1998 International Computer Symposium
Workshop on Computer Networks, Internet, and Multimedia
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

IN SUPPORTING OF DISTRIBUTED CO-EDITING ENVIRONMENT

Chin-Hwa Kuo, Chia-Lin Chio, Chen-Hsiang Yu, and Wen-Yang Hsia

Department of Computer Science and Information Engineering

Tamkamg University, Tamshui, Taiwan ROC

E-mail: chkuo@mail.tku.edu.tw

ABSTRACT

In the present paper, we focused on the development of multi-participants co-editing scheme. Due to the participant common interests may have intersection, data conflict occurs. To resolve this issue, conventional approaches face the tradeoff between data consistency and concurrent operation. We analyzed the possible operations of text editing. We found out that many undo processes can be avoided. In other words, data consistency is still ensured without sacrificing system performance. Based on the analysis, we developed an algorithm, called Temporal And Spatial Data conflict detection (TASD), in solving data conflicts effectively. The proposed algorithm can resolve many data conflicts without the undo operation.

1. Introduction

Due to the advances of computer and networks, today a usual personal computer is able to manipulate multimedia processing and networking. As we have seen, not only text, images, graphics but also audio and video can be transmitted digitally through the Internet [7]. In other words, new networked multimedia applications, such as shared white board, co-editing, and co-design will eventually get into our daily life. Hence, there is no doubt that the way of people remote communication and work will dramatically change in the near future. In the present paper, we elaborate the issue of multi-participant co-editing in support of collaborative working environment.

Research of the collaborative computing is classified into the category of computer supported cooperative work (CSCW) or groupware [6]. Briefly speaking, the technique issue of CSCW intends to integrate computer and communication network technologies in supporting distributed cooperative working environment [11][12][13]. As defined in [8], CSCW is the field concerned with the design of computer-based systems to support and improve the work of groups of users engaged in common tasks or objectives, and the understanding of the effects of using such systems. With this kind of working environment many advantages can been foreseen.

The development of a CSCW involves *technique* as well as *culture* issues [16]. In the technique part, CSCW specialists classify the systems that support group working according to the *time/space matrix* [7][17]. There are four classes, from asynchronous collocated, asynchronous distributed, synchronous collocated, to synchronous distributed working environments. In the present paper, we deal with the issue of *data consistency* in the multi-participant synchronous distributed working environment. In the co-editing environment, due to the common interests of the participants may have intersection, data conflicts occur. Thus, a management mechanism is required dealing with this issue and improving the working environment as What You See Is What I See (WYSIWIS). In conventional information protection systems, data consistency issue is ensured by using access matrix [10]. That is a matrix, which defines the access right of every user to each object. It can be classified into two modes, namely *serialization mode* and *locking mode* [9][19]. In the serialization mode, the data consistency relies on the operation sequence of the participants. In the locking mode, the request command is required before any operations. After operation, the unlock process is also required. Both modes have advantages and drawbacks. In the serialization mode, as shown by *Karsenty and Beaudouin-Lafon,* some sort of mechanisms required to deal with command operations sequence issue [2]. The transform mechanism is used to eliminate the unnecessary *undo* processes [1]. In the locking mode, the data conflict phenomena is significantly reduced. However, the chance for concurrent operation is also excluded. Based on these two methods, some extensions were made, e.g., the operation transformation scheme [3], turn-taking protocol [3], and dependency-detection [19]. The above

approaches face the tradeoff between data consistency and concurrent operation, i.e., ensure data consistency we may sacrifice concurrent operation and vise visa.

We propose a novel approach, which eliminate the above mentioned tradeoff. We allow the participants work *concurrently*. Note that data conflict occurs as the participants work on the same *temporal* and *spatial* point. In other words, some undo processes can be eliminated when the participants only complete the same spatial point at different time. We classify the possible operations of text editing into two categories, namely, General Modification Operations (GMO) and Attribute Modification Operations (AMO). The GMOs are that operation will change text content, such as insert and delete. The AMOs are that operation will not affect text content, for example change font, underline etc. Based on this analysis, we design a Serial Judgement Method to justify events temporal sequence. Then we develop a *Temporal And Spatial Data conflict detection* (TASD) algorithm to resolve data conflict issue in the multi-participant shared working environment. Many undo processes can be avoided by our approach. In other words, data consistency is still ensured without sacrificing system performance.

To realize our algorithm, we design the event data structure and transmission data format. During operation, the reception events are stored in the buffer called Received-Queue. The events generated in the local end are stored in the buffer called Finished-Queue. Comparing the contents within these two buffers, the system is able to efficiently justify the data conflict occurs or not in each event. If occurs, further analysis procedure follows. In this manner, TASD algorithm resolves many unnecessary undo processes.

The rest of this paper is outlined as follows: In section 2, we present an overview of a CSCW and our system architecture. In section 3, we present our design work in detail. Examples are given to demonstrate the effectiveness of our scheme. An implementation on Windows 95 platform is given in section 4. Finally, we conclude our work and describe future work in section 5.

2. System Overview of Collaborative Computing

2.1. Synchronous collaborative computing

A typical session processing subsystem can be depicted as Fig. 1. It consists of the following mechanisms: (1) network agent, (2) synchronization agent, (3) floor management control agent, and (4) application agents, such as co-editing, co-drawing, voice, and video tools. With these elements, one is able to integrate many valuable networked multimedia applications. The same system arcitecture can be applied to both synchronous and asynchronous cases. In the present paper, we focus only on the *synchronous distributed* co-editing case.

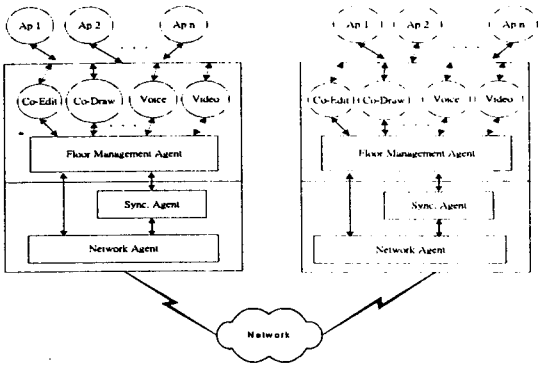


Fig. 1 Networking architecture for CSCW

2.2. The conflict issues

If more than two participants have the same area of interest, it has potential of data conflict. The simple solution to this problem is performing the *undo* process. However, the operation of undo process will decrease the working efficiency, in particular, the unnecessary undo. Conflict resolving mechanisms do require. To facilitate the following presentation, we term that the completion of an operation during co-editing in the local end as *event*, e.g., insert, delete operation. As shown in Fig. 3, the participant first markups the area of interest and then performs the operation. An event consists of information such as mark up area, operation type, and so on at the local end. When the participant presses the enter key then an event is generated and transmitted. Hereafter, we simply name the markup area as *area*. When one manipulates on characters D, E, F, the location of character C is called the *starting position* (*start_po*) of the area.

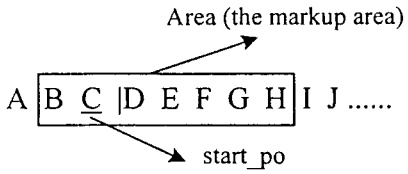


Fig.3 The relationship of area and start_po

Assuming that the participants have equal access rights and work concurrently, i.e., an open floor control policy. We analyze these data conflicts from both *temporal* and *spatial* domains. This relationship is depicted in Fig. 4. If the relationship of two events locates in the second quadrant or the third quadrant, due to no spatial intersection which implies no data conflict. For the events which has spatial intersection, it will locate in the first quadrant and the fourth quadrant. If it locates in the fourth quadrants, then two events did not occur at the same time. This also implies no data conflict. Thus, only the events locate in the first quadrant have data conflict.

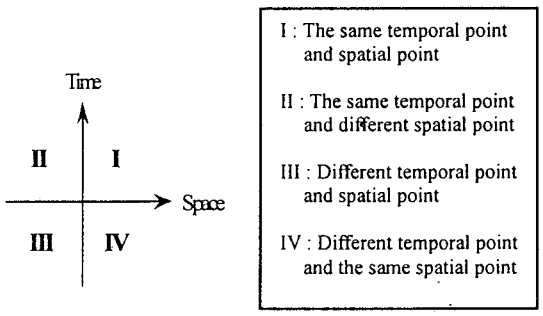


Fig. 4 Temporal and spatial relationship of two events

In other words, if both events the temporal points are different, the data conflict may not occur. Note that data conflict occurs if both events have the same temporal and spatial point. If not the case, the conflict should be resolved. Indeed in many cases, the undo is unnecessary. Hence, the data conflict mechanism requires managing these different cases. In the following section, we analyze the different cases of text co-editing and develop an algorithm to resolve data conflict. We propose an algorithm called Sequence Justification Method (SJM) to justify the events temporal sequence. If the sequence exits, the data conflict can be resolved without performing undo process. When the sequence does not exist, we analyze the spatial relationship of both events. In some cases, data conflict can be resolved without performing undo process. If data conflict cannot be resolved from the above process, the undo process is taken.

3. Design of co-editing system

We make use of *hybrid* strategy to deal with data conflict issue in the co-editing environment. In usual case, each site sends/receives the corresponding information and processes independently. As data conflict occurs, if this cannot be resolved by the local computing machine, a pre-assigned chair machine is responsible for handling such a situation. In this manner, both local machines and chair machine are responsible different data conflict cases. And then the unnecessary undo processes can be reduced. We design a mechanism called *message queue* to accomplish the above tasks.

3.1 Message Queue

To resolve data conflict, we assign two buffers to each participant computer. The reception events are stored in the buffer called *Received-Queue* (RQ). The events generated in the local end are stored in the buffer called *Finished-Queue* (FQ). The operation of the FQ and RQ is shown in Fig. 6. The event will be stored in the participant local FQ and transmit to the other participant's RQ. The events in RQ need to be compared with the events in FQ. If no conflict, the event is called the *complete event*, which is pushed onto FQ.

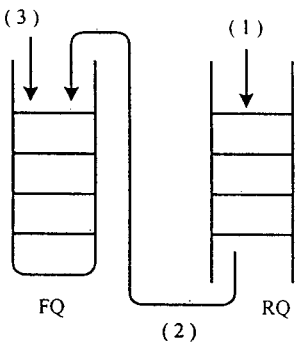


Fig. 6 The processing of events in FQ and RQ

(1) Receive Events from other sites
(2) Events finished and pushed into FQ
(3) Events happened in local site and finished

3.1.1. Operation of message queue

The operation of message queue can be stated as the follows:

(1) After the completion of an operation, the event is sent into FQ at the local end and also sent out to other sites.

(2) The received event will be stored into RQ.
(3) Comparing the received event in RQ and each event in FQ:
   (a) If no conflict detected, push the received event into FQ.
   (b) If conflict occurs, further analysis is required. If the event in RQ requires performing undo, the event will be deleted from RQ and this message is sent to the other sites. If the required undo event is in FQ, the same procedure will be performed first. Then, the corresponding event in RQ will be operated and pushed into FQ.

3.1.2. Maintenance of message queue

As we mentioned before, we make use of FQ to store a *complete event* and the RQ to temporarily keep a reception event. These complete events require to be kept for later comparison. Otherwise, a late coming event may still conflict with the previous complete event. The size of RQ is finite and it can be selected as proportional to the number of the participants in a session. In order to keep all complete events, the FQ may need infinite large buffers. Thus, it is impractical.

The following algorithm that realizes the above cases to ensure the operation efficiency of message queue.

```
if ( RQ == FULL )
   System Suspend to deal with RQ and
   clear FQ ;
else
{
   if ( FQ >= Max_Size )
      System Suspend to deal with RQ and
      clear FQ ;
   else
      progress( ) ;
}
```

3.2. Detection of temporal conflict

3.2.1. Serial Justification Method(SJM)

In the determination temporal sequence of two events, event X and event Y, we use the notation order(X,Y) represents X is a complete event in FQ and Y is an event in RQ. The temporal sequence of X is ahead of Y. The SJM relies a selected chair machine in the session in charge of management the order(X,Y) set. If the determination of temporal sequence of events is required, the corresponding site sends an order(X,Y) to chair machine. At the same time, the chair machine creates a list to record this order set. If the other sites happened have the same case, they also send order(X,Y) to chair machine. On the other hand, the other sites may record the order (Y, X). Then, as shown in Fig. 8, the chair machine makes another list for order(Y, X).

List 1 for order(X,Y)	
FQ	RQ
X	Y
X	Y
⋮	⋮

List 2 for order(X,Y)	
FQ	RQ
Y	X
Y	X
⋮	⋮

Fig. 8 Lists for order(X , Y) and order(Y , X)

In a N-participant session, assuming that two sites operate on the same area, site A generates event E_a and site B generate event E_b. Then site A transmits order (E_a, E_b) and site B transmits order (E_b, E_a). These lists are stored in the chair machine for later comparison. If both events do not occur at the same time, by inspecting the order set, some undo processes can be avoided. This process can be summarized as follows. Assuming that event E_a occurs before event E_b, then the lists in the chair machine satisfies

(1) Chair machine has (N-1) order(E_a, E_b), where N is the number of participants.
(2) There is no order (E_b , E_a) list in the chair machine site.

Note that after the chair machine received (N-1) order(E_a , E_b), it also needs make sure that event E_a existed in the FQ of site B. The SJM algorithm is as follows for N-participant case.

```
Serial Judgement Method (SJM) Algorithm:
while(number( order(Ea , Eb )) < N – 1 ) {
   if( number( order(Eb , Ea ) ) > 0 )
      return FALSE ;
}
if ( Event Ea of site B exists in its FQ )
   return TRUE ;
else
   return FALSE ;
```

3.3. Detection of spatial conflict

We have described the case of temporal conflict. If two events have temporal conflict, further spatial analysis does require. We detail the cases in this section. Common text editing operations can be classified into two categories:
(1) General Modification Operation (GMO):

The GMO are the operations, which change text content. It includes insert, delete, copy, cut, and paste. Due to copy, cut, and paste operations are variants of insert or delete, so they will be excluded in the later analysis. We only discuss the conflict of the first two operations.

(2) Attribution Modification Operation (AMO):
The AMO are the operations, which do not affect text content. It includes the operations such as change-font, change-size, change-color, and change-shape.

3.3.1. Analysis of the spatial conflict

From the spatial domain viewpoint, depending on the starting points of the corresponding events, we have two cases to be discussed.

(1) *Both events have the same starting position*
Assuming that two events E_a and E_b generated simultaneously from Site A and Site B respectively, then Site A sends order (E_a , E_b) and Site B sends order (E_b , E_a). In this case, data conflict usually occurs. However, some exceptions exist, e.g., both events perform the same operation. Then, the undo process is unnecessary.

Algorithm 1: Events have the same start_po

```
if (Event1.op != Event2.op || Event1.len !=
      Event2.len || Event1.data != Event2.data)
    if (SJM Algorithm returns FALSE)
/* We cannot detect the serialization of these
            two events by "SJM" Algorithm */
        Conflict;
    Else
        No Conflict
Else
/* The operations of these two events are equal */
    No Conflict;
```

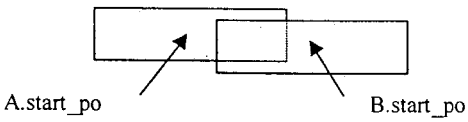
(2) *Events have different starting positions*
In the case of both events have different starting positions, the conflict resolving can be illustrated as follows.

Step 1: When a new message received, it is extracted from the RQ. This event is compared to all the events in the FQ. If there is no intersection, the event is pushed into FQ. Otherwise, record the area and go to step 2.

Step 2: Based on the classification of the operations, we have three cases:
(a) Both events are GMO, go to step 3.
(b) Both events are AMO, go to step 4.
(c) One event is GMO and the other event is AMO, go to step 5.

Step 3: GMO vs. GMO
Both events start positions (*start_po*) and lengths are extracted. Assuming event *A* has starting position is ahead of event *B*.

(a) Both A.start_po. and B.start_po. are not in the intersection area:
Due to A.start_po. and B.start_po. do not locate in the intersection area, the delete or insert operations of A or B will not affect the other event.



(b) Only A.start_po is in the intersection area:
The result is the same as (1). The operations of A or B do not affect each other.

(c) Only B.start_po is in the intersection area:
As the starting position of B located inside the area of A, the conflict may occur. If event A performs insert operation, conflict will not occur. If event A performs delete, depending on the delete length to justify the data conflict issue. In this case, if the location of A.start_po adds the delete length of event is greater than the location of B.start_po, this implies data conflict. On the other hand, it implies no data conflict.

(d) Both A.start_po and B.start_po are in the intersection area:
In the case of both events starting position located in the same area, conflict may occur.

Step 4: AMO vs. AMO
In this case, both events are AMOs. Both events operation can be perform even they have intersection area. All participants operate up to six different AMOs without any data conflicts. However, if both events do the same AMO, the conflict occurs.

Step 5: GMO vs. AMO
In this case, one event performs GMO and the other event performs AMO.

(a) Insert vs. AMO : In this case, data conflict will occur when the starting position of event insert is inside the area of AMO.

(b) Delete vs. AMO : We have the following four cases.

(1) Both Delete.start_po and AMO.start_po are outside the intersection area. They do not affect each other. Then there is no data conflict.

(2) Only Delete.start_po is inside the intersection area. We need to justify whether the operation of AMO affects Delete.start_po.

(3) Only AMO.start_po is insidet the intersection area. We only need to consider the delete operation is ahead of the AMO.

(4) Both Delete.start_po and AMO.start_po are inside the intersection area. We consider when Delete.start_po ≠ AMO.start_po.

Step 6:
(1) Push the resolving event into FQ.
(2) Otherwise, undo.

After the analysis of every situation, we summarize our work. In the case of both events have different starting position, the data conflict resolving algorithm is given in the following table.

***Algorithm 2**: Events have different start_po (Assuming that Event A is ahead of event B)*

```
Case: Event A and event B are both GMO
  If ( CoArea.start ≤ B.start_po | |
       B.start_po < ( CoArea.start +
  CoArea.length ) )
      If ( A.op == Delete )
        If ( A.start_po + A.length ≥
  B.start_po )
          If (SJM Algorithm returns
  FALSE)
              Conflict ;
            Else
              No Conflict ;
        Else
          No Conflict ;
      Else
        No Conflict ;
    Else
      No Conflict;

Case: Event A and event B are both AMO
  If (A.op != B.op)
    If ( A.start_po + A.length ≥
  B.start_po )
      If (SJM Algorithm returns
  FALSE)
          Conflict ;
        Else
          No Conflict ;
      Else
        No Conflict ;
    Else
      No Conflict;
  Else
    No Conflict;
Case: One of Event A and event B is
  GMO, and the other is AMO
  Let the GMO event be G' and the AMO
  event be A'
  If (G'.op == Insert)
  /* Insert V.S. AMO */
    If ( A'.start_po ≤ G'.start_po ≤
  A'.start_po + A'.length)
      If (SJM Algorithm returns
  FALSE)
          Conflict ;
        Else
          No Conflict ;
      Else
        No Conflict;
    Else
    /* Delete V.S. AMO */
      If ( CoArea.start ≤ B.start_po &&
  B.start_po < ( CoArea.start +
  CoArea.length ) )
        If ( A' == A )
        /* AMO is ahead of GMO */
          If ( A'.start_po + A'.length >
  G'.start_po )
            If (SJM Algorithm returns
  FALSE)
                Conflict ;
              Else
                No Conflict ;
            Else
              No Conflict;
          Else
          /* GMO is ahead of AMO */
            If ( G'.start_po + G'.length ≥
  A'.start_po )
              If (SJM Algorithm returns
  FALSE)
                  Conflict ;
                Else
                  No Conflict ;
              Else
                No Conflict;
        Else
          No Conflict;
```

3.4. Temporal and spatial data conflict detection algorithm

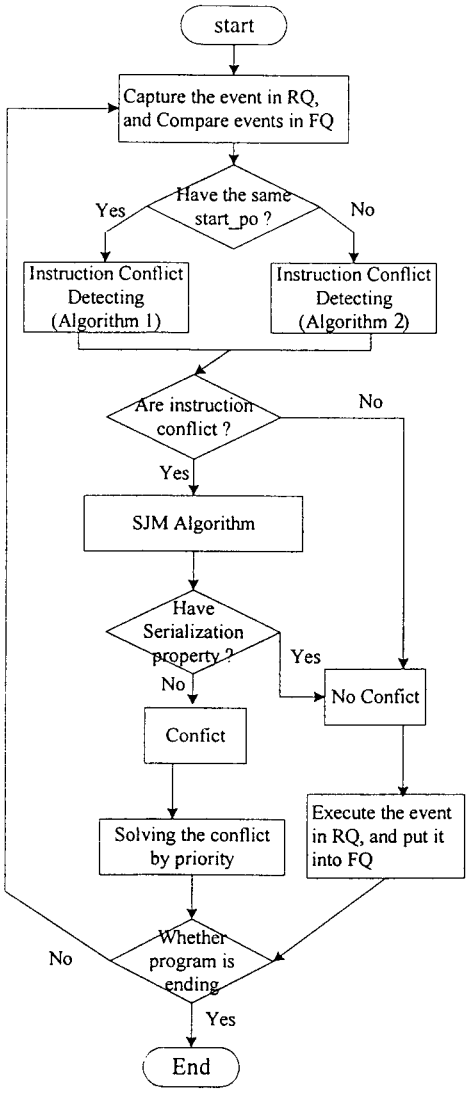


Fig. 9 TASD algorithm

4. Implementation

Each event has unique data format. The functions of each field are as follows:

Site_ID	Obj_ID	Area	OP	PO	Seq_Num	Data

(1) Site_ID: a 8-bit field to identify the participant.
(2) Obj_ID: a 4-bit field to identify each co-editing file on active.
(3) Area: the markup area, which includes area start and area length.
(4) OP: 4-bit field to classify the operation.
(5) PO: include event start_po. and operation length.
(6) Seq_Num: sequence number.
(7) Data: modified data.

We made use of Visual C++ to implement the co-editing environment on Windows 95 platform. The purpose of current version is for illustrating our design algorithm as well as for learning more experiences. Although audio and video tools are of benefit to data conflict avoidance, we have not integrated audio tool or video tool in this version yet. As shown in Fig. 10, first there are three working windows, i.e.,
(1) Co-editing window (common document window),
(2) Chatting window (public shared window), and
(3) Private window (personal editing window).

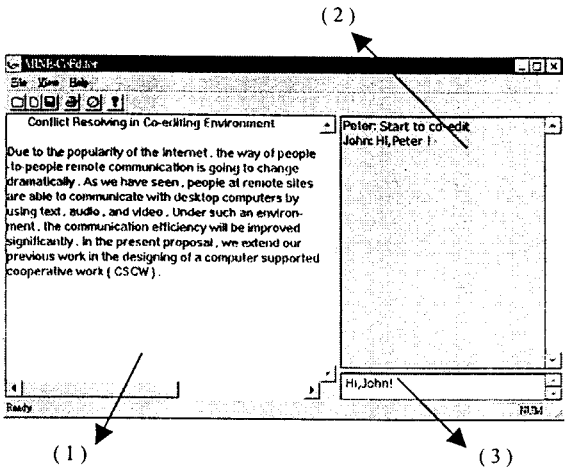


Fig. 10 MINE co-editor GUI

5. Conclusions

Data consistency is an important for shared co-editing system. Most approaches of this problem face tradeoff data consistency and concurrent processing. That is to ensure consistency some sorts of tern taking procedures are enforced. Thus, the system performance is decreased. In case of concurrent operations of multi-participants, data conflict may frequently occur. Thus, this also implies that sacrifice system performance. In the present paper, we analyze the operations of an editing process. These operations are classified into two categories, namely, general modification operation and attribute modification operation. And we design a message queue storage mechanism. Thus, whether an event, i.e., the editing process of a participant, is complete will be pre-processed on each participant machine. Finally, a data conflict resolving algorithm is developed. Many unnecessary undo processes are avoided. In other words, the developed system allows multi-participants co-edit at the

same time synchronously without sacrificing system performance. The developed algorithm is implemented on Windows 95 platform in Multimedia Information Networking (MINE) laboratory at Tamkang University.

In the present work, the operation of each event requires a markup pre-process. This preprocess may cause inconvenience to the participants who are not familiar with the operations. This easy way of this problem is to set default granularity. Then, the area is a complete sentence or a word. At this moment, we are not clear about its inference. This requires further investigation.

References

1. A. Prakash and M. Knister, "A Framework for Undoing Actions in Collaborative Systems", *ACM Conference on Computer-Supported Cooperative Work*, pp. 273-280, 1992.
2. Alain Karsenty and Michael Beaudouin-Lafon, "An Algorithm for Distributed Groupware Applications," *In Proc. of 13th IEEE Int. Conference On Distributed Computing System*, pp. 195-202, May 1993.
3. C.A. Ellis, S.J. Gibbs, " Concurrency Control in Groupware System," *In Proc. of ACM SIGMOD Conference on management of Data*, pp. 399-407 May 1989.
4. Chin-Hwa Kuo, Chi-Ming Chung, Hsi-Tsung Wang, "Designing of a CSCW System", *Proceeding of the 9th International Conference on System Research, Informatics and Cybernetics*, Baden-Baden, Germany, August 18-23, 1997.
5. C.-C. Chang, T.-C. Wu, "Controlling the Access Requests in an Information Protection System", *Information Processing & management*, vol. 29, No. 1, pp. 61-68, 1993.
6. C.A. Ellis, S.J. Glbbs, and G.L. Rein, "Groupware: some issues and experiences." *Communication of the ACM*, vol.34, NO.1, 1991.
7. Eve M. Schooler, "Conferencing and Collaborative Computing", *Multimedia System*, 4:210-225,1996.
8. Francois Fluckiger, *Understanding Networked Multimedia Applications and Technology*, Prentice Hall 1995.
9. Greenberg, S. and Marwood, D. "Real time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface." Research Report 94/534/03, 1994.
10. HongHai Shen and Prasun Dewan, "Access Control for Collaborative Environments", *CSCW 92 Proceedings*, pp.55-58, November 1992.
11. Hans-Peter Dommel and J.J. Garcia-Luna-Aceves, "Floor Control for Multimedia Conferencing and Collaboration", *Multimedia System 5*, pp.23-38, 1997.
12. Hans-Peter Dommel and J.J. Garcia-Luna-Aceves, "Floor Control for Multimedia Applications", position paper for the SIGCOMM'95 Technical Symposium (Workshop on Middleware).
13. K. Selcuk Candan, V.S. Subrahmanian, and P. Venkat Rangan, "Towards a Theory of Collaborative Multimedia", *Proceedings of MULTIMEDIA '96*: pp.279-282.
14. M. Stfik, G. Foster, D.G. Bobrow, K. Kahn, S. Lanning, and L. Suchman, "Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings", *Communication of the ACM*, vol. 30, No.1, 1987.
15. M. Handley and J. Crowcroft, "Network Text Editor (NTE) – A scalable shared text editor for the MBone", *ACM Computer Communication Review*, Vol. 27, pp. 197-208, 1997.
16. Mark O. Pendergast, " A Comparative Analysis of Groupware Application Protocols ", *ACM Computer Communication Review*, Vol. 28 , pp. 28-40 , 1998
17. Matthias Jarke and Clarence A.Ellis, " Distributed Cooperation in Integrated Information System ", *International Journal of Intelligent and Cooperative Information System*, Vol. 2 , No.1 , pp. 85-103 , 1993
18. Ralf Steinmetz and Klara Nahrstedt, *Multimedia: Computing, Communications and Applications*, Prentice Hall 1995.
19. Walter Reinhard, Tean Schweitzer, and Gerd Volksen, "CSCW Tools: Concepts and Architectures", IEEE COMPUTER, pp. 28-36, 1994.
20. Yi-Hsiang Huang, "Access and Concurrency Control for Collaborative Computing", MS Thesis Graduate Institute of Information Engineering Tamkang University, 1997.