International Journal of Hybrid Information Technology
Vol.8, No.10 (2015), pp.383-394
http://dx.doi.org/10.14257/ijhit.2015.8.10.35

High Efficient Consistency Maintenance Strategy of Real-time String Text Editing Systems

Liping Gao and Wenfeng Tang

School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai, 200093, China
Email: lipinggao@st.usst.edu.cn, wenfengtang@st.usst.edu.cn

Abstract

The idea of address space transformation provides a new way for concurrency control. During concurrent processing, it retraces the document status back to the state when the operations are generated to maintain consistency. However the previous concurrency processes strategy is based on single characters, the transmission cost during processing is too high, especially when the network is unstable. Due to this problem, this paper presents a consistency maintenance strategy based on string editing operations, and proposes the string splitting mechanism combined with the idea of the address space transformation in order to maintain consistency.

Keywords: *consistency maintenance, address space transformation, string operation, splitting mechanism*

1. Introduction

With the rapid development of computer technology and the wide application of network communication, remote collaboration becomes more and more prevalent [1-4]. However the delay of network communication is high and uncertainly. Thus how to ensure a quick respond time in coordination system becomes a factor to be considered. It is impossible to eliminate the network latency, and we can only improve the response speed on the system level.

An important field of the collaborative environment is the text editor field; in the collaborative text editing field, there are causally dependent and concurrent relationships among the operations. How to maintain the operations' intention, are the focus of the research in the text editor field.

Most of the previous text editing systems are to process a single character, and provide corresponding control strategies to maintain consistency and achieve some results, such as the COT algorithm, GOTO algorithm, AST algorithm and so on[2, 3, 5], but few involve in the string editing system.

The string text editing has practical examples, such as the copy and paste operations. In the single character text editing system, a single character is transported between the sites. In the editing environment with little number of characters, the algorithm can maintain consistency at all the sites successfully. However, when processing environment with massive characters, the algorithm has an obvious shortage, especially in the case that network is unstable.

This paper is divided into the following sections: The second chapter describes some related work in text editors, the third chapter arises the consistency maintenance issues, the fourth chapter describes the storage structure of the nodes in the string text editor, the fifth chapter gives consistency maintenance strategies and algorithms, the sixth chapter gives specific examples to verify the correctness of the algorithm, the seventh chapter analysis the efficiency of algorithm, the eighth chapter gives the acknowledgement from the practical aspects, and the final chapter give a summary and outlook of article.

2. Related Works

2.1. Consistency Maintenance Strategy

Data replication technology is widely used in group editing systems [1]; a group of users share the document and edit the text freely. In an ideal group system, whether the operations are concurrent or not, what the execution order is, the document gets the same result, which meets the CCI consistency model [6].

In the single character text editing system, previous works propose special strategies to solve the inconsistency caused by the concurrent operations, including operation transformation (OT) and address space transformation (AST). Both of them will solve the inconsistency caused by concurrency.

Operation Transformation strategy has been developed rapidly during the past twenty years, and now supports a variety of applications, including group undo[7-9], group awareness [9], operation notification and compression [10], spreadsheet and table centric applications[4], *etc*.

AST is proposed to solve the problem from the view of document status; by retracing the document back to the state when the operation is generated [6, 10, 11]. Compared with the operation transformation, this strategy looks more intuitive.

2.2. Address Space Transformation

In this paper, AST is used to solve the problem. AST algorithm is based on the mark and retrace algorithm and provides a new idea to solve the complicated conflict. State vector timestamp is proposed to judge the causal relationship between the operations.

When an operation reaches a site, the document status may have been changed different from the operation's generating state. During the processing of the AST strategy, it hides the possible impact and retraces the document status back to the status when the operation is generated by mark and retrace process, and then the operation can be performed in the new document, After the execution of the operation, the document status is then retraced to the concurrent status. In the implementation process, the function "retracing" will hide the execution effect that may affect the operation, and the function "rang-scan" is used to find a specific location for execution, and function "retrace" is called to include the operation results into the document [4].

A document has a linear structure containing a number of characters, with each character be associated with multiple operations, but each operation has only one character node. The timestamp as well as the mark of the effective or ineffective are stored in corresponding character node, which indicates whether the current node is visible or not at the current moment.

3. Consistency Maintenance

3.1. Conflict Problems

In the string text editor, the basic operations are Insert and Delete operations. In the presence of the string editor, the Insert position is not only before or after a string, but also has the case that insert a string into an existing string; the Delete operation may not only delete a particular string, but also deletes a specific character from a string. For the situations that similar to the single character, the string can be seen as a whole and transplant the single character processing strategies directly for the concurrence control; as for the latter, the single character processing algorithms cannot solve the problem, and need to find a new strategy.

Supposing that the operations O_1 and O_2 are from different sites, O_1=insert ("hello", 1), O_2=insert ("world", 1), O_1 and O_2 want to insert different strings in the same position. In addition, if an operation has inserted a character "hello" before this statement, the document has changed, execute the operation directly will lead to inconsistency, the solutions to these problems have not been totally dependent on the strategy processing the signal character.

3.2 Modified Operation Definition

Given the complexity of the string operations, the definitions of the operations need to be modified to adapt to the new situations:

Insert ("string", pos, left):"string" represents the string to be inserted;

"pos" represents the Insert position;

"left" is a symbol, there are two meanings: if left is 0, it means that there will be a string to be inserted in the position "pos"; if left is not 0, it means that the string will be inserted into the existing string, and the "left "represents that the string will be inserted in the position counting from left.

Delete (pos, n, left):

"pos" indicates the position of the strings to be deleted;

"left" is the symbol, if it is 0, it means to delete the string at the position "pos";if left is not 0, it means to delete n characters from the string at the position "pos" from the position "left" .

For example, there is a document "hello world", O1=insert ("hello", 2,0) and the document becomes "hello hello world"; O2=insert ("China ", 1,2) and the document will be "hChinaello world"; O3=delete(1,2,0) and the document becomes "world"; O4=delete(1,2,2) and the document becomes "hlo world".

3.3 Analysis of Operation Execution Cases

According to the previous definition of the operations, the Insert operation has two parameters, one representing the location to be inserted, and another representing the string to be inserted. Thus, there will be the following situations to be considered: if “left” is 0, the treatment is relatively simple; while when "left" is not 0, two situations will be considered, the parameter "pos" is equal or not equal. The following examples are corresponding to the discussion above.

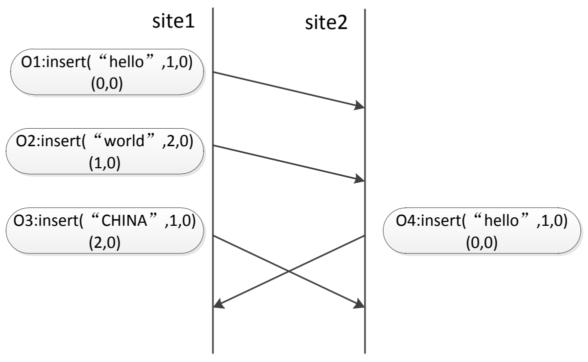

Figure 1. The Overall String Insertion

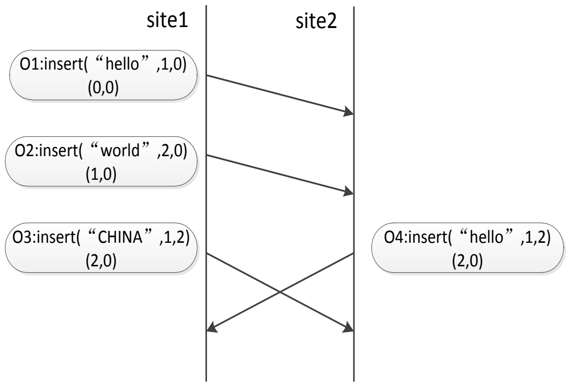

Figure 2. Insert String into the Same String

In Figure 1, the string is inserted between strings other than the internal string. Based on the definition of the previous operation, O_3 and O_4 insert the string "CHINA" and "USA" both in the first position in the first position. In Figure 2, this situation corresponds to the case that the parameter "pos" is equal. O_3 means to insert the string "CHINA" in the second position of the first string, O_4 means to insert the string "USA" in the second position of the first string.

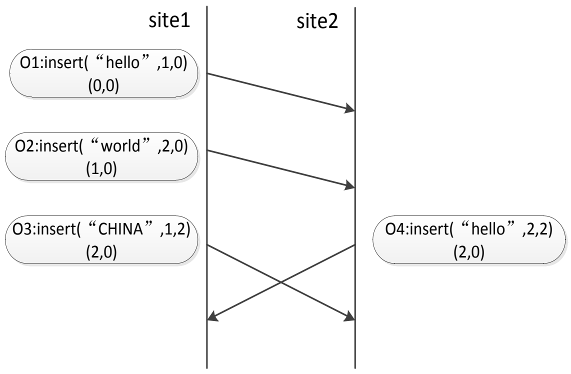

Figure 3. Insert String into Different Strings

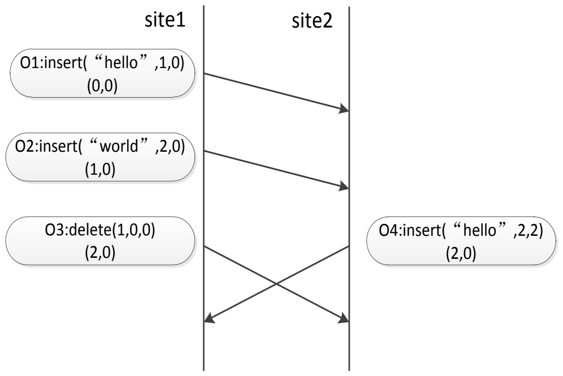

Figure 4. Delete the Entire String

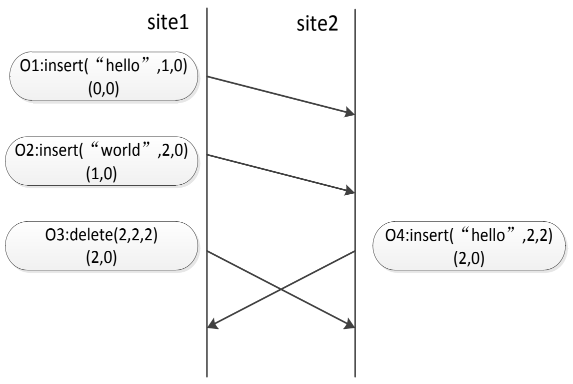
site1
site2
O1:insert("hello",1,0) (0,0)
O2:insert("world",2,0) (1,0)
O3:delete(2,2,2) (2,0)
O4:insert("hello",2,2) (2,0)

Figure 5. Delete a Specific Character from a String (a)

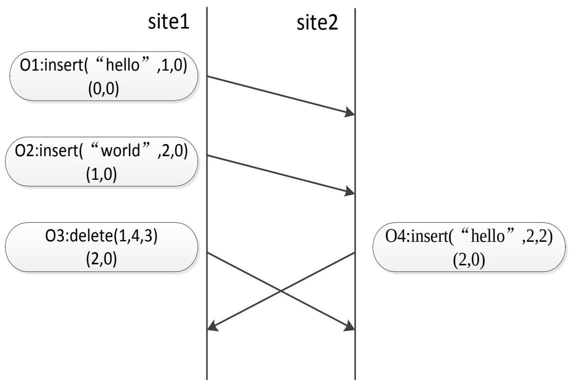
site1
site2
O1:insert("hello",1,0) (0,0)
O2:insert("world",2,0) (1,0)
O3:delete(1,4,3) (2,0)
O4:insert("hello",2,2) (2,0)

Figure 6. Delete Particular Character from a String (b)

In Figure3, the situation is the case the parameter "pos" is unequal, and is in different strings. O_3 inserts the string "CHINA" in the second position of the first string; O_4 inserts the string "USA" in the second position of the second string.Figure4, Figure5, and Figure6 show more complex situations that the insertion and deletion are mixed.

4. Storage Structure

While handling the string, this paper adopts such an approach: when the users of each site edit the document, when a word is completed, each character of the word is packaged and then broadcasted to all the other sites. The space or punctuation between words is the separators.

This paper extends the storage structure and the operation is stored based on the following agreement: when the Insert operation is carried out between the strings ,the string is directly stored as a node, and if the Insert operation is carried out in a string, the operation will be attached to the node representing the string; as for the Delete operation, this paper adopts a similar strategy, and the Delete operation is attached to a specific string; as for the delimiters between strings, this paper treats them as independent node.

Figure1 shows some operations, the storage structure is show as follows:

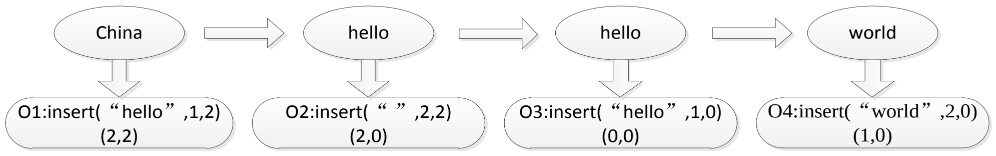
China
hello
hello
world
O1:insert("hello",1,2) (2,2)
O2:insert(" ",2,2) (2,0)
O3:insert("hello",1,0) (0,0)
O4:insert("world",2,0) (1,0)

Figure 7. Diagram of Operation Storage

For each node, the structure is shown below:

String	flag	ptonext	ptodown

Figure 8. Node Structure

"string" indicates the content of the node ;

"flag" is the node's identifier and indicates whether the node is effective or not and its value is "effective" or "ineffective";

"ptonext" indicates a pointer to the next node , if the next node do not exist, it will be null;

"ptodown" indicates a pointer pointing to a node associating with it, if the associating node do not exist, it will be null too.

5. Control Strategies

5.1 Analysis of Operation Relationship

This paper has extended the operation previously and the traditional definition of concurrency is not suitable for the string text editing system, and it needs to be redefined.

Assuming that there are two operations, O_1=insert ("string1", pos1, left1), O_2=insert ("string2", pos2, left2), and then there should be three conditions as follows:

1. pos1 = pos2 ,left1 = left2
2. pos1 = pos2 ,left1 != left2
3. pos1!= pos2

Now the relationship is defined as follows:

Assuming that operation O_1 = insert ("string1", pos1, left1), O_2 = insert ("string2", pos2, left2), then $O_1 \parallel O_2$, if and only if pos1 = pos2, left1 = left2.

Then, according to the definition, it should be O_1-> O_2, $O_3 \parallel O_4$ in Figure1.

5.2 Analysis of Operation Execution

In the string processing, the user's operation is based on strings, and the AST transplantation strategy is an algorithm that replaces the single character with the whole string.

This paper proposes the splitting strategy to resolve the inconsistencies that may appear. When there exists operation involving in internal insertion, split the string basing on the operation parameters and then use the AST algorithm to solve the problem. And in order to keep the causal relationship between the characters after splitting, the timestamp of the character after the split remains the same to that divided before.

5.3 Design of Consistency Maintenance Algorithm

For the case shown in Figure 1, this paper modifies AST algorithm to adapt to the new environment. The string is viewed as an indivisible unit; each operation maintains the original time-stamp. For the case shown in Figure 2, this paper views the string in the position "pos" as a new text document and operate the operations at the new document. The string is split based on the Insert position, and then two substrings are got and the substrings get the same stamp of the string split before. Of course, there exists Insert operations at different positions, as shown in Figure 3, the strategy is similar to case 1, and the difference is that the Insert is not in the same document and there exists no concurrency. In Figure 4, the string to be deleted is viewed as a whole and then the AST strategy is transplanted directly. In Figure 5, the string "world" will be split at the position "left" and "left+n" respectively.

Delete is similar to Insert, but the length of the string to be removed will be considered, just like the cases shown in Figure 5 and Figure 6. If the delete operation is done in one string, split the string into three substrings according to the position parameters "left" and "left + n", and the substring in the middle will be marked as ineffective. If two or more strings are involved, split the string according to the parameter "left" first, and find the last string to be deleted according the length of the to-be-removed string and split the string, and the strings between the second substring of the first string and the first substring of the last string all set to be ineffective.

As for the copy-paste operation, follow the Insert operation strategy to handle with it. If the paste string targets only one string, it can be solved according to the strategy discussed previous, but if the paste targets more than one string, there needs to split the operation into more insert by the non-word character.

The operation will be executed according to the following algorithm: Firstly, judge the type of the operation, if it is Insert and the Insert position is between two strings, then call the function "Insert_Execute"; if the Insert position is in one string, then call the function "Divide_Insert"; for the Delete, the operation's type also needs to be considered, when deleting the entire string, call the function "Delete_Execute"; when deleting some characters from one string, the function "Divide_Delete" ; if it is a paste operation, the function "Divide_Paste" will be called.

According to the analysis above, the basic algorithm is presented as follows:

```
Procedure1：String_Execute(S, O) // Handling operations, S represents the current document status, and O is the operation to be executed.
Begin
  If (O is insertion)
    If(O.left=0)  //Insert among the strings
        Insert_Execute(S, O)
    Else      //Insert internal the string
            S=HB.pos
            Divide_Insert(S, O)
Else if (O is deletion) //O is deletion
    If(O.left=0)  //delete the entire string
        Delete_Execute(S, O)
    Else  //delete characters from the string
        Divide_Delete(S, O)
Else if (O is paste) //O is copy-paste
    Divide_Paste(S, O)
End
```

This paper redefines the operation, it is necessary to convert the operation to a form suitable for AST algorithms, and the function "Convert" achieves this goal, after that the AST control algorithms can be called. The algorithm is described as follows:

```
Producer2：Insert_Execute(S, O) // Insert between the strings, the document status is S, insertion O
Begin
  O' <- Convert (O)      //convert the operation into a form suitable for AST
  Control-Algorithm (S, O')  //call the AST control algorithms
End;
```

The case that inserts a string into an existing string needs to split the string based on the insert position, the timestamps are the same to that split before. After the division, the substrings are viewed as a new document, and then the situation is equal to the situation that inserts strings between strings, the algorithm is described as follows:

```
Producer3：Divide_Insert(S,O)   //Insert  the string internal the string S and O.string is to be inserted
Begin
  s1, s2 <- Divide(S) //split the string basing on the insertion position and get two substrings
  S'= {s1, s2}
  Insert_Execute(S', O)
End
```

The strategy that deletes the entire string adopts the single character strategy, and the characters are marked ineffective. The algorithm is described as follows:

```
Producer4：Delete_Execute(S, O) // Delete a string, the document status is S, O is deletion
Begin
    Set HB.pos ineffective // mark the characters to be deleted ineffective
End
```

When there needs to delete some characters from a string, first judge whether the Delete involves multiple strings: if not, the string at the delete position is split into three substrings, and the middle substring is marked ineffective; if so, judge the number of the involved string firstly, and then find the first and the last string involved, then split them, and all the characters involved should be set ineffective. The algorithm is described as follows:

```
Producer5：Divide_Delete(S, O) // Delete some characters from the strings; the string need to be deleted is O.string
Begin
If (O.left+O.n <= S.len) //delete some strings internal the string
        s1, s2, s3 <- divide(s) //split the string
        Set s2 ineffective
Else //Delete involves more than one string
        s1, s2 <- divide(s)
        If (n-(S.len-left+1) < HB.(pos+1).len) // deletion involves two non-null strings
                s' = HB.(pos+1)
                s3, s4 <- divide(s')
                Set s3 ineffective
        Else //Delete involves two strings
                Count = judge (n-(S.len-left+1))
                S' = HB. (pos +count+1)
                s3, s4 <- divide(S')
                Set HB.pos, HB.pos+1 ...HB.(pos + count+1) ineffective
                Set S3 ineffective
End
```

While handling copy-paste, the strings will be split into multiple Inserts and then be inserted into the document orderly. When inserting, the function “Insert_Execute” will be called and the algorithm is as follows:

```
Producer6：Divide_Paste(S, O) // function to achieve pasting a string in a document
Begin
    O1, O2...Oi <- divide (O) // operation is split into three of insertions, and executed orderly
    Repeat
    Insert_Execute(S,Oj)
    Until j=i
End
```

6. Case Analysis

According to the algorithm above, the results of all the cases before will be got. In Figure1 at site1, O_1, O_2 and O_3 execute according to the algorithm and the document is “Chinahelloworld”, and then O_4 arrives. For $O_3$and O_4 are concurrent, and the “Insert_Execute” is called to put the “hello” after “China”, then the document is “CHINA hello hello world”;at site2 O_1, O_2 and O_4 execute and get the document “hellohelloworld”, and then O_3 arrives and “Insert_Execute” is called to put “CHINA” before “hello”,then the document is “hellohelloworld”, which consistent with that at site1.Similar to Figure1, the execution in Figure2 and other examples before will follow the algorithm and maintain the intention of all the operation. Thus we will not give describe.

7. Efficiency Analysis

The analysis of the algorithm shows that the cost of the algorithm is mainly in two aspects, one is the retracing process, and the other is the execution of the operation.

The first section is the analysis of the execution efficiency. For Insert operation the function "range-scan" selects the Insert position, and for the Delete operation, there only needs to modify the effective or ineffective flag of the nodes, and it is completed in time O(1), the complexity of the function "range-scan" is O (m), where m is the number of ineffective nodes [6]. The process seeking the position is the process of traversing the list and the complexity is O (n), where n is the number of the nodes in the list.

The following is the analysis of the efficiency of retracing process. The operations that have not been executed are stored in the request queue Q, and will be removed from the queue after their execution. Assuming that the length of Q is h, that's to say the retracing process will modify at most h nodes' identifier. Assuming that the number of operations attaching to each node is d, then the complexity of first retracing process is O (d*h). The first retracing changes the identifier of the node, and will recover them in the second retracing; the complexity of recovery process is O (h +d).Based on the above analysis, the actually complexity of the execution of the Insert operation is O (m +d* h+ h +d), and the Delete is O (d*h+ h +d).

Assuming that there are k strings, with each probabilistic length of the string is n, h and d are consistent with the definition previous. In the single character editing system, the complexity of Insert operation is O (m*n +d*h*n +h*n + d*n), and the Delete operation is O (d*h*n + h*n + d*n). And in a string text editing system, the Insert operation complexity is O (m +d*h + h + d), and the Delete operation is O (d*h +h +d). Obviously the latter will be more efficient.

8. Experiments

To verify the algorithm, an experiment is made. The experiment simulates a document editing process; three users edit the same document simultaneously. First simulates the single character mode. Then the users simulate the string mode, the same three users, reading strings from the previous document, and write to a new document. The time of the completion of the same document is recorded in both editing modes. Five groups are experimented.

The experimental environment is described in the following: Platform: ubuntu11.10, Language: Linux C, CPU clock speed: 2.26GHz, Memory: 2G. The following chart presents the experiments results.

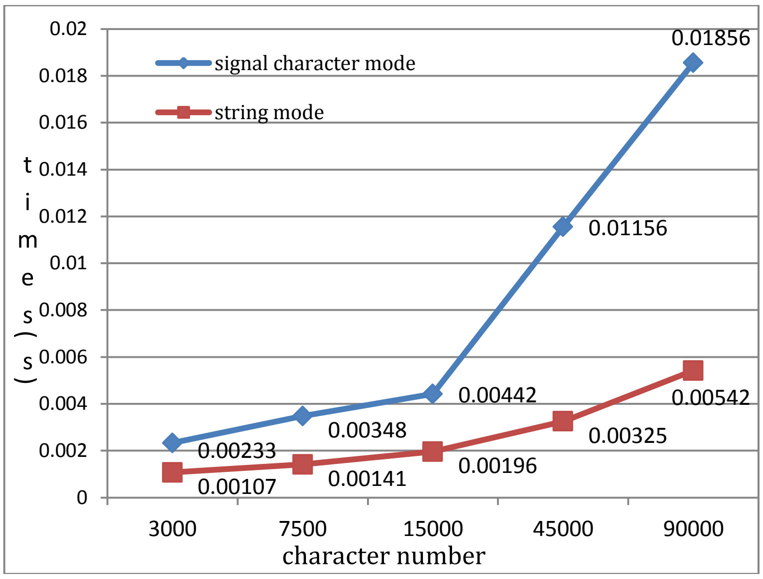

Figure 9. Consuming Comparison between Character Mode and String Mode

From Figure 9, we can see that the string mode is a less time consuming mode in the completion of the same document compared to the signal character mode, and it is more efficient, especially when there are a large number of characters, the advantage is more obvious.

9. Conclusions and Future Work

This paper proposes a splitting strategy combined with AST to solve the problem in the string text editing system, and compared to the previous strategy, it improves the processing efficiency greatly and at the same time meets the CCI consistency model [7].

This article firstly analyses various issues that need to be considered in the string text editing system, and lists various situations that may appear in string handling. Then it discusses the solutions to these problems and finds that when handling the whole string operations, The next part of the paper details a lot of solutions to such problems and proposes an algorithm based on splitting strategy, and also analyses the efficiency of the algorithm to prove the efficiency improvement of the algorithm.

This paper extends the definitions of operations; at the same time the paper continues the logic of the dependent and concurrent operations, but to the concurrent relationship, the paper redefines it to make the definition be more complete after the introduction of mark attaching to it. In the algorithm design phase, this paper analyses the questions raised at the beginning of the article respectively, and verifies it through specific examples and practical experiment.

A complete text editing system is not only able to execute the operations, but also should have error modification. All the contents in this article only settle the process to execute the operation, but do not involve the undo operations, this is the work needs to be done later.

Acknowledgement

The work is supported by the National Natural Science Foundation of China(NSFC) under Grant No. 61202376, "Chen Guang" project sponsored by Shanghai Municipal Education Commission and Shanghai Education Development Foundation under Grant No.10CG49, Innovation Program of Shanghai Municipal Education Commission under Grant No. 13YZ075, Shanghai Key Science and Technology Project in Information Technology Field under Grant No. 14511107902, Shanghai Leading Academic Discipline Project under Grant No. XTKX2012, and Shanghai Engineering Research Center Project under Grant No. GCZX14014 and C14001.

References

[1] C. A. Ellis and S. J. Gibbs, “Concurrency Control in Groupware Systems”, **(1989)**, ACMO-89791~317-S/89/0005/0399.

[2] C. Sun and C. S. Ellis, “Operation Transformation in real-Time Group Editors: Issues, Algorithms, and Achievements”, In Proc of 1998 ACM Conference on Computer-Supported Cooperative Work, Seattle, USA, **(1998)** November 14-18.

[3] D. Li and R. Li, “Transparent sharing and interoperation of heterogeneous single-user applications”, In Proc. of the ACM Conf. on Computer-Supported Cooperative Work, **(2002)** November, pp. 246-255.

[4] H. Gu, X. Xie, Q. Lv, Y. Ruan and L. Shang, “E-tree: Effective and efficient event modeling for real-time online social media networks”, In Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011 IEEE/WIC/ACM International Conference on, vol. 1, **(2011)**, pp. 300-307.

[5] N. Gu, J. Yang and Q. Zhang, “Consistency Maintenance Based on the Mark & Retrace Technique in Groupware Systems”, GROUP’05, **(2005)** November 6-9, Sanibel Island, Florida, USA.

[6] C. Sun, “Undo as Concurrent Inverse in Group”, Editors ACM Transactions on Computer-Human Interaction, vol. 9, no. 4, **(2002)** December.

[7] G. Abowd and A. Dix, “Giving undo attention”, Interact, Comput., vol. 4, no. 3, **(1992)**, pp. 317–342.

[8] N. Vidot, M. Cart, J. Ferrie and M. Suleiman, “Copies convergence in a distributed real-time collaborative environment”, In Proc. of the ACM Conf. on Computer-Supported Cooperative Work, **(2000)** December, pp. 171-180.

[9] H. F. Shen and C. Sun, “A flexible notification framework for collaborative systems”, In Proc. of the

[10] ACM Conf. on Computer-Supported Cooperative Work, **(2002)** November, pp. 77-86.
[11] H. Gu, M. Gartrell, L. Zhang, Q. Lv and D. Grunwald, "AnchorMF: towards effective event context identification", In Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, ACM, **(2013)**, pp. 629-638.
[12] H. Gu, H. Hang, Q. Lv and D. Grunwald, "Fusing Text and Frienships for Location Inference in Online Social Networks", In Web Intelligence and Intelligent Agent Technology (WI-IAT), IEEE/WIC/ACM International Conferences on, vol. 1, **(2012)**, pp. 158-165.

Authors

Liping Gao, She graduated from Fudan University, China with a PhD 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.

Wenfeng Tang, He is a postgraduate student in University of Shanghai for Science and Technology. He obtained his BSc degree in Electronic Information Engineering from Henan University of Science and Technology, China. His current research interests include CSCW, collaborative design and collaborative computer.