We distinguish two classes of CRDTs for sequences. The tombstone based CRDT approaches use death certificate on removed elements. Henceforth, these elements are hidden to the users, but still remain in the underlying model and undermine the performance. For instance, the number of tombstones in popular or controversial pages in Wikipedia pages would be very high due to vandalism and recoveries. The representatives of this class of CRDTs are WOOT [7], WOOTO [16], WOOTH [1], CT [3], Treedoc [8], PPS [18], RGA [9] and [19].

To get rid of these marked elements, the tombstone based CRDTs require additional protocols related to garbage collecting mechanisms. In [4, 9] they propose approaches to safely purge the tombstones. However, it requires a global knowledge over collaborators. In [9], they describe an approach that obtains the global knowledge by maintaining a vector clock with one entry per participant although it is too costly in large distributed network where collaborators can enter and leave the system. In [4], they describe the *core nebula* approach that permits to reach the consensus over participants. However, this approach constrains the network topology to make the consensus reachable and uses a costly catch up protocol.

The alternative to tombstones is the variable-size identifiers CRDT approach. These CRDTs directly encode the total order of elements into their unique identifiers. Thus, they do not require tombstones. However, the identifiers can grow. In Logoot [17] and Treedoc [8], identifiers have a linear space complexity compared to the number of insert operations. Furthermore, their size heavily depends on the position of insertions. For example, performing insertions repetitively at the beginning of a document leads to identifiers with a size equals to the number of insert operations. Consequently, such approaches require an additional re-balance protocol, like tombstone CRDTs. However, their underlying allocation strategies can be replaced by LSEQ [6] that provides a sub-linear upper-bound on the space com-

plexity regardless of the editing behaviour.

The LSEQ approach uses two antagonist allocation strategies. Although paper [1] argues about the importance of concurrency in CRDTs experiments, the original paper of LSEQ does not consider this aspect. In this paper, we showed on synthetic documents that its original configuration does not scale in the number of users. The added value of synthetic documents over real collaborative documents is the total flexibility on their parameters, especially on the editing behaviour. Indeed, the number of available real collaborative documents (including concurrency) is very limited. Moreover, such experiments are costly to set up, and often biased by the nature of the task to perform.

In this paper, we propose to replace the random strategy choice component of LSEQ by a hash-based choice strategy to reach an implicit consensus on which strategy to employ. This configuration called h-LSEQ, does not require additional computation and extends the sub-linear complexity bound of LSEQ from a single user to multiple collaborators regardless of the latency. Thus, h-LSEQ constitutes a safe allocation strategy for sequences CRDTs, even in concurrent cases.

6. CONCLUSION

In this paper, we presented an improvement of the allocation strategy LSEQ called h-LSEQ. Contrarily to the original approach, our configuration supports multiple users regardless of the latency. Indeed, replacing the random strategy choice by the hash-based choice strategy allows to reach a global sub-linear upper-bound on space complexity. As a consequence, distributed collaborative editors can safely use h-LSEQ and show better scalability than current trending editors such as Google Docs, Etherpad, etc.

The hash-based choice strategy is a common function over participants that maps a depth in the underlying tree to an allocation strategy identifier. This surjective function initialized with a shared seed allows to reach an agreement on strategies with no additional synchronization cost. Consequently, there is no loss in the identifiers' space due to antagonist allocation strategies.

This paper highlights the importance of multiple users analysis in CRDTs for sequences, particularly in the case of multiple underlying allocation strategies. We also showed that latency does not badly affect the size of identifiers in variable-size identifiers CRDTs. However, we performed experiments with synthetic documents in order to show the general behaviour of the allocation without considering semantically consistent documents.

Future works include the formal proof of the sub-linear upper-bound on space complexity of h-LSEQ. A probability analysis of the worst-case scenario is mandatory to show that it seldom happens. We plan to experiment h-LSEQ on a corpus of real documents including multiple collaborators and concurrency. Finally, we aim to build a real distributed collaborative editor based on a variable-size identifiers CRDT using h-LSEQ as allocation strategy.

References

[1] M. Ahmed-Nacer, C.-L. Ignat, G. Oster, H.-G. Roh, and P. Urso. Evaluating CRDTs for Real-time Document Editing. In ACM, editor, *11th ACM Symposium on Document Engineering*, pages 103–112, Mountain View, California, États-Unis, Sept. 2011.

[2] A. Andersson and M. Thorup. Dynamic ordered sets with exponential search trees. *J. ACM*, 54(3), June 2007.

[3] 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 '10, pages 3:1–3:10, New York, NY, USA, 2010. ACM.

[4] M. Letia, N. Preguiça, and M. Shapiro. Crdts: Consistency without concurrency control. *Arxiv preprint arXiv:0907.0929*, 2009.

[5] 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*, pages 111–120. ACM, 1995.

[6] B. Nédelec, P. Molli, A. Mostefaoui, and E. Desmontils. LSEQ: an Adaptive Structure for Sequences in Distributed Collaborative Editing. In ACM, editor, *13th ACM Symposium on Document Engineering*, Sept. 2013.

[7] G. Oster, P. Urso, P. Molli, and A. Imine. Data consistency for p2p collaborative editing. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 259–268. ACM, 2006.

[8] N. Preguiça, J. M. Marquès, M. Shapiro, and M. Letia. A commutative replicated data type for cooperative editing. In *Distributed Computing Systems, 2009. ICDCS'09. 29th IEEE International Conference on*, pages 395–403. Ieee, 2009.

[9] H.-G. Roh, M. Jeon, J.-S. Kim, and J. Lee. Replicated abstract data types: Building blocks for collaborative applications. *Journal of Parallel and Distributed Computing*, 71(3):354–368, 2011.

[10] Y. Saito and M. Shapiro. Replication: Optimistic Approaches. technical report, 2002.

[11] Y. Saito and M. Shapiro. Optimistic replication. *ACM Comput. Surv.*, 37(1):42–81, Mar. 2005.

[12] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski. A comprehensive study of Convergent and Commutative Replicated Data Types. Rapport de recherche RR-7506, INRIA, Jan. 2011.

[13] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski. Conflict-free replicated data types. *Stabilization, Safety, and Security of Distributed Systems*, pages 386–400, 2011.

[14] C. Sun and C. Ellis. Operational transformation in real-time group editors: issues, algorithms, and achievements. In *Proceedings of the 1998 ACM conference on Computer supported cooperative work*, CSCW '98, pages 59–68, New York, NY, USA, 1998. ACM.

[15] 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 (TOCHI)*, 5(1):63–108, 1998.

[16] S. Weiss, P. Urso, and P. Molli. Wooki: A p2p wiki-based collaborative writing tool. In B. Benatallah, F. Casati, D. Georgakopoulos, C. Bartolini, W. Sadiq, and C. Godart, editors, *Web Information Systems Engineering – WISE 2007*, volume 4831 of *Lecture Notes in Computer Science*, pages 503–512. Springer Berlin Heidelberg, 2007.

[17] S. Weiss, P. Urso, and P. Molli. Logoot: a scalable optimistic replication algorithm for collaborative editing on p2p networks. In *Distributed Computing Systems, 2009. ICDCS'09. 29th IEEE International Conference on*, pages 404–412. IEEE, 2009.

[18] Q. Wu, C. Pu, and J. Ferreira. A partial persistent data structure to support consistency in real-time collaborative editing. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pages 776–779, 2010.

[19] W. Yu. A string-wise crdt for group editing. In *Proceedings of the 17th ACM international conference on Supporting group work*, GROUP '12, pages 141–144, New York, NY, USA, 2012. ACM.