2009 29th IEEE International Conference on Distributed Computing Systems

Logoot : a Scalable Optimistic Replication Algorithm for Collaborative Editing on P2P Networks

Stéphane Weiss, Pascal Urso and Pascal Molli
Nancy-Université
LORIA
Campus Scientifique Vandoeuvre-lès-Nancy
{weiss,urso,molli}@loria.fr

Abstract

Massive collaborative editing becomes a reality through leading projects such as Wikipedia. This massive collaboration is currently supported with a costly central service. In order to avoid such costs, we aim to provide a peer-to-peer collaborative editing system. Existing approaches to build distributed collaborative editing systems either do not scale in terms of number of users or in terms of number of edits. We present the Logoot approach that scales in these both dimensions while ensuring causality, consistency and intention preservation criteria. We evaluate the Logoot approach and compare it to others using a corpus of all the edits applied on a set of the most edited and the biggest pages of Wikipedia.

1. Introduction

Collaborative editing (CE) systems allow distant users to modify the same data concurrently. The major benefits are: reducing task completion time, getting different viewpoints, etc... Wikis, online office suites and version control systems are the most popular collaborative editing tools.

Several collaborative editing systems are becoming massive: they support a huge number of users to obtain quickly a huge amount of data. For instance, Wikipedia is edited by 7.5 million of users and got 10 million of articles in only 6 years. However, most of CE systems are centralized with costly scalability and poor fault tolerance. For instance, the Wikimedia Foundation spent 2.7 million dollars between 2007 and 2008 for maintaining wiki servers[1]. To overcome these limitations, we aim to provide a peer-to-peer (P2P) CE system.

P2P systems rely on replication to ensure scalability. A single object is replicated a limited number of times in structured networks (such as Distributed Hash Tables) or a unbounded number of times in unstructured P2P networks. In all cases, replication requires to define and maintain consistency of copies. With a limited number of replicas, it is possible to maintain strong consistency models such as sequential consistency. For instance, some P2P replication systems are based on consensus algorithm [1]. However, if the number of replicas grows, the communication cost becomes too expensive. Collaborative editing can rely on weaker consistency criteria that generate less traffic and that are more efficient. For instance, Git [2] distributed version control system relies on causal consistency, Usenet [3] on eventual consistency, and CoWord [4], a real-time editing systems relies on CCI consistency .

CCI consistency have been proven suitable for replicated collaborative system [5]. CCI consistency means Causality, Convergence, and Intention preservation. Thus, CCI consistency implies causal consistency and eventual consistency. Intention preservation means that an operation effect observed on a copy, must be observed in all copies whatever any sequence of concurrent operations applied before.

Many algorithms have been proposed for maintaining CCI consistency. Some approaches [6], [7] do no support P2P constraints such as churn. The others [8], [9], [10] rely on data "tombstones". In these approaches, a deleted object is replaced by a tombstone instead of removing it from the document model. Tombstones cannot be directly removed without compromising the document consistency. Therefore, the overhead required to manage the document grows continuously.

In this paper, we present a new optimistic replication algorithm called Logoot that ensures CCI consistency for linear structures, that tolerates a large number of copies, and that does not require the use of tombstones. This approach is based on non-mutable and totally ordered object position identifiers. The time complexity of Logoot is only logarithmic according to the document size. We validate the Logoot algorithm with real data extracted from Wikipedia. In this paper, we show and analyze the results of this experiment.

2. P2P Collaborative Editing System

We make the following assumptions about P2P Collaborative Editing Systems (P2P CE) and their correction criteria.

1. http://wikimediafoundation.org/wiki/Donate/Transparency/en

1063-6927/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDCS.2009.75

A P2P CE network is composed by a unbounded set of peer P. Objects edited by the system are replicated on set of replicas R (with $0 < |R| \leq |P|$). Each replica has the same role and is hosted on one peer. A peer can host many replicas, each one of a different object. Peers can enter and leave the network arbitrary fast. We assume that each peer possesses a unique comparable site identifier.

The modifications applied on a replica are eventually delivered to all other replicas. We make no assumption about the kind of dissemination routine through the P2P network or the propagation time of modifications. When a modification is delivered to a replica, the modification is applied. Thus, the replica diverge in the short term. This kind of replication is known as optimistic replication [11] (or lazy replication).

According to [12], a collaborative editing system is considered correct if it respects the CCI criteria:

Causality: This criterion ensures that all operations ordered by a precedence relation, in the sense of the Lamport's *happened-before* relation [13], will be executed in the same order on every copy (causal consistency).

Convergence: The system converges – i.e. all replicas are identical – when the system is idle (criteria also known as eventual consistency).

Intention: The expected effect of an operation must be observed on all replicas. One definition of operations intention for textual documents is :

delete A line is eventually removed from the document if and only if it has been deleted on, at least, one replica.

insert A line inserted on a replica eventually appears on every replica. Moreover, the order relation between the document lines and the newly inserted line must be preserved on every replica (as long as these lines exist).

Along these criteria, we add a numerical scalability criteria from [14].

Scalability: The system must handle the addition of users or objects without suffering a noticeable loss of performance.

3. Related Work

In this section, we present the optimistic replication approaches that are known to scale according to the number of users in the network.

WOOKI [9] is a P2P wiki system based on Wooto, an optimization of Woot [15]. The main idea of Woot is to treat a collaborative document as a Hasse diagram that represents the order induced by the *insert* operations. Therefore, the Wooto algorithm computes a linear extension of this diagram. WOOKI barely respects the CCI correction criteria. Indeed, the causality is replaced by preconditions. As a result, the happened-before relation can be violated in some cases. The convergence is ensured by the algorithm by using tombstones.

TreeDoc [10] is a collaborative editing system which uses a binary tree to represent the document. Deleted lines are also kept as tombstones. The authors propose a kind of "2 phase commit" procedure to remove tombstones. Unfortunately, this procedure cannot be used in an open-network such as P2P environments. However, this approach proposes also an interesting general framework called Commutative Replicated Data Type (CRDT) to build distributed collaborative editors ensuring CCI criteria.

[7] proposes a distributed optimistic replication mechanism in the CRDT framework, that ensures the CCI criteria but using tombstones and vector clocks. Vector clocks (aka state vector) have a size proportional to the number of replicas in the network, and thus are not scalable when the number of users grow.

The Operational Transformation approach [16], [17], [5], [18] is a framework for building distributed collaborative editor. Except MOT2 [8], all algorithms in this framework require the use of vector clocks and, thus, do not scale. MOT2 is a P2P peer-wise based reconciliation algorithm. This algorithm assumes the existence of transformation functions satisfying some properties. To our best knowledge, the only transformation functions for text document adapted [19] for MOT2 are the Tombstone Transformation Functions [20] which are based on tombstones.

Thus, all of the above approaches that are usable on P2P networks are based on tombstones. According to the scalability definition, the tombstone cost is not acceptable on massive editing systems. For instance, for the most edited pages of Wikipedia[2], the tombstone storage overhead can represent several hundred times the document size. Tombstones are also responsible of performance degradation. Indeed, in all published approaches, the execution time of modification integration depends on the whole document size – including tombstones. Therefore, letting the number of tombstones growing degrades the performance.

4. Proposition

Our idea is based on the CRDT [10] framework for collaborative editing. In the CRDT framework, modifications produced locally are re-executed on remote replicas. There is no total order on operations, thus, concurrent operations can be re-executed in different orders. The main idea of this framework is to use a data type where all concurrent operations commute. Combined with the respect of the causality relationship between operations, this commutation ensures the convergence criteria.

To achieve commutativity on a linear structure, the authors propose a solution based on a total order between elements

2. http://en.Wikipedia.org/wiki/Wikipedia:Most_frequently_edited_articles

in the document. More precisely, there is two kinds of modification :

- $insert(pid, text)$ that inserts the line content $text$ at the position identifier pid.
- $delete(pid)$ that removes the line at the position identifier pid.

In the original paper, a tree structure is introduced to maintain the total order between positions identifier. However, safely removing elements from this tree requires tombstones. In [7], authors refer also to the CRDT framework but use vector clocks to obtain this order.

Our idea is to use a position identifier based on a list of integers for each line. With such an identifier, a line can be removed from the document model without affecting the order of the remaining lines.

4.1. Logoot model

A Logoot document is composed by *lines* defined by: $\langle pid, content\rangle$ where $content$ is a text line and pid a unique *position identifier*. There is two virtual lines called l_B and l_E to represent the beginning and the ending of the document.

The main idea to insert a *line* is to generate a new position A such as $P \prec A \prec N$ where P is the position of the previous line and N the position of the next line.

```
1 ⟨pid_0, l_B⟩
2 ⟨pid_1, "This is an example of a Logoot document" ⟩
3 ⟨pid_2, "Here, pid_1 ≺ pid_2 " ⟩
4 ⟨pid_3, "And pid_2 ≺ pid_3 " ⟩
5 ⟨pid_∞, l_E⟩
```

To allow operations to commute, position identifiers must be unique. Also, since a user can always insert a line, we must be able to generate a position A such as $P \prec A \prec N$ for any P and N.

In the following definition we assume that each site maintains a persistent logical clock $clock_s$ incremented each time a line is created.

Definition 1.
- *An* identifier *is a couple* $\langle pos, site\rangle$ *where* pos *is an integer and* $site$ *a site identifier.*
- *A* position *is a list of identifiers.*
- *A* position identifier *generated by a replica* s *is a couple* pos, h_s *where* $pos = i_1.i_2.\ldots.i_n.\langle p, s\rangle$ *is a position and* h_s *is the value of* $clock_s$.

Thus, every position identifier is unique since the last identifier of the list $i_1.i_2.\ldots.i_n.\langle p, s\rangle$ which contains the unique site identifier and the value of the logical clock of this site.

To obtain a total order between positions, we use the following definition.

Definition 2.
- *Let* $p = p_1.p_2\ldots p_n$ *and* $q = q_1.q_2\ldots q_m$ *be two positions, we get* $p \prec q$ *if and only if* $\exists j \leq m.\ (\forall i < j.\ p_i = q_i) \wedge (j = n+1 \vee p_j <_{id} q_j)$
- *Let* $id_1 = \langle pos_1, site_1\rangle$ *and* $id_2 = \langle pos_2, site_2\rangle$ *be two identifiers, we get* $p_1 <_{id} p_2$ *if and only if* $pos_1 < pos_2$ *or if* $pos_1 = pos_2$ *and* $site_1 < site_2$.

We only compare positions – and not logical clocks – since there can not be, in the same model, two lines with the same position (see lemma 3).

Finally, the logical view of a Logoot document looks like:

```
1 ⟨⟨0, 0⟩, NA, l_B⟩
2 ⟨⟨1, 1⟩, 0, "This is an example of a Logoot document" ⟩
3 ⟨⟨1, 1⟩.⟨1, 5⟩, 23, "How to find a place between 1 and 1" ⟩
4 ⟨⟨1, 3⟩, 2, "This line was the third made on replica 3" ⟩
5 ⟨⟨MAXINT, 0⟩, NA, l_E⟩
```

4.2. Modifying a Logoot document

On collaborative editing systems such as wiki or VCS, an edit on a document is not a single operation but a set of operations (a patch). Lines inserted by a patch are often contiguous. Thus, to apportion the line positions, we define the generateLinePosition(p, q, N, s) function which generate N positions between a position p and a position q using numbers in base $MAXINT$ (the maximum of the unsigned integer plus 1). To obtain short positions, it firstly select the smallest equal length prefixes of p and q spaced out at least N. Then it apportions randomly the constructed positions.

```
1  function generateLinePositions(p, q, N, s)
2
3  list := {};
4  index := 0;
5  interval := 0;
6
7  while (interval < N)
8    index++;
9    interval := prefix(q, index) − prefix(p, index);
10 endwhile
11 step := interval / N;
12 r := prefix(p, index);
13 for j:=1 to N do
14   list.add(constructPosition(r + Random (1, step),p,q,s));
15   r := r + step;
16 done
17 return list;
```

The function prefix$(p,\ i)$ returns a number in base $MAXINT$ which each digits is $p_i.pos$ the integers of the first i^{th} identifiers of p (filled with 0 if $|p| < i$). The function constructPosition$(r,\ p,\ q,\ s)$ constructs a position $\langle\langle r_1, s_1\rangle.\langle r_2, s_2\rangle.....\langle r_n, s_n\rangle\rangle$ where r_i is the i^{th} digit of r. We use the following rules to define each s_i: 1) if $i = n$ then $s_i = s$, 2) else if $r_i = p_i.pos$ then $s_i = p_i.site$, 3) else if $r_i = q_i.pos$ then $s_i = q_i.site$ 4) else $s_i = s$

For instance, on a site s, insertion positions between $p = \langle\langle 2, 4\rangle\rangle$ and $q = \langle\langle 10, 5\rangle\langle 20, 3\rangle\rangle$ are apportioned in the set

- $\{\langle\langle x, s\rangle\rangle | x \in]2, 10[\}$ if $N < 8$

- $\{\langle\langle 2,4\rangle\langle x,s\rangle\rangle | x \in [0, MAXINT[\} \vee$
  $\{\langle\langle y,s\rangle\langle x,s\rangle\rangle | x \in [0, MAXINT[, y \in]2,10[\} \vee$
  $\{\langle\langle 10,5\rangle\langle x,s\rangle\rangle | x \in [0,20[\}$ if $N \geq 8$

To choose the value of the value of the integer in the position, any arbitrary choice can be made. However, to restrain two different replicas to generate concurrently the same choice, and thus to reduce the grow rate of the position list, we apply a random function.

To delete a line, we simply generate a delete operation which contains the position identifier of this line. Then, we can completely remove this line from the document.

4.3. Integrating remote modifications

Both line insertion and removal can be integrated in a logarithmic time according to the number of lines in the document and constant according to the number of the user in the P2P network. Indeed, we simply use the binary search algorithm to find the position in the document corresponding to the position identifier.

Also, the integration of a delete operation can safely remove the line from the document model, since the total order between remaining lines is not affected. Moreover, this removal will free a position identifier that can be reused. This mechanism reduces the growing rate of position identifier as shown in section 5.

4.4. Correctness of the approach

To ensure convergence in the CRDT framework, concurrent operations must commute. If line positions are unique, non mutable, and totally ordered, the different replicas can apply any series of insert operations in any order and obtain the same result.

The following lemma states that there cannot be two different lines with the same position on one model.

Lemma 1. *If causality is preserved, the position of a line is unique on each model.*

Proof: The last element of the line position contains the unique identifier of the site which generates the line. As a result, two different replicas cannot generate the same position.

A replica can only generates a position different from every other position in its model.

A replica can generate an insert operation o_i of line l_2 with the same position than a line l_1 it has previously generated. However, this is possible only if l_1 was previously deleted on that replica by a (remote or local) delete operation o_d. Then, we have the following happens before relationship $o_d \rightarrow o_i$. Thus, if causality is preserved l_2 can only be inserted on a replica where l_1 was deleted. □

The following theorem states that Logoot ensures consistency criteria. Its correctness is based on the CRDT correctness proof of [10].

Theorem 2. *If causality is preserved, Logoot ensures consistency.*

Proof: Since the couple composed of a site identifier and a clock value is unique, each position identifier is unique.

According to Lemma 3, each position is unique, and thus position identifier are totally ordered.

Finally, since Logoot position identifiers are unique, non mutable and totally ordered, every couple ($insert/insert$, $insert/delete$ and $delete/delete$) of concurrent operations commutes. Thus, Logoot data type is a CRDT. □

The following theorem states that Logoot respects the intentions of the insert and delete operations as defined Section 2.

Theorem 3. *If causality is preserved, Logoot ensures intentions.*

Proof: Since position identifiers are unique and non mutable, delete operation intention is respected.

According to Lemma 3, each position is unique. Thus, the Logoot is always able to compute a new position between two lines. Since positions are non-mutable and totally ordered, the line order observed on the generation replica will be preserved on each other replica. □

4.5. P2P constraints

In order to be deploy on a P2P network, our approach needs to satisfy some constraints. It must scales in terms of peers and support the churn of the network (i.e. peers which enter and leave the network arbitrarily fast).

Logoot position identifiers support these constraints since their size and space complexity are constant according to peers number (no vector clock). We only assume that each site has a unique identifier. Additionally to position identifiers, Logoot only require a causal dissemination mechanism. To obtain it, a scalable broadcast such as the lightweight probabilistic broadcast [21] in association with causal barriers [22] can be used.

Also, the Logoot framework supports network churn since it does not require any group membership mechanism or consensus algorithm. It also does not require to know the number of peers.

The following section discuss about Logoot scalability in terms of number of edits.

5. Evaluation

The size of Logoot position identifiers is unbounded. Theoretically, position identifiers can grow each time a line

is inserted. Thus, if no line is ever removed, the size of the document model overhead can be $\sum^{n} i = O(n^2)$ where n is the total number of inserted lines. However, due to the randomized nature of the algorithm, such a worst case can only arrive with a probability equal to $1/MAXINT^n$ which is negligible. In practice, lines are often removed and position identifier are apportioned, thus the overhead size remains low.

In order to effectively measure the Logoot overhead, we have replayed the modifications made on some Wikipedia pages in a Logoot document. For instance, in the proportionally worst result of our test bed (case 2, Figure 4), their is a total of 43352 identifiers, to be compared with the number of inserted lines $n = 623863$.

On another hand, in tombstones-based approaches, the size of the document model is also unbounded. The Wooto and Treedoc approaches have a constant overhead for each inserted line, thus strictly proportional to n. Comparing to the number of tombstones, the size of each position identifier remains low. Our approach is sightly less efficient only in a specific case (see Section 5.2.3).

5.1. Methodology

In our implementation, we use 8-bytes integers for site identifiers and positions, and 4-bytes integers for the logical clock, hence, a position identifier contains at least 20 bytes.

For the Wooto approach, we use an 8-bytes integer for the site identifiers and 4-bytes integers for the logical clock and the degree. Then, the overhead for each line is 16 bytes.

About the TreeDoc approach, we do not consider the tree overhead, we only count one 8-bytes integer for the site identifier and one 4-bytes integer for the counter. Finally, the overhead for each line is 12 bytes.

To replay the histories of some Wikipedia pages, we use the MediaWiki API[3] to obtain an XML file containing several revisions of a specific Wikipedia page. Then, using a diff algorithm [23], we compute the modifications performed between two revisions. Modifications are simply re-executed in our model. Since our approach generates each position randomly, we re-executed ten times each page history to obtain average values.

We also measure the overhead for Wooto and TreeDoc. The result obtained for TreeDoc does not take account of the "stabledel" and "gc" procedures which aim to remove tombstones. We motivate this choice by the fact that these procedures require to know the exact number of replicas which is unknown, unbounded and unstable in P2P networks.

The overhead of Wooto and TreeDoc are directly computed from the number and the type of operations performed on the document. Indeed, their overhead are directly proportional to the number of inserted lines in the document since deleted lines remain as tombstones.

We have applied this schema on the top pages of three categories[4] :

- The most edited encyclopedic pages,
- The most edited pages,
- The biggest pages.

For each of the treated pages, we present the average – over the last 100 edits – overhead of the Logoot, Wooto and Treedoc approaches. We present the average size of the page and the number of patches (i.e. edits on the page).

5.2. Results

Figure 1 shows the relative (size of the overhead divided by the size of the visible document on a logarithmic scale) overhead of the three different approaches on the most edited encyclopedic page of the English Wikipedia. Figure 2 shows the absolute overhead. The Logoot overhead remains constant all along the editing session, while the overhead of tombstones-based approaches continuously grows.

Finally, the Logoot overhead is inferior to the document size while tombstones-based approaches require more than 100 times the document size and continuously grows. While the "George W. Bush" page contains only about 553 lines, the number of deletions is about 1.6 million. As a consequence, tombstones-based systems are not well-suited for such documents since we obtain 1.6 million tombstones for only 553 lines.

Most of the modifications done on Wikipedia pages consists in updating the content of some existing lines. To ensure user's intentions, distributed editing systems handle such an update as deleting the old content and inserting the new content. Thus, the number of tombstones grows quickly.

Also, the figure 1 shows several peaks which are mainly due to vandalism acts. Indeed in some of the most edited encyclopedic pages of Wikipedia, there is a lot of vandalism acts done by users, including erasing the whole content of the page. Every vandalism is reverted by re-introducing the previously erased content or removing malicious content introduced. This process adds each time a lot of tombstones (up to the page size). Introducing a specific undo mechanism that reuses tombstones such as [19] should reduce the overhead due to tombstones.

5.2.1. Most edited encyclopedic Pages. The Figures 3, 4 and 5 show the average relative overhead (i.e. the size of the overhead divided by the size of the visible document)

3. http://www.mediawiki.org/wiki/API:Query_-_Properties#revisions_.2F_rv

4. According to http://en.Wikipedia.org/wiki/Wikipedia:Most_frequently_edited_pages and http://en.Wikipedia.org/wiki/Special:LongPages on end of November 2008. However, due to some technical issues (i.e. invalid characters, missing patch, ...), we skipped some of the top pages, but the first page of each category is presented.

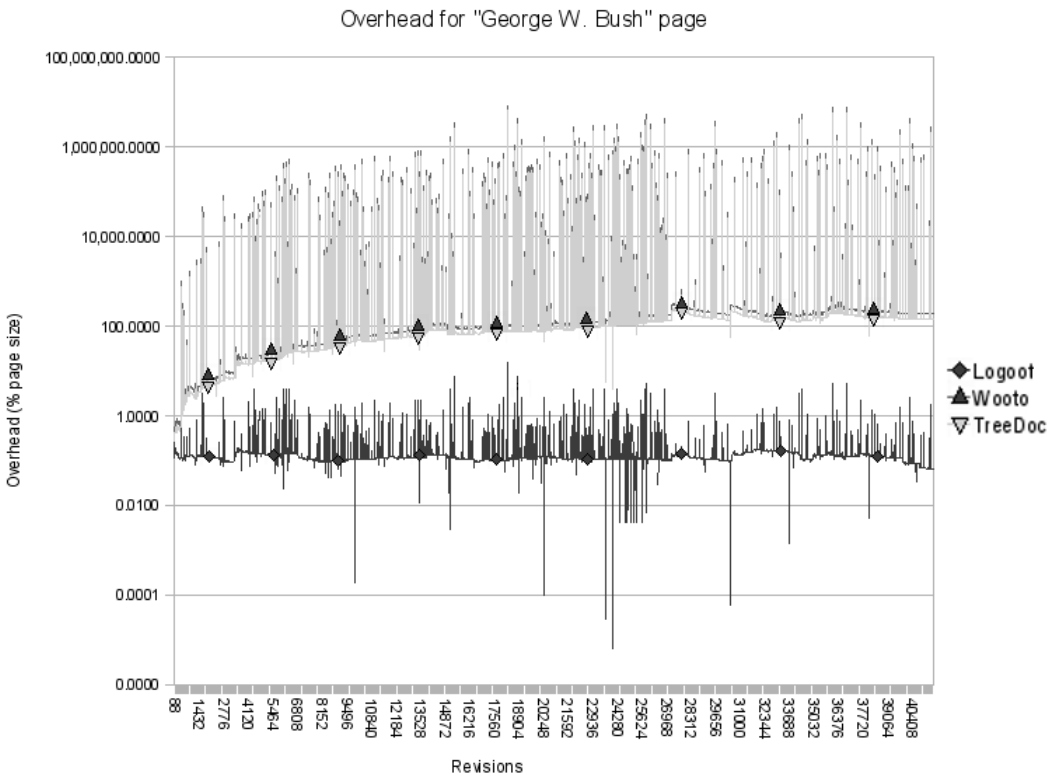

Figure 1. Relative overhead for “George W. Bush” page.

computed on the 100 last revisions of each page for the approaches Logoot, Wooto and TreeDoc. The column “Size” indicates the average size of each pages for the 100 last revisions. Finally, in the column “Number of Patches”, we show the number of edits done on each pages.

The Figure 3 presents the results obtained on the most edited encyclopedic pages. The histories of such pages show a lot of edits as well as vandalism acts. A huge number of deletions has been performed on such pages, hence, tombstones-based approaches show an important overhead. On the contrary, the Logoot overhead remains low.

5.2.2. Most edited Pages. These pages (Figure 4) are discussion pages or special pages mostly edited by bots. In such pages, there is no or very few vandalism acts but a lot of edits.

For all these pages, the Logoot approach is more efficient than tombstones-based approaches. However, we can notice that the difference is far more important for pages were data are very volatile for instance like case 1 (a communication channel to detect and block vandals) or like case 4 (a sandbox). The other cases represent discussion pages. Users ask questions, and other users reply by modifying the page. Each topic is removed after one week. They are edited in the same way : mostly adding content at the end of the page and removing week old topics content at the beginning. Thus, there is, in these pages, a lot of tombstones but the Logoot position identifiers grow as well.

5.2.3. Biggest Pages. These pages (Figure 5) are often lists of elements. If these lists are always edited in the same way (for instance adding elements at the end of the page), they represent the worst cases for our approach. Indeed, the Logoot position identifier will grow the quickest, especially if insertions are done in many different occasions. Effectively, in cases 4, 9, and 10, our approach is less efficient than tombstones-based approaches.

However, these results show that our approach is in average, even in these disadvantageous real cases, less costly than tombstones-based approaches.

5.3. Limits of the experimentation

Since Wikipedia uses a centralized wiki, we can expect a slightly different behavior in a P2P system. For instance, Wikipedia reduces the impact of concurrent modifications. In a P2P environment, preventing users to make concurrent modifications is not a realistic hypothesis. Therefore, concurrent modifications are automatically merged. This will certainly produce an “inconsistent” document which requires

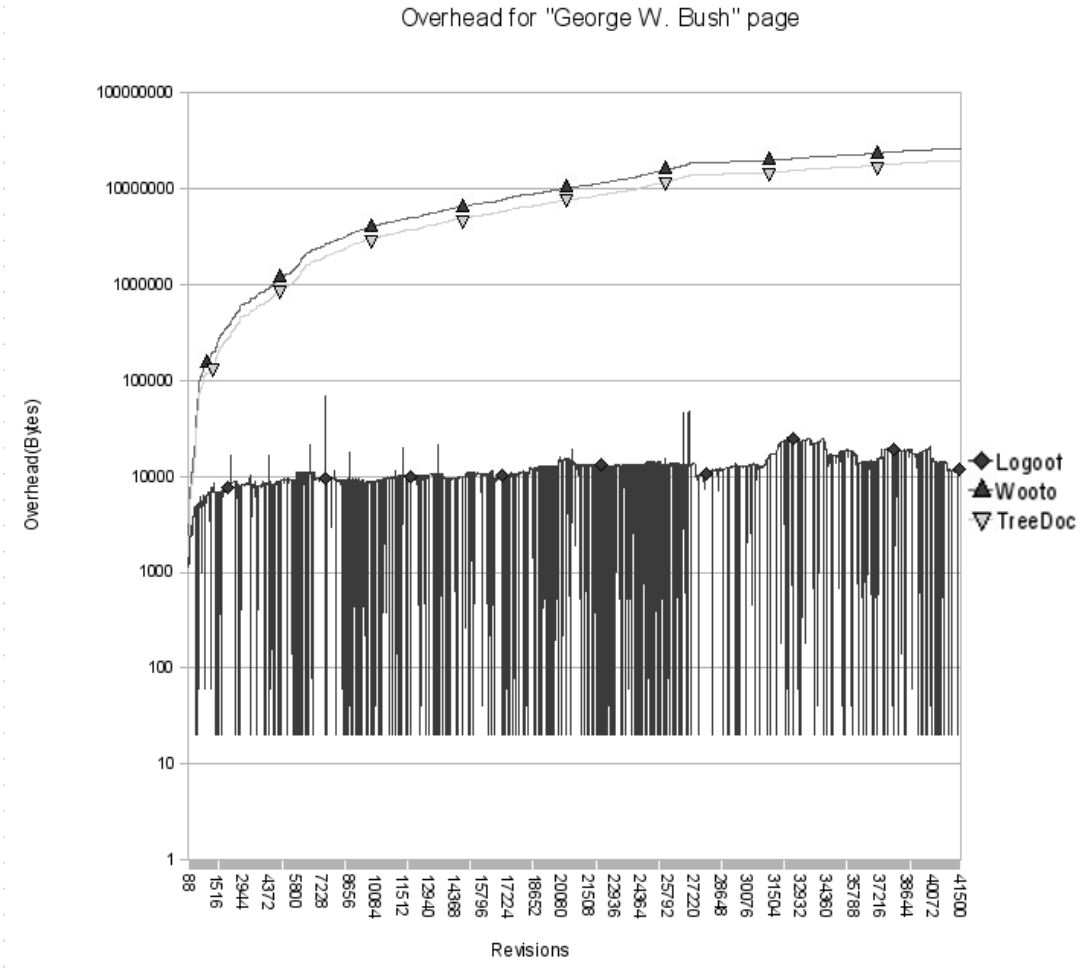

Figure 2. Absolute overhead for “George W. Bush” page.

the intervention of some user to correct it. Therefore, the number of edits will certainly be more important in a P2P wiki than in a centralized wiki.

In Wikipedia, some pages are protected to reduce the number of vandalism acts. However, such protection mechanism is not compatible with P2P constraints. Therefore, in a P2P wiki, the number of vandalism acts is certainly more important. Therefore, we expect to obtain more edits and vandalism acts on a P2P wiki system.

Contrary to the Woot approach, CRDT approaches, including ours, requires a causal broadcast to achieve convergence. However, a causal delivery implies an overhead on each message sent by each replica. The two main mechanism to achieve a causal delivery are vector clocks [24] and causal barriers [22]. Vector clocks are not usable in P2P networks since their sizes are proportional to the number of replica. Causal barriers have a smaller size, that depend only on the degree of concurrency of the operations in the network. On collaborative editing system, this degree remains low : less than 3 edits per second on the whole English Wikipedia in average[5]. However, a realistic measure of the communication overhead can only be achieved with a corpus of concurrent collaborative editions.

6. Conclusion

In this paper, we have presented the Logoot algorithm. Logoot is an optimistic replication algorithm that ensures CCI consistency on linear structures. Logoot can be used on structured or unstructured P2P networks. It does not require tombstones. Therefore, the space overhead remains linear during the life of the document and no garbage collector is required.

We validated the logoot algorithm on a corpus extracted from Wikipedia. The experimentation demonstrates that the Logoot unbounded list of identifiers associated to each line

5. 2.69 in October 2008 according to http://en.wikipedia.org/wiki/Wikipedia:Statistics

	Pages	Overhead (in percent)			Number of Patches	Size (in bytes)
		Logoot	Wooto	TreeDoc		
1	George W. Bush	8.33	16128.75	14590.79	41563	133146
2	List of World Wrestling Entertainment employees	39.24	8413.41	6310.05	27152	16673
3	United States	8.30	5875.07	4406.31	24781	158242
4	Jesus	9.83	4179.09	3134.32	20271	125669
5	2006 Lebanon War	13.62	927.12	695.34	17780	139458
6	Islam	15.92	2996.30	2247.22	15315	101278
7	Roman Catholic Church	5.92	1129.51	847.13	14378	170380
8	Deaths in 2006	18.51	1747.24	1310.43	14029	21880
9	Canada	17.88	4431.19	3323.39	13992	112589
10	Akatsuki (Naruto)	9.81	389.89	292.42	13929	60638
	Average	14.74	4621.76	3715.74	20319	106639

Figure 3. Most edited encyclopedic pages

	Pages	Overhead (in percent)			Number of Patches	Size (in bytes)
		Logoot	Wooto	TreeDoc		
1	Wikipedia: Administrator intervention against vandalism	27.78	287530.03	215647.52	438330	2369
2	Wikipedia: Reference desk/Miscellaneous	520.21	7492.31	5619.23	148283	133204
3	Wikipedia: Reference desk/Science	186.14	3431.45	2573.59	142722	190858
4	Wikipedia: Introduction	43.74	4195621.30	3146715.98	132693	317
5	Wikipedia: Help desk	58.11	9266.41	6949.81	126509	96256
	Average	167.20	900668.3	675501.23	197707	1011.98

Figure 4. Most edited pages

	Pages	Overhead (in percent)			Number of Patches	Size (in bytes)
		Logoot	Wooto	TreeDoc		
1	Line of succession to the British throne	23.65	488.30	366.23	3317	376760
2	United States at the 2008 Summer Olympics	52.65	314.71	236.03	2314	314748
3	List of sportspeople by nickname	19.14	82.34	61.75	2332	309576
4	List of Brazilian football transfers 2008	27.08	11.33	8.5	752	287128
5	List of college athletic programs by U.S. State	34.60	48.56	36.42	868	305294
6	List of Chinese inventions	5.11	37.71	28.29	2344	293228
7	List of suicide bombings in Iraq since 2003	13.51	24.55	18.42	1260	215763
8	China at the 2008 Summer Olympics	61.55	134.15	100.61	1552	268720
9	List of urban areas in Sweden	40.04	39.61	29.71	19	108353
10	Table of United States Core Based Statistical Areas	63.55	61.54	46.15	31	252236
	Average	34.09	124.28	93.21	1478.9	320899

Figure 5. Biggest pages

stays acceptable in practice. The experimentation also shows that Logoot has better average performances than the WOOT and Treedoc algorithms.

In the future, we plan to evaluate Logoot overhead in time and to extend it to manage more structured data such as XML documents. We are also working on a group undo feature.

References

[1] T. Schütt, F. Schintke, and A. Reinefeld, "Scalaris: reliable transactional p2p key/value store," in *ERLANG '08: Proceedings of the 7th ACM SIGPLAN workshop on ERLANG*. New York, NY, USA: ACM, 2008, pp. 41–48.

[2] L. Torvalds, "git," (April 2005), http://git.or.cz/.

[3] R. Salz, "InterNetNews: Usenet transport for Internet sites," in *USENIX conference proceedings*. San Antonio, Texas, tats-Unis: USENIX, t 1992, pp. 93–98. [Online]. Available: citeseer.ist.psu.edu/salz92internetnews.html

[4] S. Xia, D. Sun, C. Sun, D. Chen, and H. Shen, "Leveraging single-user applications for multi-user collaboration: the coword approach." in *CSCW*, J. D. Herbsleb and G. M. Olson, Eds. ACM, 2004, pp. 162–171.

[5] C. Sun and C. A. 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*. New York, New York, tats-Unis: ACM Press, Novembre 1998, pp. 59–68.

[6] M. Suleiman, M. Cart, and J. Ferrié, "Concurrent operations in a distributed and mobile collaborative environment," in *Proceedings of the fourteenth International Conference on Data Engineering - ICDE'98*. Orlando, Floride, tats-Unis: IEEE Computer Society, Fvrier 1998, pp. 36–45.

[7] H.-G. Roh, J. Kim, and J. Lee, "How to design optimistic operations for peer-to-peer replication," in *JCIS*, 2006.

[8] M. Cart and J. Ferri, "Asynchronous reconciliation based on operational transformation for p2p collaborative environments," in *CollaborateCom*, 2007.

[9] S. Weiss, P. Urso, and P. Molli, "Wooki: a p2p wiki-based collaborative writing tool," in *Web Information Systems Engineering*. Nancy, France: Springer, December 2007.

[10] M. Shapiro and N. Preguia, "Designing a commutative replicated data type," INRIA, Rapport de recherche INRIA RR-6320, October 2007. [Online]. Available: http://hal.inria.fr/inria-00177693/fr/

[11] Y. Saito and M. Shapiro, "Optimistic replication," *ACM Computing Surveys*, vol. 37, no. 1, pp. 42–81, 2005.

[12] 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)*, vol. 5, no. 1, pp. 63–108, Mars 1998.

[13] L. Lamport, "Time, clocks, and the ordering of events in a distributed system." *Commun. ACM*, vol. 21, no. 7, pp. 558–565, 1978.

[14] B. C. Neuman, "Scale in distributed systems," in *Readings in Distributed Computing Systems*. IEEE Computer Society Press, 1994, pp. 463–489.

[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*. Banff, Alberta, Canada: ACM Press, nov 2006, pp. 259–267.

[16] C. A. Ellis and S. J. Gibbs, "Concurrency control in groupware systems." in *SIGMOD Conference*, J. Clifford, B. G. Lindsay, and D. Maier, Eds. ACM Press, 1989, pp. 399–407.

[17] M. Suleiman, M. Cart, and J. Ferrié, "Serialization of concurrent operations in a distributed collaborative environment." in *GROUP*, 1997, pp. 435–445.

[18] D. Li and R. Li, "An approach to ensuring consistency in peer-to-peer real-time group editors," *Computer Supported Cooperative Work*, vol. 17, no. 5-6, pp. 553–611, 2008.

[19] S. Weiss, P. Urso, and P. Molli, "An undo framework for p2p collaborative editing," in *CollaborateCom*, Orlando, USA, November 2008.

[20] G. Oster, P. Urso, P. Molli, and A. Imine, "Tombstone transformation functions for ensuring consistency in collaborative editing systems," in *The Second International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2006)*. Atlanta, Georgia, USA: IEEE Press, November 2006.

[21] P. T. Eugster, R. Guerraoui, S. B. Handurukande, P. Kouznetsov, and A.-M. Kermarrec, "Lightweight probabilistic broadcast," *ACM Trans. Comput. Syst.*, vol. 21, no. 4, pp. 341–374, 2003.

[22] R. Prakash, M. Raynal, and M. Singhal, "An adaptive causal ordering algorithm suited to mobile computing environments," *J. Parallel Distrib. Comput.*, vol. 41, no. 2, pp. 190–204, 1997.

[23] E. W. Myers, "An o(nd) difference algorithm and its variations," *Algorithmica*, vol. 1, no. 2, pp. 251–266, 1986.

[24] F. Mattern, "Virtual time and global states of distributed systems," in *Proceedings of the International Workshop on Parallel and Distributed Algorithms*, M. C. et al., Ed. Chteau de Bonas, France: Elsevier Science Publishers, Octobre 1989, pp. 215–226.