Consistency Maintenance of Collaborative Shared Documents in Unstable Network Environment

Liping Gao[1,2], Dan Wang[3], Naixue Xiong[1], Bing Wang[4]

[1] Computer Engineering, University of Shanghai for Science &Technology, China
[2] Shanghai Key Laboratory of Data Science, Fudan University, China
[3] Computer Center University of Shanghai for Science &Technology, China
[4] Information Office, University of Shanghai for Science &Technology, China
lipinggao@usst.edu.cn, wangdandjxy@sina.com, nxiong@coloradotech.edu.cn, wbing73@usst.edu.cn

Abstract

Aim to the instability of the mobile internet, the paper proposes an algorithm to achieve the consistency maintenance of the shared documents during the collaborative editing process in the broken network environment. Unlike with the previous studies, we divide network statuses into three types: break-before, break-after and connect-after, in order to record the document and net states, the paper puts forward two new definitions about *cursor* and network statuses. In the break-before phase, the algorithm updates all the collaborative sites' cursors regularly to record the latest consistency point. In the break-after phase, the algorithm packs the operations according to *cursor* and network attributes. While in the connect-after process, the algorithm adapts the address space transformation strategy to resend and execute lost operations as well as constructing necessary performing condition for ABST or other algorithms that support asynchronous cooperation. In this paper, a case analysis is given to describe the execution process of strategy, and correctness proof is also given to validate the whole strategy.

Keywords: Unstable network, Mobile network, CSCW, Consistency maintenance

1 Introduction

The tremendous development of computer technology visually changes people's life. With the popularization of the wireless network, the work territory has stepped over the office to the outside. However, due to the increasing workload and design scale, it is inevitable that people face a series of problems like huge amount of operations and time-consuming, etc. especially when they are solving a complex project only by one person. So the emerge of CSCW (Computer Supported Cooperative Work) [1] technique comes to support the collaborative work of a team.

Real-time group editors allow multiple users in distributed areas to participate in the editing and make modification of the shared data objects. To achieve high responsiveness, the full replication architecture [2-6] is adopted, which suggests to remain a local replica and the local operations is executed immediately after their releases from the user interface. Then, because of the reordering execution processes of the operations, how to transform remote operations or documents to make local replica consistent with other collaborative sites, is a big challenge, which is called consistency maintenance.

There are two optimistic concurrency control methods in the collaborative editing techniques, which are Operational Transformation (OT) [9] and Address Space Transformation (AST) [6, 11]. Almost all the researches before in this filed made the assumption that the network is stable. In recent years, with the development of the mobile devices (such as pad, cell phone, etc.), text editing systems have become popular for these devices. However, its potential weakness of unstable network is against the premise of OT or AST technique, which needs an ideal, non-blocking network environment. It assumes that no operations are lost during the broadcasting process, and all remote operations are received only once time and must be executed correctly. The researchers mainly focus on the horizontal development of the technology, for example, they paid more attention to optimize the efficiency of algorithm or to find new conflicts to solve, omitting a fact that these works must be done in stable network. On the contrary, for the vertical development of the technology, there are fewer researches to mention it, which study and solve the consistency maintenance conflicts that appear in the unstable network. Therefore, the conflict caused by unstable network must be a fatal problem in the development of the collaborative applications especially for the mobile devices.

*Corresponding Author: Liping Gao; E-mail: lipinggao@usst.edu.cn
DOI: 10.3966/160792642019012001001

Shao presented an efficient transformation algorithm ABST [7] (An admissibility-based sequence transformation) algorithm based on OT architecture. Users can work in parallel style during disconnection periods on the local replica, when the net is re-connected, sequences of update operations made by different users are merged to produce a consistent result of the shared documents. It not only improves editing efficiency, but also provides possible technology for consistency maintenance in mobile group editing environment. Yet, ABST just focuses on how to transform the operations while disconnection, ignores the analysis of whether the documents states of the collaborative sites are identical or not. This paper aims to analyze the document status when the network is down, transforms the corresponding operations to build the necessary execution pre-condition of the ABST algorithm, and maintains the document consistency with the ABST and AST algorithms in different network statuses. And the algorithm we provide in this paper totally solve the problem of lost operations in unstable network environment. In other words, the consistency maintenance algorithm we provide in this paper, not only help ABST algorithm construct main precondition in the unstable network, its ability to track and catch lost operations, and handles lost operations to make all collaborative sites' copies same, also help other algorithms that support asynchronous cooperation get important operating precondition (copies of all collaborative sites' are the same).

The rest of this paper is organized as follows: Firstly, section 2 introduces the related works about consistency maintenance techniques. Section 3 makes a brief introduction of ABST and gives the necessary precondition of the algorithm, and proposes the basic concepts and notations to be used in this paper. After that, section 4 presents the consistency maintenance algorithm in the unstable network environment. Section 5 analyzes the efficiency of this algorithm. Section 6 gives an example to describe the whole workflow of the strategy. Section 7 analyzes the correctness of the consistency maintenance algorithm proposed in this paper. Finally, Section 8 concludes the paper and discusses possible future research directions.

2 Related Work

In 1989, Elis [9] proposes the technique of operation transformation firstly and constructed the basic model of OT algorithm. Then a collaborative editing system Grove adopts the dOPT strategy, which is also OT-based. In the Next time, Ressel [15] finds that there may exist inconsistent problem at collaborative sites in some specific conditions, and modifies the previous research work based on OT. Compared to the early consistency model, Sun introduces the CCI [8] (Convergence Causality Intention) model and pointed out an important correct standard—Intention Preservation. Therefore, a lot of consistency maintenance algorithm, which are satisfied with the CCI model are raised, such as adopted [9], GOT [12, 14, 16], GOTO [8], COT [17], SCOT4 [18], LBT [19], TIBOT [20], ABT [21], ABST [7, 22], ABTS [10, 23], etc. Because of the complex transformation rules and poor efficiency, and the difficulty of correctness proof of OT, a new Mark & Retrace based method called AST is put forward, which searches the target position by retracing the document's address space to the state when the operation is generated, instead of transforming the operations. The AST strategy improves the executing efficiency significantly, and its correctness is easier to be verified.

In the mobile network, Shao. [10] puts forward a string-wise consistence maintenance algorithm ABTS which is based the ABT framework. And, in his later work ABTSO is proposed to optimize the executing details of delete operations, which further improves the performing efficiency and practical degree on mobile devices. The ABST algorithm supports mobile and asynchronous collaboration, under the premise of the mobile network status is smooth. It preserves the users' intention according to merge and swap operation sequences between local and remote sites. In 2014, Xia. [13] creates a collaborative mobile commenting system, taking into account the small memory and poor internet connection, which introduces to use the partial replication architecture to replace the full replication architecture. It uses comment nodes to save the detailed contents of user's requests, gets the skeleton node according to the replicating rule, and constructs the skeleton tree as well as the local partial replica finally. Compared to the full replication architecture, it saves the memory and updating time of local replica greatly.

However, the previous consistency maintenance researches in the mobile network, mostly consider the disadvantages of small storage space and poor internet connection. In practice, lost operations may appear caused by the unstable network, while ABST algorithm just deals with the operation sequences generated in no linking network. If there exist some lost operations to make the operation sequence incomplete, the correctness of ABST cannot be guaranteed.

3 The ABST Algorithm

In order to find the necessary precondition of ABST, firstly we overview its basic idea, and analyze whether the precondition can be destroyed by broken network. Besides, we introduce the basic concepts of document, History Buffer and primitive operation, etc., which are related with the consistency maintenance algorithm in this paper.

3.1 ABST Basis

ABST [7] is a novel OT-based algorithm optimized for supporting mobile and asynchronous collaboration. It focuses on how to transform real-time collaboration to asynchronous collaboration and deal with the operations when facing a negative network connection. In simple terms, the algorithm refuses to broadcast local operations to remote sites in the poor network, instead, it packages all the operations, to a request sequence until it meets an optimistic network, and then it broadcasts the sequence to the remote sites. Receivers transform the operation sequence to its execution status and merge it with local history buffer after executed, and then, the HB is split to HBi and HBd, which is composed of the Insert and Delete operations respectively.

Take Figure 1 as the example, two users Alice and Bob are co-authoring a document with the initial state of "abcd". Both of them choose to edit on their local replica, without operation transmission between two sites. We define the current network state as net-broken, which the collaborative sites cannot communicate with each other. After breaking, Alice executes operation sequence T1, and updates the local replica to "pabq"; Bob changes the local document from "abcd" to "yzd" by performing T2 and T3. Suppose that the period of the local operation sequences' broadcasting, the transforming and the executing after receiving to produce a consistent document finally, is regard as net-survive. Still considering the scenario in Figure 1, after Alice receives two sequences of operations T2 and T3 submitted by Bob, T2 and T3 will be transformed by comparing the operations in the local history buffer to produce T2' and T3', and after the execution of T2' and T3', the document status is updated to "pyzq".

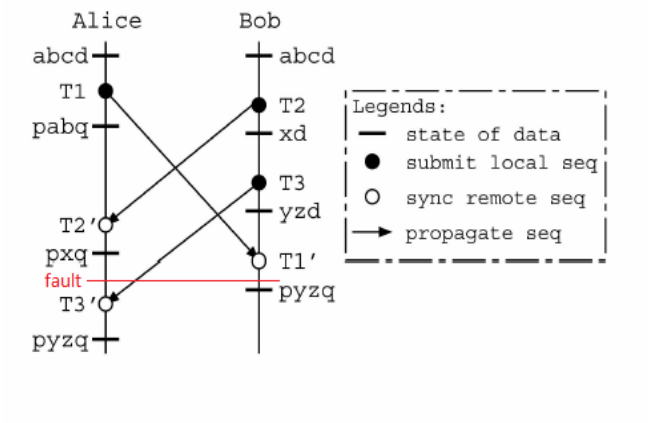


Figure 1. Alice submitted sequence T1 and Bob submitted two sequences T2

However, the research assumption of ABST is that the network status is ideal, clear and explicitly divided by users. It must ensure that all collaborative sites' copies are the same when the network is broken. In another words, at net-broken time, every collaborative site should have submitted all local operations and executed all remote operations successfully. But, in the unstable network environment, it is difficult to ensure this precondition. For example, at the fault moment (which is indicated with a red line in Figure (1): T2 arrives at site Alice and is executed in the form of transition to get the document's state "pxq"; besides, T1 follows and changes the document's state to "pyzq". Suppose that if the network breaks at this time, T3 is in the broadcasting process and it has not arrived at site Alice. In this condition, it is obvious that the states of two sites' documents are not identical. According to the ABST, the operation requests, which are below the fault line should be packaged to wait for posting to remote sites once network links again. However, because of the loss of T3, even all the operations generated after the fault line can be transformed and executed correctly at the collaborative sites, the final documents of different sites are inconsistent yet. So, how to capture lost operations and manage them to get consistent replica is a main problem we should resolve.

For this problem, we propose a consistency maintenance algorithm to support the collaborative editing in unstable mobile network environment, and confirm the correctness of this algorithm through a case analysis.

3.2 Document Model and Primitive Operations

The consistency maintenance algorithm presented in this paper is combined with Address Space Transformation (AST) [6] to dispose the problem of lost operations. In order to construct the necessary execution precondition of ABST, we model the shared data as a linear string (or sequence), and consider two primitive operations ins (p,c) and del (p,c), which insert and delete one object (or character) "c" at position p, respectively. As shown in Table 1, unlike the past definitions of operations, we append some new attributes to the operations. For each operation o, we define the following six attributes: o.id is the id of the collaborative site, which generates it originally; o.v is the timestamp when o is generated, some appropriate changes will be made to the definition of timestamp v in order to adapt to the AST strategy. Different from the ABST [7] using a one-dimensional array to define the type of timestamp. We stipulate the o.v as N-dimensional Array <1, 2…, n>, with n=1, 2, 3,…. To simplify discussion, the elements in o.v are ordered by the id (e.g. v<1, 3> expresses site1 and site2 has executed 1 and 3 operations, respectively); o.type is either ins or del; o.pos is the target position; and o.c is the target object that o inserts or deletes; o.net uses 0 and 1 to distinguish between the operations at the moment of break-before and break-after network. For example, Oa=(1,v,ins,2,h,0), Oa.v=<3,0> represents Oa is the third request operation generated at site1 in the period of smooth net connection as well as in the break-before network, and it attempts to insert character "h" at the second position of the document.

Table 1. Basic definition

Symbol	Brief Description
o.id	the id of site that originally generates o
o.v	the timestamp when o is generated
o.type	the operation type of o, either ins or del
o.pos	the position of o relative to the data model
o.c	the character inserted or deleted by o
o.net	to distinguish the network station by using 0 or 1
DOC	the local linear replica
HB	an operation sequence stores the executed operations
HBset	storing the operations selected from checking process
cursor	storing the length of operation set which has been executed at all sites

In the replicated architecture, any participating sites of the collaborative editing team must generate the local document replica, which reduces the responsiveness dramatically by allowing local operations to be applied immediately. The local replica is defined as Doc; and two linear structures are defined at each site, with one storing the executed operations (i.e. HB), while another storing the operations set, which are selected from checking process. (i.e. HBset). The indexes of both HB and HBset are from 1. Following AST [6] strategy, we restore the operations together with their timestamps to the character's linear node, and using the "effective/ineffective" mark to indicate the validity of each node in the linear structure.

3.3 Definitions of Operations Relationships

Definition 1: Given two operations O1 and O2, $O1=O2$, iff:

$O1.id = O2.id;$

$O1.v[id] = O2.v[id]$.

In this paper, every local operation is executed immediately on local replica after generation, and the corresponding timestamp will be modified automatically. Assuming that at site1, a new operation O is generated, and its timestamp is O.v=<id=1, id=2…, id=n>. After the execution of O, id=1 will be increased to 2. In simple terms, every value in O.v records the execution order of operations at corresponding site. So the timestamps of two operations generated at the same site cannot be equal. It should be noted that the reason why we propose this equal relationship is to compare and integrate operations during the packaging process, which will be introduced in detail later.

Definition 2: There are two local replicas Doc1 and Doc2, which are maintained at site1 and site2 respectively, and their history buffers are HB1 and HB2. So if Doc1=Doc2, it must satisfy the following conditions:

(1) $HB1.length=HB2.length$;

(2) $HB1 \cap HB2=HB1$ *and* $HB1 \cap HB2=HB2$;

(3) $(HB1 \cap HB2).length=HB1.length$ *and* $(HB1 \cap HB2).length=HB2.length$.

Because the consistency maintenance algorithm adopted in this paper is based on AST and ABST. In the previous researches, the CCI [8] (Convergence Causality Intention) model has been used to prove the correctness of the groupware system. The Convergence feature of CCI guarantees that when the same operations have been executed at all sites, all copies of the shared document should be identical, even executed in different orders. The above two techniques successfully satisfy CCI model. It is complex to compare Doc1 with Doc2 just according to the detail content by using compare algorithm [26-28]. Hence, we just need to compare the history buffers of different collaborative sites to prove whether the document copies are identical, which judges if these collaborative sites executed the same operation set.

3.4 Dynamic Cursor

We propose the definition of dynamic *cursor*, expressed as *cursor* to track the last operation set in HB when replicas of collaborative sites are consistent after comparing. So *cursor* is defined as numeric type and stores the length of operation set which has been executed at all sites. With the increasing number of the operations in the real-time group editing system, the value of the *cursor* should be updated frequently. An example is given to explain the usage of the *cursor*. Assume that at some points, the history buffers of site1 and site2 are HB1={O1, O2, O5, O3} and HB2={O5, O2, O1, O4}, the initial value of *cursor* is 0, after compared with each other, the HBset is created as {O1, O2, O5}, which is composed of the operations that have been executed in both sites even in different orders. And, the value of the *cursor* is changed by HBset.length, that is *cursor*=3.

The advantage of the *cursor* is that: It tremendously reduces the checking time and totally improves algorithm efficiency because we only need to check the operations behind the *cursor*, no need to check the operations again that have been checked before.

4 Consistent Maintenance Algorithm in Unstable Network

The distinguishing feature of the mobile network is instability, since it may be affected by electricity, signal and hardware performance, etc. As shown in Figure 2, in this article, we divide network into three statuses: break-before, break-after and connect-after. The main algorithm and mission in each status are not the same, but the ultimate purpose is to make all document copies of the collaborative sites identical by using a series of transformation when network connects again, no matter what states of the replicas at the net-broken point are, even some operations are lost at the point. It should be pointed out that we just focus

on how to capture and manage the lost operations to get identical copies, so the consistency maintenance algorithm used in break-before is un-constraint, and we suggest AST strategy in our research work.

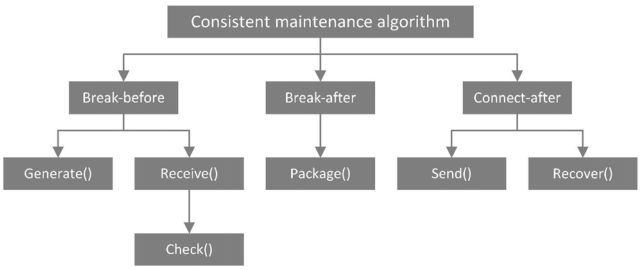


Figure 2. The main process in different network states

4.1 Break-before Period

In the period of break-before, the local operations, and the history buffer with the *cursor* should be broadcasted to remote sites. After the remote sites receiving the operations and the HB, they first compare it with local HB and then update the value of *cursor*. But, in practice, HB is an operation sequence and its size is larger than 1. So, if the local HB and the new operation are broadcasted to remote sites simultaneously, long time is needed for the process, which will reduce the efficiency of collaborative work. For this reason, the paper suggests to post HB automatically at a specific time interval.

Algorithm 1. Generate(Ou, HBu)

```
1. Execute(Ou);
2. Ou.net=0; // the value of .net is initialized
3. HBu.Append(Ou);
4. Broadcast(Ou);
   //The following is a broadcast timer which
   stipulates a specific time interval for posting local
   HB, researchers can define the interval in need; it
   sends local HB to remote sites automatically if
   current time satisfies the broadcasting interval.
5. If (Current-Time is Broad-Time)
6. {Broadcast(HBu);}
7. //Or return Generate algorithm to manage new
   operation
8. Else
9. {Return;}
```

If any operation at local site is being executed when remote HB arrives, for operation priority, the operation is firstly handled to update the local HB and then the history buffers are compared to refresh the value of the *cursor*.

Algorithm 2. Receive(Ou, HBu)

```
1.  If Ou is casually ready
2.  {
3.     Execute(Ou);
4.     HB.Append(Ou);
5.  }
6.  If remote HB is received
7.  {
8.     Check(HBlocal, HBremote, cursor);
9.     cursor=newCursor;
10. }
11. Else
12. {
13.   Return;
14. }
```

In order to improve the checking efficiency, we set the checking length as the shorter one of the length of the two HBs of the collaborative sites. According to the operation definition in this article, because of the value of user.id and timestamp is unique, there is no equivalent operation in one HB, which ensures the correctness of check function (HBlocal, HBremote, *cursor*).

Function 1. Check (HBlocal, HBremote, cursor)

```
1.  If (HBlocal.lengh>=HBremote.length)
2.     {length=HBremote.length;}
3.  Else {
4.     Length=HBlocal.length;
5.  }
6.  If(length==cursor)
7.     {  break;  }
8.  Else{
9.     For(i=cursor+1 to length)
10.       {For(j=cursor+1 to length)
11.          {If(HBlocal[i].id==HBremote[j].id  and
                 HBlocal[i].v[id]==HBremote[j].v[id])
12.            {  HBset.Append(HBremote[j]);  }
13.          }
14.       }
15. }
16. If(HBset.length==length-cursor)
17.    {
18.       cursor=length;
19.    }
20. Else{
21.    length=cursor + HBset.length;
22.    Goto step 5 -23;
23.   }
```

4.2 Break-after Period

In the former introduction of the operation definitions, O.net is used to distinguish the net status when an operation is generated, with 0 and 1 representing the net statuses of break-before and break-after respectively. So once the network is dis-connected, the operations generated during the period of break-after should be packed with the value of net as 1. Then for all collaborative sites, as Figure 3, packing the local HB at position started from *cursor* as two packages *SQ*0 and *SQ*1 according to the value of net, with *SQ*0 composed of operations with O.net=0 and *SQ*1 composed of operations with O.net=1.

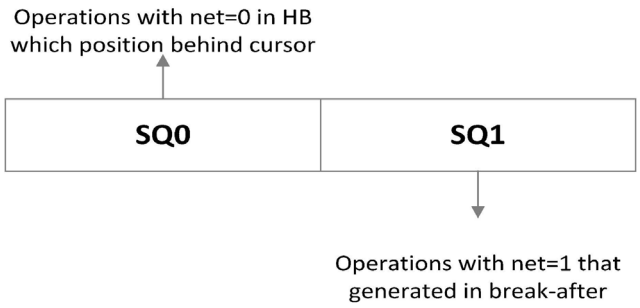


Figure 3. The differences between SQ0 and SQ1

Algorithm 3. Package(SQ0, SQ1)

```
1. Execute(O);
2. O.net=1;//Since the operation is generated during
   break-after period, it is executed immediately and
   its net value is set to 1
3. SQ1.Append(O);
4. For(i=cursor to HB.length)
5. {
6.     if(HB[i].net=0)
7.         {SQ0.Append(HB[i]);}
8.     if(HB[i].net=1)
9.         {SQ1.Append(HB[i]);}
10. }
```

4.3 Connect-after Period

After the re-connection of the net, the operations packaged in *SQ*0 and *SQ*1 of local site should be broadcasted to remote sites. When the remote sites receive these packages, it firstly identifies the lost operations in *SQ*0 and executes them to construct the execution precondition of the performing operations in *SQ*1. Because the main problem we resolve in this paper is how to execute the lost operations caused by net -broken, the concrete transforming process of the operations in *SQ*1 can be referred to the examples in ABST [7].

Algorithm 4. Send (SQ0, SQ1)

```
1. BroadCast(SQ0);// Preferentially post SQ0 to the
   collaborative sites to update the value of the
   cursor.
2. BroadCast(SQ1);// The size of SQ1 is bigger than
   SQ0, actually.
3. Excute(Ou);
4. Ou.net=0;//It emphasis that the net value is set to 1
   after connect-after.
```

Like the send(*SQ*0, *SQ*1) algorithm, remote sites firstly deal with the operations in *SQ*0. The puzzling problems caused by the broken network can be divided into two points:

(1) It is difficult to judge the relationships [8] between the lost operations and the local operations;

(2) If there exist lost operations that are generated before the operations which have been executed at remote sites, how to transform the lost operations when it is received after re-send is difficult to define.

Take Figure 4 as the example. There are two sites: siteA and siteB, and the initial value of *cursor* is 0 at two sites. Assume that O2 and O3 have been post to siteB and siteA and executed separately. O1 is a lost operation, which is generated before O2 at siteA, this scenario satisfies the second point discussed above in section 4.

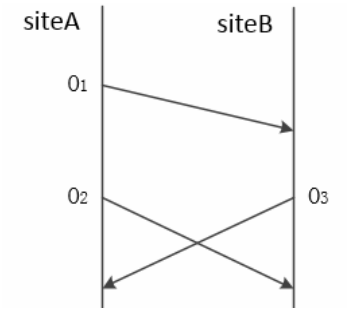


Figure 4. Broadcasting process between two sites

At this time, A.*SQ*0={O1, O2, O3}, B.*SQ*0={O3, O2}, O1 should exclude the effect of O3 to guarantee that siteB can perform O1 correctly.

Similarly, O1 and O3 have been executed at siteB and siteA already. Because of the disconnecting network, O2 becomes the lost operation, which cannot be broadcasted to siteB, this conforms to the first point discussed in section 4.

Now, A.*SQ*0={O1, O2, O3}, B.*SQ*0={O1, O3}, and after receiving O2 at siteB, O2 should be transformed against O3 due to the concurrent relationship between them.

Considering the complex relationship between lost operations and executed operations we prefer AST algorithm to deal with the lost operations. In the recover process, we use operation sequence Larry to store the operations whose id are remoteid in *SQ*0local, and Rarry to store the operations in *SQ*0remote whose id are remoteid.

Algorithm 5. Recover (SQ0remote, SQ1remote)

```
1.  Receive(SQ0remote);
2.  For(i=1 to SQ0local.length)
3.  {//Select the operation where id=remoteid in local
    SQ0 and append it to Larry.
4.  if(SQ0local[i].id==remoteid)
5.     Larry.Append(SQ0local[i]);
6.  }
7.  For(j=1 to SQ0remote.length)
8.  {// Select the operation where id=remoteid in
    remote SQ0 and append it to Rarry.
9.  If(SQ0remote[j].id==remoteid)
10.    Rarry.Append(SQ0remote[j]);
11. }
12. If(Larry.length==Rarry.length)
13. {//It indicates that all operations generated before
    the break have been executed in all the
    collaborative sites, but the value of the cursor has
    not been updated immediately.
14.    cursor==cursor+SQ0local.length;
15. }
16. Else
17. {//There exist lost operations, which results in the
    inconsistency of the document copies of the
    collaborative sites.
18.    For(k=1 to Rarry.length)
19.    { //Search and manage the lost operation.
20.      If(Rarry[k] not in Larry)
21.      {  Retracing(Doc, Rarry[k].v);
22.         Excute(Rarry[k]);
23.         Rarry[k].v[remoteid]++;
24.         HBlocal.Append(Rarry[k]);
25.         Retracing(Doc, v);
26.      }
27.    }
28. cursor=cursor+SQ0local.length+Rarry.length
              -Larry.length;
29. }
30.    Receive(SQ1remote);
31.    ABST(SQ1remote);
```

5 Efficiency Analysis

We divide the network statuses into three types: break-before, break-after and connect-after, and employ different control algorithms in different statuses, so the processing of the three statuses is parallel.

The check (HBlocal, HBremote, *cursor*) algorithm used in break-before period mainly records the latest synchronization point, where the document copies of the collaborative sites are identical, and updates the value of *cursor* by comparing the operations in their history buffers. The definition of *cursor* proposed in this paper significantly improves the efficiency and shorten the comparing time. Suppose that it costs d times' comparisons to update the value of *cursor*, its time complexity is expressed as O(d·(HBlocal.length-*cursor*)·(HBremote.length-*cursor*)). So the more frequently the check algorithm runs, the faster the *cursor* updates. However, with the increasing of comparing times, the comparing length will be shorter gradually. Therefore, in reality, the overall efficiency is higher than we analyze above.

The main work in break-after period is to divide the operations into two packages according to the net statuses. Its time complexity is linear which is expressed as O(HB.length-*cursor*).

In the connect-after period, the collaborative sites identify and execute the lost operations. The step 1-11 in algorithm 4 Recover(*SQ*0remote, *SQ*1remote) mainly selects the operations with given id, and its comprehensive time complexity is less than step 16-27 which focuses on finding and managing the lost operations. Because we utilize AST control algorithm to manage lost operations in the connect-after phase, the amortized costs of the Insert and Delete are both O(logn) (where n is the number of the document's character nodes). However, it has to traverse all the operations of Rrray in order to compare with operations in Larry for searching lost data, and the running efficiency during this period is O(Rarry.length·Larry.length). The comprehensive time complexity of the Recover algorithm is O(Rarry.length·Larry.length·logn), which can be simplified as O(logn) when the lost operations are less.

6 Case Analysis

As shown in Figure 5, the initial document's state is "", and the value of *cursor* is 0. "T1" is the time to update the value of *cursor* after broadcasting local HB to remote sites. "Mark" indicates the lost operations during the net dis-connecting period. "net" is used to indicate whether an operation is produced before the breakpoint or not, with 0 indicating the status of break-after, while 1 otherwise. This paper focuses on dealing with the operations, which are lost during the broadcasting process or not send for constructing the necessary precondition of the ABST algorithm correctly. So, we just pay attention to how to manage the operations with O.net=0.

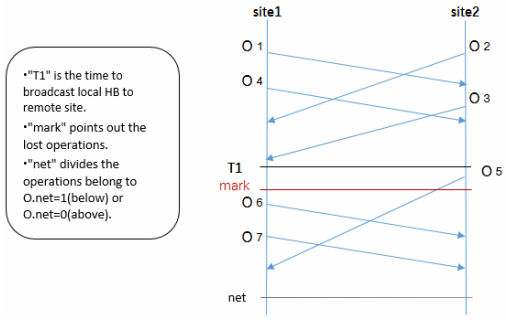


Figure 5. An example of consistency maintenance algorithm

Firstly, the detailed description of the operations in Figure 5 is as follows.

Site1 generates operations:

O1=Ins(b,1), O4=Ins(c,2), O6=Ins(d,1), O7=Ins(e,2);

Site2 generates operations:

O2=Ins(a,1), O3=Del(b,2), O5=Ins(x,2).

In the following, we will analyze the document states, HB and *cursor* of site1 and site2 at time point T1. And the values of main parameters on two sites at the point T1 can refer to the Table 2.

Table 2. The values of parameters on two sites at the point "T1"

parameters	Site1	Site2
HB	{O1, O4, O2, O3}	{O2, O1, O3, O4}
cursor	4	4
Doc	"ac"	"ac"
net	0	0
v	(2,2)	(2,2)

Site1: O1 and O2 are generated and executed immediately at local site. After that the document state is changed to "bc", and the timestamp vector is set as (2,0). When O2 arrives at site1 with the timestamp (0,1), the control process first retraces the Address Space of site1 to get the document state "bc" (characters which tagged by underline represent ineffective nodes), then determines the accurate insert position b by using function Rang-Scan [6], and then updates the document's state and timestamp as "abc" and (2,1) respectively. Finally, the Address Space is retraced to the current time to get the ultimate state "abc". O3 is the last arrived operation which transforms the document's state to "ac" and alters the timestamp vector as (2,2).

Site2: After the execution of O2 generated at local site, the timestamp and document's state is changed to (0,1) and "a" respectively. Afterwards, O1 arrives at site2 with timestamp (1,0) and the state of document is changed from "a" to "ab" by using the AST strategy like site1, and the timestamp vector is updated as (1,1). O3 is executed instantly, and then O4 arrives and executed, resulting in the timestamp result of (2,2), and the final document's state of "ac".

Because the current time does not satisfy the posting condition, so there is no HB broadcasted or received, or the value of *cursor* has no change; At this time, HB1={O1, O4, O2, O3}, HB2={O2, O1, O3, O4} and both of two states of the documents of the sites are "ac".

(1) In the break-before phase, the local site broadcasts HB to remote sites, and updates the local value of *cursor* to unite with the old *cursor*. Take the scenario at site1 as the example, after HB2 is received from site2 with HB2={O2, O1, O3, O4}, it is compared with local HB1={O1, O4, O2, O3} by using algorithm Check(HBlocal, HBremote, *cursor*), and the detailed process is as follows.

The value of checking length is set as the shorter one between HB1.length and HB2.length, i.e. length=4.The operations are selected, which satisfies that both O.id and O.v[id] equal to that of the operations in HB2 are selected from HB1[*cursor*+1] to HB1[length] (the index of the linear sequences appear in this article all starts from (1), and added them into HBset, finally resulting HBset={O1, O4, O2, O3}. Because HBset.length=length, only one time is needed to search the operations performed at two sites. There are four operations that have been executed already and document statuses at site1 and site2 are identical even in different orders. After the checking process, the value of local *cursor* is updated to *cursor*+length immediately, that is *cursor*=4.

Table 3. The values of parameters on two sites when net-broken

parameters	Site1	Site2
HB	{O1, O4, O2, O3, O6, O7}	{O2, O1, O3, O4, O5}
cursor	4	4
Doc	"deac"	"axc"
net	1	1
v	(4,2)	(2,3)

(2) In the break-after phase, there may exist several operations lost or not be sent during the posting process (such as the operations between "mark" and "net" in Figure 5), and local site may continue generating a set of operations (such as the operations below the "net" line, and this problem had been resolved in ABST). So at site1, the value of O.net of the new operations, which are generated in the period of break-after, will be set as 1. Next, the operations from HB1.[*cursor*+1] to the end will be packed according to the value of O.net to create two packages *SQ*0 and *SQ*1. Since HB1={O1, O4, O2, O3, O6, O7}, *SQ*0.site1 is set to {O6, O7} and *SQ*1.site1 to {}.

(3) In the connect-after phase, the local site broadcasts *SQ*0 and *SQ*1 to the collaborative sites. In site1, firstly, *SQ*0.site2={O5} is received from remote site2, and after the comparison between *SQ*0.site1 and *SQ*0.site2, we can deduce that O5 is lost during the broadcasting process. How to make consistent copies by calling Recover() algorithm is described in the following:

Step 1: Select the operations with id=2 in *SQ*0.site2 and *SQ*0.site1, add them to the operation sequences Rarry and Larry separately. Thus we get Rarry={O5}, Larry={};

Step 2: Judge Rarry.length to decide whether it is equal to Larry.length. If so, the value of *cursor* is updated to *cursor*+*SQ*0.length. If not, find the operations that do not exist in Larry but in Rarry. As for every operation in Rarry [n] (n=1, 2,…), apply AST strategy to execute them. In this example, Rarry [1] (i.e. O5) should be executed on site1, and after the

execution of O6 and O7, the document's state and the timestamp are "deac" and (4,2) respectively. Rarry [1] (namely O5) with the timestamp vector (2,3) should be executed at site1, on the basis of function Retracing() to get the state of the document as "deac" and ensure the position for inserting character "x" is in the middle of "ac", finally O5 is executed and the document's state is retraced to "deaxc".

Step 3: The value of *cursor* should be updated from 4 to (4+*SQ*0.site1.length+Rarry.length-Larry.length).

Similar to site1, we get Rarry={O6, O7} and Larry = {} at site2, so that O6 and O7 are executed with the AST strategy. O6 changes the document's state and timestamp to "daxc" and (3,3), while O7 updates the content of the local copy to "deaxc" with the timestamp (4,3). Finally, the value of *cursor* is refreshed to (oldcursor+*SQ*0.site2.length + Rarry. length-Larry.length) the oldcursor represents the last value of *cursor*. And the Table 4 shows the comparison data on two collaborative sites.

Table 4. The values of parameters on two sites after process recover()

parameters	Site1	Site2
HB	{O1, O4, O2, O3, O6, O7, O5}	{O2, O1, O3, O4, O5, O6, O7}
cursor	7	7
Doc	"deaxc"	"deaxc"
net	0	0
v	(4,3)	(4,3)

7 Correctness Analysis

In the break-before and the break-after phases, we adapt AST [6] strategy to dispose the remote operations and new operations generated at local. In the connect-after phase, we also use AST to re-execute the lost operations caused by net broken. For the operations generated during the period of break-after, they will be packed as an operations sequence (the *SQ*1 introduced above) and be transformed through ABST [7] control algorithm. The correctness of AST and ABST algorithm have been confirmed in the related articles, therefore, both of them conform to the CCI model. Thus, next we mainly validate the correctness of the consistency maintenance algorithm proposed in this paper. Firstly, we overview the specific content of CCI [8] model:

(1) **Convergence**: when the same operation set has been executed at all sites, all copies of the shared document are identical.

(2) **Causality-preservation**: for any pair of operations Oa and Ob, if Oa→Ob, Oa is executed before Ob at all sites.

(3) **Intention-preservation**: for any operations O, the effects of executing O at all sites are the same as the intention of O, and the effect of executing O does not change the effects of independent operations.

For Convergence, in the three network statuses introduced in this article, the operations generated in the period of break-after cannot be broadcasted to remote sites, of course, there may exist some lost operations during the posting process in the break-before phase because of the unstable network. But the function check(), which introduced in the break-before, mainly compares local HB with remote HB to update the value of *cursor*, we can confirm that the operations before *cursor* must have been executed at all collaborative sites. Once the net is broken, the package() process involves in the break-after will pack the operations behind *cursor* as two operation sequences according to the value of net. We append the operations with the value of net is 0 into operation sequence *SQ*0, and the operations in *SQ*0 are divided into two types: one is broadcasted and executed correctly at remote sites, while the other is the lost operations we raise in this paper. Finally, in the connect-after phase, remote sites capture the lost operations in the received *SQ*0 through Recover() algorithm, and then execute them to refresh the local HB and *cursor*. To sum up, in the whole process, we find, re-post and execute lost operations by referring to the value of *cursor*, promising that every collaborative site executes the same operation set, thus guaranteeing the Convergence.

For Causality-preservation, because we use AST strategy to manage the lost operations, it only transforms the address space of document according to the timestamps of the operations, there is no need to consider the relationship between operations.

For Intention-preservation, in the above example, according to the process of disposing lost operations in *SQ*0, we get the consistent copies at all collaborative sites (both site1 and site2 create the identical copies with the state of "deaxc"). When receiving the operation sequence *SQ*1, because of the necessary precondition of running ABST algorithm has been created in the above consistency maintenance technique, the *SQ*1 can be executed correctly by ABST algorithm. All collaborative sites will get the identical document statuses

The key technique to implement consistency maintenance in our proposed algorithm, is the *cursor*. It records the latest consistency point of copies of collaborative sites. The lost operations can also be obtained by using the *cursor* changed last time even the value of the *cursor* is not updated immediately. Of course, it needs some time to process these lost operations and un-updated cursor's value in the connect-after status. Although in the negative case, due to the fast increasing number of the operations, the *cursor* still holding the initial value, in other words, the checking process is just finished in Recover() algorithm, that will lead to low efficiency. However, it is worth that we can get the consistent shared documents in the end.

8 Conclusion and Further Work

With the popularity of the mobile network and fast development of variable network conditions technique [24-25], more and more people prefer to undertake work with the mobile terminals, which is portable and has no constraint of position. The research and application development in mobile collaborative field have become the development trend of the computer science and technology. Not only on mobile terminals, but also on desktop and laptop computers may appear the condition of the network is slow, and some users prefer to work remotely by using PC or Cloud storage [26, 29, 31-32]. So the conflicts in unstable network made by these kinds of situations can be resolved through the consistent maintenance algorithm proposed in this article.

The greatest contriburion of this paper is : proposes a consistency maintenance algorithm to support the collaborative editing work in unstable network. This algorithm constructs the precondition for the correct execution of the ABST strategy, combined with the basic ideas about AST, however, if we just design a cooperative system by using exsiting algorithm that support asynchronous cooperation in unstable network, it is difficult to gurantee that the program of broadcasting doesn't exsit lost operations, so the consistency maintenance algorithm we provide just can catch and handle lost operaions.

We plan to extend this work along two directions: (1) because ABST is an OT-based algorithm, and we employ the AST to manage the lost operations, there may need more storage space to save the character nodes and the primary operations. If we can develop the consistency maintenance algorithm, which is AST-based, huge storage space can be saved; (2) The influential factors of mobile network (such as power, geographic position and time, etc.) should also be considered into our consistency maintenance algorithm, we may made some detectors to analysis these index [30].

Acknowledgements

We would like to thank the reviewers, whose valuable critique and comments helped to improve this paper. Moreover, this work is supported by the National Science Foundation of China (NSFC) under Grant No. 61202376 and 61572325, Shanghai Natural Science Foundation under Grant No. 17ZR1419100, the Open Project Program of Shanghai Key Laboratory of Data Science (No. 201609060003), Shanghai Key Science and Technology Project in Information Technology Field (No. 14511107902), Shanghai Leading Academic Discipline Project (No. XTKX2012), Shanghai Engineering Research Center Project (GCZX14014, C14001).

References

[1] C. A. Ellis, S. J. Gibbs, G. L. Rein, Groupware: Some Issues and Experiences, *Communications of Association for Computing Machinery*, Vol. 34, No. 1, pp. 39-58, January, 1991.

[2] Y. H. Feng, L. P. Gao, N. Gu, F. Wang, Locking Intention Preservation Based on Address Space Transformation Technique, *International Conference on Pervasive Computing & Applications,* Alexandria, Egypt, 2008, pp. 497-502.

[3] H. S. Gu, X. Xie, Q. Lv, Y. Ruan, L. Shang, Etree: Effective and Efficient Event Modeling for Real-time Online Social Media Networks, *International Conference on Web Intelligence & Intelligent Agent Technology*, Lyon, France, 2011, pp. 300-307.

[4] B. Shao, D. Li, T. Lu, N. Gu, An Operational Transformation Based Synchronization Protocol for Web 2.0 Applications, *Association for Computing Machinery Conference on Computer Supported Cooperative Work*, Hangzhou, China, 2011, pp. 563-572.

[5] G. Oster, P. Urso, P. Molli, A. Imine, Data Consistency for P2P Collaborative Editing, *The 20th Anniversary Conference on Computer Supported Cooperative Work*, Banff, Alberta, Canada, 2006, pp. 259-268.

[6] N. Gu, J. M. Yang, Q. W. Zhang, Consistency Maintenance Based on the Mark & Retrace Technique in Groupware Systems, *International Association for Computing Machinery Siggroup Conference on Supporting Group Work, Sanibel Island*, FL, 2005, pp. 264-273.

[7] B. Shao, D. Li, N. Gu, A Fast Operational Transformation Algorithm for Mobile and Asynchronous Collaboration, *IEEE Transactions on Parallel and Distributed System*, Vol. 21, No. 12, pp. 1707-1720, December, 2010.

[8] C. Z. Sun, C. A. Ellis, Operational Transformation in Real-time Group Editors: Issues, Algorithms, and Achievements, *Association for Computing Machinery Conference on Computer Supported Cooperative Work*, Seattle, WA, 1998, pp. 59-68.

[9] C. A. Ellis, S. J. Gibbs, Concurrency Control in Groupware Systems, *Association for Computing Machinery Sigmod International Conference on Management of Data*, Portland, OR, 1989, pp. 399-407.

[10] B. Shao, D. Li, N. Gu, ABTS: A Transformation-based Consistency Control Algorithm for Wide-area Collaborative Applications, *International Conference on Collaborative Computing: Networking, Applications and Work Sharing*, Washington, DC, 2009, pp. 1-10.

[11] B. Shao, D. Li, N. Gu, A Sequence Transformation Algorithm for Supporting Cooperative Work on Mobile Devices, *Association for Computing Machinery Conference on Computer Supported Cooperative Work,* Savannah, GA, 2010, pp. 159-168.

[12] Y. Xu, C. Z. Sun, M. Li, Achieving Convergence in Operational Transformation: Conditions, Mechanisms and Systems, *Association for Computing Machinery Conference*

on Computer Supported Cooperative Work, Baltimore, MD, 2014, pp. 505-518.

[13] H. H. Xia, T. Lu, B. Shao, G. Li, X. Ding, N. Gu, A Partial Replication Approach for Anywhere Anytime Mobile Commenting, *Computer Supported Cooperative Work and Social Computing, Baltimore*, MD, 2014, pp. 530-541.

[14] Y. Xu, C. Z. Sun, Conditions and Patterns for Achieving Convergence in OT-based Co-Editors, *Transactions on Parallel and Distributed Systems*, Vol. 27, No. 3, pp. 695-709, March, 2016.

[15] M. Ressel, D. Nitsche-Ruhland, R. Gunzenhäuser, An Integrating, Transformation-oriented Approach to Concurrency Control and Undo in Group Editors, *Association for Computing Machinery Conference on Computer Supported Cooperative Work*, Boston, MA, 1996, pp. 288-297.

[16] C. Z. Sun, X. Jia, Y. Yang, D. Chen, Achieving Convergence, Causality Preservation, and Intention Preservation in Real-time Cooperative Editing Systems, *Association for Computing Machinery Transactions on Computer-Human Interaction*, Vol. 5, No. 1, pp. 63-108, March, 1998.

[17] D. Sun, C. Z. Sun, Operation Context and Context-based Operational Transformation, *The 20th anniversary conference on Computer Supported Cooperative Work*, Banff, Alberta, Canada, 2006, pp. 279-288.

[18] N. Vidot, M. Cart, J. Ferrie, M. Suleiman, Copies Convergence in a Distributed Real-time Collaborative Environment, *Computer Supported Cooperative Work,* Philadelphia, PA, 2000, pp. 171-180.

[19] R. Li, D. Li, A Landmark-based Transformation Approach to Concurrency Control in Group Editors, *International Association for Computing Machinery Siggroup Conference on Supporting Group Work,* Sanibel Island, FL, 2005, pp. 284-293.

[20] R. Li, D. Li , C. Z. Sun, A Time Interval Based Consistency Control Algorithm for Interactive Groupware Applications, *The 10th International Conference on Parallel and Distributed Systems,* Newport Beach, CA, 2004, pp. 429-436.

[21] D. Li, R. Li, An Admissibility-based Operational Transformation Framework for Collaborative Editing Systems, *Computer Supported Cooperative Work*, Vol. 19, No. 1, pp. 1-43, February, 2010.

[22] B. Shao, D. Li, N. Gu, An Optimized String Transformation Algorithm for Real-time Group Editors, *The 15th Institute of Electrical and Electronic Engineers International Conference on Parallel and Distributed Systems*, Shenzhen, China, 2009, pp. 376-383.

[23] W. J. Huang, T. Lu, H. Y. Zhu, G. Li, N. Gu, Effectiveness of Conflict Management Strategies in Peer Review Process of Online Collaboration Projects, *Association for Computing Machinery Conference on Computer-Supported Cooperative Work & Social Computing*, San Francisco, CA, 2016, pp. 717-728.

[24] P. Guo, J. Wang, X. H. Geng, C. S. Kim, J.-U. Kim, A Variable Threshold-value Authentication Architecture for Wireless Mesh Networks, *Journal of Internet Technology*, Vol. 15, No. 6, pp. 929-935, November, 2014.

[25] J. Shen, H.-W. Tan, J. Wang, J.-W. Wang, S.-Y Lee, A Novel Routing Protocol Providing Good Transmission Reliability in Underwater Sensor Networks, *Journal of Internet Technology*, Vol. 16, No. 1, pp. 171-178, January, 2015.

[26] Y.-J. Ren, J. Shen, J. Wang, J. Han, S.-Y Lee, Mutual Verifiable Provable Data Auditing in Public Cloud Storage, *Journal of Internet Technology*, Vol. 16, No. 2, pp. 317-323, March, 2015.

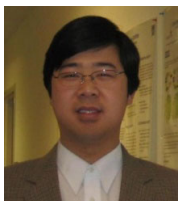
[27] Z. Xia, X. Wang, X. Sun, Q. Liu, N. Xiong, Steganalysis of LSB Matching Using Differences between Nonadjacent Pixels, *Multimedia Tools and Applications*, Vol. 75, No. 4, pp. 1947-1962, February, 2016.

[28] Y. Liu, N. Xiong, Y. Zhao, A. V. Vasilakos, J. Gao, Y. Jia, Multi-layer Clustering Routing Algorithm for Wireless Vehicular Sensor Networks, *Institution of Engineering and Technology Communications*, Vol. 4, No. 7, pp. 810-816, April, 2010.

Biographies

Liping Gao graduated from Fudan University, China with a Ph.D. in 2009 in Computer Science. She received her BSc and master degree in Computer Science from Shandong Normal University, China in 2002 and 2005 respectively. She is doing her research work in University of Shanghai for Science and Technology as an assistant professor. Her current research interests include CSCW, heterogeneous collaboration, consistency maintenance and collaborative engineering.

Dan Wang is a postgraduate student in University of Shanghai for Science and Technology. She obtained his BSc degree in Computer Science from Dalian Minzu University, China. Her current research interests include CSCW, collaborative design and collaborative computer.



Naixue Xiong is a Professor at School of Computer Science, Colorado Technical University, Colorado Spring, CO, USA and University of Shanghai for Science and Technology, China. He received his both PhD degrees in Wuhan University (about software engineering), and Japan Advanced Institute of Science and Technology (about dependable networks), respectively. Before attending Colorado Technical University, he worked in Wentworth Technology Institution, Georgia State University for many years. His research interests include Cloud Computing, Security and Dependability, Parallel and Distributed Computing, Networks, and Optimization Theory.

Bing Wang graduated from SooChow University, China with a master degree in 2006 in School of Computer Science and Technology. He received his BSc degree in Computer Science from XinYang Normal University. He is doing his research work in University of Shanghai for Science and Technology as an engineer. His current research interests include CSCW, Big Data and Educational Informationization.