Proceedings of the IASTED International
Conference Internet and Multimedia Systems and Applications
October 18-21, Nassau, Bahamas

FRAMEWORK FOR COLLABORATIVE MULTIMEDIA APPLICATION BASED ON CORBA

SEUNGGEUN LEE
Dept. of Computer Science & Engineering,
Inha University, Inchon, Korea
sglee@selab.inha.ac.kr

SEIHOON LEE
Dept. of Computer Engineering
Inha Technical College, Inchon, Korea
seihoon@true.inhatc.ac.kr

CHANGJONG WANG
Dept. of Computer Science & Engineering,
Inha University, Inchon, Korea
cjwangse@dragon.inha.ac.kr

Abstract

In recent years, as Information superhighway and advancement of computing power, it has been increasing requirements about CSCW that is based on the distributed multimedia. And, because the policy of interaction in collaborative application is implemented in application, application must be implemented again if the policy of collaborative application is changed. It is big problem in environment that policy is changed dynamically.

We design CORBA based framework for collaborative distributed multimedia application. Framework supports session management, event notification and transmission of multimedia stream data for collaborative multimedia application. And framework support the functionality which developer can redesign policy of collaborative application dynamically. Especially, because CORBA offers a useful Framework to develop the distributed application, the developments of distributed multimedia application based on CORBA is progressed actively.

Keyword : Collaborative Systems, Distributed Multimedia, CSCW, CORBA

1. Introduction

In recent years, as Information superhighway and advancement of computing power, it has been increasing requirement about CSCW(Computer Supported Cooperative Work) that is based on the distributed multimedia of video conference or VOD[1,2,3]. Session management manages to interaction of multi users by coordination policies[3,4] in CSCW. Currently, developers typically implement subsystems to perform session management on a per-application basis[2]. So, coordination policy has problem which is not changeable after application developing. Also, this trend has in extra three characteristics of current session management systems, which are problematic. The first characteristic is that applications developers typically implement subsystems to perform session management when they build an application. There is little cooperation between applications or code reuse between developers. Second session management itself is subordinate to the centerpiece task the applications in a particular collaborative application are often not very robust, flexible, or powerful. Usually the session management facilities provide the minimal level of functionality to allow the application to perform in a collaborative setting. The last characteristic is per-application re-implementation of session management typically means that three are no facilities for altering session management behavior on a global scale. All of these problems are similar to those encountered by early applications before the advent of common programming interface for developing, for example, graphical applications. Common APIs enables applications built on those APIs to be developed more quickly and with greater standardization than would be possible otherwise.

In this paper, we'll design CORBA based framework for collaborative distributed multimedia application. On the designed framework, application developer can define function of coordination policies of session management dynamically. We apart coordination policy from application implementations. This method is participants explicitly adopt roles, and coordination policies are specified in terms of roles in widgets. Policies are implemented each collaboration site during execution. This method become to changeable coordination policies not only developing course, but also

execution, and make a solution the existed problem of session management.

2. Related Work

2.1 CORBA Telecom

The CORBA suggested by OMG is the applications development standard using distributed object technology, its current version is 2.3[5]. In the part of application development, developers are able to improve the interoperability of system and get advantages of simplifying the system implementation. Especially, CORBA Telecom is defined for the development of audio/video application. Fig 1 shows the basic CORBA Telecom architecture[6].

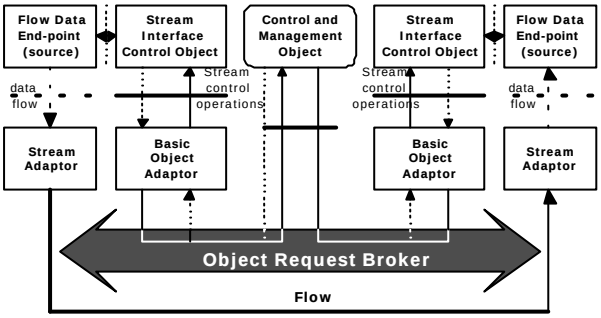

Fig 1. CORBA Telecom architecture

CORBA Telecom defines Flow Object specified a sequence of frames, Stream Object which is a set of Flows between Objects. The control of audio/video data is accomplished by Stream Interface Control Object through ORB. The actual Audio/Video Flow is specified in the form of accomplishing through Stream Adapter out of ORB. The definitions of CORBA Telecom make us use the benefits of distributed multimedia application development. As a result, it is expected to be more active in the distributed multimedia application development.

2.2 Collaborative System

Multiple users create opportunities for collaboration, rich and potentially unexpected interactions occur between users and applications-all of these contributes to the dynamic nature of collaborative software. The dynamism present in collaborative systems approaches intentionally the fluidity and richness of interactions among people in the physical world. Many applications require flexible session control[2,3,7] to allow the participants to dynamically join and leave a session, to take multiple roles simultaneously, and to smoothly shift between different roles. Many social rules are better built into the software as mechanical protocols to ensure that they are followed by all the participants, while some others are better left to the participants as social protocols to obtain more flexibility. This is opposed to traditional single-user or distributed systems in which human-computer interaction patterns are generally more predictable and can be governed by pre-coded mechanical rules.

Coordination policies are usually sensitive to the work style and organizational structure of individual groups. Different groups are often governed by different policies, and even the same group may need different coordination. Development of collaborative systems is typically heuristic in that the definition of coordination policies often involves intensive interactions between end users, developers and scientists from many fields. This is in contrast to a once-and-for-all solution that defines everything right from the beginning. With regard to this dynamic nature of collaborative systems, it is vital to start with a software architecture that is flexible enough to model collaborations and to accommodate the evolution of coordination policies both during the development of experimental systems and at runtime by the end users.

3. Design of CORBA based Framework

3.1 Overview

This framework is composed of session manager, event messenger, and communication manager. Fig 2 describes proposed framework for CORBA based distributed collaborative multimedia applications.

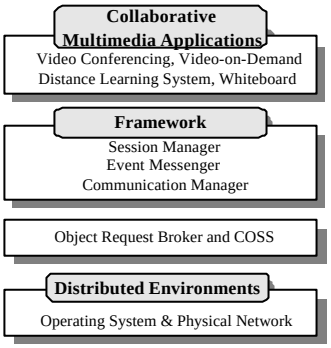

Fig 2. Overall structure of proposed framework

Session manager performs the interoperation with other users and controls of event messenger and communication manager according to coordination policy defined by user. Event messenger delivers the messages reached by session manager to the users who are the same collaboration and receives the messages reached by other users. Communication manager takes charge of Audio/Video data processing to manage Stream Object and Flow Object.

The framework is designed with CORBA Object on ORB. Actually video conferencing or distance learning system applications is running in framework. Through the development regardless of coordination policies which define the interoperation with other users, the changes of

coordination policy in the running have no effects to applications.

3.2 Session Manager

On the base of user definition policy, Session manager performs the activity conforming to each policy. Session manager is composed of Group Factory, Notification Module, Group Manager and Group Context. Group Factory maintains group lifecycle and lists of active groups. For request of group creation by user, it creates Group Manager and Group Context which maintain group information. If users want to participate the active group, they can get the object reference of group manager. Group manager maintains group context and perform handling for interaction among participants. Group Context maintains information of participants, of administrator, floor control. If participant join/leave the group or changing floor control, Group Context is switched and event is delivered to all participants by Notification Module. Fig 3 is Session Manager.

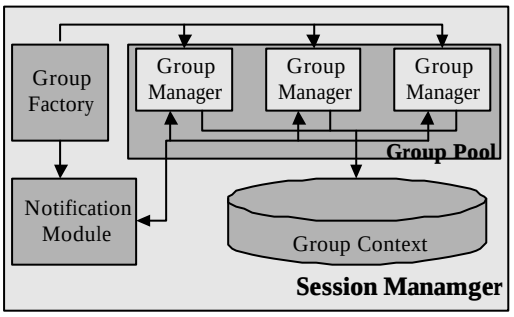

Fig 3. Session manager

Group Context maintains group states. Group context consists of participants ' list, chairman information, token owner information and other information about group. If group context switching is raised, engine send changed information to all participants through event messenger. Group participants play a role of chairman, member and inspector. Role is received by chairman when he takes part in group. The operations of group destroy and token control can be controlled by chairman.

Users can send message to participants when he has a token. In order to send messages, users request a token to chairman. If inspector request a token, Group Manager sends a error message to user through Event daemon. Fig 4 is interface of Group Management

```
interface GroupManager {
  boolean giveRole( in string passwd, in string ClientID, in Role role );
   boolean requestToken( in string ClientID );
  boolean releaseToken( in string ClientID );
  boolean giveToken( in string ClientID );
  boolean fetchToken( in string passwd, in string ClientID )
   Participant getParticipantfomation(in string ClientID);
   boolean TokenOwner( in string GroupID );
   ParticipantList getParticipantList(in string ClientID);
  boolean SendMessageToAll(in string ClientID, in any Message);
   Participant GetChairmanInfo(in string ClientID);
   Participant TokenOwnerInfo(in string ClientID);
    Participant GetGroupmemberInfo(in string ClientID);
   Participant GetInspectorInfo(in string ClientID);
 }
```

Fig 4. Group Manager interface

And, Group Manager contains coordinator which maintains each coordination policy of group. Group administrator, who mainly made group, defines group policy with PDL. PDL is defined for the description of group policy. The defined group policy by PDL(Policy Definition Language) is translated from rules and facts by translator. Fig 5 lists the templates.

```
(deftemplate ROLELIST (multislot rolelist))
(deftemplate STAGELIST (multislot stagelist))
(deftemplate OBJECTLIST (multislot objectlist))
(deftemplate EVENTLIST (multislot eventlist))
(deftemplate STAGES (slot (stage)))
(deftemplate RULE (slot tagetlist)
                  (slot mode)
                  (slot datatype)
                  (slot WHEN)
                  (multislot by)
)
```

Fig 5. Templates used by translator

Translator uses templates for this process. Rule and fact are inserted to knowledge base, and used for interactions among participants. Reasoning engine is matching user events with rule, and call actor which is implementation of policy of collaboration. It is implemented in Java object.

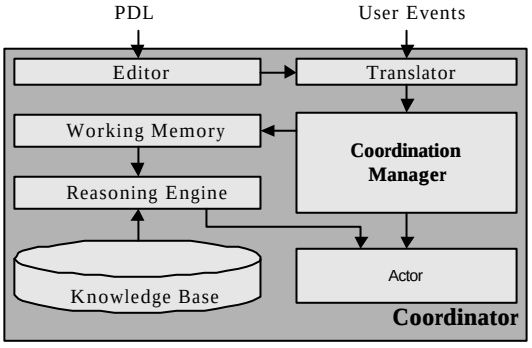

Fig 6. The structure of Coordinator

It can be called by Reasoning Engine in runtime. If group policy is changed in runtime, Reasoning Engine only calls the other actor by updating Knowledge Base. So, although group policy is changed, applications don't have to be changed or re-implemented. We use JESS engine(Java Expert System Shell) in Reasoning engine.

3.3 Event messenger

In CORBA Event Service, events raised in supplier are sent to all consumers such as broadcasting type. This

broadcasting type is expensive because some consumer may not need some events. And, total system efficiency is dropped. To solve this problem, we designed event messenger that sends events to users who are interested in those. To accomplish these operations, event messenger uses filtering process. Event's consumers register event type that he wants to receive to service using interface.

We designed two interfaces for filtering process. RegistrationAdmin interface defines the operations that event supplier/consumer registration and maintenance information in service. FilterAdministration interface defines the operations that filter creation/remove, filter/filter list maintenance.

3.4 Communication Manager

We design a stream based communication manager which support continuous media transmission during the user participated in distributed multimedia application. This manager's functionality is subset of CORBA Telecom.

Communication manager provides multicast transmission based on stream among clients. Stream communication service is composed of connector manager, communication factory. Port object, Stream object, and Connection object are dynamically created by request of communication manager and provide of multicast of stream. In order to transmit continuous multimedia data, user must create Port object using Communication Factory. Connector manager creates server side port object to receive stream data. Two ports, user's port and server side port, are bound by stream object. Stream object controls transmission of stream data between user port and server side port. If session manager or each user request transmission of continuous media, connector manager creates two port object and Stream object.

One Port object is data supplier and the other Port object is data consumer. Connection object is created after Stream objects are created, and maintains reference of Stream objects. Also, it performs the copy operation input stream to output stream. This Connector object is used for multicasting transmission of steam data.

Ports perform real media transfer. The basic control of ports is executed by stream object. The method related to stream transmission used by devices using transport interface. Port control interface is described with CORBA IDL, and provides control method of stream endpoint' behavior. Port control interface provides operations. : Lock(), unlock(), Start(), Stop().

Transport interface provides method used in transmission/reception of continuous media data in devices. This methods are added after compilation and are write_frame(), write_header(), read_frame(), read_header() and so on. Stream object provides abstraction of stream to application or other services. Applications control transmission of stream using this method. Fig 7 describes communication manager.

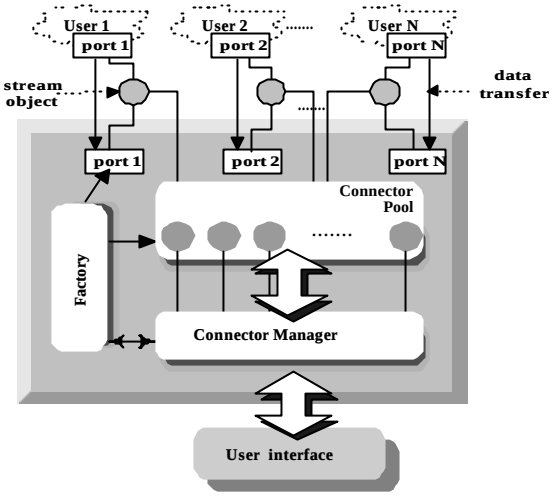

Fig 7. Design of Communication manager

4. Lecturing System using framework

In this chapter, we implement video lecturing system using designed framework. Environments for implementation are following.

- ORB Software : IONA OrbixWeb 2.3
- Developing Tools : Java Development Kit 1.1.7
- Real time Movie : software encoding/decoding module of MPEG-1

Lecturing system consists of whiteboard and video conferencing. The video/audio data are transferred by media stream server and communication manager.

Lecture request session manager in order that lecture creates group named by "Overview of CSCW". We supposed one lecturer two student, one participant and stage of instructions : Teaching, Questioning, Discussing, Reporting. Group attribute is protected mode, so participant must use password. Group policy are defined by follow.

- *If stage of instruction is Lecturing, request of floor control*

from students is denied.

• *If student has no floor control, voice can't be transmitted to the others.*

Then, As Fig 8, Group policy is describe using PDL. Fig 8 is translated with rules and facts by translator.

```
Classroom : "Overview of CSCW"
 ROLES : Lecturer Student Attendant
 STAGES : Teaching Question Discussion Reporting
 OBJECTS :
    OBJECT : video Stream USING_BY : ALL W
    OBJECT : sound Stream USING_BY : ALL W
     OBJECT : whiteboard Stream USING_BY : ALL W
 P_N : 4
     Lecture => 1
     Student => 2
     Attendant => 1
 PROTECTED : $as!123
  TOKEN_CONTROL
 RULES :
        RULE :
           LOCAL : IN :- REQUEST_TOKEN WHEN Teaching BY Student
                    => "Deny Request"
            LOCAL : IN :- sound WHEN Teaching BY Student Attendant
                     =>"Prohibit Speech"
```

Fig 8. Group policy described by PDL

Students can join Session View interface. Fig 9 is Session View. Session View presents information of active group list, function of group registration, unregistration, leave, join and query of group information.

Fig 9. Session View

Fig 10 is execution view of educational system. Lecturing system is composed by video stream and audio stream. Additionally, whiteboard is used in instruction. Video stream is transmitted regardless of floor control, but audio stream and whiteboard data can be transmitted only by participant having floor control.

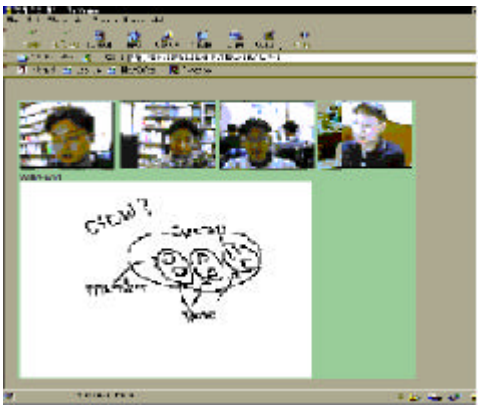

Fig 10. Lecturing System

5. Conclusion

Our approach is different from previous work in that we explicitly divide coordination policy and implementation. This separate process called by coordinator orchestrates the application tools as well as other computation modules to make a collaboration. Because Coordinator uses reasoning engine, coordination policy is added easily and it is adapted in interaction among participants dynamically. Also, session manager and communication manager help developers to develop CSCW in web environment. The point of our work is not to implement communication manager based full CORBA Telecom specification, but to provide a testbed for further study of how coordination policies are defined and how defined component of framework support development of collaborative distributed multimedia application.

References

[1] L. Fuentes and J. M. Troya, A Java Framework for Web-based Multimedia and Collaborative Applications, IEEE Internet Computing, 1999, 55-64

[2] D. Li, R. Muntz, COCA: Collaborative Objects Coordination Architecture, ACM CSCW'98 Proceedings, 1998.

[3] N. Kim, C. Wang, "CAFÉ: CORBA-based Framework for Distributed Multimedia Applications," Proceddings of the XV. IFIP World Computer Congress, 1998.

[4] H. P. Dommel, J. J. Garcia-Luna-Aceves, Group Coordination Support for Synchronous Internet Collaboration, IEEE Internet Computing, 1999, 74-80.

[5] OMG, CORBA Telecom Specification, OMG Document, 1998.

[6] OMG, The Common Object Request Broker: Architecture and Specification Revision 2.0, OMG Document, 1997

[7] W. Edwards, Policies and Roles in Collaborative Applications, ACM CSCW'96 Proceedings, 1996.

Acknowledgements

The authors wish to acknowledge the financial support of University Fund by Korean Ministry of Information & Communication in 1998.