A FRAMEWORK FOR SYNCHRONOUS AND UBIQUITOUS COLLABORATION

Kangseok Kim

Submitted to the faculty of the University Graduate School
in partial fulfillment of the requirements
for the degree
Doctor of Philosophy
in the Department of Computer Science
Indiana University
November 2007

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Doctoral Committee

Geoffrey Fox, Ph.D. (Principal Advisor)

Dennis Gannon, Ph.D.

Kay Connelly, Ph.D.

Sun Kim, Ph.D.

September 26, 2007

Acknowledgements

First, I would like to thank my advisor, Dr. Geoffrey C Fox, for his guidance, insight, support, and encouragement during my Ph.D. research. I will never forget the encouragement he gave me. Thank you! Dr. Fox. I am grateful to Dr. Marlon Pierce and Dr. Shrideep Pallickara for their valuable comments and help given to me during my rehearsal Ph.D. presentations. I also want to thank my research faculty Dr. Dennis Gannon, Dr. Kay Connelly, and Dr. Sun Kim for their valuable comments and questions. I am indebted to Global-MMCS project team members Dr. Wenjun Wu, Dr. Ahmet Uyar, and Dr. Hasan Bulut for their help and valuable work. I would like to thank Dr. Bryan Carpenter and Dr. Xiaohong Qiu for their valuable work and contributions on collaborative chess game project. I appreciate Mehmet Aktas, Mehmet Nacar, Beytullah Yildiz, Ahmet Sayer, Ahmet Mustacoglu, Ahmet Topcu, Ali Kaplan, Galip Aydin, Harshawardhan Gadgil, Jongyoul Choi, Seunghee Bae, and other colleagues at Community Grids Lab for their help, comments, and discussion. I greatly enjoyed working with you. Thank you! The computer science department and Northeast Parallel Architectures Center (NPAC) at Syracuse University, the computer science department and School of Computational Science and Information Technology (CSIT) at Florida State University, and the computer science department and Community Grids Lab (CGL) at Indiana University provided ideal collaborative work environment for conducting my research. I will miss those places, and faculty, students, and staff alike. I also thank TeraGrid (NCSA and SDSC) staff for providing me with the necessary computing facilities for conducting my research. Finally, I feel great gratitude to my parents and brothers for their devotion, love, support, and encouragement.

Abstract

With the advance of a variety of software/hardware technologies and wireless networking, there is coming a need for ubiquitous collaboration which allows people to access information systems independent of their access device and their physical capabilities and to communicate with other people in anytime and anywhere. Also, with the maturity of evolving computing paradigms and collaborative applications, a workspace for working together is being expanded from locally collocated physical place to geographically dispersed virtual place. Current virtual conferencing systems are not suitable for building integrated collaboration systems to work together in the same collaboration session. They also lack support for ubiquitous collaboration. As the number of collaborators with a large number of disparate access devices increases, the difficulties for protecting secured resources from unauthorized users as well as unsecured access devices will increase since the resources can be compromised by inadequately secured human and devices. Collaboration generally includes sharing resources. Mechanisms for dealing with consistency in application shared among collaborators will have to be considered in an unambiguous manner.

In this dissertation we address a number of issues related in building a framework for synchronous and ubiquitous collaboration as well as heterogeneous community collaboration. First, to make ubiquitous collaboration more promising, we present a framework built on heterogeneous (wire, wireless) computing environment. Second, to provide a generic solution for controlling sessions and participants' presences in heterogeneous community collaboration, we present a set of session protocols defined in

XML. Third, to provide a solution for controlling accesses to resources, we present a flexible and fine-grained access control mechanism based on Role Based Access Control model. Fourth, to provide a solution for maintaining shared state consistency at application level, we present a floor control mechanism which coordinates activities occurred in synchronously cooperating applications being shared among collaborators. The mechanism with strict conflict avoidance and non-optimistic locking strategy allows all participants to have the same views and data at all times. Finally, we give detailed experimental measurements to demonstrate the viability of the control mechanisms.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Collaboration is about interaction among people and between people and resources. With the advance of a variety of software/hardware technologies and wireless networking, there is coming a need for ubiquitous collaboration and access which allows people to access information systems independent of their access device and their physical capabilities and to communicate with other people in anytime and anywhere. Also, with the maturity of evolving computing paradigms and collaborative applications, a workspace for working together is being expanded from locally collocated physical place to geographically dispersed virtual place. Mobile computing paradigm [91] made ubiquitous access possible with the integration of wireless communication technology in anytime and in anywhere. With grid computing paradigm [30, 45, 46] which is about sharing resources, resources are distributed into workspaces and shared among geographically dispersed collaborators. With pervasive computing paradigm [22, 82], it is becoming possible to make workspaces virtually suitable for collaborating users in the goal of all the time and everywhere instead of accommodating collaborating users to collocated workspace. During our work, we saw the improvement of computing performance, the increase of network bandwidth, and the advance of wireless networking. We believe from Moore's law [80] and our development experience that the computing performance of mobile devices as well as desktop computers will continue to improve and networks' bandwidth will continue to increase. Thus the infrastructure improvements of software, hardware, and networking will make ubiquitous collaboration and access more prevalent and make the vision of Mark Weiser for 21st Century Computing [82] more promising as well in the future. In our work we have been designing and building virtual workspace on roaming cell phones as well as traditional desktops by integrating heterogeneous collaboration systems into a single easy-to-use collaboration system.

1.1 Motivation

The following scenario illustrates the needs of ubiquitous collaboration and access, and motivates the research issues described in this thesis. Researchers in Community Grids Lab (CGL) [13] at Indiana University often travel to attend offline real conference in a shared location. Students in CGL sometimes need to discuss with researchers. Researchers have to find a virtual conferencing system compatible with a conferencing system in CGL to discuss with students while traveling. Further, roaming researchers may have to find a place in which a compatible system is located. As this occurs, an integrated collaboration system, which combines heterogeneous virtual conferencing systems into a virtual conferencing system, will facilitate collaboration between the researchers and students. Virtual conferencing systems over Internet are rapidly increasing. Also, with increasing mobile devices, to integrate diverse mobile devices into a globally virtual conferencing system is becoming increasingly important. Current virtual conferencing systems lack support for ubiquitous collaboration and access.

Students in CGL are going to have a session for their colleague's research presentation. Some students join the presentation session in a shared conference room of CGL and others join at remote locations by using CGL's conferencing collaboration tool – Global-MMCS system (Global Multimedia Collaboration System) [32, 37]. The presenter starts her presentation with the conferencing collaboration tool. During her presentation, she may use an application like shared whiteboard to discuss design issues of the research which she is doing on grid computing. In shared workspace with the application, people in offline shared real room see the same whiteboard canvas, while people in online virtual room see their own canvases. Each student in the online virtual room has their own canvas and a set of interfaces to the shared whiteboard application but they see the same results (or views) as others do. Her advisor, researchers, and colleagues in CGL want to make comments on her research by directly manipulating the shared application showing the same views among participants in her research presentation session. Thus, the presenter needs to control their accesses to the shared application by enforcing who is allowed to access the application, and the conditions under that the privileges for the use of the application occur to restrict unauthorized access for the protected application.

As participants in her research presentation session try to manipulate the shared application at the same time, she has to be able to provide the right to access the shared application for only one participant in the session at any time to ensure the consistency of the shared application state. The shared application, that requires mutual exclusions in real time, has to be assigned to only one participant who requests it under a set of well-defined rules. Participants in offline session can use the rules of etiquette or social protocols to gain the manipulation of the shared application in an order by the rules or protocols. However, participants in online session can not use the etiquette rules or social protocols. Therefore, she will need some rules to substitute the etiquette rules and social protocols by defining the time and the way which a participant in collaboration gains access to the shared application.

1.2 Problem Statement

Conference collaboration systems typically provide a group of users with a set of well-defined interactions to access applications and resources, and communications among them. In such collaboration systems a group of users generally work sharing collaborative applications and resources in their workgroups (sessions). Therefore it is necessary to maintain consistent state information among sessions and collaborating users in a conference in a coordinated way. The state information includes managing workgroups, presences of and connectivity among collaborators in the workgroups. To maintain the consistent state information among users joining sessions as well as among shared collaborative applications, a set of event messages have to be disseminated to collaborators in a well-defined and unified manner. Also, especially in user mobility enabled collaboration, mobile hosts may be disconnected from the conferencing collaboration for arbitrary periods of time until reconnected into the collaboration. During the disconnected periods of time, new users may join the collaboration or sessions in the collaboration may be destroyed from the collaboration, and hence disconnected hosts (or users) may have inconsistent state information different from existing other hosts connecting (or joining) to the collaboration. Therefore, such a scheme to support operations during disconnection which is typical in mobile computing will be inappropriate for synchronous collaboration. Thus, we use query-dissemination interaction event messaging mechanism with publish-subscribe messaging service provided by our messaging and service middleware – NaradaBrokering [44, 83, 95, 96]. The mechanism provides a flexibility for adapting dynamic changes of collaboration states (creation and destroy of workgroups, and presences of participating users in workgroups) through the dissemination of event messages among users joined in a collaboration. The query-dissemination interaction serializes the procedures of the mechanism (query-update-dissemination-and same view (information)) never resulting in inconsistent state. Thus, the interaction mechanism can provide uniform access to collaborative applications identifying users among corresponding workgroups.

There are some well-known A/V conferencing and data collaboration systems like H.323 / T.120 [87], SIP (Session Initiation Protocol) [61, 65, 85], and Access Grid [1]. However, they are not suitable for building integrated conferencing and data collaboration systems to work together in the same collaboration session. For example, SIP has limited conference control and thus needs additional conference control mechanisms to support A/V conferencing and data collaboration. A/V conferencing in H.323 and data collaboration in T.120 are not well integrated and are designed in a relatively complicated OSI (Open Systems Interconnection) model. Also, these only deal with homogeneous conferencing and thus can not connect to other heterogeneous collaboration systems. In order to get the heterogeneous collaboration endpoints to work together, a common conferencing signaling protocol has to be designed to support interactions among heterogeneous collaboration endpoints. To build integrated collaboration system in the same session, the heterogeneous signaling procedures have to be translated into the common conferencing signaling procedures. To describe the protocol of the common signaling procedures, XML seems like good candidate because it makes the signaling protocol easy to read and understand and to interact with other web based components as opposed to binary format.

Fundamentally collaboration includes sharing resources. The cooperation on the resources shared among a group of users may hence produce new results on the shared resources. On the contrary, security is about restricting unauthorized access to resources and thus it is essential that security of the collaboration environments as well as of collaborative applications running on them is ensured while providing the openness only to users that are authorized to access them. Therefore, difficulties to deal with the conflicting goals of allowing and restricting access for resources among a group of users may happen in collaboration environment. In collaboration environment collaborating users are generally assigned a role, and collaborative applications have different types of roles which are assigned to a group of users. Access control scheme in collaboration system hence needs fine-grained access control for providing accesses for individual users in group, and for a finer granularity of accesses on individual resources shared in group.

In traditional face-to-face offline session, participants generally follow rules of etiquette or social protocol when they interact with each other. For example, if all the participants try to draw on a shared whiteboard, then the conflicts which may result in inconsistent state can be solved by a moderator or social protocols. However, in online session or CSCW (Computer Supported Cooperative Work), the social protocols may not be able to be used for coordinating the interaction of participants since they are not collocated. For example, if all the participants try to send drawing events through a communication channel in a distributed collaboration system, then the conflicts are not able to be solved by the social protocols used in face-to-face offline session. Therefore, policies and mechanisms used in an offline session may need a mapping into those able to be used in an online session with user interfaces between participants and CSCW environment.

When users perform concurrent activities on shared synchronous resources such as collaborative applications, floor control [17, 18] is necessary. Floor control is the problem of coordinating activities occurred in synchronously cooperating resources shared among participants in an online conference session. The floor control mitigates race conditions within online sessions on who is allowed to manipulate shared data or to send synchronous events. A set of well defined policies and mechanisms are needed for efficiently coordinating the use of resources in CSCW. The policies for floor control typically describe how participants in CSCW request resources, and how the resources are assigned and released when participants share a synchronous resource such as audio-video control event in conferencing, drawing events in shared whiteboard or moving events in chess game. Also, mechanisms including user interfaces (human-computer interaction) between participants and CSCW environment are needed to implement and enforce the policies. The floor control mechanisms have to be able to provide the floor on shared resource for only one participant in a synchronous online session at any time. No single floor control scheme is appropriate for all collaboration applications. The simplest scheme is free-for-all (no floor control) for applications like text chat. Therefore floor control needs to provide significant flexibility ranging from free-for-all to application specific floor control mechanism for avoiding uncoordinated activities to shared collaboration applications.

1.3 A Framework for Synchronous and Ubiquitous Collaboration

For ubiquitous collaboration and access as well as heterogeneous community collaboration, CGL has developed a virtual conferencing collaboration system – Global-MMCS (Global Multimedia Collaboration System) [32, 37] by integrating heterogeneous collaboration systems into a single easy-to-use collaboration system. The Global-MMCS provides the services of videoconferencing, instant messaging, and streaming to various clients, and integrates different collaboration communities into a global collaboration platform. In the virtual Global-MMCS, after a conference is activated, users can join the conference by starting their Global-MMCS client. Also, a conference chairperson can create sessions where a conference is composed of a set of sessions (online workgroups of collaborators working with sharing various collaboration applications). Through Global-MMCS, roaming users with cell phone devices as well as remote users can communicate with other users.

To handle cooperation and communication among heterogeneous communities, and to provide collaborative applications in the heterogeneous community collaboration, we built a framework on heterogeneous (wire, wireless) computing environment for ubiquitous collaboration as well as heterogeneous community collaboration.

To provide various collaboration sessions in a conference for users, XGSP (XML based General Session Protocol), which is a protocol for streaming session control messages written in XML, is used. The XGSP accounts for policy, presence, session creation, initiation, teardown, and so on.

To support group communications, a series of XGSP event messages are generated and disseminated to all the participants in a conference through our message and service middleware system – NaradaBrokering developed by CGL which supports publish/subscribe messaging model with a dynamic collection of brokers and provides services for TCP, UDP, Multicast, SSL, and raw RTP clients.

To provide a solution for controlling accesses to resources by defining which resources are available, who is allowed to access the resources, and the conditions under that the privileges for the use of the resources occur, roles based on users' privileges and devices' capabilities are used to allow users to manipulate the protected resources in the collaboration – XGSP-RBAC (XGSP Role Based Access Control).

To coordinate activities to resources and maintain shared state consistency at application level by mitigating race conditions within online sessions on who is allowed to manipulate shared data or to send synchronous events, we used a request-response interaction scheme between a moderator and a floor requester with human-computer interaction. Also, we used two-player turn-taking scheme for collaborative chess game application. To allow all participants to have the same views and data at all times for synchronous collaboration, we used non-optimistic floor control mechanism which strictly avoids conflicts – XGSP-Floor (XGSP Floor control).

The control mechanisms (XGSP, XGSP-RBAC, and XGSP-Floor) were integrated into a framework for synchronous and ubiquitous collaboration.

1.4 Contributions

The main contribution of this thesis includes the following:

1. *Building of a Framework for Synchronous and Ubiquitous Collaboration*: This includes another colleague's contribution on desktop. To facilitate ubiquitous

collaboration in anytime and anywhere among heterogeneous communities, we built a collaboration framework on heterogeneous computing platforms (cell phone and desktop).

2. *Defining of XGSP (XML General Session Protocol)*: This includes another colleague's contribution. We defined a set of session protocols which control sessions and participants' presences in a conference. The protocols are defined in XML. This protocol is used to provide various collaboration sessions (heterogeneous community collaboration sessions) for participants in a conference.

3. *Designing and implementing of XGSP-RBAC (XGSP Role Based Access Control)*: As an intermediate control entity between collaborators and collaboration resources, we used the concept of the role which is based on the users' privileges and devices' capabilities to manipulate protected shared collaboration applications. For fine-grained access control for the instance of individual resource, we defined fine-grained actions in our collaborative applications as the smallest interactive major events (semantic events). We designed and implemented XGSP-RBAC mechanism with the use of role, flexible, and fine-grained controls. With our shared whiteboard applications, we showed the performance of the XGSP-RBAC mechanism as well.

4. *Designing and implementing of XGSP-Floor (XGSP Floor control)*: We defined a policy and implemented a mechanism for the policy for coordinating accesses to collaborative applications, and maintaining shared state consistency at application level. We presented the functionality of the XGSP-Floor tool that provides a user interface (human-computer interaction) for control of floor to a moderator and participants in a session with desktop and cell phone devices. Also, we showed a synchronous collaboration, which means all participants in collaboration always have the same views and data in real time, with a major event conflict detection function and a non-optimistic locking mechanism.

5. *Building of application filter for cooperation of heterogeneous types of whiteboard applications*: The application filter converts a type of representation on one computing platform to other types of representations on other heterogeneous computing platforms with different screen sizes and different representation formats. The architecture is built as a derivative of shared display model and shared event model [31, 34, 110] for image and drawing object respectively. To demonstrate the effectiveness of the application filter, we showed the experiment (functionality and performance) for the use of the application filter with shared whiteboards on heterogeneous computing platforms.

6. *Building of application proxy for Instant Messenger*: The proxy has responsibility for getting responses from Jabber server [54] and performs any

necessary conversions for the clients on mobile device. As an intermediary, the proxy retains communication interfaces and thus can offload some computational needs (parsing of XML for XMPP (Extensible Messaging and Presence Protocol) [111] which is a presence protocol used in Jabber). The architecture is built as a derivative of shared event model [31, 34, 110].

7. *Building of collaborative applications on cell phone device*: To show the viability of our work, we built Text Chat, Instant Messenger, Java Whiteboard, and SVG (Scalable Vector Graphics) [79] Whiteboard on cell phone device as collaborative applications.

8. *Modeling of control mechanisms (XGSP-RBAC and XGSP-Floor)*: To prove the correctness of the control mechanisms, we modeled the mechanisms and verified the modeled mechanisms by Colored Petri Nets with time [64] in terms of mutual exclusion, dead lock, and starvation. The formal verification result shows that the modeled mechanism ensures consistent shared state at application level among collaborators.

1.5 Organization

The remaining chapters of this thesis are organized in the following manner:

In Chapter 2, we describe an architecture for collaboration framework built on heterogeneous (wire, wireless) computing environment, and then present XGSP (XML based General Session Protocol) for controlling sessions and participants' presences in a conference by defining a general protocol in XML.

In Chapter 3, we present a generic moderator-mediated interaction (request-response) mechanism – XGSP-RBAC (XGSP Role Based Access Control) for controlling accesses to applications and its supporting architecture integrated into collaboration framework.

In Chapter 4, we present XGSP-Floor (XGSP Floor control) for coordinating accesses to applications. Then we describe the functionality of an XGSP-Floor tool that provides a user interface for control of floor to a moderator and participants in a session with desktop and cell phone devices, a major event conflict detection function and a non-optimistic locking mechanism.

In Chapter 5, we show modeling of control mechanisms, in this thesis referred to as XGSP-RBAC and XGSP-Floor mechanisms. Then, we show formal verification to prove the correctness of the modeled control mechanisms. For the abstract representation of the modeling, we use Colored Petri Nets with time.

Finally, in Chapter 6, we conclude by summarizing main points drawn from our work and then present future works.

Chapter 2

Collaboration Framework Architecture and XML based General Session Protocol (XGSP)

2.1 Overview

In this chapter we describe an architecture for collaboration framework built on heterogeneous (wire, wireless) computing environment that handles cooperation and communication among heterogeneous communities and provides collaborative applications in the heterogeneous community collaboration, and overall architecture to support it. A key function of the framework is to provide a generic solution for controlling sessions in a conference and accesses to resources, maintaining shared state consistency at application level and maximizing the use of various collaborative capabilities to collaborator by defining a general protocol in XML [29]. Another function of the framework is to provide a structure for development and deployment of collaborative applications that can be used to support asynchronous collaboration by allowing different users of a session to access the same resource at different times, and synchronous collaboration by enabling the users to share the same resource in real time.

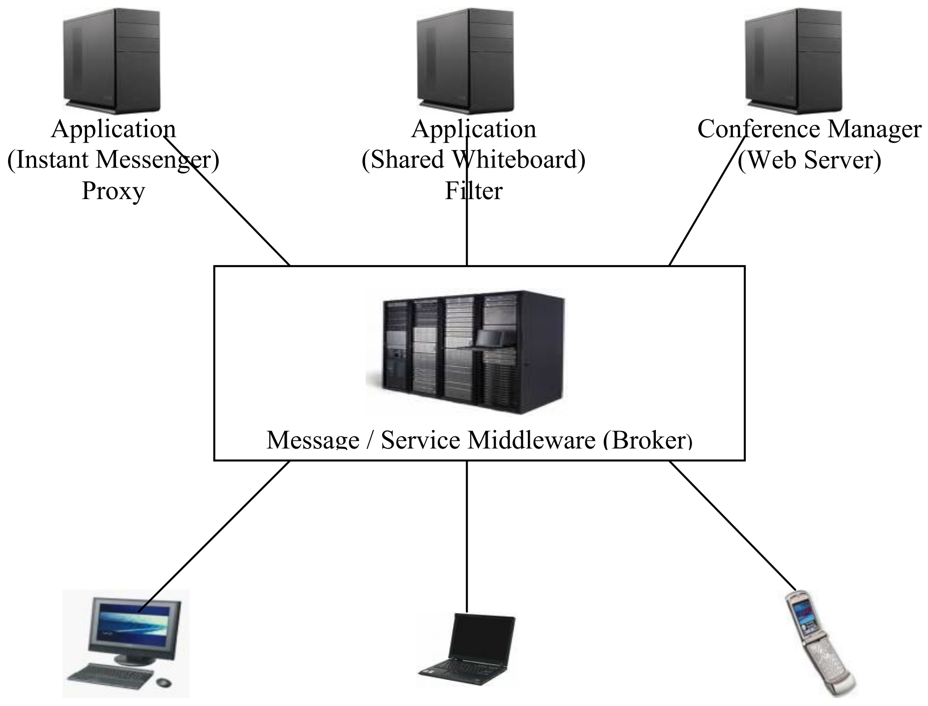


Figure 2.1: A Broad Architecture View.

In our work we have used J2ME (Java 2 Micro Edition) [62] for our software development in mobile (cell phone) computing environment and J2SE (Java 2 Standard Edition) [55] for our software development in non-mobile (stationary device like desktop PC) computing environment. For communication service, we have used NaradaBrokering [44, 83, 95, 96] for messaging and service middleware system as overlay built over heterogeneous networks to support group communications among heterogeneous communities and collaborative applications. The NaradaBrokering from Community Grids Lab (CGL) [13] is adapted as a general event brokering middleware, which supports publish/subscribe messaging model with a dynamic collection of brokers and provides services for TCP, UDP, Multicast, SSL, and raw RTP clients. The NaradaBrokering also provides the capability of the communication through firewalls and proxies. It is an open source and can operate either in a client-server mode like JMS [56] or in a completely distributed peer-to-peer mode [89, 99]. In this thesis we use the terms "message and service middleware (or system)" and "broker" interchangeably.

Figure 2.1 shows broad architecture view for our collaboration. In the Figure 2.1, the application proxy is used for Instant Messenger (IM). The IM is used to send notifications to user who is not connected or not joined in conference collaboration. IM on cell phone interacts with the application proxy (Jabber proxy) via a broker, and then the proxy communicates with Jabber open server [54] via socket connection. The proxy has responsibility for getting responses from Jabber server and performs any necessary conversions for clients on mobile devices. As an intermediary, the proxy retains communication interfaces and thus can offload some computational needs. The IM is an instant messaging client capable of interfacing with messenger services like MSN and Yahoo Messenger using Jabber open server. The Jabber open server is an instant messaging and presence managing platform based on XML, XMPP (Extensible Messaging and Presence Protocol) [111] and open standards. The benefits of using Jabber server include presence management, message processing based on XML, transparent interoperability, structured information data, and open formats. With such an approach using open or commercial technology, we can build a sustainable high functionality system taking advantage of the latest technologies and enable multiple collaborative applications to re-use the same basic technologies in a modular fashion with appropriate interface for collaborative applications.

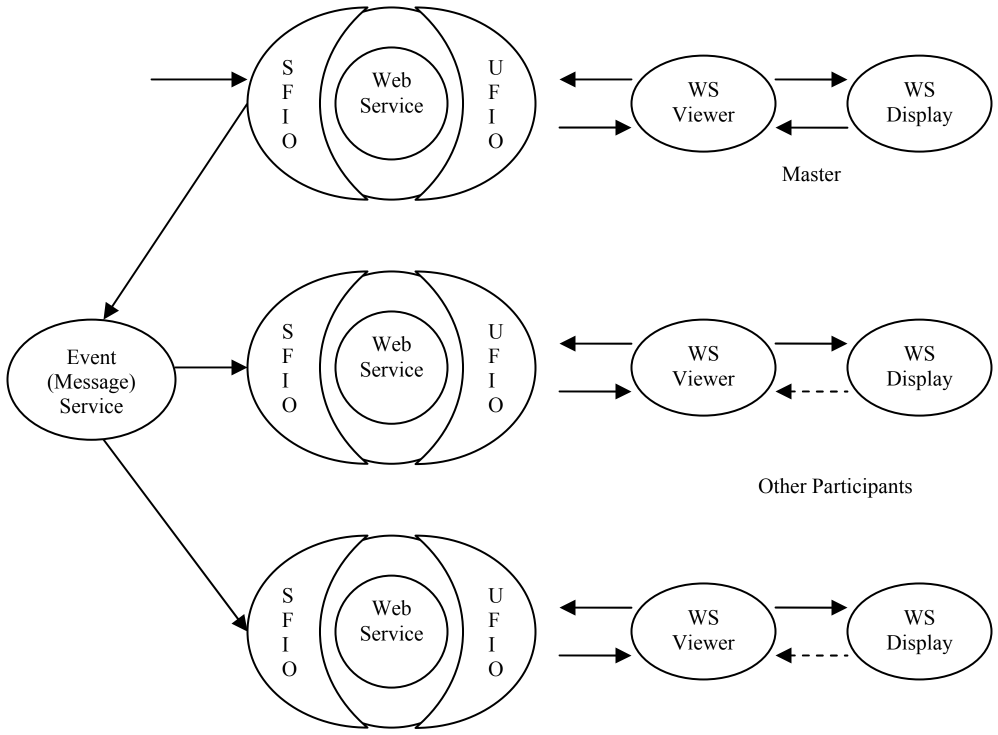


Figure 2.2: Shared Input Port Model (also called Shared Event Model or MMMV (Multiple Model Multiple View) Collaboration Model). UFIO and SFIO are User Facing and Service Facing Input/Output Ports.

Figure 2.2 shows MMMV (Multiple Model Multiple View) collaboration model (also called shared event model, or shared input port model) [31, 34, 110]. In the model, each client in collaboration shares one copy of the web service with the master. Then sharing is achieved by intercepting the pipeline before the master web service and directing copies of the messages (events) on each input port of the master web service to the replicated copies – shared event, where the pipeline means stage flow from one object to transform other object. Only the user-facing ports in the model are partially shared with data from the master transmitted to each replicated web service. The example is PowerPoint where all the clients have a copy of the application and the presentation to be shared before joining collaboration. Then events such as slide change can be sent to all participating clients for same view among them. Another example is shared SVG browser [110] which uses JavaScript event model to trap user input to a browser. The user input events playing the role of input ports are directly sent to all participating clients. CGL has a collaborative chess game application [100, 109, 110] for desktop devices. The application uses the shared event model. Each of players and observers in the game has their own chessboard and plays the game through a shared major event – moving object event which results in same view among them. The Jabber [54] can also be an instance of the model. Figure 2.3 shows the architecture of the Jabber as a derivative of shared event model.

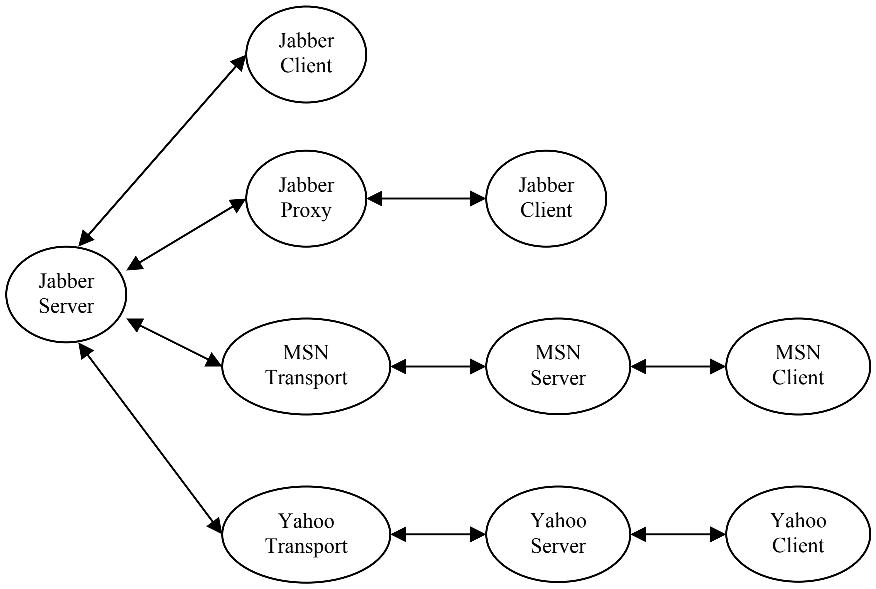


Figure 2.3: Jabber Instant Messenger Architecture View with Application (Instant Messenger) Proxy.

In Figure 2.1, the application filter is a kind of agent able to cooperate and to coordinate heterogeneous types of applications on heterogeneous platforms, but is not considered as intelligent agent. The purpose of the application filter use is to convert one type of representation to other types of representations on heterogeneous platforms with different screen (or canvas) sizes and different representation formats. Arriving objects into the application filter are immediately filtered and converted to other types of objects, and are broadcasted to all participating clients through our messaging and service middleware. The communication channel (publish/subscribe) of the application filter enables one type of collaborative application to exchange event objects with other types of collaborative applications, where the event objects may be different according to the types of applications, e.g. drawing and image sharing events in shared whiteboard application or moving object event in chess game application. Note that the homogeneous collaborative applications with the same type of representations can communicate directly through the broker without the use of the application filter. Therefore arriving messages are converted to (N-1) types of event messages where N is the number of heterogeneous types of application supported in collaboration. In this architecture, each application does not know how to convert its own representation into other representations. The application filter is responsible for translating data from one type of representation into other types of representations. In Figure 2.4, shared display collaboration model [31, 34] shows that clients share the graphical image display and the state is maintained (shared) between the clients by transmitting the changes in the display through the event message service. The supporting heterogeneous clients require that sophisticated shared display environments automatically change size and display representation formats to suit each client. The shared display model has one key advantage – it can immediately be applied to all shared objects. But it has two obvious disadvantages – it is rather difficult to customize and requires substantial network bandwidth as complete graphical image display between desktop and cell phone applications as well as among desktop applications has significant network transit overhead.

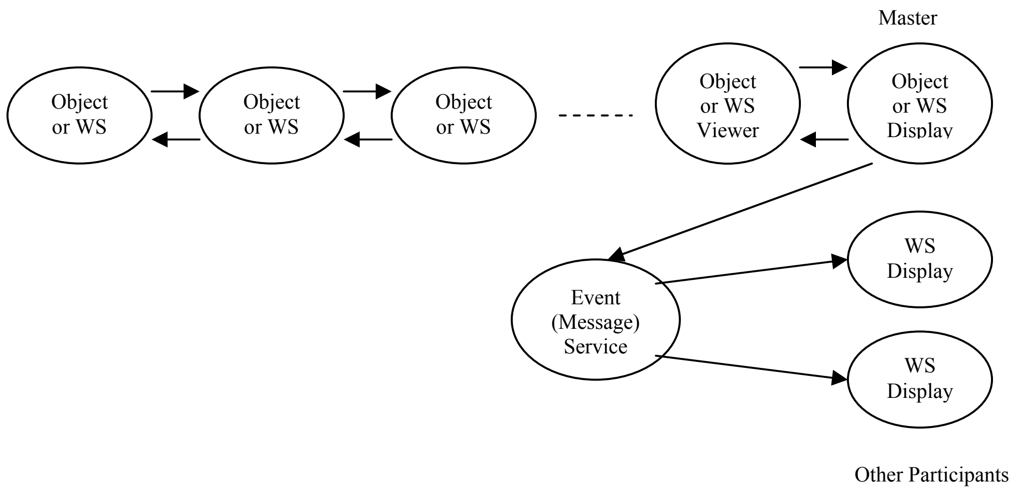


Figure 2.4: Shared Display Collaboration Model

Figure 2.5 shows an instance of the application filter obtained by applying the shared display model. To demonstrate the effectiveness of the application filter, we show the experimental results of the application filter with our collaborative application – shared whiteboards on heterogeneous computing platforms (cell phone and desktop) in section 2.6.3.

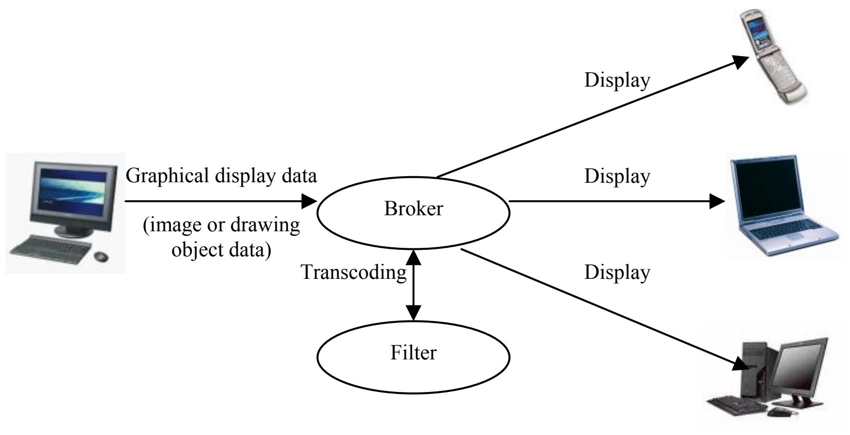


Figure 2.5: Application (Shared Whiteboard) Filter Architecture View.

This chapter is organized as follows. In the remainder of this chapter, we put research issues and our solution about them in section 2.2. Section 2.3 describes related works. Section 2.4 presents the architecture of collaboration framework and the implementation of it. Section 2.5 describes XML based General Session Protocol (XGSP). Section 2.6 presents experimental results. Finally, we conclude by summarizing main points drawn from building collaboration framework and the XML based General Session Protocol.

2.2 Problem Statement and Solutions

Conference collaboration systems typically provide a group of users with a set of well-defined interactions to access applications and resources, and communications among them. In such collaboration systems a group of users generally work sharing collaborative applications and resources in their workgroups (sessions). The collaborative applications can be synchronous or asynchronous applications shared among users in the workgroup, and hence the sessions can be synchronous or asynchronous workgroups according to the applications. Therefore it is necessary to maintain consistent state information among sessions and collaborating users in the conference in a coordinated way. The state information includes managing and coordinating workgroups, and presences of and connectivity among collaborating users in the workgroups. To maintain the consistent state information among users joining sessions as well as among shared collaborative applications, a set of event messages have to be disseminated to collaborating users in a well-defined and unified manner. Also, especially in user mobility enabled collaboration, mobile hosts may be disconnected from the conferencing collaboration for arbitrary periods of time until reconnected into the collaboration. During the disconnected periods of time, new users may join the collaboration or sessions in the collaboration may be destroyed from the collaboration, and hence disconnected hosts (or users) may have inconsistent state information different from existing other hosts connecting (or joining) to the collaboration. Therefore, such a scheme to support operations during disconnection which is typical in mobile computing will be inappropriate for synchronous collaboration. Thus, we use query-dissemination interaction event messaging mechanism with publish-subscribe messaging service provided by our messaging and service middleware. The mechanism provides a flexibility for adapting dynamic changes of collaboration states (creation and destroy of workgroups, and presences of participating users in workgroups) through the dissemination of event messages among participants in a collaboration. In the mechanism users only need to send a set of queried event messages to a chairperson node on which a conference chairperson resides. After the chairperson node processes the queries and disseminates the results of the queries to users joined (or connected) in the collaboration through the broker which is responsible for reliable and ordered consistent delivery of messages, the users can have the consistent collaboration state information in the collaboration. The query-dissemination interaction serializes the procedures of the mechanism (query-update-dissemination-and the same view (information)) never resulting in inconsistent state. Thus, the interaction mechanism can provide uniform access to collaborative applications identifying users among corresponding workgroups.

There are some well-known A/V conferencing and data collaboration systems like H.323 / T.120 [52, 87], SIP (Session Initiation Protocol) [61, 65, 85] and Access Grid [1]. However, they are not suitable for building integrated conferencing and data collaboration systems to work together in the same collaboration session. For example, SIP has limited conference control and thus needs additional conference control mechanisms to support A/V conferencing and data collaboration. A/V conferencing in H.323 and data collaboration in T.120 are not well integrated and are designed in a relatively complicated OSI (Open Systems Interconnection) model. Also these only deal with homogeneous conferencing and thus are not able to connect to other heterogeneous collaboration systems. In order to get the heterogeneous collaboration endpoints to work together, a common conferencing signaling protocol has to be designed to support interactions among heterogeneous collaboration endpoints. To build integrated collaboration system in the same session, the heterogeneous signaling procedures have to be translated into the common conferencing signaling procedures.

To describe the protocol of the common signaling procedures, XML seems like good candidate because it makes the signaling protocol easy to read and understand and to interact with other web based components as opposed to binary format.

2.3 Related Work

In this section we examine existing conferencing collaboration system. NetMeeting [84] provides audio/video conferencing and data collaboration functions for Internet and corporate intranet. The functional capabilities for conferencing collaboration include sharing information, sending text messages, transferring binary file, recording meeting notes and communicating with other people in real time through Internet telephony audio/video conferencing. NetMeeting uses H.323 [52, 87] protocol for audio/video conferencing and modified T.120 [103] protocol for data collaboration. The NetMeeting was replaced by Windows Meeting Space [108] running on Windows Vista. However, now the Window Meeting Space has just collaboration capabilities not including conferencing capability.

GroupKit [43, 73, 74] is a Tcl/Tk (Tool Command Language / Graphical User Interface Toolkit) [101] library to build real time groupware applications such as multi-user drawing tools, shared text editors, and conference meeting tools. The GroupKit presents three strategies such as run-time process and communication architecture for creation, interconnection and management of conference sessions, overlays for easily adding general components needed for building groupware applications, and flexible interface and interaction policies for accommodating work styles of groups. The

GroupKit toolkit provides development environments to build shared view applications such as shared window, whiteboards, and editors.

H.323 [52, 87] developed by standardization sector of ITU-T (International Telecommunication Union) is a series of recommendations for packet-based multimedia group communications systems specifying the components to be used within an H.323-based environment. It provides conference management functionality for audio/video conferences using the call signaling (connection establishment/teardown) functionality of H.225 [48] and the control functionality of H.245 [50] which provides control management for exchanging terminal capabilities and creating media channels. These protocols provide call set-up and call transfer of real time connections to support small-scale multipoint conferences. The protocol H.243 [49] defines some commands for system operation between the MCU (Multipoint Control Unit) and H.320 [51] audiovisual terminals to implement audio mixing, video switch and cascading MCU, where MCU provides support for conferences of three or more H.323 terminals. T.120 recommendation [103] is used for data management of a conference. This standard contains a series of communication and application protocols and services that provide support for real time, multi-point data communications. The multi-point facilities are important building blocks for a whole new range of collaborative applications including desktop data conferencing, multi-user applications and multi-player gaming. However, in H.323, T.120 is completely independent of H.225 and H.245. In fact, A/V and data collaborations should be integrated in the same framework so that the architecture can be easily implemented and maintained.

The Session Initiation Protocol (SIP) [61, 65, 85] developed in MMUSIC WG (Multiparty Multimedia Session Control Working Group) of IETF (Internet Engineering Task Force) was designed as a general session protocol for establishing, maintaining, and tearing down Internet sessions including multimedia conferences. SIP provides basic functions including: user location resolution, capability negotiation (session parameter), and call management (invitation to session). All the capabilities in SIP are basically equivalent to the service H.225 and H.245 in H.323 protocol. The major difference is that SIP was designed in a text format and took request/response protocol style like HTTP. But H.225 and H.245 were defined in a binary format and kept a style of OSI (Open Systems Interconnection). Therefore, SIP has some advantages of interaction with web protocol like HTTP. More importantly, SIP does not define the conference control procedure like H.243 and T.120. Additional conference control mechanisms have to be implemented on the base of SIP to support the A/V and data collaboration.

Virtual Rooms Video Conferencing Systems (VRVS) [104] provides some kinds of integration of different A/V endpoints. However, VRVS is not open project and thus has few introduction documents which briefly describe its architecture and conference control framework. From the brief introduction, VRVS builds its collaboration service on top of pure software reflector [104] which interconnects each user to a virtual room by a permanent IP tunnel. The use of the reflector technology assures the quality needed for videoconferences transmission (audio, video, and data flows). The VRVS is capable of supporting MBONE [78] tools, H.323 terminal, and data sharing collaborations, like shared web browsing and virtual network computing (VNC) for shared view on desktop.

2.4 Collaboration Framework Architecture

Collaboration framework is a basic structure to hold consistent view or information of users' presence and sessions together, and to support diverse collaborative applications to collaborators joining in a conference at remote locations. It also has a capability that allows a user to join a conference using networked heterogeneous (wire, wireless) computing devices anytime and anywhere and to use collaborative applications in the conference. It is important to users joining a conference that it seems to be in offline real conference room even when using heterogeneous computing devices at remote locations. It is typical today and will be more typical in the future that all users can access information independent of their access devices and physical capabilities anytime and anywhere.

To maintain consistent information of presences and sessions in a conference, we use a request (query) and response (dissemination) mechanism that requires a user to inquire queries (request event messages) to a chairperson node (conference chairperson) and a conference manager in order to engage in presence and various collaboration activities, and the chairperson node and conference manager to disseminate the queried information to all the participants through the messaging and service middleware. A set of protocols are defined in section 2.5 for maintaining consistent collaboration state information among participants in conference collaboration.

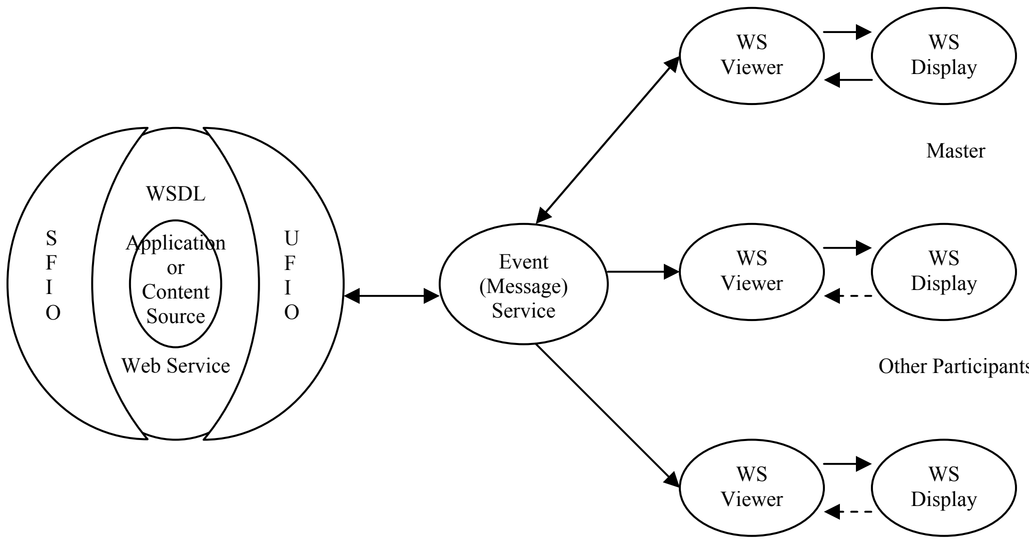


Figure 2.6: Shared Output Port Model (also called SMMV (Single Model Multiple View) Collaboration Model).

Figure 2.6 shows SMMV (Single Model Multiple View) collaboration model (also called shared output port model) [31, 34, 110]. The shared display model in Figure 2.4 is a subset of SMMV. This SMMV collaboration model can be used for collaborative applications. Our collaboration framework follows this approach for collaborative applications built on it with our messaging and service system. In Figure 2.7, we show an instance obtained by applying the model, and used for session/membership data and display of the information data in the collaboration framework.

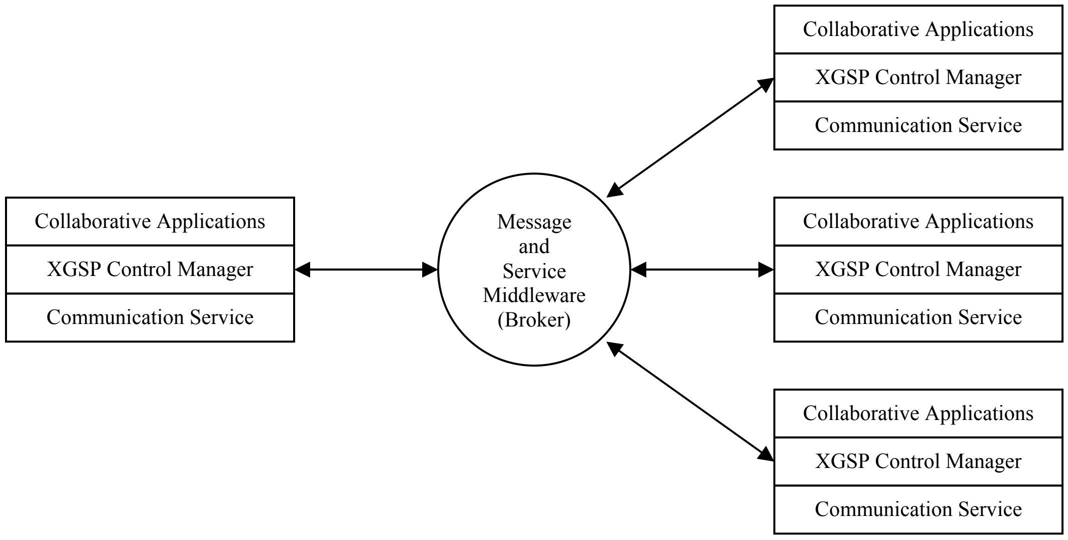


Figure 2.7: XGSP Service Architecture.

As shown in Figure 2.8, the collaboration framework is structured as three layers and six major components: control manager, session / membership control manager, access / floor control manager, policy manager, request and reply event message handlers, and communication channel. We describe the components in turn.

2.4.1 Control Manager

A control manager is an interface component located between sessions and managers in collaboration framework for providing conference management services such as presence, session, and access and floor control managements for participants in collaboration. Presence of participants, creation/destroy of sessions, and activation/deactivation of actions to access resources are serviced through this manager into each of control management services. The control manager also has factories for all kinds of applications, and hence can create new application instances and invoke, start, and destroy them.

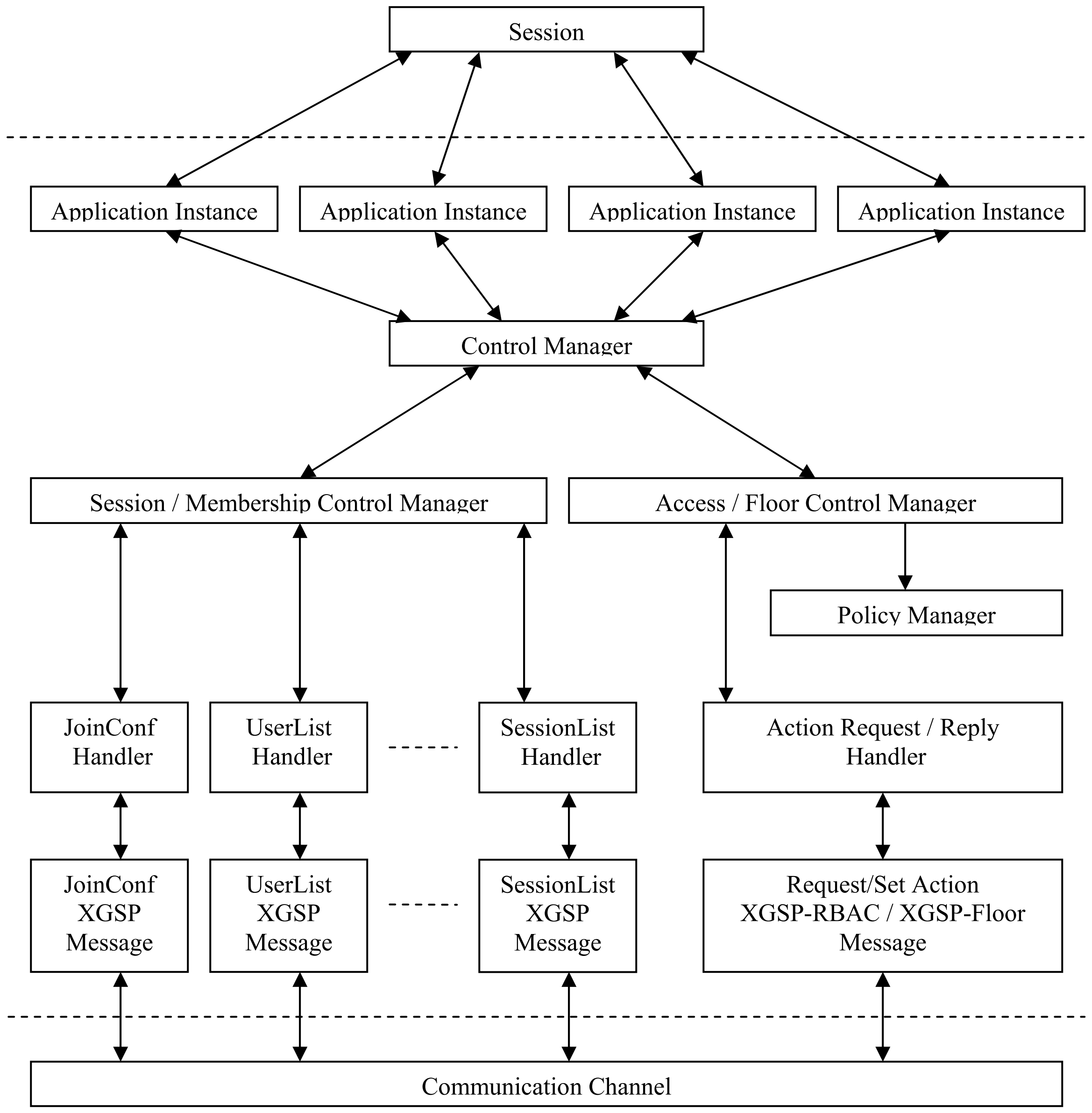


Figure 2.8: Collaboration framework architecture consists of three layers (collaborative applications, managers, and communication service) and six major components.

2.4.2 Session and Membership Control Manager

This manager manages information about who is currently in the conference and has access to what applications, and which sessions are available in the conference. The session and membership control manager has a set of control logics that are used to manage presences of and connectivity among collaborating users in collaborating workgroups, and organize the workgroups. The control logics communicate through a set of predefined protocols (session control protocols) for streaming control messages to exchange presence information of collaborating users and state information of various collaborative sessions. The session control protocols account for policy, presence, session creation, initiation, teardown, and so on. To describe presences, connectivity, and states of sessions, XML is used as a protocol definition language of the session and membership control. The XML based General Session Protocol (XGSP) is described in section 2.5 in more detail.

2.4.3 Access and Floor Control Manager

The access and floor control manager component in collaboration framework is responsible for handling accesses to collaborative applications through the request and reply event message handlers which are one of components in the framework. A user requests an access to use resources like collaborative applications to a chairperson or moderator through a request event message handler. The chairperson or moderator responses a decision (grant, deny, or queued) to the requesting user who wants to access resources through a reply event message handler. The chairperson or moderator also broadcasts the decision to make the change of access state to each resource globally visible to all the participants in a session. A GUI (Graphic User Interface) on the framework, which is used to display access state information for resources, is used to request accesses to resources. Within the access and floor control manager, policies are read from a file, a request is validated through a policy manager and one of classified access types is returned into the manager through an access type decision service. With the returned access type, a chairperson or moderator makes a decision and the decision is dispatched to the requesting user. Also the decision is broadcasted into each node to update the access state information for the resource. The XGSP Role Based Access Control mechanism (XGSP-RBAC) and the XGSP Floor control mechanism (XGSP-Floor) are described in chapter 4 and 5 respectively in more detail.

2.4.4 Policy Manager

Access and floor control policies are written in XML and put into the conference manager shown in Figure 2.1 for globally consistent use. When a new user joins a conference, the conference manager pushes the policy into the node (or host) of the new user as a stream message, and the policy is stored in local policy store (a file) of the node during joining (connecting) in the conference. The policies describe which roles (users in them) in collaboration are allowed to perform which actions on which target applications. As a request event message for accessing applications arrives, the policy manager pulls the policy from the policy store. The policy manager is responsible for validating the request event messages based on the access and floor control policies pulled from a local policy store.

2.4.5 Request and Reply Event Message Handlers

An event message handler is a subroutine that handles request and reply event messages. The control manager manages the associations between incoming and outgoing event messages with each of event message handlers. According to the associations, generated outgoing (request) event messages are first processed by the associated request event message handlers in each node (or host). Incoming (reply or response) event messages are also serviced by the associated reply/response event message handlers. The messages are sent to a broker via the communication channel shown in Figure 2.8. The broker disseminates the messages to other participants connected to the collaborating workgroup (session).

2.4.6 Communication Channel

The communication channel is responsible for controlling interactions among participants and communications among collaborative applications. The channel uses topic-based publish-subscribe mechanism that defines a general API for group communication. The API for the topic-based publish-subscribe mechanism is used as an interface for group communication of sessions in a conference and between collaborative applications and a broker. In the topic-based publish-subscribe mechanism, the topic information contained within messages is used to route the messages from publisher to subscriber. The topic information has two kinds of naming schema: a name separated simply by slash("/") strings like /XGSP/Conference-ID/Application-Session-ID can be used and another naming schema can be described using a set of tag=value pairs, a set of properties associated with the message, verbose text, or XML. The messages containing topic information are sent to a broker through the communication channel. And the messages are disseminated through router nodes, referred to as brokers to subscribers which registered a subscription to the topic.

In the next section, we describe the XML based General Session protocol (XGSP) [106] for event messages used for communication among collaborating users.

2.5 XML based General Session Protocol (XGSP)

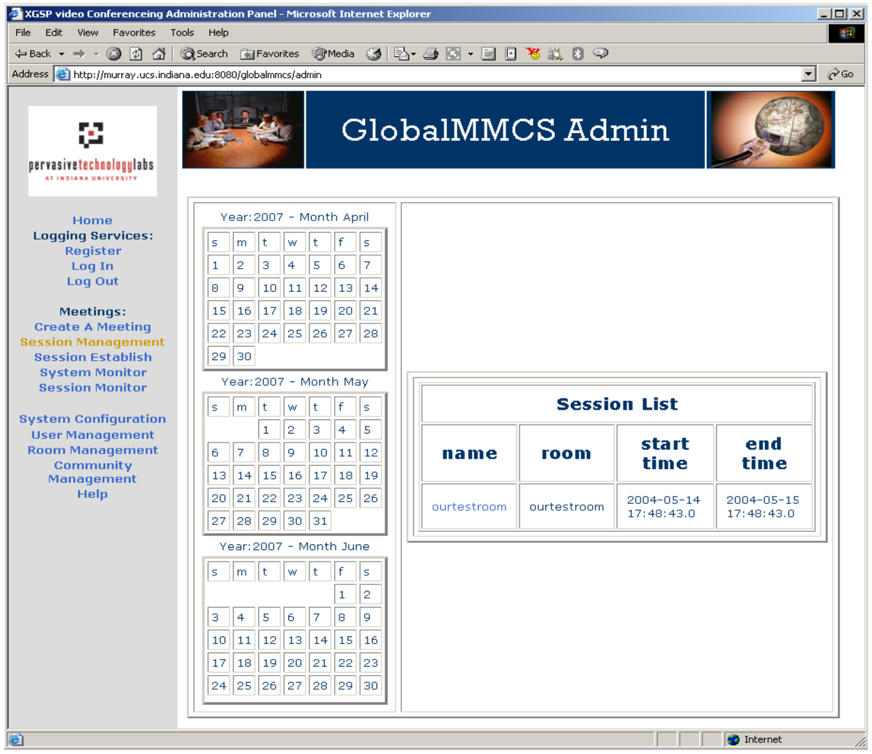
Collaboration can be defined as interaction for cooperation on shared resources between people working at remote locations. The interaction in collaborative computing requires a simple and universal access means and mechanism for people to easily access information or to conveniently communicate with other people. Interactions and cooperation for collaboration can be generally provided through the unit of conference and sessions. A conference is composed of a set of sessions, where a session means online workgroup of collaborating users working with sharing various collaborative resources. A conference needs control logic to maintain state information among sessions and presence information among participants in a conference. The control logic is used to manage presences of and connectivity among collaborating users in the online workgroup (session), and organize the online workgroups (sessions or conference). The control logic needs a protocol for streaming control messages to exchange presence information of collaborating users and states of various collaborative sessions. To describe control logics of presences, connectivity, and sessions' states, we use XML as a protocol definition language of session control. The

XML based General Session Protocol (XGSP) [106] is a protocol for streaming control messages written in XML to provide various collaboration sessions in a conference for users according to the presences and connectivity. The session control protocol account for policy, presence, session creation, initiation, teardown, and so on. In the next subsections we describe conference, session, and presence management in turn with our session control protocol - XGSP.

2.5.1 XGSP Conference Management

In Figure 2.1, the conference manager manages information related to all the conferences. The conference manager resides on web server running on tomcat. The manager maintains registries of all scheduled conferences, registries of collaborative applications such as A/V, text chat, whiteboard, and chess game, user accounts and access and floor control policy for the applications. Through a meeting calendar [2], the conference manager provides a set of meeting lists to users. Users can make meeting reservations via their browsers or emails. The conference manager can grant or deny the requests of users according to the capability of conference servers. The manager also activates or deactivates conferences at the starting and ending time of them.

After a conference is activated, users can join the conference by starting their node manager (or called Global-MMCS client). Then the node manager generates a series of XGSP event messages and broadcasts them to conference manager and all the participants in the conference. The Figure 2.9 shows the administration web portal page of the conference manager for the conference management of Global Multimedia Collaboration System (Global-MMCS) developed by Community Grids Lab (CGL) at Indiana University. The Global-MMCS system [32, 37] based on the collaboration framework provides the services of videoconferencing, instant messaging, and streaming to various clients, and integrates different collaboration communities into a global collaboration platform.


XGSP video Conferenceing Administration Panel - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://murray.ucs.indiana.edu:8080/globalmmcs/admin
Go
pervasivetechnologylabs AT INDIANA UNIVERSITY
GlobalMMCS Admin
Home
Logging Services:
Register
Log In
Log Out
Meetings:
Create A Meeting
Session Management
Session Establish
System Monitor
Session Monitor
System Configuration
User Management
Room Management
Community Management
Help
Internet

Year:2007 - Month April

s	m	t	w	t	f	s
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

Year:2007 - Month May

s	m	t	w	t	f	s
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

Year:2007 - Month June

s	m	t	w	t	f	s
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Session List

name	room	start time	end time
ourtestroom	ourtestroom	2004-05-14 17:48:43.0	2004-05-15 17:48:43.0

Figure 2.9: GlobalMMCS Admin Example Web Portal Page.

2.5.2 XGSP Session Management

All the participants in a conference can join sessions predefined as a default session in the conference or created by a conference chairperson. The predefined default session has a default application like A/V, shared whiteboard, text chat, and instant messenger

(Jabber client) if participant has an account of Jabber IM [54]. The procedures in creating and terminating application sessions include behaviors of conference manager and node managers. The conference manager monitors the life-cycles of the sessions. If the session has a session server, the manager commands the application session server to work for the management of the session. How A/V application sessions are handled is described in more detail [33]. Each node manager keeps a directory of application sessions. When the node manager of a conference chairperson creates an application from the application session directory, the node manager sends a XGSP event message ("Create Application Session") to all other node managers in the conference. Each node manager adds the application session information in the XGSP event message into a local application directory. When the application session is terminated, the node manager of the conference chairperson also sends a XGSP event message ("Destroy Application Session") to all other node managers. They close application instances in the session and remove the session information from their local application session directory. An example XML stream for creating and terminating an application session is shown in Figure 2.10 and 2.11 respectively. If a chairperson creates a new application session, the session information is broadcasted through a broker in the following XML stream.

```
<?xml version="1.0" encoding="UTF-8"?>
<CreateAppSession>
        <ConferenceID>ourtestroom</ConferenceID>
        <ApplicationID>wb</ApplicationID>
        <SessionID>NewAppSession</SessionID>
        <Creator>kskim</Creator>
</CreateAppSession>
```

Figure 2.10: XML Stream for Creating an Application Session.

But if a chairperson destroys a session, the old session information is broadcasted in the following XML stream. Thus the destroyed session is removed from session repository and GUI (Graphic User Interface) in each node (or host).

```
<?xml version="1.0" encoding="UTF-8"?>
<DestroyAppSession>
        <ConferenceID>ourtestroom</ConferenceID>
        <ApplicationID>wb</ApplicationID>
        <SessionID>OldAppSession</SessionID>
        <Destroyer>kskim</Destroyer>
</DestroyAppSession>
```

Figure 2.11: XML Stream for Destroying an Application Session.

When a user wants to join an application session, the user can select a session from session directory which is displayed as a GUI in her node manager. The application factory in the node manager creates an application instance. During the creation of the application instance, some parameters like a topic name which is used in a broker and initial default action allowed for accessing the application are passed to the application instance. An example XML stream for joining an application session is shown in Figure 2.12. Before joining a collaborative application session, a user has to establish a session by sending the XML stream to a chairperson node if she wants to join the session and uses application instances in the session. Figure 2.13 shows a XML stream instance disseminated as leaving a session.

```
<?xml version="1.0" encoding="UTF-8"?>
<JoinAppSession>
        <SessionID>NewAppSession</SessionID>
        <UserID>kskim</UserID>
</JoinAppSession>
```

Figure 2.12: XML Stream for Joining a session NewAppSession

```
<?xml version="1.0" encoding="UTF-8"?>
<LeaveAppSession>
        <SessionID>NewAppSession</SessionID>
        <UserID>kskim</UserID>
</LeaveAppSession>
```

Figure 2.13: XML Stream for Leaving a session NewAppSession

2.5.3 Presence and Session Establishment

Session control based on XGSP integrated into the collaboration framework and conference manager is implemented via a request-response (query-dissemination) mechanism that requires a user to establish a session on a chairperson node (conference chairperson) and conference manager in order to engage in presence and various collaboration activities. A user needs to send a join-conference message to conference manager before the user can establish a session on the conference manager in order to receive policies for setting session controls and accessing to resources. Also, a user needs to send the join-conference message to a chairperson node before the user establish a session on the chairperson node in order to engage in presence information of other collaborating users and existing various collaborative applications in the session. The initial presence message, join-conference XML stream, is to signal her availability for communications to all other participants and conference manager in conference collaboration. An example of the initial presence stream is shown in Figure 2.14, 2.15, and 2.16.

- Before joining in a conference, a user has to send her initial presence in join-conference XML stream to a chairperson node and conference manager. In Figure 2.14 and 2.15 we show an example join-conference XML stream for a chairperson and desktop users with role names chairperson and non-mobile user respectively. A join-conference XML stream for mobile users is shown in Figure 2.16 with a role name mobile-user.

```
<?xml version="1.0" encoding="UTF-8"?>
<JoinConf>
	<ConferenceID>ourtestroom</ConferenceID>
	<User>
		<RoleName>chairperson</RoleName>
		<UserID>kskim</UserID>
		<UserName>kangseok-kim</UserName>
	</User>
</JoinConf>
```

Figure 2.14: Join XML stream of Chairperson on Desktop PC showing conference ID, user's role name, user ID and user name

```
<?xml version="1.0" encoding="UTF-8"?>
<JoinConf>
	<ConferenceID>ourtestroom</ConferenceID>
	<User>
		<RoleName>nonmobile-user</RoleName>
		<UserID>kskim</UserID>
		<UserName>kangseok-kim</UserName>
	</User>
</JoinConf>
```

Figure 2.15: Join XML stream of Desktop users showing conference ID, user's role name, user ID and user name

```
<?xml version="1.0" encoding="UTF-8"?>
<JoinConf>
	<ConferenceID>ourtestroom</ConferenceID>
	<User>
		<RoleName>mobile-user</RoleName>
		<UserID>kskim</UserID>
		<UserName>kangseok-kim</UserName>
	</User>
</JoinConf>
```

Figure 2.16: Join XML stream of Cell phone users showing conference ID, user's role name, user ID and user name

- Conference manager informs a XML stream binding policies that are used for requests of resources. The example stream is shown in Figure 2.17.

```
<?xml version=”1.0” encoding=”UTF-8”?>
<ReplyPolicy>
        <ConferenceID>ourtestroom</ConferenceID>
        <User>
                <UserID>kskim</UserID>
                <UserName>kangseok-kim</UserName>
        </User>
        <Policy>
                <XGSP-RBACPolicy>
                …………………
                </XGSP-RBACPolicy>
        </Policy>
</ReplyPolicy>
```

Figure 2.17: Policy XML stream from Conference Manager showing conference ID, user ID, user name, and access / floor control policy

Upon joining a collaboration conference, a user needs to request a presence list of participants and existing sessions in the conference.

- A user has to send a query requesting presence and available session list to a chairperson node. Figure 2.18 shows a presence request XML stream and Figure 2.19 shows a session request XML stream.

```
<?xml version=”1.0” encoding=”UTF-8”?>
<RequestUserList>
    <UserID>kskim</UserID>
    <ConferenceID>ourtestroom</ConferenceID>
</RequestUserList>
```

Figure 2.18: Presence request XML stream

```
<?xml version=”1.0” encoding=”UTF-8”?>
<RequestSessionList>
        <UserID>userID</UserID>
        <ConferenceID>confID</ConferenceID>
</RequestSessionList>
```

Figure 2.19: Session request XML stream

- A chairperson node informs users' presence list and an available session list. Figure 2.20 shows a presence reply XML stream and Figure 2.21 shows a session reply XML stream.

```
<?xml version=”1.0” encoding=”UTF-8”?>
<ReplyUserList>
        <UserID>kskim</UserID>
        <ConferenceID>ourtestroom</ConferenceID>
        <UserList>
                <User>
                        <RoleName>chairperson</RoleName>
                        <UserID>kskim2</UserID>
                        <UserName>kangseok-kim2</UserName>
                </User>
                        User presence list joining in conference ID ourtestroom
        </UserList>
</ReplyUserList>
```

Figure 2.20: Presence reply XML stream

```
<?xml version=”1.0” encoding=”UTF-8”?>
<ReplySessionList>
        <UserID>kskim</UserID>
        <ConferenceID>ourtestroom</ConferenceID>
        <SessionList>
                        Session list in conference ID ourtestroom
        </SessionList>
</ReplySessionList>
```

Figure 2.21: Session reply XML stream

- When a participant leaves a conference, her leave-conference XGSP event message is disseminated in the XML stream of Figure 2.22. Thus her presence is removed from the membership directory and GUI in each node (or host) participating in collaboration.

```
<?xml version=”1.0” encoding=”UTF-8”?>
<LeaveConf>
	<ConferenceID>ourtestroom</ConferenceID>
	<User>
		<RoleName>chairperson</RoleName>
		<UserID>kskim</UserID>
		<UserName>kangseok-kim</UserName>
	</User>
</LeaveConf>
```

Figure 2.22: Leave Conference XML stream

2.6 Experimental Results

In this section we show experimental results to demonstrate the viability of our approach with a variety of performance measurements. We show network performance between our messaging and service middleware – NaradaBrokering and collaboration framework built on cell phone and desktop devices. We also show the performance of XGSP event mechanism integrated into our collaboration framework. In a case study of shared whiteboard application with image annotation and the application filter which is one of components in our collaboration system, we show the viability of our architectural approach to support effective collaboration between heterogeneous collaboration applications. The main purpose is to show the effectiveness of cooperation of collaborating users using heterogeneous collaboration applications and our architectural approach for integration of heterogeneous collaboration applications in heterogeneous computing and network environments.

2.6.1 Baseline Performance Result

In this section we show the baseline performance results of network (wire, wireless) used for communication between our messaging/service middleware (broker) and collaboration framework built on cell phone and desktop devices. Note that the results are not to show better performance enhancement but to quantify the network performance of wireless cell phone and wired desktop devices for a variety of datasets. The quantified results will be used as a reference of the experimental results of the performance measurements used in the following sections. In our experiments, we measured the round trip time involved in performing communication between collaboration framework and a broker for a variety of datasets in heterogeneous networked environments over a variety of locations. The experiment results were measured from executing collaboration framework running on Palm OS 5.2.1H Powered Treo600 [102] cell phone platform with 144 MHz ARM Processor and 32MB RAM connected to cellular network, and running on Windows XP platform with 3.40 GHz Intel Pentium and 2 GB RAM connected to Ethernet network. The collaboration framework on cell phone and desktop is located in Community Grids Lab at Indiana University. The broker ran on a 2.4 GHz Linux with 2 GB RAM located in Community Grids Lab at Indiana University, a 1.2 GHz Linux with 8 GB RAM located in NCSA (National Center for Supercomputing Applications) at UIUC (University of Illinois at Urbana-Champaign) and a 1.2 GHz Linux with 8 GB RAM located in SDSC (San Diego Supercomputer Center) at UCSD (University of California at San Diego).

Figure 2.23, 2.24 and 2.25 show the round trip time to transfer bytes data between collaboration framework and a broker through wired and wireless network respectively including the corresponding execution time of the broker. As the size of data increases, the time for transferring the data increases as well, as shown in the figures. Note that where the results in Figure 2.23 are in the range of only milliseconds, the results in Figure 2.24 and Figure 2.25 are in the range of seconds. This measurement results will be used as a baseline for the performance measurements in the following sections.

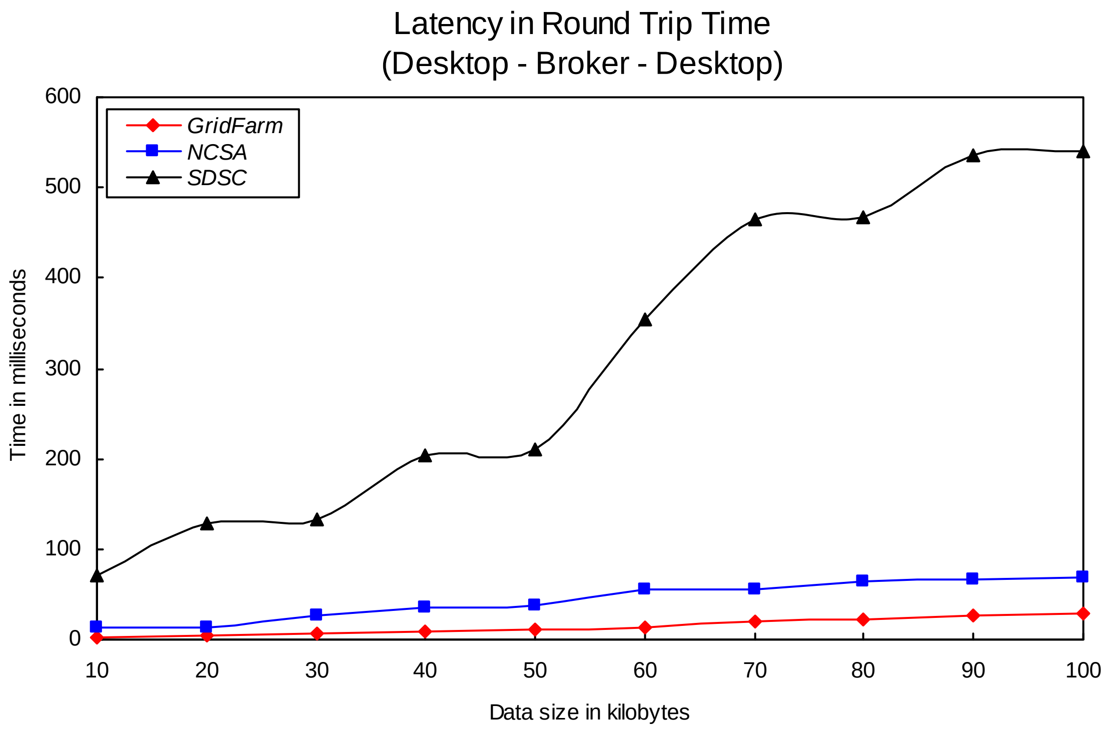


Figure 2.23: Latency in Round Trip Time between Desktop and Broker

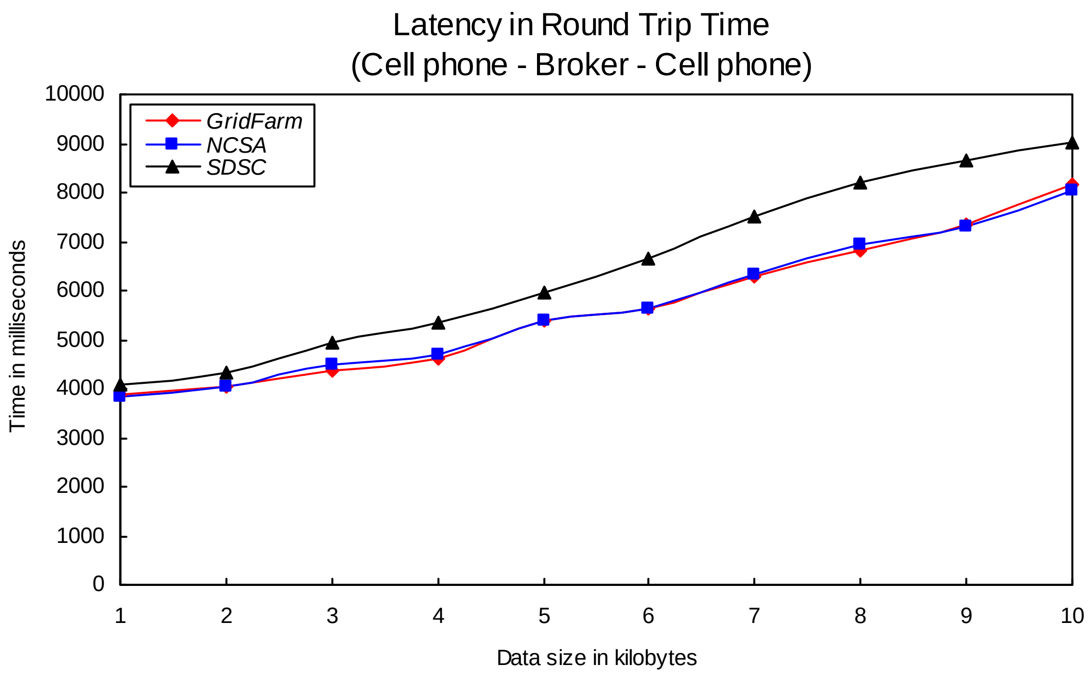


Figure 2.24: Latency in Round Trip Time between Cell phone and Broker

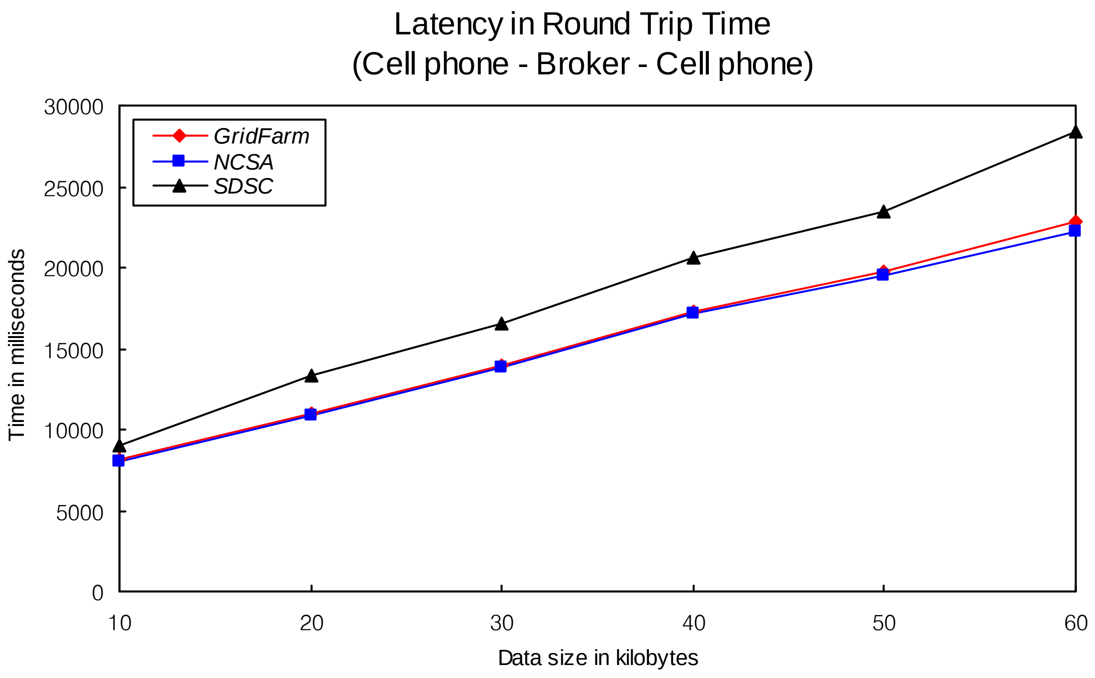


Figure 2.25: Latency in Round Trip Time between Cell phone and Broker

2.6.2 Performance of XGSP Query/Dissemination Mechanism

In this section we show the performance of XGSP query/dissemination interaction mechanism. The XGSP query/dissemination interaction mechanism is based on a set of queries and responses (dissemination through our messaging and service middleware) using session protocols written in XML. To maintain consistent session state information among participants in conferencing collaboration, a user needs to request a presence list of participants in the conference and a list of existing sessions in the conference. A chairperson node (chairperson) disseminates participants' presence information including the presence of the requesting user and all existing session information to all the participants in the conference through a broker. The queries and dissemination for presences and sessions in our implementation involve the XML stream sequences in Figure 2.18, 2.19, 2.20, and 2.21.

The primary costs to measure the performance of the query-dissemination interaction mechanism in XGSP to maintain consistent session state information among collaborating users are as follows.

- Transport cost: The time to transmit the request (query) and receive the response.
- Processing (or Display) cost: The time to process the response including the display (viewing) time of session list in a request node (or host).

The lower graphs in Figure 2.26 and 2.27 show the execution time to process the response in a request node as a function of sessions' number. The time includes parsing XML message replied from a chairperson node and displaying the session information in the parsed message on the device of a request node. The graphs show the processing time increases as the number of sessions increases as expected. The increasing time is more remarkable in the request node on cell phone platform due to low processing performance of the platform. Note that we did not measure the time needed to perform the XGSP query-dissemination mechanism for a presence list of users. The XML stream structure and size of the membership data in our experiment are almost similar to those of sessions. For brevity, we show the results needed to perform the transfer of only session information. The upper graphs in Figure 2.26 and 2.27 show the latency of request-response transport for a variety of session sizes including the corresponding execution time of a chairperson node. The transport cost increases as well when the number of sessions increases as expected. It also shows the execution overhead time incurred by a request node is a smaller part than the overhead time incurred by the networking cost in overall time. In the previous section we showed that the wired networking cost in Figure 2.23 are in the range of only milliseconds and the wireless networking cost in Figure 2.24 and 2.25 are in the range of seconds in our test bed. We note the consistency of session state information among participants in collaboration and independence of the session state information from heterogeneous computing devices in this experiment. The main benefit obtained by applying query-dissemination mechanism and SMMV (Single Model Multiple View) [31, 34, 110] collaboration model in Figure 2.6 from this experiment is consistency and independence of which allow users on heterogeneous computing devices to share the same workspace by disseminating session information through our message/service event system – broker.

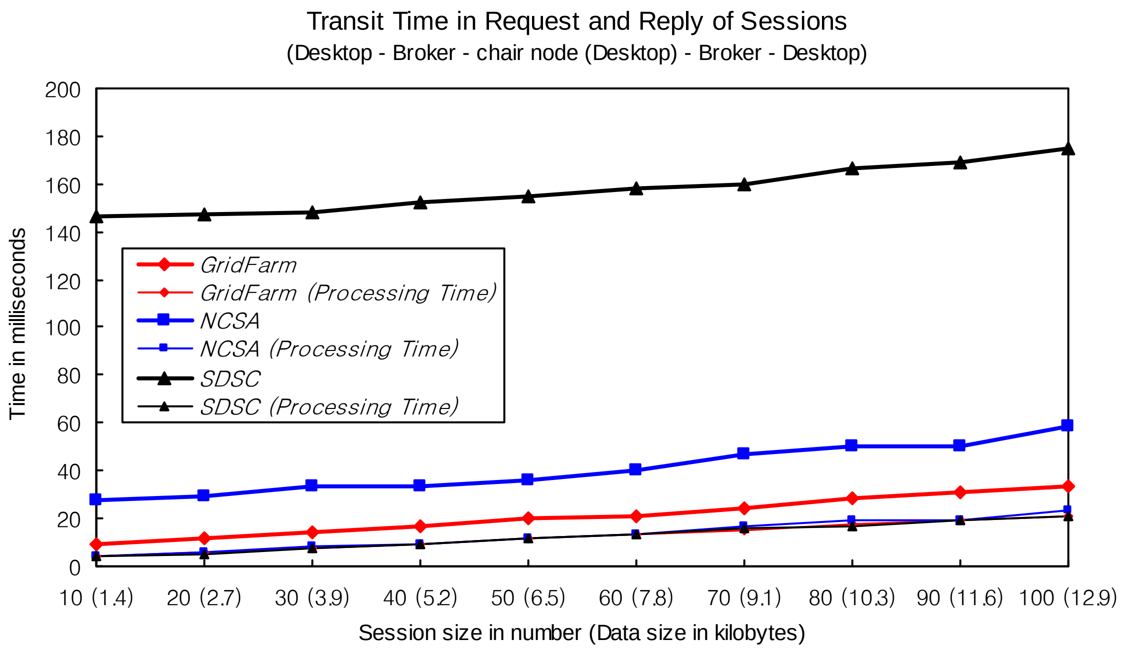


Figure 2.26: Latency in Time of XGSP Request and Response between Request Node (Desktop) and Chairperson (Response) Node (Desktop).

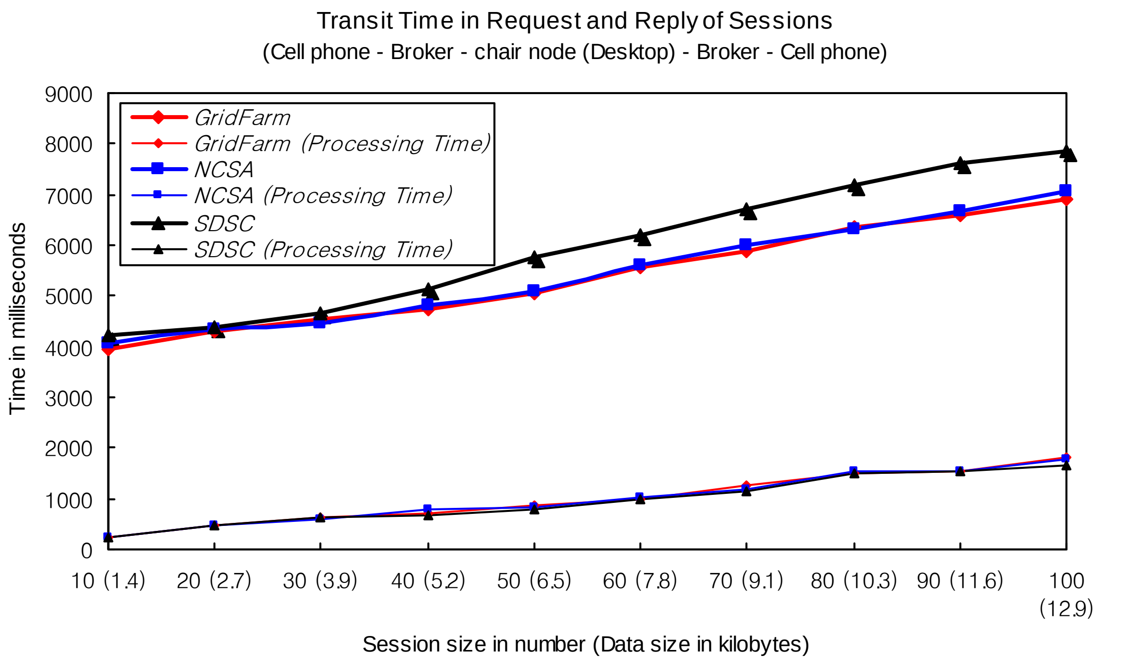


Figure 2.27: Latency in Time of XGSP Request and Response between Request Node (Cell Phone) and Chairperson (Response) Node (Desktop).

2.6.3 A Case Study of Shared Whiteboard Applications with Image Annotation and Its Application Filter

In this section we present experimental results we measured to analyze the performance of the application filter which is one of our architectural components. In this case study we use our collaborative application – shared Java whiteboard with image annotation.

2.6.3.1 Shared Whiteboard and Application Filter

The shared whiteboard is a collaborative enabled drawing application implemented using J2ME on cell phone, and J2SE on desktop. The whiteboard offers a set of drawing tools and a common canvas shared virtually among all users joined in a session. One of the users interacts with a cell phone while another user interacts with a desktop in our experiment. In this collaboration with heterogeneous computing devices, the cell phone device can not directly display objects (drawings and images) represented from the desktop device because of different canvas sizes (160x144 vs. 1024x768) and supported different image format representations (PNG vs. JPEG). Instead, a converted object is retrieved from the application filter through which the object can be represented into the cell phone device in accommodating the capabilities of the device. The application filter is connected to a broker as an application server and is located in the same place where the broker is residing to reduce the latency of the object transfer time for filtering. The filter is implemented with partly the functionality of XGSP integrated into our collaboration framework to dynamically intercept the session information and all the functions in the filter are written in J2SE. To filter (convert) a large graphical image data sent from whiteboard on desktop into a shrunk image data for cell phone, an image is created from the large binary image data and then a BufferedImage [11] class is created, which is a subclass of Java Image API [55] because the class provides a more structured internal design with much greater access to the image data. The created BufferedImage is scaled (shrunk) to the canvas size of a cell phone device and converted to PNG (Portable Network Graphics) image file type for the cell phone. The binary data of the PNG image is sent to a cell phone through a broker and represented into the PNG image format on the cell phone. The image conversion (filtering) diagram is shown in Figure 2.28. A screenshot of this case study is shown in Figure 2.29.

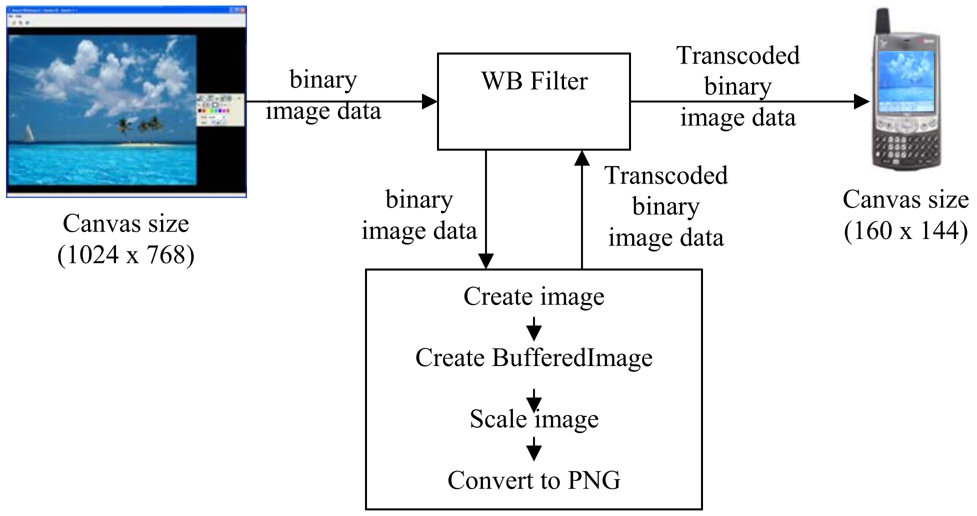


Figure 2.28: Image Filtering Structure

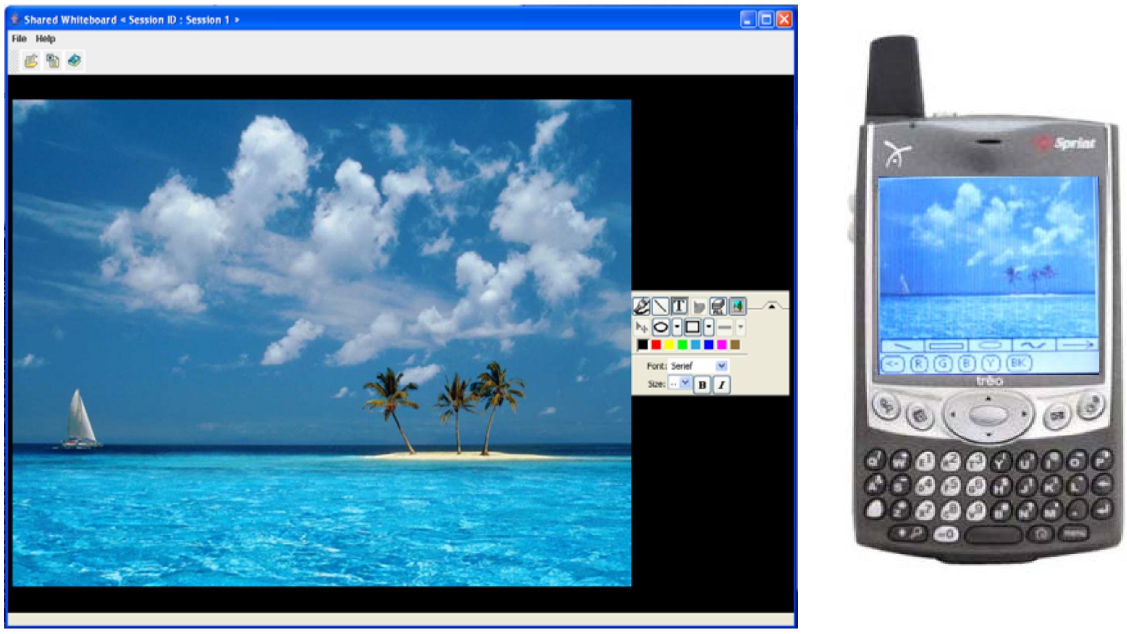


Figure 2.29: 800x600 JPEG Image with 60 KB on Desktop vs. 158x134 PNG Image with 50 KB on Cell phone

Figure 2.29 shows a collaboration of shared whiteboard applications on desktop and cell phone devices. The Figure shows an 800x600 JPEG image with 60 KB sizes loaded from the whiteboard application on desktop and a 158x134 PNG image with 50 KB sizes on cell phone transferred from the desktop after transcoded by the application filter.

2.6.3.2 Performance and Analysis

We measured the latency incurred in transferring binary image data and drawing objects such as line, rectangular, and oval from desktop to cell phone using shared whiteboards and an application filter in the collaboration between desktop and cell phone user. The performance results in time are shown in Figure 2.30 and 2.31. The latency is divided in two components: time of network delay and processing time in the application filter. One interesting observation from these results is: improved performance in time and increased image detail loss. The transfer time is reduced as the size of image data increases, relatively compared to transfer time performed without filtering. The performance enhancement in time is due to transformation from large graphic image size to small size for adapting to the image format and screen size of cell phone device. In our experiments, 1 MB image size is transformed into 52 KB image size in application filter. Then the 52 KB PNG byte image is sent to a cell phone through a broker. Note that we did not measure the latency to transfer more than 60KB data between cell phone and broker since the cell phone device (Treo 600) used in our experiments does not support the transfer of more than 60 KB data size. After the JPEG image is transcoded and scaled by the application filter as shown in Figure 2.29 and Table 2.1, much details of the original JPEG image were lost – one of drawbacks in transcoding between desktop and cell phone. By exploring different transcoding and scaling algorithms, the problem as well as technical limitation occurring as porting applications from desktop computers (moderate screen size) into mobile devices (small screen size) [71] may be able to be overcome in the future.

	Original Size	**Shrunk Size**	**Scaled**
Image 1	61365 bytes (60 KB) (800 x 600)	50664 bytes (158 x 134)	0.2 x 0.22
Image 2	400523 bytes (400 KB) (1792 x 1200)	55812 bytes (158 x 134)	0.09 x 0.11
Image 3	1208494 bytes (1 MB) (1764 x 1180)	52675 bytes (158 x 134)	0.09 x 0.11

Table 2.1: Original Size vs. Shrunk Size

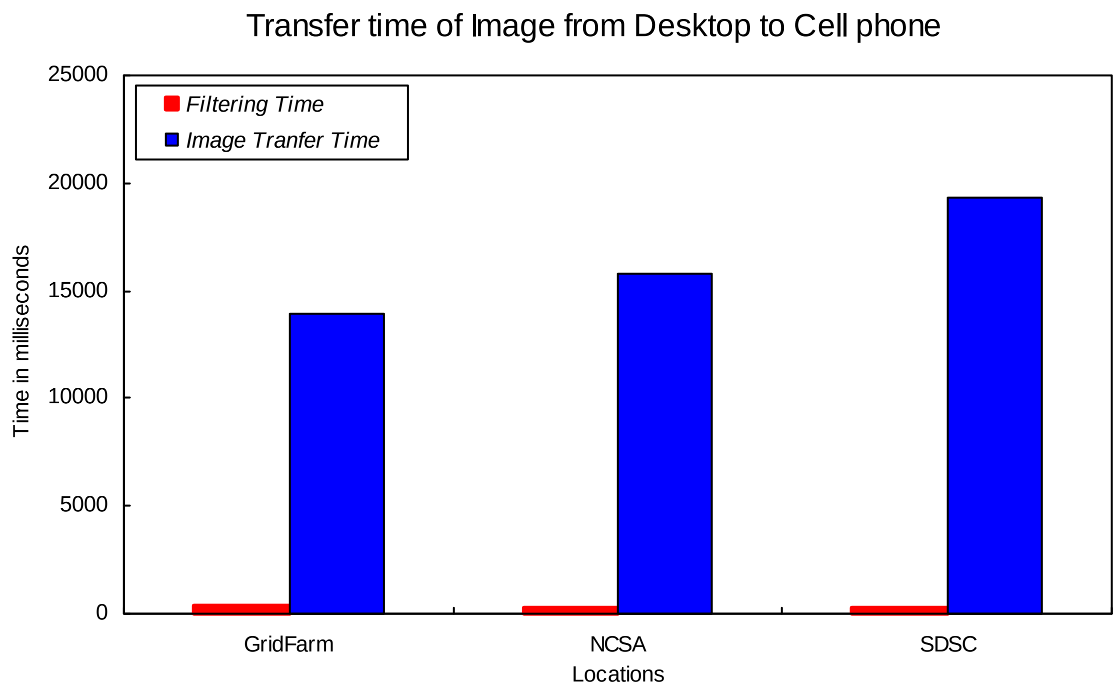


Figure 2.30: Transfer time of Images from Desktop to Cell phone

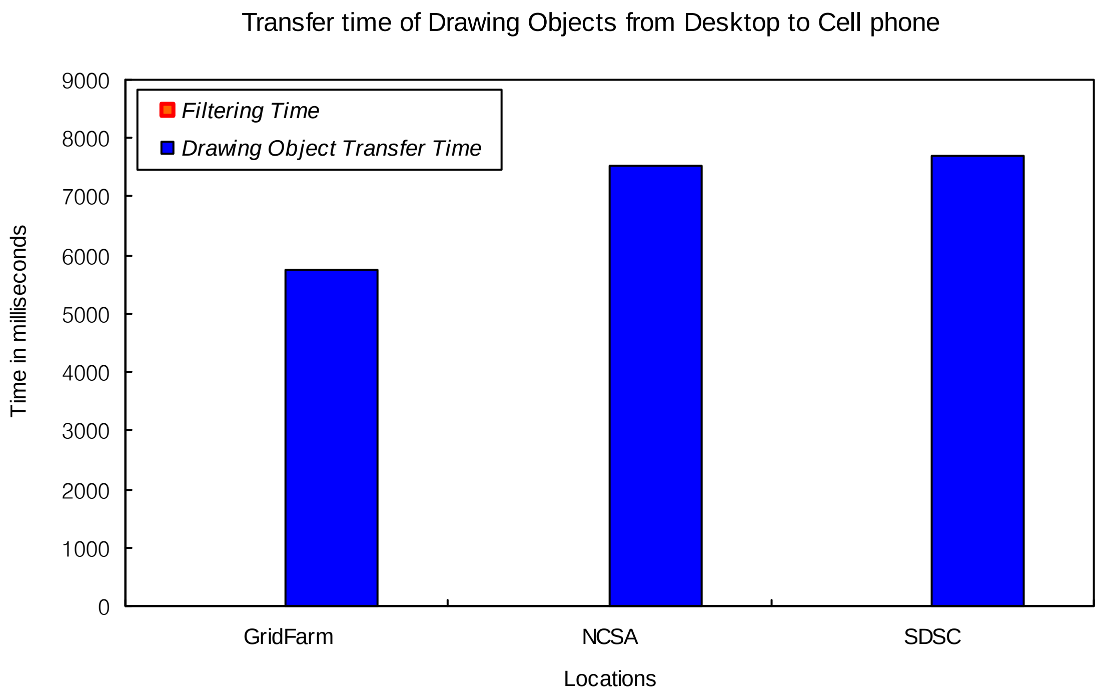


Figure 2.31: Transfer time of Drawing Objects from Desktop to Cell phone

2.7 Summary

In this chapter we presented the effectiveness of architectural functionality and components of collaboration framework built on heterogeneous (wire, wireless) computing environment differing by orders of magnitude in bandwidth and latency. The framework handles cooperation and communication among heterogeneous communities, and provides collaborative applications in the heterogeneous community collaboration. A key function of the framework is to provide a generic solution for controlling sessions in a conference by defining a general protocol in XML.

In our experiments with the collaboration framework, we encountered a few problems. The first problem was found in the experiment of the XGSP query/dissemination interaction event mechanism with cell phone device. The cell phone device needed 12 different screens to display 100 sessions used as a test bed in our experiment. It will cause inconvenience for cell phone user to search for her preferred sessions in scrolling the session lists down while her roaming. A mechanism to provide preferred sessions for roaming users should be taken into consideration in future work.

The second problem was found in the experiment of the application filter with shared whiteboard application. Much details of image displayed on a cell phone device through transcoding of an image transferred from a desktop were lost because the graphical image was shrunk to accommodate the screen size of cell phone. To improve the quality of the transcoded image from desktop into cell phone, we should consider different transcoding and scaling algorithms in future work for heterogeneous collaboration applications on heterogeneous computing platforms.

The third problem occurred with a failure like network disconnection of a chairperson or moderator node (or host). If a chairperson or moderator node fails or is disconnected, and is not able to recover from the failure for some amount of time, one of participants in collaboration capable of having the role capability of the chairperson or moderator has to be elected. We tested it with an event driven message mechanism. However, when the network connection of a chairperson or moderator node was lost, it did not work since the event messages could not be disseminated in disconnected network. One approach to overcome the problem by exploring different fault-tolerant role delegation mechanism with role hierarchy policy will be presented in future work of Chapter 6.

Chapter 3

XGSP Role Based Access Control (XGSP-RBAC)

3.1 Overview

In the previous chapter we presented an architecture for collaboration framework built on heterogeneous (wire, wireless) computing environment differing by orders of magnitude in bandwidth and latency. Also we presented XML based General Session Protocol (XGSP) for controlling sessions and participants' presences in a conference by defining a general protocol in XML.

In collaboration environment, applications generally have different types of roles which are assigned to a group of users. Chess game application example includes two players, and observer roles. Then, collaborating users can have different access rights for such collaborative applications based on roles which are assigned to them. In this chapter, we present a generic moderator-mediated interaction (request-response) mechanism –

XGSP-RBAC (XGSP Role Based Access Control) for controlling accesses to resources and its supporting architecture integrated into our collaboration framework. The XGSP-RBAC uses the concept of the role [15, 16, 75, 92] as an intermediate entity between collaborating users and collaboration resources and defines policies in XML to describe access privileges on collaboration resources.

The following scenario illustrates the needs of access control and motivates the design issues described in this chapter. Students in CGL at Indiana University are going to have a session for their colleague's research presentation. Some students join the presentation session in a shared conference room of CGL and others join at remote locations by using CGL's conferencing collaboration tool – Global-MMCS system (Global Multimedia Collaboration System) [37]. The presenter starts her presentation with the conferencing collaboration tool. During her presentation, she may use an application like the shared whiteboard, which was introduced in the previous chapter, to discuss design issues of the research which she is doing on grid computing. In shared workspace with the application, people in offline session see the same whiteboard canvas, while people in online session see their own canvases. Each student in the online session has their own canvas and a set of interfaces to the shared whiteboard application but they see the same results (or views) as others do. Her advisor, researchers, and colleagues in CGL want to make comments on her research by directly manipulating the shared application showing the same views among participants in her research presentation session. Thus, the presenter needs to control their accesses to the shared application by enforcing who is allowed to access the application, and the conditions under that the privileges for the use of the application occur to restrict unauthorized access for the protected application.

The rest of this chapter is organized as follows. We put research issues and our solution about them in section 3.2. Section 3.3 describes related works. Section 3.4 presents the architecture of XGSP Role Based Access Control (XGSP-RBAC) integrated into our collaboration framework and the implementation of it. Section 3.5 discusses the experimental results obtained from the practical evaluation of XGSP-RBAC mechanism. We finally conclude by summarizing main points drawn from the XGSP-RBAC.

3.2 Problem Statement and Solutions

Fundamentally collaboration is about interaction among people and between people and resources. The cooperation on the resources shared among a group of users may hence produce new results on the shared resources. On the contrary, security is about restricting unauthorized access to resources and thus it is essential that security of the collaboration environments as well as of collaborative applications running on them is ensured while providing the openness only to users that are authorized to access them. Therefore, difficulties to deal with the conflicting goals of allowing and restricting accesses for resources among a group of users may happen in collaboration environment. The examples of the difficulties include protecting secured computing environments and resources from unauthorized users as well as unsecured remote devices since the environments and resources can be compromised by inadequately secured entities – human, devices, software, data, and so on.

The activities in collaboration system include the interactions for the use of resources as well as for cooperation among a group of users working at remote locations. The interaction for resources involves not only the use of applications but also the use of hardware devices, software, and data. Some resources in the interaction activities may require authorized access, meaning the resources can be accessed by only authorized users. For the resources an access control policy and a mechanism to enforce the policy should be implemented defining which resources are available, who is allowed to access the resources, and the conditions under that the privileges for the use of the resources occur.

In traditional system such as file system, access rights in access control schemes are usually static permissions that are permanent during the interactive activity in the system [18]. Access control schemes need flexible access rights adapting to the state change of collaborative resources that may be occurred from cooperation in collaboration system. Collaboration system thus needs a scheme to enable collaborating users or collaborative applications to control accesses during their activities at run time.

In collaboration environment collaborating users are generally assigned a role, and collaborative applications have different types of roles which are assigned to a group of users. Access control scheme in collaboration system hence needs fine-grained access control for providing accesses for individual users in group, and for a finer granularity of accesses on individual resources shared in group. In other words, an access control scheme for collaboration environment should allow independent specification of each access right of each user on each protected resource [94]. For example, it should allow fine-grained drawing actions and support protection for each of them in whiteboard application.

In this chapter we show a moderator-mediated interaction (request-response) mechanism, which uses role entity between collaborating users and collaboration resources for ease of administration, fine-grained access control, and flexible adaptation of collaboration environment's changes.

3.3 Related Work

In this section we examine existing access control schemes for collaboration system.

3.3.1 Access Control Matrix

Access control matrix is a scheme that describes current allowed accesses using a matrix. It characterizes the access rights of each subject associated with respect to each object in a system [7, 75]. In the scheme, the subjects, which are active protected entities such as processes and users, operate on objects (protected entities) using a matrix which describes the access rights of each subject over each object in a system. Figure 3.1 shows an example access control matrix. Most of collaboration systems in which groups of users work together can have thousands of objects and thousands of subjects. Then the storage requirements will be too much. Thus, the access control matrix scheme can be impractical due to the storage requirements in the collaboration system. Instead, variants of the access control matrix such as access control lists and capabilities enable systems to use more convenient and more optimized mechanisms which eliminate the storage problem.

	File 1	**File 2**	**File 3**
Alice	Read	Read, Write, Own	Write
Bob	Read, Write, Own		Read, Write, Own

Figure 3.1 Access Matrix Model

3.3.1.1 Access Control Lists

To implement the access control matrix this mechanism stores each column with the object which associates it with lists of a set of pairs (subjects and rights) it contains. In this implementation, on a dynamic collaboration system with many subjects, the storage requirement for ACLs (Access Control Lists) may still be very large in some degree. This mechanism lack flexible ability to examine all access rights a subject has. This need to examine the access rights of every other object with respect to a subject. The corresponding access control lists for objects like files in Figure 3.1 are

ACL (File 1) = {(Alice, {Read}), (Bob, {Read, Write, Own})}

ACL (File 2) = {(Alice, {Read, Write, Own})}

ACL (File 3) = {(Alice, {Write}), (Bob, {Read, Write, Own})}.

3.3.1.2 Capability Lists

This mechanism stores each row with the subject which associates it with lists of a set of pairs (objects and rights) it contains. The list is known as a capability list. Likewise this mechanism lack flexible ability to examine all subjects over an object. This needs to examine the capability lists of all subjects. The corresponding capability lists for subjects in Figure 3.1 are

Capability (Alice) = {(File 1, {Read}), (File 2, {Read, Write, Own}), (File 3, {Write})}

Capability (Bob) = {(File 1, {Read, Write, Own}), (File 3, {Read, Write, Own})}.

Access control matrix scheme lacks fine-grained control of access to objects, and thus allows subjects to have more access rights than ones needed when performing an operation to objects in a system. That is, the scheme does not follow the principle of least privilege [75] to reduce introducing compromises. Also, collaboration includes cooperation on objects shared among groups of users. The scheme provides access rights for the cooperation of users in collaboration system but needs more flexible support for a finer grained manipulation of access rights for individual users. An example framework using ACLs is a SUITE [94] which is a multi-user editing framework. Shen and Dewan [94] extended the conventional access matrix scheme in several ways: the use of collaboration rights, the support of negative rights which is explicit denial of a right, the use of inheritance rules and conflict resolution rules. Another example is a Globus Security Infrastructure (GSI) [38] which provides a coarse-grained access control approach and uses a mapping list. The mapping list is used to map user's local account name to DN (Distinguished Name) on the user's certificate. When a user wants to use a service, the mapping list is consulted and the access for the service is granted or denied depending on whether she or he appears on the list with the correct credentials. An example framework using Capability is a XPOLA (eXtensible Principle of Least Authority) [67] which provides fine-grained authorization solution for Grid services to follow the principle of least privilege. Another example is a Community Authorization Service (CAS) [12, 69, 70] which will be described in section 3.3.4. The CAS implements the capability scheme using an authorization server called CAS server.

3.3.2 RBAC (Role Based Access Control)

RBAC model [15, 16, 75, 92] is a scheme that describes access rights using the notion of roles predefined in organizations. It characterizes the relationship between users and access right for resources with respect to roles based on job functions in organizations. The relationship includes permission assignment and user assignment; access rights for resources are assigned to roles (permission assignment) and users who are authorized to assume the associated roles are assigned to the roles (user assignment). That is, access rights are connected not to a user but to a role which is an intermediate entity located between users and access rights as shown in Figure 3.2. In the scheme, access for resources is authorized to the users that have roles predefined in organizations if the roles are allowed to access the resources.

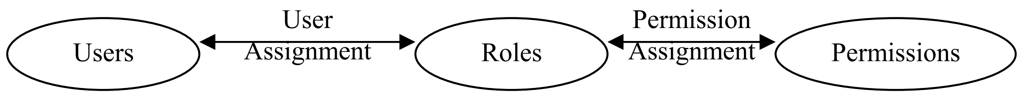


Figure 3.2: Assignment Relationship between Users, Roles, and Permissions in RBAC

As RBAC scheme is applied to the collaboration system that can have thousands of users, it is more scalable than the schemes using the notion of subject-object because the number of roles is generally smaller than that of users in an organization [3]. Also the scheme reduces the administrative overhead costs occurring with management of users since administrator can easily assign or revoke users' role membership from one role to other roles without modifying access rights assigned to roles. Whereas RBAC scheme is very effective for collaboration systems with respect to ease of administration and scalability issues, it is not effective with relation to flexibility and fine-grained control issues. RBAC scheme lacks fine-grained access control for providing accesses for individual users in groups and for a finer granularity of accesses on individual resources. In traditional system such as file system, access rights in RBAC as well as subject-object schemes are usually static permissions that are not changed during the interactive activity in the system. Collaboration system includes sharing resources and cooperation on them among groups of users and thus needs a scheme to enable users or collaborative applications to control access during their activity at run time. To make collaboration system flexible for giving users or their applications authorization to decide access for resources, OASIS [112] role-based access control model addresses the issues of role activation and deactivation based on first-order logic which specifies parameters of conditions to determine the activationdeactivations. An example framework using RBAC scheme is PERMIS (Privilege and Role Management Infrastructure Standards) [23, 24, 25, 26, 90] which will be described in section 3.3.3.

3.3.3 PERMIS (Privilege and Role Management Infrastructure Standard)

The Privilege and Role Management Infrastructure Standards (PERMIS) [23, 24, 25, 26, 90] is a RBAC authorization infrastructure to utilize a scalable X.509 Attribute Certificate (AC) [53] based Privilege Management Infrastructure (PMI). The PMI uses AC which holds a binding between a user and her privilege attributes. The ACs are issued to users and a resource gatekeeper reads the privilege attributes in the users' ACs to see if they are allowed to access resources. PERMIS system uses RBAC mechanism based on the X.509 AC for authorization infrastructure. The PERMIS RBAC mechanism provides a set of Java API for a resource gatekeeper to inquire if access for resources should be allowed. The PERMIS also provides XML based policies with fine-grained control capabilities [23]. The rules in the policies specify subject policy (defines the domains of users that may be granted roles), SOA (Source of Authority) policy (lists the SOAs that are trusted to assign roles to subjects), role hierarchy policy (defines the relationship of roles that has a directed graph structure) and role assignment policy (defines which roles can be assigned to which subjects by which SOAs), delegation policy (defines whether the assigned roles can be delegated), target policy (defines the domains of targets), action policy (defines actions for access to a target), and target access policy (defines which roles are allowed to perform which actions on which targets).

In the PERMIS scheme, roles are assigned to subjects with X.509 AC. And the roles and policies are stored in one or more LDAP [68] repositories. Then a user can create a proxy certificate with unique identifiers (the Object Identification (OID) number of the policy in the LDAP repository and the URI of the LDAP repository) to submit an access for resources. Through the identifiers, the PERMIS authorization service grants or denies the access request from the user.

The privilege verification subsystem in PERMIS defines two key components to authorize access to the target based on ACs as depicted in Figure 3.3. Access control Enforcement Function (AEF) authenticates a user and asks Access control Decision Function (ADF) if the user is allowed to perform the requested action on target resource. ADF accesses LDAP to retrieve policy and role ACs for the user and make a decision based on them.

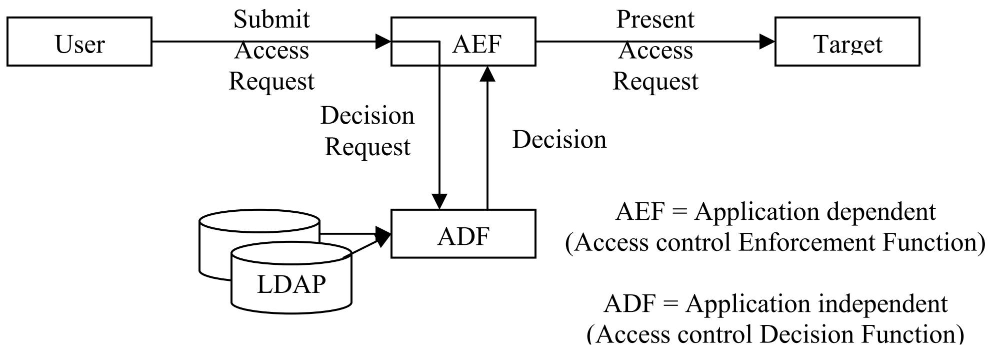

User
Submit Access Request
AEF
Present Access Request
Target
Decision Request
Decision
ADF
LDAP
AEF = Application dependent (Access control Enforcement Function)
ADF = Application independent (Access control Decision Function)

Figure 3.3: Privilege Verification Subsystem

3.3.4 CAS (Community Authorization Service)

Community Authorization Service (CAS) [12, 69, 70] implements the capability scheme using an authorization server called CAS server. Resource providers establish a trust relationship with the administrator of a community served by CAS and then delegate a fine-grained access control policies to the administrator as depicted in Figure 3.4. A user issues a request to the CAS server in her community. The CAS server issues a proxy credential with capabilities (access right lists granted to access resources) to the user. Then the user uses the proxy CAS credential to access the resources.

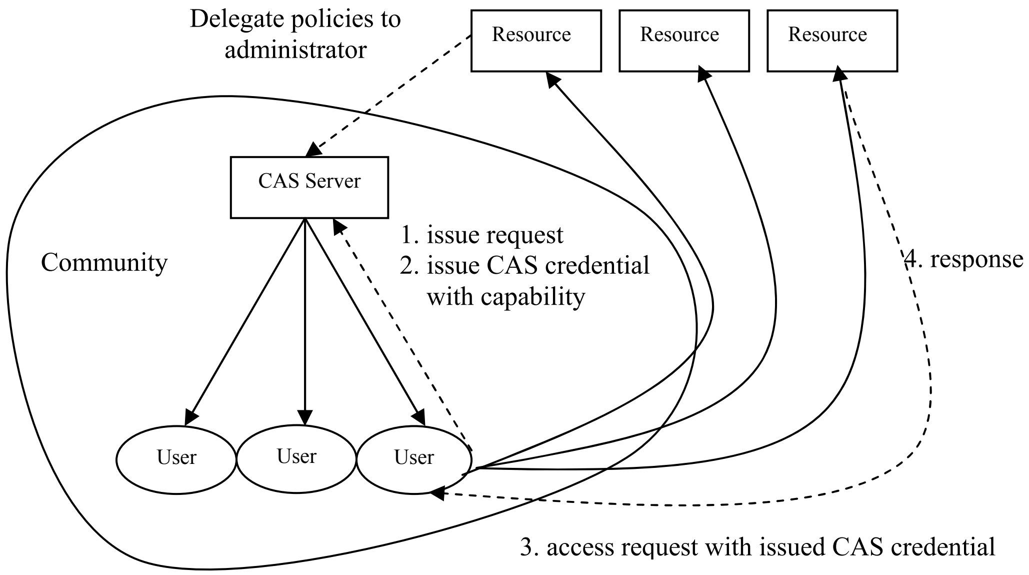

Delegate policies to administrator
Resource
Resource
Resource
CAS Server
Community
1. issue request
2. issue CAS credential with capability
4. response
User
User
User
3. access request with issued CAS credential

Figure 3.4: Community Authorization Service (CAS)

The example resource that can be accessed through CAS is GridFTP [42, 105]. The paper [4] implements RBAC scheme using the CAS server. Since centralized characteristic of the CAS server, CAS service may have scalability problem in very large VOs (virtual organizations) [46] which form a group of users and a collection of resources shared among them, and also single point of failure problem of the CAS server.

3.4 XGSP-RBAC (XGSP Role Based Access Control)

The basic idea behind RBAC [15, 16, 75, 92] is the notion of role used as an intermediate entity between users and protected resources. The intermediate entity – a role is assigned to a group of user with which collaboration is associated and is assigned a set of access rights to perform operations on resources in the collaboration. XGSP-RBAC uses the concept of the role as an intermediate control entity between collaborating users and collaboration resources. The XGSP-RBAC provides effectiveness with respect to ease of administration, flexible adaptation to the state change of collaboration resources, and fine-grained access control. It uses XML for policy specification as well.

- ✓ Collaboration roles in XGSP-RBAC are a representation to categorize users joining a conference for collaboration. The roles are based on the users' privileges and devices' capabilities allowed to manipulate the protected resources in the collaboration.

- ✓ In XGSP-RBAC collaboration, the use of role simplifies the administrative management of access rights for resources since a user can easily be reassigned from one role to other roles without modifying the access control policy. Also, the use gives an administrator flexibility adapting to the change of collaboration environment by allowing a user to take multiple roles simultaneously, assigning new roles to the user, or revoking roles from the user. The XGSP-RBAC scheme provides flexibility adapting to the state change of collaborative resources that may be occurred from cooperation among collaborators at run time in collaboration system.

- ✓ A fine-grained access control for the instance of individual resource is used in collaboration. For example, the actions (access rights) to perform operations on the whiteboard which is a shared application in our collaboration are fine-grained into line, rectangular, oval, pen (a series of contiguous lines) drawings, and so on. Also, a fine-grained access control on individual user in a role can be used. For example, a moderator in collaboration can give access rights for resources to a specific user in a role (a user in a workgroup or in a session) since XGSP-RBAC uses moderator-mediated interaction mechanism. But a moderator needs to give a user the least of privilege needed in collaboration session (principle of least privilege [75]) in the fine-grained access control on individual resource.

- ✓ To specify access control policies and exchange request-response messages of access control for resources between normal user node (request node (or host)) and moderator node (response node), XML is used for streaming request-response messages of access control for resources and for specification of policies since it is easy to understand and use with pre-existing industry standard parsers.

XGSP-RBAC is a role based access control mechanism mediated by a moderator in collaboration, where policy is written in XML and stored in a local policy store – a file residing in each node (or host). The policy is dispatched to each node from the conference manager shown in Figure 2.1 of Chapter 2 at joining time in a conference. The XGSP-RBAC architecture is composed of four major components: activation/deactivation service, access control decision service, local policy store, and authentication and secure delivery service. At request time for accessing collaboration resources, a user sends a request message in XML stream to moderator node (or moderator). XGSP-RBAC mechanism makes its decisions according to the policy read from the policy store of moderator node at decision time. If the request is validated by the access control decision service, then a moderator in collaboration grants or denies the requesting user's access to the collaboration resources. At decision response time, a moderator responds a decision to the requesting user in XML stream as well.

The following subsections provide protected resource access policy, collaboration role and fine-grained action definition, secure end-to-end delivery of messages for authentication and encryption-decryption of messages, and the architecture of XGSP-RBAC integrated into our collaboration framework.

3.4.1 XGSP-RBAC Policy

XGSP-RBAC policy specifies which roles (users in them) in collaboration are allowed to perform which actions on which target resources. The XGSP-RBAC policy (resource access policy) is described in terms of roles, protected resources (collaborative applications), and fine-grained actions permitted on the protected resources. Also, an access type is placed on the resource access policy based on the characteristics of collaborative applications. The access type in our collaboration means rules categorized to access collaborative applications. The access type includes shared, exclusive, released, and implicit types. The access type shared means the fine-grained action in a collaborative application can be shared among collaborating users. The access type exclusive means the fine-grained action is not able to be shared among collaborating users. It hence means a floor control mechanism has to be able to provide the floor for the action on the shared application for only one participant in the synchronous online session at a time. The access type release means the action with the type can be used for releasing the action a user holds. For example, in our whiteboard application, the action slave has the access type released. The access type implicit means the action with the type can be granted without the mediation of moderator according to the resource access policy. In the whiteboard application, a moderator has actions with the access type. The grant mechanism with this type is similar to the capability scheme of access control matrix holding a capability token (a set of access rights). In our collaboration system, a role is a collection of representations capable to operate on collaborative applications with heterogeneous computing devices. We used chairperson, moderator, non-mobile users (desktop users), mobile-users (cell phone users), and chess players (white player, black player, and observers) as a set of example roles in our collaboration system. Actions are a set of operations permitted on the protected resources. The type of actions is dependent on the type of resources and the capabilities supported by heterogeneous computing devices (desktop and cell phone). For example, the role non-mobile-user (desktop user) can have actions including capability moving drawing objects (line, rectangular, oval, pen) in our shared whiteboard application with image annotation while the role mobile-user (cell phone user) is not able to have the capability moving the objects because the whiteboard application on mobile device (cell phone) does not support the capability. Note that we did not define the role hierarchy policy in the XGSP-RBAC policy and implement the mechanism to enforce the policy, and hence we will design and implement it with fault-tolerant role delegation issue as a next phase in future work. The example XGSP-RBAC policy, used in our collaboration system, is shown in Figure 3.6.

As described in Chapter 2, a user has to join a conference by sending her initial presence in join-conference XML stream to a moderator node and a conference manager before the user can establish a session in the conference on the conference manager in order to receive policies for setting session policies up and accessing to resources. The conference manager informs a XML stream binding a policy that is used for requests of protected resources and then she can be an active member of the predefined role assigned in the collaboration. An example of the policy binding stream is shown in Figure 3.5.

- Conference manager informs a XML stream binding a policy that is used for requests of resources.

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplyPolicy>
        <ConferenceID>ourtestroom</ConferenceID>
        <User>
                <UserID>kskim</UserID>
                <UserName>kangseok-kim</UserName>
        </User>
        <Policy>
                <XGSP-RBACPolicy>
                …………………
                </XGSP-RBACPolicy>
        </Policy>
</ReplyPolicy>
```

Figure 3.5: XML Stream Binding a Policy from Conference Manager showing conference ID, user ID, user name, and resource access policy (XGSP-RBAC Policy).

3.4.2 Collaboration Role and Fine-grained Action in XGSP-RBAC

Collaboration roles in XGSP-RBAC are a representation to categorize collaborating users joining a conference session for collaboration. The roles are based on the users' privileges and devices' capabilities to manipulate protected shared collaborative applications. In this section we present how collaboration roles used in XGSP-RBAC are represented. For the representation we use functional notion to show the relationship between roles, and action privileges.

```
<XGSP-RBACPolicy>
 <ResourceAccesspolicy>
  <ResourceAccess>
  <RoleName>mobile-user</RoleName>
  <ApplicationRegistries>
   <ApplicationRegistry>
    <ApplicationID>wb</ApplicationID>
    <MainClass>cgl.myprofessor.whiteboard.Whiteboard</MainClass>
    <Actions>
     <Action>
          <ActionName>slave</ActionName>
          <Capabilities>read</Capabilities>
         <AccessType>released</AccessType>
     </Action>
     <Action>
          <ActionName>master</ActionName>
          <Capabilities>read+write</Capabilities>
         <AccessType>exclusive</AccessType>
     </Action>
     <Action>
          <ActionName>line</ActionName>
          <Capabilities>linedrawing</Capabilities>
         <AccessType>shared</AccessType>
      </Action>
      <Action>
          <ActionName>rect</ActionName>
          <Capabilities>rectdrawing</Capabilities>
         <AccessType>exclusive</AccessType>
      </Action>
      <Action>
          <ActionName>oval</ActionName>
          <Capabilities>ovaldrawing</Capabilities>
         <AccessType>shared</AccessType>
      </Action>
      <Action>
          <ActionName>pen</ActionName>
          <Capabilities>pendrawing</Capabilities>
         <AccessType>exclusive</AccessType>
      </Action>
      <Action>
          <ActionName>clear</ActionName>
          <Capabilities>clear</Capabilities>
         <AccessType>exclusive</AccessType>
      </Action>
     </Actions>
     </ApplicationRegistry>
    </ApplicationRegistries>
  </ResourceAccess>
 </ResourceAccesspolicy>
</XGSP-RBACPolicy>
```

Figure 3.6: An Example of XGSP-RBAC Policy with the Role Name mobile-user and Application Name whiteboard

In role abstraction domain of the function we express the collaboration roles to be assigned to users joining sessions. In action representation domain of the function we express actions permitted to manipulate protected collaborative applications in sessions. The function representation is shown in Figure 3.7. The definition of the collaboration actions depends on the type of applications. As an example we use shared whiteboard application for the definition of actions in Backus-Naur Form (BNF) below. In BNF we also define collaboration roles and actions as follows.

CollabApp ::= WB

CollabRole ::= Chairperson | Moderator | Non-mobile User | Mobile User

CollabAction ::= Master | Slave | Line | Rect | Oval | Pen | Eraser | Clear | Load | Move

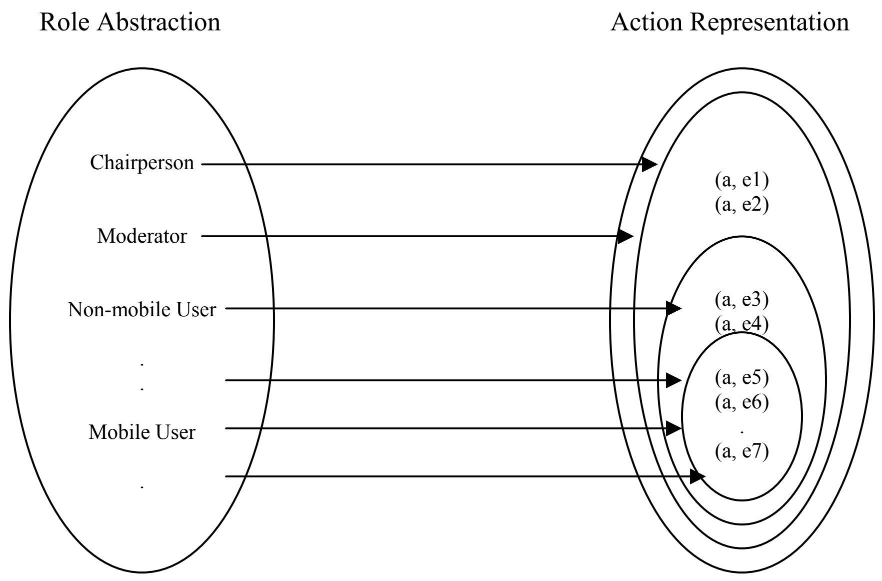


Figure 3.7: A collaboration action is represented as a pair (a, e) ∈ A x E, where a ∈ A is an application and e ∈ E is the authorized smallest major event defined by a, and A is a set of applications, E is a set of the smallest major events defined by an application in A.

We define fine-grained actions in our collaborative application as the smallest interactive major events (semantic events [110]). For example, in the whiteboard application, drawing a line includes clicking, dragging, and releasing a mouse on the whiteboard canvas. For a user working alone with the whiteboard, user input events (low level events such as mouse click, drag, and release) can be interactive major events between the user and whiteboard application. For users working with others sharing the application, the smallest major event means "drawing a line" (semantic event) and the user input events will then be an event data (mouse click – the first point of the line and mouse release – the second point of the line). CGL built a shared SVG (Scalable Vector Graphics [93]) browser and a collaborative chess game application with SVG [100, 109, 110]. In the collaborative chess game application, the smallest major events are to click on an object, to move, and to release the object during moving the object. After the completion of each move (as the mouse is released), the semantic event (moving an object) is dispatched to another player as the smallest interactive major event. Then the user input events will be an event data for moving an object in the chess game affecting the chess board (view-sharing) of another player as well as observers. Therefore, the major events can be different according to the types of applications. The fine-grained action in our collaboration means an interactive smallest major event affecting the shared view (or result) among users in collaboration.

3.4.3 Secure and Authorized End-to-End Delivery of Messages

In this section we present a security framework [97] for secure and authorized end-to-end delivery mechanism of messages between entities (publishers and subscribers) in our messaging system based on publish-subscribe paradigm. The messages delivery for communication between the entities is based on the knowledge of topic. Publisher publishes messages over the topic while subscriber registers a subscription to the topic. The capabilities for creation, advertisement, discovery, and restriction of topics are provided by Topic Discovery Node (TDN) [96] which is regarded as a specialized node in the system. Topic owner creates and advertises topics, and enforces constraints related to the discovery of the topics through the TDN. The TDN advertises the signed topic which is regarded as a secure topic in the system. Publisher encrypts the content payload of a message with the secret key that is retrieved from Key Management Center (KMC) [97] and signs the encrypted payload involving computing the message digest of it and encrypting this hashed value with private personal-key. Also the publisher signs signed-payload with a secret token that is generated from KMC. An authorized subscriber verifies the signature to ensure the message's integrity and decrypts the encrypted payload with the previously distributed secret key.

As shown in Figure 3.8, the security framework is structured as five major components: Certificate Authority (CA), Topic Discovery Node (TDN), Key Management Center (KMC), publisher and subscriber. We describe the components in turn.

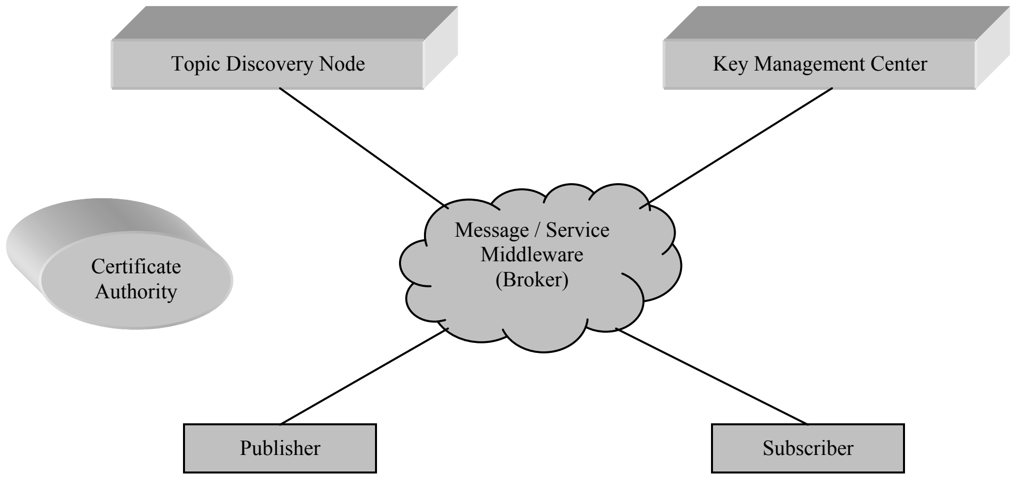


Figure 3.8: The security framework consists of five major components: Certificate Authority (CA), Topic Discovery Node (TDN), Key Management Center (KMC), Publisher and Subscriber

3.4.3.1 Certificate Authority (CA)

A CA is responsible for issuing certificates to entities and managing revocation lists pertaining to compromised entities within our messaging system. The CA notifies brokers and KMCs within the system about any additions to the revocation lists.

3.4.3.2 Topic Discovery Node (TDN)

This node [96] provides topic discovery and creation scheme for the creation, advertisement, and authorized discovery of topics by entities within our messaging system. Through this node, topic creators can advertise their topics and enforce constraints related to the discovery of the topics.

3.4.3.3 Key Management Center (KMC)

A KMC [97] is a specialized node within the system which is responsible for managing information pertaining to secure topics. The KMC generates secret symmetric key for encrypting-decrypting the content payload of messages and security token for establishing entity's rights and duration of them over a secure topic. Also this maintains the list of authorized entities and information related to the entities.

3.4.3.4 Topic Publisher

Publisher encrypts the content payload of message with the secret key that is received from KMC. The publisher signs the encrypted message and security token together by computing the message digest of the encrypted content payload and then encrypting this computed message digest with its private key. After performing the procedures, the publisher disseminates the message through our messaging system.

3.4.3.5 Subscriber

Subscriber includes the security token related to the secure topic in its subscription request. Through verifying header and payload signatures of received message and decrypting the message, the subscriber consumes the message.

3.4.4 XGSP-RBAC Architecture

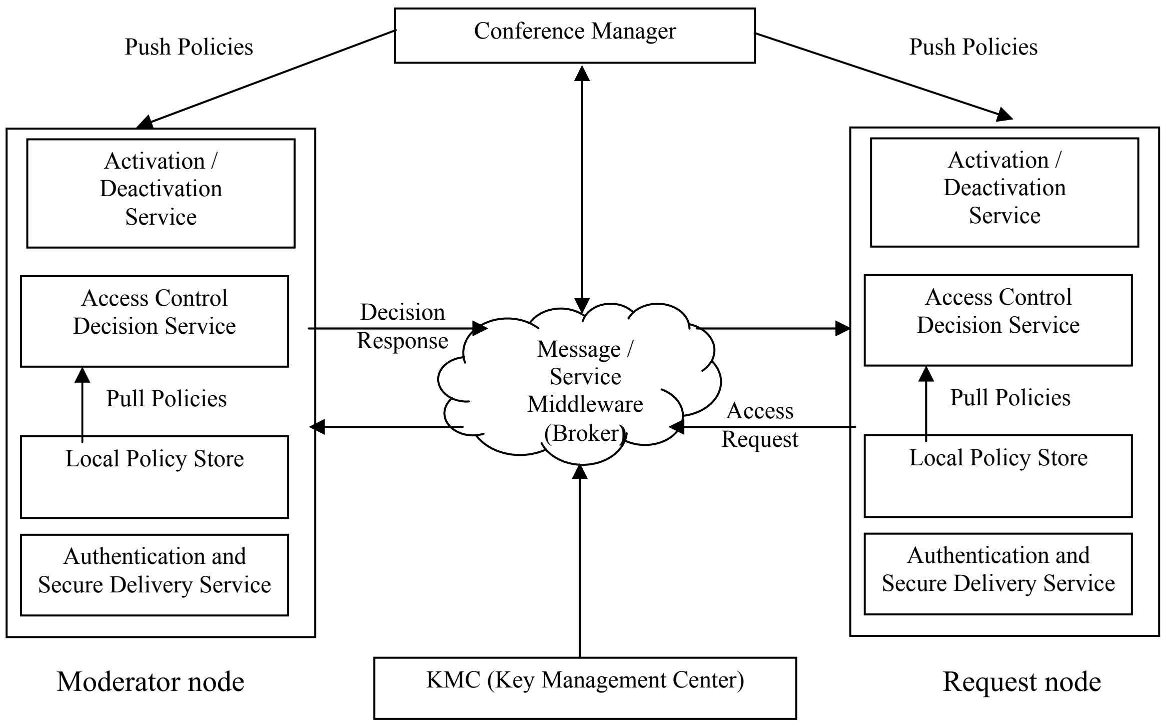
As shown in Figure 3.9, the XGSP-RBAC manager integrated into our collaboration framework is structured as four major components: activation/deactivation service, access control decision service, local policy store, authentication and secure delivery service. We describe the components in turn.

3.4.4.1 Activation / Deactivation Service

When a user requests an action for accessing a protected resource in a session, the request is transformed into a XML stream as shown in Figure 3.10 and the XML stream is sent to a moderator node through a broker from the communication channel of the request node. Then, the request from the request node is passed to the access control decision service in the access/floor control manager of a moderator node through the action request/reply handler shown in Figure 2.8 of Chapter 2 to ask if the request action is allowed to perform an operation on the requested resource. The following two streams show the action request and grant decision response stream between a request node and a moderator node.

- **Access Request Stream**

A list of actions available for accesses of protected resources in a session is represented with actions which other active users currently hold in the access control GUI of each node. The GUI will be shown in Figure 4.7 and Figure 4.9 of Chapter 4. The human-computer interaction with the GUI transforms the access request of a user to perform an operation over a protected resource into a XML stream. The following example XML stream in Figure 3.10 transformed from the human-computer interaction enables a user (user id: kskim) to request an action (action: pen) over a protected resource (application: whiteboard) in a session (application session ID: NewSession).



Conference Manager
Push Policies
Push Policies
Activation / Deactivation Service
Access Control Decision Service
Pull Policies
Local Policy Store
Authentication and Secure Delivery Service
Decision Response
Message / Service Middleware (Broker)
Access Request
Activation / Deactivation Service
Access Control Decision Service
Pull Policies
Local Policy Store
Authentication and Secure Delivery Service
Moderator node
KMC (Key Management Center)
Request node

Figure 3.9: XGSP-RBAC manager integrated into collaboration framework is structured as four major components: activation/deactivation service, access control decision service, local policy store, authentication and secure delivery service.

```
<?xml version=”1.0” encoding=”UTF-8”?>
<RequestAction>
        <AppSessionID>NewSession</AppSessionID>
        <UserID>kskim</UserID>
        <ActionDescription> pen</ActionDescription>
</RequestAction>
```

Figure 3.10: Action Request XML Stream

➢ **Access Grant / Deny Stream**

To check the access privilege of a user over a protected resource, 3-tuple <role name, protected application name, request action name> is consulted in the access control decision service of moderator node. If the role of the requester is allowed to perform the request action according to the resource access policy in the XGSP-RBAC policy, then the request action to access the protected application is granted. Otherwise, the request action is denied. The XML stream in Figure 3.11 enables a user (user ID: kskim) to execute the request action (action: pen) over a protected resource (application: whiteboard) in a session (application session ID: NewSession). Then, the granted action with the name of the user is represented in the access control GUI of each node as an active action of the user in the session. The GUI will be shown in Figure 4.7 and Figure 4.9 of Chapter 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<SetAppAction>
        <AppSessionID>NewSession</AppSessionID>
        <UserID>kskim</UserID>
        <ActionDescription> pen</ActionDescription>
</SetAppAction>
```

Figure 3.11: Grant Decision Response XML Stream

3.4.4.2 Access Control Decision Service

Policy manager in collaboration framework shown in Figure 2.8 of Chapter 2 reads the XGSP-RBAC policy from a local policy store, e.g. a file. The requested action is validated against the policies in the XGSP-RBAC policy read from the policy store. The validation is to check if the action is allowed for the role assigned to the user and for the resources considering all the conditions specified within the resource access policy. If the request is invalid, it is denied. If the request is valid, access type decision service returns an access type value to the access control decision service. The access control decision service makes a decision based on the returned access type value. The decision from the service is passed to moderator. Then the moderator makes a decision on the request. The decision is transformed into a XML stream as shown in Figure 3.11 and the XML stream is sent to the request node through a broker from the communication channel of moderator node.

3.4.4.3 Local Policy Store

When a user joins a conference, the conference manager shown in Figure 3.9 sends a XGSP-RBAC policy to the user by the XML stream as shown in Figure 3.6. The policy is stored in a file residing in the user's node. This ensures that the policy is up-to-date and consistent among collaborating users. Note that our mobile device, Treo600 [102] cell phone, does not support writing the policy into itself. The phone then throws a security exception. Thus we held the policy as a string during an online session.

3.4.4.4 Authentication and Secure Delivery Service

As described in section 3.4.3, this service encrypts the content payload of decision response message with the secret key that is received from KMC. This service signs the encrypted message and security token together by computing the message digest of the encrypted content payload and then encrypting this computed message digest with its private key. After performing the procedures, a moderator node disseminates the encrypted decision through a broker. The request node consumes the decision response from moderator node through verifying header and payload signatures of received decision response message and decrypting the message.

Note that we did not implement the encryption mechanism of messages for roaming users with cell phone. In future work we will design and implement the authentication service for users joining a conference during roaming with cell phone devices, and the encryption service of messages sent to and from the cell phone devices.

3.5 Performance and Analysis

In this section, we discuss an experiment with our collaborative application built in heterogeneous (wire and wireless) computing environment to show the viability of XGSP-RBAC mechanism. The main purpose of the experiment is to identify key factors that influence the performance of XGSP-RBAC mechanism comparing overheads incurred from wired-networked environment with those incurred from wireless-networked environment. In the experiment, we measured mean network transit time (request-response time), mean waiting time in a queue and mean access control decision service time in a moderator node involved in performing communication (an access request for resources and a decision response) between the request nodes and response node (moderator node) for mean interarrival time among access requests in heterogeneous networked environments over a variety of locations.

In the experiment, we utilized two desktop devices, one cell phone and one broker. The collaboration framework on cell phone and desktops is located in Community Grids Lab at Indiana University. The broker ran on a 2.4 GHz Linux with 2 GB RAM located in Community Grids Lab at Indiana University, a 1.2 GHz Linux with 8 GB RAM located in NCSA (National Center for Supercomputing Applications) at UIUC (University of Illinois at Urbana-Champaign), and a 1.2 GHz Linux with 8 GB RAM located in SDSC (San Diego Supercomputer Center) at UCSD (University of California at San Diego). The experiment results were measured from executing collaboration framework and the shared whiteboard application built on the framework running on Palm OS 5.2.1H Powered Treo600 [102] cell phone platform with 144 MHz ARM Processor and 32MB RAM connected to cellular network, and running on Windows XP platform with 3.40 GHz Intel Pentium and 2 GB RAM and Windows XP platform with 3.40 GHz Intel Pentium and 1 GB RAM connected to Ethernet network respectively. The application codes on the cell phones are written in J2ME (Java 2 Micro Edition) [62] and the application codes on the desktops are written in Java 1.5 [55]. A conference managing server (conference manager) is operated as an apache web server. The XML activities on non-mobile (desktop) devices are parsed by and handled with JDOM [57] that is a Java implementation of Document Object Model (DOM) [27]. The XML activities on mobile devices (cell phones) are parsed by and handled with kXML [66] that is a J2ME implementation of DOM. The following subsections show experimental scenario, overhead timing considerations, and analysis about the performance measurements.

3.5.1 Experimental Scenario

Our experiment is carried based on the XGSP-RBAC mechanism which is described in section 3.4. The access request for resources from a request node and the decision response from a moderator node in the experiment involve the XML streams in Figure 3.10 and Figure 3.11 respectively. The experiment is also carried with the simulation program which is behaved by Coloured Petri-nets (CP-nets) [64]. The simulation program uses the exponential function provided by the CP-nets to generate access requests with pre-known mean interarrival time. The access request arrival times form a Poisson process since the interarrival times of the requests are independent random variables with exponential distribution with pre-known mean interarrival rate. In our experiment, we suppose the requests randomly arrive with the pre-known arbitrary mean interarrival rate. The experimental scenario overview is depicted in Figure 3.12. Note that we did not use the decision behavior of a moderator (human) since the behavior of a human does not reflect the consistent reaction in time that may affect the latency of requests waiting in a queue. The decision result from the access control decision service will thus be directly sent to request nodes without the decision interruption of a moderator. We discuss the overhead costs in the next subsection and how these affect XGSP-RBAC mechanism as involved with cell phone devices since cell phone devices are sensitive to the network delay as shown in Figure 2.24 and Figure 2.25 of Chapter 2.

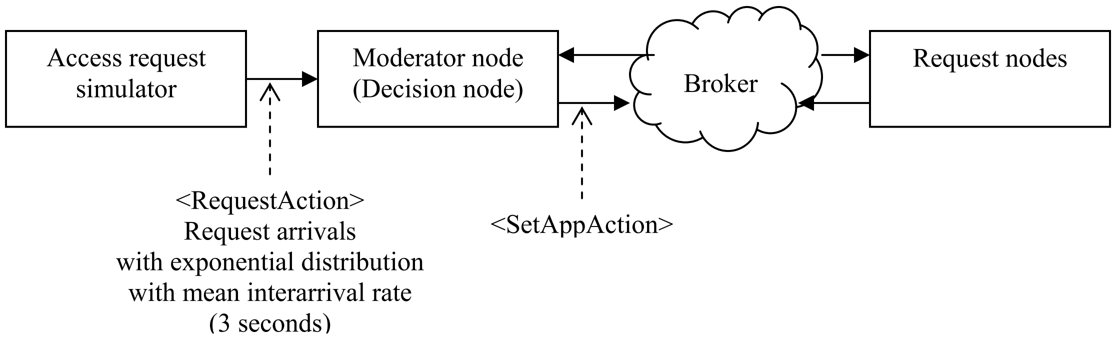

Access request simulator
Moderator node (Decision node)
Broker
Request nodes
<RequestAction>
Request arrivals with exponential distribution with mean interarrival rate (3 seconds)
<SetAppAction>

Figure 3.12: Experimental Scenario Overview

3.5.2 Overhead Timing Considerations

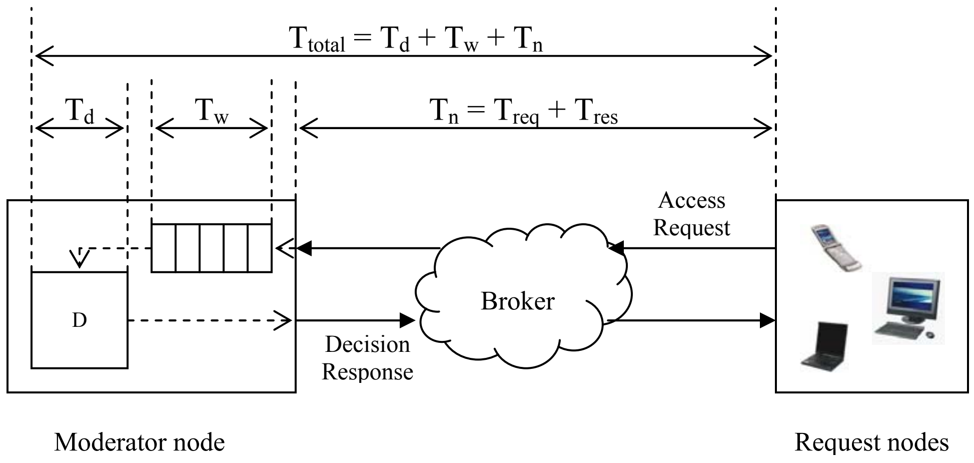


Figure 3.13: Total latency = Decision time (T_d) + Waiting time (T_w) + Network transit time ($T_n = T_{req} + T_{res}$), where D means an access control decision service

Figure 3.13 shows a breakdown of the latency for serving a request. The cost in time for XGSP-RBAC mechanism has three primary overheads.

- Transit cost ($T_n = T_{req} + T_{res}$) – The time to transmit an access request (T_{req}) to and receive a decision response (T_{res}) from moderator node.
- Access control decision service cost (T_d) – The processing time to make a decision on an access request for resources at moderator node. This cost includes reading a XGSP-RBAC policy from a file and validating access requests from the policy.
- Waiting cost (T_w) – The time between arriving at a queue and leaving the queue (being served by the access control decision service) at moderator node. The queue is implemented as FIFO (First-In, First-Out) order. The arrival of new request is modeled as Poisson processes with arrival rate λ where the interarrival times between interarrival requests are independent random variables with exponential distributions with mean interarrival rate $1/\lambda$. The arrival rate λ means the average number of arrivals in unit time. To get independent random variables with exponential distributions with some mean interarrival rate in terms of the arrival time variable of new request, we simulated the exponential distribution of arrival times with an automated simulation tool [14]. The simulation tool randomly generates independent new access requests with an arbitrary mean interarrival rate which is already known before the simulation of the new requests' arrival.

Examining overhead costs and total cost, we measured the mean overhead cost for 100 access requests in heterogeneous networked environment over a variety of locations. The results are summarized in Table 3.1 with the mean completion time of a request.

3.5.3 Experimental Result and Analysis

In this section we present an experimental result we have measured to analyze the overheads incurred from controlling fine-grained accesses in XGSP-RBAC mechanism. The simulator generates new access requests on behalf of users on request nodes. The access request generation process follows an exponential distribution. The generated request events, according to the order delivered from the simulator, are stored in a request queue. The experiment is run through the mean request interarrival time (3000 milliseconds) which is an average interarrival time between two successive requests issued by the simulator.

Figure 3.14 depicts mean completion time of a request vs. mean request interarrival time for three different network combinations involved in our collaboration over three different locations: collaboration using only desktop devices (wired network), collaboration using only cell phone devices (wireless network), collaboration using desktops and cell phones together (wired and wireless network). The comparison shows when cell phone devices using wireless network are involved in our collaboration, the mean completion time of a request is increased since the wireless network has high latency. In the case of the use of cell phone, we may need to make the granularity of fine-grained actions larger to reduce the wireless network overhead. The shared whiteboard application uses fine-grained actions with the smallest major events as described in section 3.4.2. When a user requests an image loading action, it may be natural to simultaneously request it with some drawing actions. This natural request with larger-grained action can improve response (delay) time of a request but decrease the amount of concurrency and introduce complexity. The degree for granularity is a balance between responsiveness and concurrency [6] and between responsiveness and simplicity. Also, without user's point of view [41] for the granularity of actions, unnatural granularity may violate the principle of least privilege because it may give a user more privilege than needed. The experimental result shows that in future work we need to observe user's behavior with applications in collaboration environment considering responsiveness vs. concurrency, responsiveness vs. simplicity, and responsiveness vs. principle of least privilege.

	GridFarm at CGL			**NCSA at UIUC**			**SDSC at UCSD**		
	milliseconds								
	D	**D + C**	**C**	**D**	**D + C**	**C**	**D**	**D + C**	**C**
Mean transit time	4.29 (7.05)	2205.4 (1840.1)	4674.5 (1704.6)	39.63 (8.69)	2159.4 (1897.6)	4581.9 (1692.7)	257.64 (12.79)	2585.9 (2058.8)	4875.2 (1741.2)
Mean waiting time	1.09 (3.98)	1.7 (4.86)	1.74 (4.97)	1.69 (4.83)	1.42 (4.53)	1.85 (5.03)	2.06 (5.35)	2.17 (5.41)	1.74 (4.97)
Mean decision time	4.85 (10.1)	4.56 (10.03)	3.9 (9.76)	4.39 (8.91)	4.68 (9.04)	3.28 (9.49)	4.23 (7.98)	3.76 (8.67)	3.13 (8.29)
Mean completion time	10.23	2211.66	4680.14	45.71	2165.5	4587.03	263.93	2591.83	4880.07

Table 3.1: Mean completion time of a request vs. Mean request interarrival time (3000 milliseconds) where D means collaboration using only desktop devices (wired network), D + C means collaboration using desktops and cell phones together (wired + wireless network), and C means collaboration using only cell phone devices (wireless network)

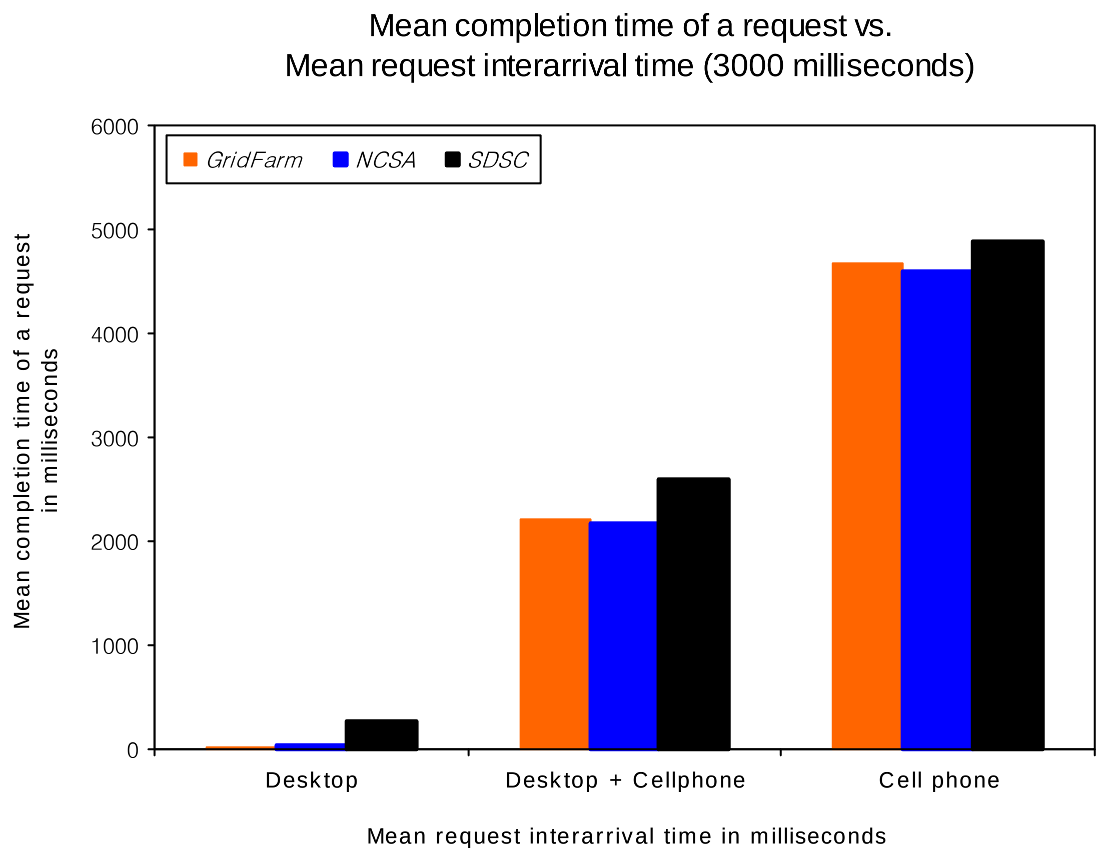


Figure 3.14: Mean completion time of a request vs. Mean request interarrival time (3000 milliseconds)

3.6 Summary

In this chapter, we presented the XGSP-RBAC mechanism integrated into our collaboration framework. The XGSP-RBAC uses the notion of role as an intermediate control entity between collaborating users and collaborative applications. The roles in XGSP-RBAC are based on users' privileges and devices' capabilities to allow users to manipulate the protected applications in the collaboration. The use of role simplifies the administrative management of access rights for applications and gives an administrator flexible adaptation to the changes of collaboration environment. Also, XGSP-RBAC mechanism provides flexibility adapting to the state change of collaborative applications that may be occurred from cooperation among collaborators at run time in collaboration system. To specify access control policies and exchange request-response messages of access control for resources, it uses XML because it is easy to understand and use with pre-existing industry standard parsers. Also, fine-grained access control for the instance of individual application as well as for individual user is used.

In future work, we will design and implement the authentication service for users joining a conference during roaming with cell phone devices, and the encryption service of messages sent to and from the cell phone devices.

We left support of role hierarchy policy for the problem of fault-tolerant role delegation which occurred with a failure like network disconnection of a moderator node in our collaboration system as described in the summary of Chapter 2. If the moderator node fails or is disconnected, and can not recover from the failure for some amount of time, one of users capable of having the role capability of moderator has to be elected by the role hierarchy policy.

Chapter 4

XGSP Floor control (XGSP-Floor)

4.1 Overview

In the previous chapter, we presented a moderator-mediated interaction mechanism (XGSP-RBAC) for controlling accesses to resources in our collaboration system. The XGSP-RBAC mechanism uses the concept of a role entity between collaborating users and collaboration resources. Also, we showed overhead costs incurred from performing the mechanism (request-response interaction in XML stream) with heterogeneous networked environments over a variety of locations.

In this chapter, we present a policy and a mechanism implementing it – XGSP-Floor (XGSP Floor control) for coordinating concurrent activities to synchronous resources and maintaining shared state consistency at application level by defining a general protocol in XML. Also, we describe the functionality of a XGSP-Floor tool that provides a user interface (human-computer interaction) for control of floor to a moderator and participants in a session with desktop and cell phone devices, a major event conflict detection function which detects whether an action in a floor request conflicts with the action of current floor holder, and a non-optimistic locking mechanism which is used in our synchronous collaboration from moderator's point of view and participant's point of view.

The following scenario, which is continued from the previous scenario in Chapter 3, illustrates the needs of access control at synchronous application level (floor control [17, 18]) and motivates design issues described in this chapter. As participants in her research presentation session try to manipulate the shared application at the same time, she has to be able to provide the right to access the shared application for only one participant in the session at any time to ensure the consistency of the shared application state. The shared application, that requires mutual exclusions in real time, has to be assigned to only one participant who requests it under a set of well-defined rules. Participants in offline session can use the rules of etiquette or social protocols to gain the manipulation of the shared application in an order by the rules or protocols. However, participants in online session can not use the etiquette rules or social protocols. Therefore, she will need some rules to substitute the etiquette rules and social protocols by defining the time and the way which a participant in collaboration gains access to the shared application – policy and mechanism.

This chapter describes the design and implementation of XGSP-Floor (XGSP Floor control) for coordinating activities occurred in synchronously cooperating applications being shared among participants in an online session. The rest of this chapter is organized as follows. We put research issues and our solution about them in section 4.2. Section 4.3 describes related works. Section 4.4 presents a policy and a mechanism of XGSP-Floor integrated into our collaboration framework and the functionality of an XGSP-Floor tool (floor control tool). Also, we present the conflict detection function and the non-optimistic locking mechanisms used in the XGSP-Floor for synchronous collaboration. Finally, we conclude by summarizing main points drawn from the XGSP-Floor.

4.2 Problem Statement and Solutions

In traditional face-to-face offline session, participants generally follow rules of etiquette or social protocols when they interact with each other. For example, if all the participants try to draw on a shared whiteboard at the same time, then the conflicts which may result in inconsistent state can be solved by a moderator or social protocols. However, in online session or CSCW (Computer Supported Cooperative Work), the social protocols may not be able to be used for coordinating the interaction of participants since they are not collocated. For example, if all the participants simultaneously try to send drawing events through a communication channel in a distributed collaboration system, then the conflicts are not able to be solved by the social protocols used in the face-to-face offline session. Therefore, policies and mechanisms used in the offline session may need a mapping into those able to be used in the online session with user interfaces between participants and CSCW environment.

Users working with synchronous collaboration applications in CSCW environment usually interact with each other using computer-mediated policies with computer-mediated tools which make collaborative works conveniently among them. Such computer-mediated policies in CSCW are generally called floor control [17, 18]. Floor control is the problem of coordinating activities occurred in synchronously cooperating resources shared among participants in an online conference session. The floor control mitigates race conditions within online sessions on who is allowed to manipulate shared data or to send synchronous events.

A set of well defined policies and mechanisms are needed for efficiently coordinating the use of resources in CSCW. The policies for floor control typically describe how participants in CSCW request resources, and how the resources are assigned and released when participants share a synchronous resource such as audio-video control event in conferencing, drawing events in shared whiteboard or moving events in collaborative chess game. Also, mechanisms including user interfaces (human-computer interaction) between participants and CSCW environment are needed to implement and enforce the policies. The floor control mechanisms have to be able to provide the floor on shared resource for only one participant in a synchronous online session at any time.

When users perform concurrent activities on shared synchronous resources such as collaborative applications, floor control is necessary. No single floor control scheme is appropriate for all collaboration applications. The simplest scheme is free-for-all (no floor control) for applications like text chat. The XGSP-Floor integrated into our collaboration framework provides significant flexibility ranging from free-for-all to application specific floor control mechanism for avoiding uncoordinated activities to shared collaboration applications. The moderator (moderator node) is responsible for maintaining the consistent state of applications in our collaboration system. An access conflict detection function detects whether an action in a floor request conflicts with the action of current floor holder. The avoidance of the conflicts ensures applications are maintaining consistent states. When a participant in collaboration requests a floor to access a shared application to a moderator node (moderator), the moderator makes a decision with the decision value validated by and returned from access-floor control decision service. The request floor is then granted or denied according to the decision. Also, by the dissemination of the decision event message through our message and service middleware - broker, all the participants in the collaboration maintain the same shared state information which results in consistent state. The conflict detections depend not only on the applications but also on fine-grained actions (major events) involved in manipulating shared application. For example, our shared whiteboard detects conflicts by fine-grained actions involved in the application. As another example, a collaborative chess game [100, 109, 110] detects conflicts by the application itself. The chess game is a collaborative game application developed by CGL. We show a mechanism for coordinating the conflicts in this chapter and show formal verification by Coloured Petri-Nets [64] to prove the correctness of our modeled consistency mechanism in Chapter 5 in terms of mutual exclusion, deadlock, and starvation.

4.3 Related Work

The coordinated use of resources within conferencing and collaboration system is fundamental to increasing synchronous collaborative applications. In this section we examine existing floor control schemes for collaboration system. Also, we examine considerations for a selection of mechanisms for dealing with consistency in the use of shared whiteboard application among collaborators in our collaboration domains – heterogeneous community collaboration, synchronous and ubiquitous collaboration.

4.3.1 Existing Floor Control Schemes

Dommel [18, 19] classified floor control schemes into two known paradigms, random-access (contention-based) and scheduled-access (token passing-based) floor controls.

- The random-access based floor control scheme includes sensing availability of a resource by users or system, or mediation by social protocols. The sensing floor scheme example is Activity Sensing Floor Control (ASFC [63]). The ASFC provides a mechanism based on sensing activities on a distributed shared resource. By sensing activities on the shared resource, decisions for floor control are autonomously made without the mediation (intervention) of moderator. If the requests are collided, they are backed off until the floor holding the resource is released. The example protocol schemes by mediation are Conference Control Channel Protocol (CCCP [76]) and floor control protocol built on MBone seminars [72]. The CCCP provides moderator-controlled interaction floor control in conference collaboration where a designated moderator gives access rights to a participant who wants to access a shared resource. The MBone videoconferencing system uses centralized moderator-controlled floor control mechanism in question board which is a tool to enable participants to ask questions in large-scale loosely-coupled MBone seminars (sessions). It also enforced the floor control mechanism using the conference bus mechanism developed at LBL (Lawrence Berkely Laboratory). The conference bus is a multicast mechanism which is used for communication between tools provided in a session.

- The scheduled-access based floor control scheme includes autonomous token passing interaction floor control scheme where the token is used to request, grant, deny, or release a floor.

Boyd [8] classified the floor control schemes according to design dimensions such as degree of interaction and granularity of control for floor control policies in multi-user applications. He introduced an interactive and fine-grained policy with user interface for floor control, called fair dragging which can be used in multi-user applications. The policy means that a user has control of an object during only short-term dragging, where dragging means pressing a button over an object which he wants to use and releasing the button when he relinquishes the use of the object. Greenberg [39, 40] classified the floor control schemes for turn-taking between participants with view-sharing applications. He discussed some floor control mechanisms implemented in the view-sharing applications and showed as an example that explicitly managing (or designing) turn-taking floor control with view-sharing (of full screen) applications in a windowed environment is difficult without disturbing the shared view to explicitly activate floor control from user's perspective because there is no room to display information about current state of the floor in the shared full screen. Also, GroupKit [40, 73] provides participants in collaboration with flexible floor control mechanisms – preemptive floor control scheme and ring-passing scheme. The preemptive floor control scheme means that a user can immediately grab the floor from the current floor holder. The ring-passing floor control scheme means that a user can seize the floor if the floor is free. Microsoft's NetMeeting [84] uses locking mechanism for the floor control of the shared whiteboard application provided in it.

Dommel [20] presented the theoretical evaluation for a comparative analysis among floor control protocols (uncoordinated social mediation, social mediation with feedback, activity sensing, direct coordination, ring-based coordination, and tree-based coordination). The evaluation showed the efficacy of the floor control protocols in considering point-to-point communication and broadcast communication with the parameters for control state management by assuming the existence of the broadcast communication. The experimental evaluation of activity sensing protocols for small group sizes is presented in Activity Sensing Floor Control (ASFC [63]). McKinlay [77] evaluated the performance (usability and effectiveness) of four different policies for the management of the turn-taking performed in a computer supported online meeting in comparing with the performance of face-to-face meeting. The used four policies are face-to-face, free-for-all, request-and-grant, and request-and-capture (where the capture means anyone can take a turn at any time). Myers [10] created more comprehensive classification of floor control policies with three independent primitive dimensions: request, acquire, and release control. By combining the primitives, he showed the use of the floor control with puzzle control program in Pebbles project [9] which studies the use of PDAs (Personal Digital Assistants) simultaneously with a desktop PC.

4.3.2 Considerations for a Selection of a Floor Control for Shared Whiteboard Application in Heterogeneous Community Collaboration, Synchronous and Ubiquitous Collaboration Domains

Mechanisms for dealing with consistency in the use of application shared among collaborators in collaboration system will have to be considered in an unambiguous manner according to increasing heterogeneous collaboration applications. When collaborators perform concurrent activities on shared synchronous collaboration application, floor control is necessary. No single floor control mechanism is appropriate for all collaboration applications. There are many different mechanisms for floor control as shown in section 4.3.1, and hence many different mechanisms for floor control of a collaborative application can be considered according to the size of group in the number of participants, individual preferences of participants in group, collaboration style (presentation, brainstorming, design meeting, and so on) or some other reasons. Also, the intrinsic latency occurred due to the increase of interactive distance, relatively to the latency occurred in collocated place, may affect on the choice of the mechanism for shared state consistency of application. By combining different parameters (group size, human considerations, technical implementation considerations, and so on) with our shared whiteboard application, many different floor control mechanisms can be considered. A few example scenarios are:

- Floor control mechanism in small group size. When participants want to send drawing event messages in a distributed collaboration system, to do so may be satisfactory in small group size since the possibility of direct conflicts occurred from manipulating shared synchronous collaboration application may be rare [41, 88]. If there are conflicts that may introduce inconsistency in small group size, the conflicts can be solved by the participants themselves who coordinate the concurrent conflicts by social protocols. If computer-mediated floor control is used, the conflicts can be avoided or resolved by synchronizing them through the computer-mediated consistency mechanism. Then the computer-mediated floor control can be strict or relaxed mechanism according to the degree of the mitigation of race conditions to ensure consistent state to participants, where the strict or non-optimistic floor control mechanism means to avoid conflicts and the relaxed or optimistic floor control mechanism means to allow updates by any host (or participant) on any object (or data) and to resolve the uncoordinated updates [18]. An example is Tivoli 1.0 which is shared whiteboard system designed for supporting small group size meetings in collocated place [88]. All the participants in the meeting with Tivoli 1.0 can have access to the shared whiteboard without official moderator. The Tivoli 1.0 does not provide a computer-mediated consistency mechanism for coordinating activities among participants. The computer-mediated consistency mechanism for coordinating activities among geographically-separated multiple users is implemented in Tivoli 2.0 with the human and technical considerations in disseminating a new shared object so that each host can have consistent shared object and each user can work on the disseminated object copy [81].

- Floor control mechanism in large group size. In small group size, to achieve the agreement of social protocols among participants may be not difficult. Free-for-all which allows participants to send drawing event messages in a distributed collaboration system when they wish may be satisfactory in small group sizes by following social protocols. In large group size, to achieve the agreement of social protocols among participants may be difficult when hosts are heterogeneous, network has high latency, shared tools are more complex, or social protocols are misunderstood among participants [21]. In these cases, if all participants send drawing event messages at the same time, the event messages may race, leading to inconsistent state to participants. Gray [59] in a modeled system showed that as a system scales up in the number of hosts (or participants), the system that performs well on a few hosts with simple transactions may become unstable since the number of conflicts in an optimistic mechanism grows quadratic with the number of hosts and transactions in the system. Therefore, to solve the conflict problems in a distributed collaboration system of large group size may be inefficient without computer-mediated floor control. Due to more conflicts of drawing event messages from different participants in large group size than those in the small group size, some form of floor controls in large group size have to be used to ensure consistent state among hosts (or participants) if inconsistency matters. Then, the choice of a floor control mechanism for the conflict management can depend on human considerations since the interactions for shared synchronous collaboration applications include people as well as computers, and technical implementation considerations [41]. In the human considerations, individual preferences of the participants in group for choosing floor control mechanisms may have to be considered even though people's preferences often do not match with performance [10]. Also, locking, serialization, and the latency of interactions over networks for the selection of a floor control may have to be considered according to people's preferences in group [41]. In the technical implementation considerations, the complexity for implementing optimistic locking and serialization, and network transactions (undo/redo) may have to be considered since optimistic mechanism is more difficult to implement than non-optimistic mechanism [41]. Also, the overheads incurred from the computational complexity of some optimistic schemes may have to be considered [41]. For example, many different floor control mechanisms for shared whiteboard application in large group size can be considered using the mechanisms recommended by Dommel [18] – negotiation, token passing, token asking, time stamping, two-phase locking, blocking, activity sensing, reservation, and dependency detection.

Our collaboration framework is designed to support the construction of heterogeneous collaboration applications on our collaboration domains – heterogeneous community collaboration as well as synchronous and ubiquitous collaboration. The choice of a floor control mechanism for coordinating concurrent activities among participants on shared whiteboard application built on the collaboration framework can be considered from the three different collaboration domains.

4.3.2.1 Considerations for a Selection of a Floor Control in Heterogeneous Community Collaboration Domain

The heterogeneous community collaboration means to integrate different collaboration communities into a global collaboration community. Our hypothesis for the number of participants in the heterogeneous community collaboration is that there may be a number of participants linked together for collaboration among heterogeneous communities as compared with small number of participants linked together in a community. In the hypothesis with the heterogeneous community collaboration domain, if the possibility of concurrent activities occurred from manipulating shared whiteboard application among a number of participants linked together for the purpose of collaboration is small or rare, the floor control mechanism for their concurrent activities may be relaxed. Otherwise, the floor control may have to be a strict mechanism due to the increase of the complexity and overheads incurred from the concurrency management. As Gray [59] predicted through a modeled system, if the number of hosts or participants in a collaboration system with a relaxed or lazy (optimistic) consistency control scheme increases, the scaled system may have consistency problems due to the occurrence of more concurrent activities which may introduce the increase of transaction operations. Master copy replication (primary copy) scheme can reduce the problems as a system scales up [59]. For example, this scheme means the master of shared whiteboard application is responsible for maintaining the consistent state information of the application in collaboration. If concurrent activities are detected at the master, the reconciliations for concurrency are disseminated to the other hosts. But this may also introduce the complexity for the reconciliations and the propagation delay of the reconciliations for consistency among hosts or participants, violating the characteristics of synchronous collaboration if specifically the concurrent activities increase. Therefore, for a selection of a floor control for shared whiteboard application in the heterogeneous community collaboration which may be increasingly scaled up, we will need to consider the possibility of concurrent activities, the complexity for managing them, and the overheads incurred from them with the growing number of hosts or participants in the heterogeneous communities.

4.3.2.2 Considerations for a Selection of a Floor Control in Ubiquitous Collaboration Domain

The ubiquitous collaboration means capability of multiple users to link together with disparate access devices in anytime and anywhere. The relaxed or strict floor control mechanism for shared whiteboard application in the ubiquitous collaboration domain can be considered differently according to network latency. Since the wireless cellular network has high latency as shown in section 2.6.1, the selection of a floor control mechanism in the collaboration linked with cell phone devices has to consider overheads – the network transactions for undo/redo operations in optimistic mechanism and the waiting time for turn-taking among participants in non-optimistic mechanism.

If concurrent activities among participants increase or are not small in optimistic mechanism, then the number of the network transactions (undo/redo) will increase, leading to the increase of complexity for managing the transactions and transformations of objects on shared whiteboard, and the increase of overhead time for processing them in specifically cell phone devices which have low computing performance. In non-optimistic mechanism, the turn-waiting time to provide a turn for only one participant at a time may increase. If concurrent activities are small, the number of network transactions for undo/reo operations will be small in optimistic mechanism and the waiting time for turn-taking may decrease in non-optimistic mechanism.

Therefore, for the selection of a floor control for shared whiteboard application run on wireless cell phone devices which have high latency and low computing performance in ubiquitous collaboration domain, we will need to consider the effects of network transactions in optimistic mechanism vs. the waiting time for turn-taking among participants in non-optimistic mechanism according to the occurrence of the increasing or decreasing number of concurrent conflicts.

4.3.2.3 Considerations for a Selection of a Floor Control in Synchronous Collaboration Domain

The synchronous collaboration means to allow all participants in collaboration to have the same views and data at all times in real time. The relaxed or strict floor control mechanism in the synchronous collaboration domain may have to be considered differently according to intermittent network disconnection of mobile devices as well. Mobile hosts may be disconnected from collaboration for arbitrary periods of time until reconnected into the collaboration. During the disconnected periods of time, connected users might generate new objects on the shared whiteboard, or some objects on the whiteboard might be removed or transformed, and hence disconnected hosts (or participants) may have inconsistent state information different from other hosts connecting (or joining) to the collaboration. Therefore, we need a scheme to provide consistent state information to disconnected users as reconnected – for example, when a disconnected host (or participant) joins a collaboration session, a moderator or an agent who is responsible for maintaining the consistent state information of shared whiteboard application in collaboration needs to send the host all up-to-date updates since the host was disconnected.

Also, in optimistic mechanism, as a disconnected host is reconnected while the moderator or agent manages redo/undo operations for consistency, the reconnected host may have an inconsistent view with the need of the additional computation for the operations and transformations, leading to the increase of computational complexity for managing them, and the degradation of computing performance on cell phone device as well. In non-optimistic mechanism, a disconnected user may also have an inconsistent view with some degree of delay as reconnected while a connected single host does actions for update, but with less complexity and computations than those in the optimistic mechanism.

Therefore, for the selection of a floor control for shared whiteboard application in synchronous collaboration domain, we will need to consider intermittent network disconnection of cell phone devices with the effects of network transactions and the computational complexity for managing them on cell phone devices in optimistic mechanism as well as non-optimistic mechanism.

The two distinct relaxed and strict floor control mechanisms exemplified with our collaboration domains how the decision for the selection of a floor control mechanism with shared whiteboard application can be made with the following considerations:

- the possibility of concurrent activities, the complexity for managing them, and the overheads incurred from them with the growing number of hosts or participants linked together for collaboration among heterogeneous communities in heterogeneous community collaboration domain

- the effects of network transactions in optimistic mechanism vs. the waiting time for turn-taking among participants in non-optimistic mechanism according to the occurrence of the increasing or decreasing number of concurrent activities in ubiquitous collaboration domain

- the intermittent network disconnection of cell phone devices with the effects of network transactions and the computational complexity for managing them on cell phone devices in optimistic mechanism as well as non-optimistic mechanism in synchronous collaboration domain

In this thesis we focus on moderator-mediated floor control with non-optimistic mechanism using conflict detection function and non-optimistic locking for coordinating concurrent activities on shared whiteboard application in our collaboration domain of large group size with a number of participants. The moderator mediates concurrent activities on shared whiteboard application among participants, and sends disconnected or newly joining hosts (or users) all up-to-date updates when the disconnected or new hosts join a collaboration session. The non-optimistic floor control mechanism is used for reducing the number of network transactions and the complexity of operations occurred from network transactions in our collaboration involved with cell phone devices which use slow wireless network and have low computing performance. But, the non-optimistic mechanism in our collaboration may increase the waiting time for turn-taking among a number of participants if the number of concurrent activities increases. In future work we will apply the moderator-mediated floor control mechanism to synchronous media applications such as audio and video applications, and consider different floor control mechanisms with different parameters for floor control of the shared whiteboard application in our collaboration domains – heterogeneous community collaboration as well as synchronous and ubiquitous collaboration.

4.4 XGSP Floor Control (XGSP-Floor)

This section describes a floor control policy, a floor control mechanism (XGSP-Floor) implementing the floor control policy, and the functionality of a XGSP-Floor control tool to provide participants in collaboration with human-computer interaction for control of a floor, where the human-computer interaction means user interfaces between participants and CSCW environment. We also describe a conflict detection function and a non-optimistic locking mechanism used in the XGSP-Floor.

4.4.1 XGSP-Floor Policy

This section describes an XGSP-Floor policy (floor control policy) that defines how the participants in synchronous collaboration session request a floor for the use of a collaborative application, and how the floor for the use of the application is assigned and released when the participants share the synchronous collaboration application. The XGSP-Floor policy is written in XML as shown in Figure 3.6 in Chapter 3. An example policy from the Figure 3.6 is shown in Figure 4.1. The element access type in the example policy describes whether a fine-grained action of an application can be shared among participants. If a fine-grained action is not able to be shared among participants, a floor control mechanism has to be able to provide a floor for the action on a shared application for only one participant in a synchronous collaboration session at a time. We define a set of predicate rules (policies) used as determinants in decision procedure of a moderator node in terms of the following three types of predicate statements (request, response, release) to provide a floor for only one participant at a time:

1. Participants in a synchronous collaboration session can request a floor for the use of a shared application in the session using the XGSP-Floor control tool described in section 4.4.3, or a moderator in the session can directly assign a floor to participants. This case, which is a mapping from offline request-response social protocol into online human-computer interaction, is for the shared whiteboard in our collaboration (moderator-mediated request-response interaction scheme). The text chat application does not need the floor request for free conversations among participants (no floor control scheme). The collaborative chess game application [100, 109, 110] uses different floor control scheme which alternates a floor in turn between the two players using the roles white-player and black-player in our collaboration role term (two-player turn-taking scheme or token-passing scheme).

2. When a participant requests a floor on an application being shared among participants, after the floor request is validated by the access and floor control decision service of a moderator node,

   If the floor is available, a moderator assigns the floor to the floor requester. Otherwise, the floor request is queued into a floor waiting queue or can be denied.

3. When a current floor holder releases a floor control or after a prefixed amount of time, the floor is assigned to a requester waiting in a floor waiting queue in FIFO order or the floor can also be directly released from a moderator.

The floor control policy is written with the XGSP-RBAC policy described in Chapter 3 and is implemented by the XGSP-Floor mechanism which is described in next section 4.4.2.

```
<XGSP-RBACPolicy>
 <ResourceAccesspolicy>
  <ResourceAccess>
  <RoleName>mobile-user</RoleName>
  <ApplicationRegistries>
   <ApplicationRegistry>
    <ApplicationID>wb</ApplicationID>
    <MainClass>cgl.myprofessor.whiteboard.Whiteboard</MainClass>
    <Actions>
      <Action>
            <ActionName>slave</ActionName>
            <Capabilities>read</Capabilities>
            <AccessType>released</AccessType>
      </Action>
      <Action>
            <ActionName>line</ActionName>
            <Capabilities>linedrawing</Capabilities>
            <AccessType>shared</AccessType>
       </Action>
       <Action>
            <ActionName>oval</ActionName>
            <Capabilities>ovaldrawing</Capabilities>
            <AccessType>exclusive</AccessType>
       </Action>
       <Action>
            <ActionName>pen</ActionName>
            <Capabilities>freedrawing</Capabilities>
            <AccessType>exclusive</AccessType>
       </Action>
    </Actions>
    </ApplicationRegistry>
  </ApplicationRegistries>
  </ResourceAccess>
 </ResourceAccesspolicy>
</XGSP-RBACPolicy>
```

Figure 4.1: An Example of XGSP-RBAC Policy with the Role Name mobile-user and Application Name whiteboard (wb).

4.4.2 XGSP-Floor Mechanism

A floor control mechanism (XGSP-Floor mechanism) is a means to implement the floor control policy described in the previous section. An XGSP-Floor mechanism regulates floors among all the participants in collaboration. In this section we present decision procedures implemented in a moderator node (which is a decision node to control accesses for applications in our collaboration system) to determine grant or deny of participants' floor requests to access applications in moderator-mediated interaction mechanism. The decision procedures follow the following five different types of stages. The broad view of the moderator-mediated interaction mechanism is depicted in Figure 4.2. Also, we show a request-response interaction scheme between a moderator and a floor requester with human-computer interaction in Figure 4.3.

4.4.2.1 Decision Procedures of XGSP-Floor Mechanism

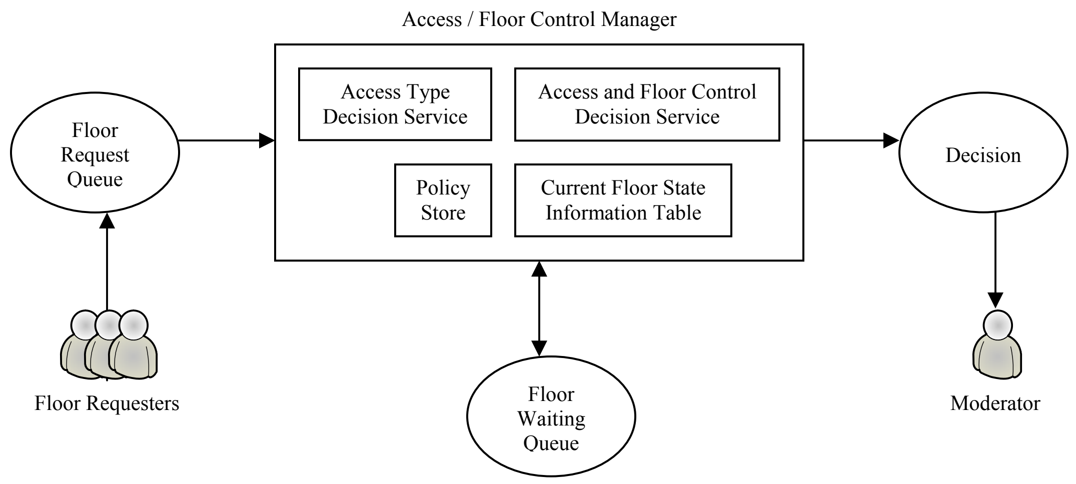

Access / Floor Control Manager
Access Type Decision Service
Access and Floor Control Decision Service
Policy Store
Current Floor State Information Table
Floor Request Queue
Decision
Floor Requesters
Floor Waiting Queue
Moderator

Figure 4.2: Decision Procedure of XGSP-Floor Mechanism

First, a moderator node has a single queue for storing floor requests from participants. The queue is implemented in FIFO (First-In, First-Out) order for mitigating race conditions of floor requests to applications and thus enforces mutual exclusion among applications. The first request in the queue is validated by policy manager and is sent to the access type decision service located in the access and floor control manager of the moderator node. Then, the first request is removed from the queue. During the activity, new floor requests are stored in the floor request queue waiting for next service.

Second, the access type decision service returns a classified access type value among Invalid, Implicit, Exclusive, Shared, or Released into the access and floor control decision service in access and floor control manager.

Third, decision activities are behaved with the same type value returned from the access type decision service. The decision activities are also classified (or branched) into the same access type activities as the returned value mentioned in the second stage. Each decision activity returns one of decision values (grant, deny, or queued) to the moderator. Then, the moderator can make a decision according to the decision values. In this stage, we present a set of predicate rules used as determinants in decision procedures of a moderator node in terms of the following two types of predicate statements:

1 Determination of types classified to access applications.

a) If the role name, application ID, and request action of a floor requester is validated by a policy manager then an access type value (among Implicit, Exclusive, Shared, or Released) by access type decision service is returned into the access and floor control decision service. In the elements, the role name means the name of role assigned to participants in our collaboration system, the application ID means an application identifier existing in application registries of our collaboration system, and the request action means the name of a fine-grained action in which participants can manipulate applications.

   i. If the return type is "Implicit", then the request is granted.
   ii. If the return type is "Exclusive", then the request is granted or queued.
   iii. If the return type is "Shared", then the request is granted or denied.
   iv. If the return type is "Released", then the request is granted.

b) If one of the elements does not exist in policy, then a type "Invalid" is returned into a moderator and the request is denied.

2 Determination of whether an action in a request exists in current floor state information table, in other words, a request action conflicts with the action of current floor holder.

i. If the return type from access type decision service is "Exclusive" and the request action exists in the floor state information table of a moderator node, then the request is queued. Otherwise, the request is granted.

ii. If the return type is “Released” and a floor waiting queue is not empty, then the request is granted and the first request in the waiting queue is granted and removed from the queue.

iii. If the return type is “Released” and a floor waiting queue is empty, then the request is granted.

Next stage is to update current floor state information table. To maintain consistent shared state at application level among collaborating participants, we need to maintain the current floor state information. This floor state information is updated to reflect an action in a request whenever the request is granted.

Finally, all the requests stored in a floor waiting queue for the use of shared applications are serviced in prefixed amount of time to avoid starvation. The floor requests to shared applications are stored in a single queue which is implemented in FIFO order. The first request in the queue is serviced when the floor of current floor holder is released or after a prefixed appropriate amount of time. Then, the request is removed from the queue and the current floor state information table is updated with the removed request. Note that when the floor of current floor holder is released after an appropriate amount of time, the mechanism uses the acknowledgement (reply message) from the revoked node. The acknowledgement prevents the floor of current floor holder from assigned to another participant before the floor of the floor holder is revoked.

4.4.2.2 Request-Response Interaction Scheme between a Moderator and a Floor Requester with Human-Computer Interaction

The moderator in our collaboration system is responsible for passing floor control to and from participants in XGSP-Floor mechanism. The moderator grants a floor either by clicking on a button on pop-up window representing a participant's request or by selecting an entry from the action list allowed for the participant displayed on a frame window invoked from a moderator node manager. The communication channel in a moderator node (on which a moderator resides) shown in Figure 2.8 of Chapter 2 disseminates the floor to all the participants in a session through a broker including a decision (grant) for the floor requester (a participant who wants to have the right for manipulating the application being shared). The granted floor event message autonomously activates the fine-grained request action of the application which the requester wants to manipulate. Other nodes, on which other participants reside, are updated to reflect the current floor holder in the session. This mechanism shows a mapping instance of a universal social protocol (request-response) from an offline session to an online session with a set of user interfaces between a participant in a session and a collaboration environment as shown in Figure 4.3. In our collaborative applications, whiteboard application uses this mechanism. Note that audio/video applications can use this request-response interaction scheme with the application specific locking mechanism as we integrate the scheme into the applications as a next phase in future work.

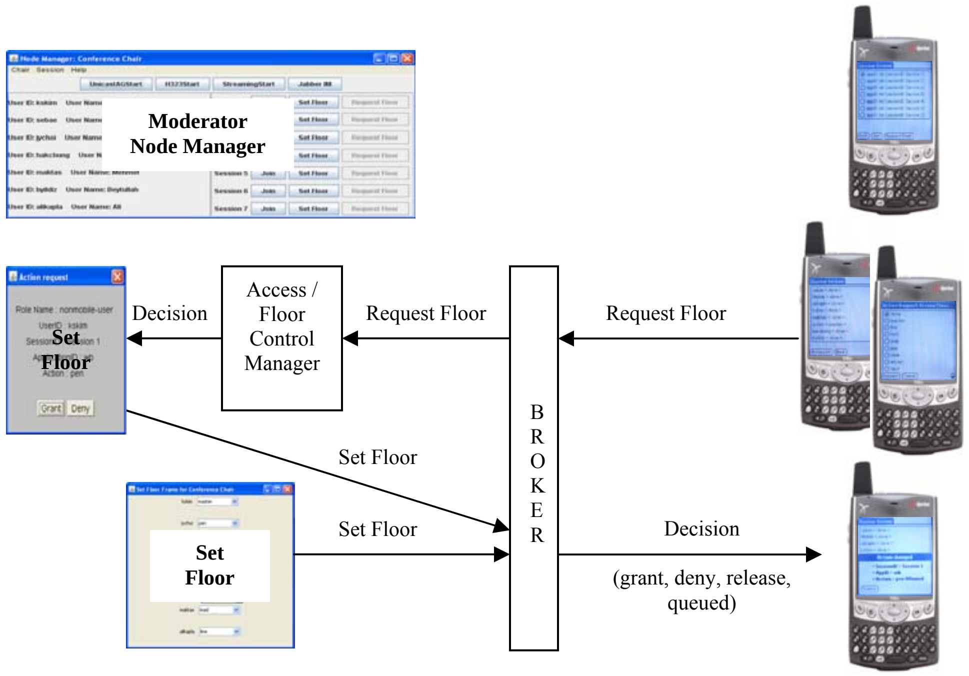


Figure 4.3: Request-Response Interaction Scheme between a Moderator and a Floor Requester with Human-Computer Interaction.

Chess game application uses different floor control mechanism able to be behaved without the mediation of the moderator like turn-taking mechanism. Thus, if one player in the chess game releases a floor or the prefixed playing time of a player is expired, then the floor is autonomously given to another player. In the two player game, the floor holder (one player in the game) directly passes the floor to another player through a broker. Other participants are regarded as an observer role which can not have a floor to play but share playing-view in the chess game. Figure 4.4 depicts the instance of the two-player turn-taking mechanism for the collaborative chess game application used in our collaboration.

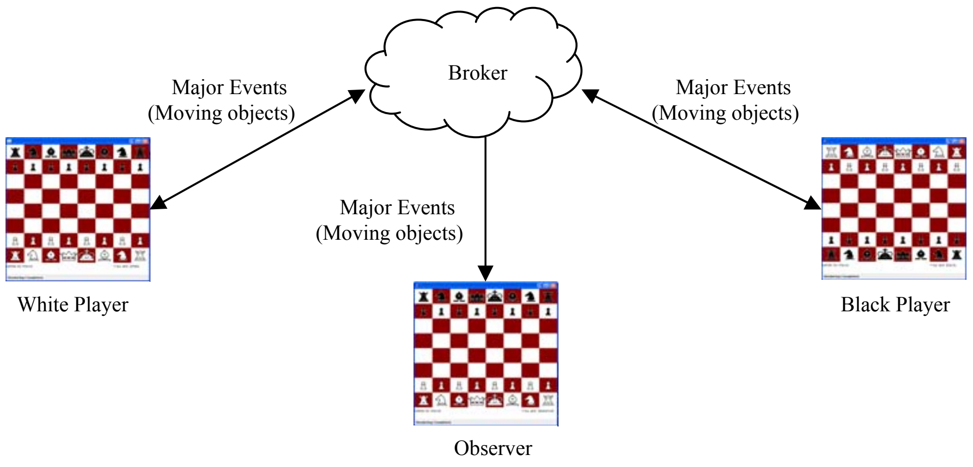
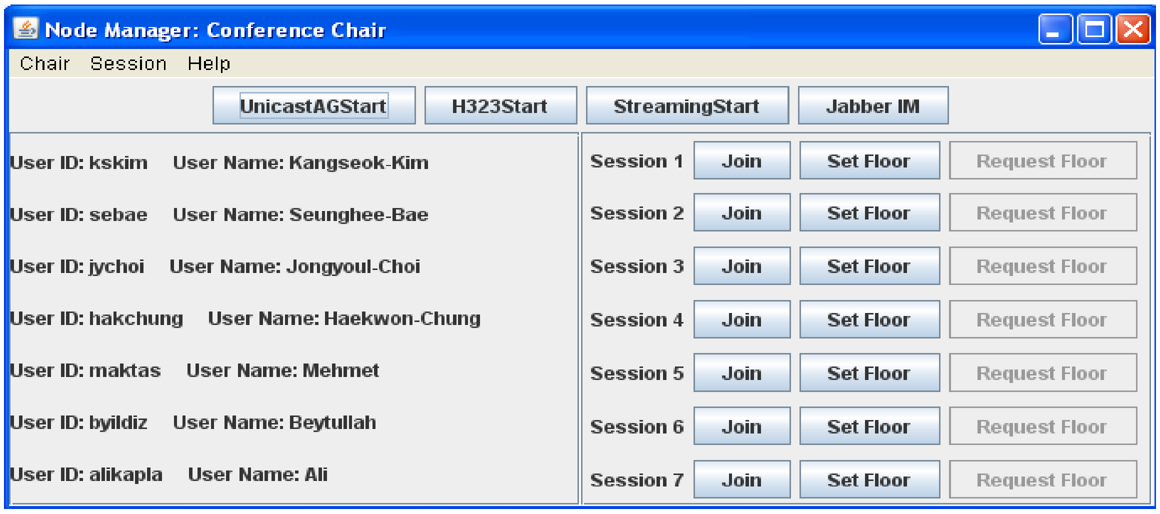


Figure 4.4: Two-player Turn-taking Mechanism for Chess Game Application

4.4.3 Functionality of XGSP-Floor Control Tool

This section describes the functionality of a floor control tool (XGSP-Floor tool interfaces) that provides a user interface (human-computer interaction) for control of floors to a moderator and participants in a session with desktop and cell phone devices. A moderator can give decisions (grant, deny, release (or revoke), and queued) for a floor to participants and the participants can request a floor to manipulate the application being shared in a session via the XGSP-Floor tool.

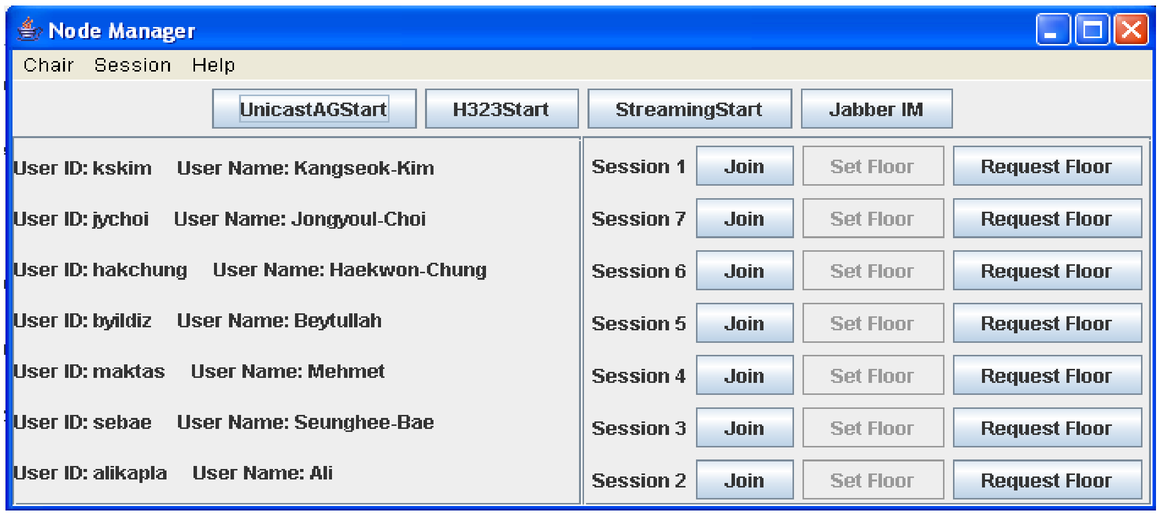
Figure 4.5 shows a node manager of a moderator on desktop. Figure 4.6 shows a node manager of participants (normal users) on desktop. The two node managers are almost similar except that the moderator node manager has a button able to control a floor and the normal user node manager has a button able to request a floor. Therefore, participants (not moderator) are not able to control the floor to give the right for manipulating the application being shared to other participants. The left display panel in the node managers shows a list of participants joined in the conference. The right display panel shows a list of sessions available in the conference. Each entry in a list of sessions has a session ID and three buttons (Join, Set Floor, and Request Floor). Participants in a conference can join a session by clicking on the "Join" button. A moderator can control floors in the window frame invoked by clicking on the "Set Floor" button in Figure 4.5. The pop-up window frame is shown in the left figure of Figure 4.7. Participants can request floors in the window frame invoked by clicking on the "Request Floor" button in Figure 4.6. The pop-up window frame for the request floor is shown in the right figure of Figure 4.7. A moderator can control floors of all the participants joined in a session via the window frame shown in the left figure of Figure 4.7 while participants can request only their own floors via the window frame shown in the right figure of Figure 4.7 but see current floor states of other participants.



User ID: kskim User Name: Kangseok-Kim
User ID: sebae User Name: Seunghee-Bae
User ID: jychoi User Name: Jongyoul-Choi
User ID: hakchung User Name: Haekwon-Chung
User ID: maktas User Name: Mehmet
User ID: byildiz User Name: Beytullah
User ID: alikapla User Name: Ali

Session 1
Session 2
Session 3
Session 4
Session 5
Session 6
Session 7

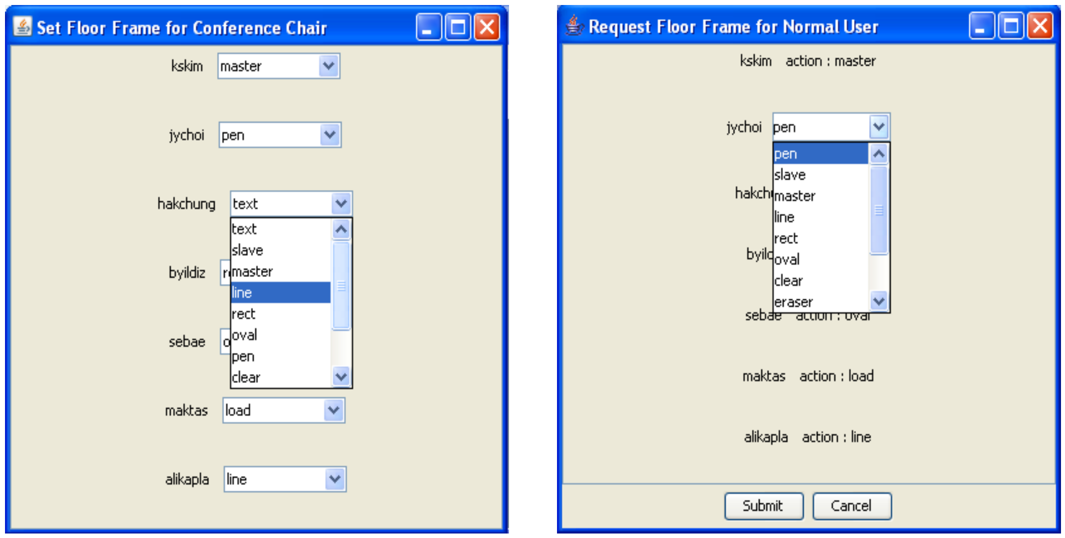
Figure 4.5: Node Manager for a Moderator on Desktop



User ID: kskim User Name: Kangseok-Kim
User ID: jychoi User Name: Jongyoul-Choi
User ID: hakchung User Name: Haekwon-Chung
User ID: byildiz User Name: Beytullah
User ID: maktas User Name: Mehmet
User ID: sebae User Name: Seunghee-Bae
User ID: alikapla User Name: Ali

Session 1	Join	Set Floor	Request Floor
Session 7	Join	Set Floor	Request Floor
Session 6	Join	Set Floor	Request Floor
Session 5	Join	Set Floor	Request Floor
Session 4	Join	Set Floor	Request Floor
Session 3	Join	Set Floor	Request Floor
Session 2	Join	Set Floor	Request Floor

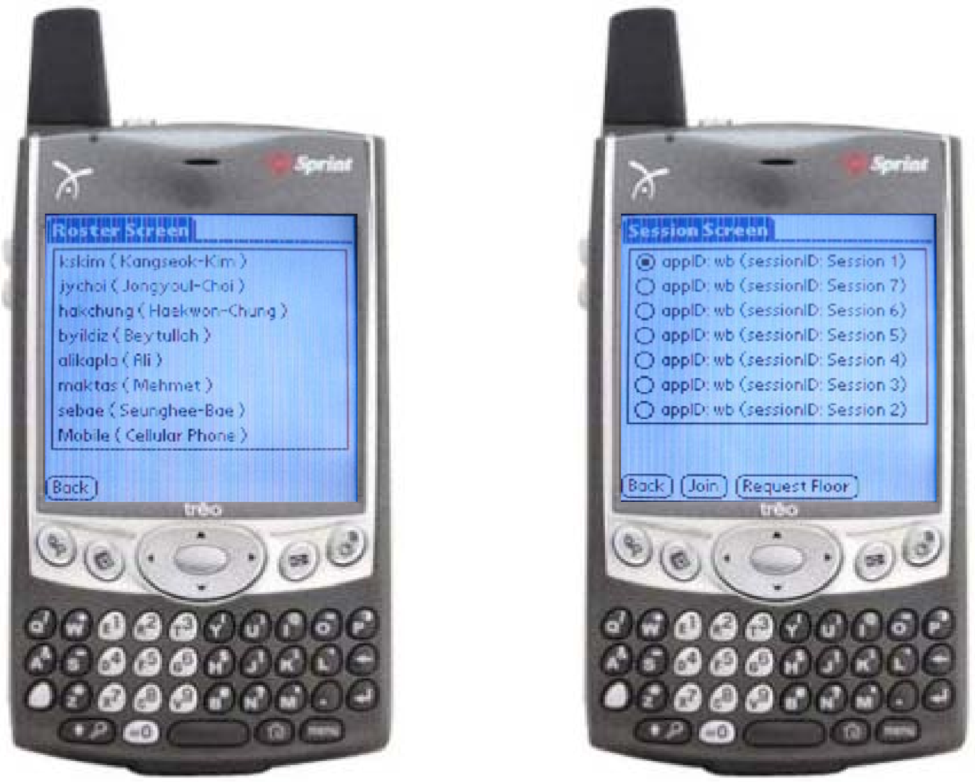
Figure 4.6: Node Manager for Normal Users on Desktop



kskim master
jychoi pen
hakchung text
byildiz
sebae
maktas load
alikapla line

kskim action : master
jychoi pen
hakchung
byildiz
sebae action : oval
maktas action : load
alikapla action : line

Figure 4.7: Set Floor Frame for a Moderator vs. Request Floor Frame for a Normal User



kskim (Kangseok-Kim)
jychoi (Jongyoul-Choi)
hakchung (Haekwon-Chung)
byildiz (Beytullah)
alikapla (Ali)
maktas (Mehmet)
sebae (Seunghee-Bae)
Mobile (Cellular Phone)

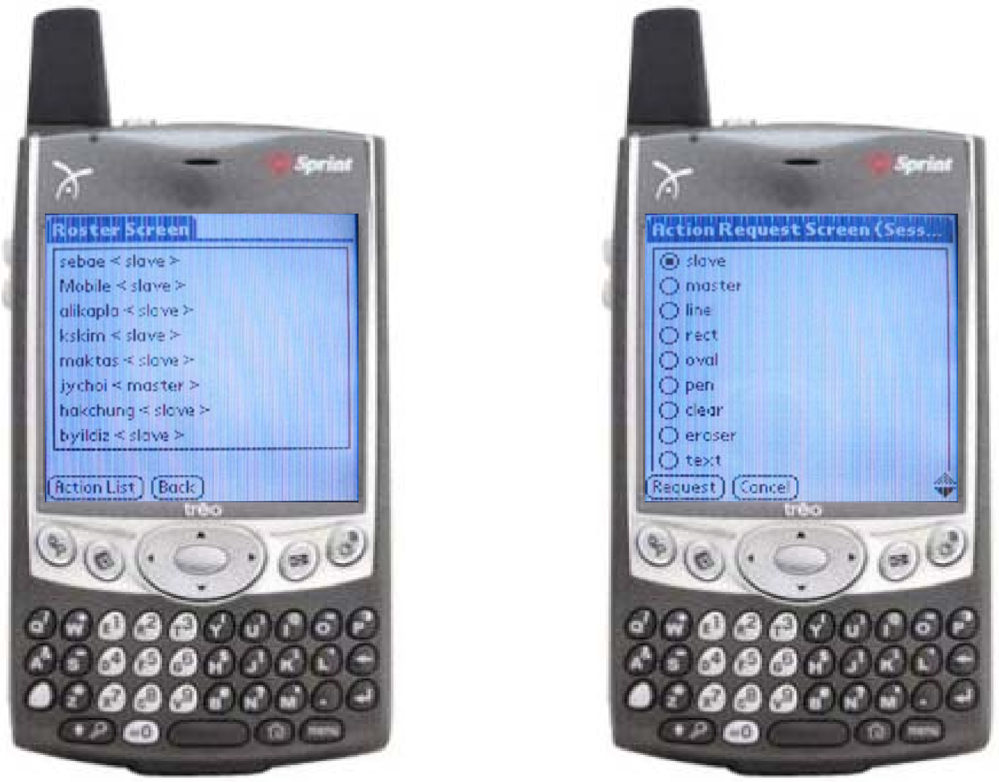
● appID: wb (sessionID: Session 1)
○ appID: wb (sessionID: Session 7)
○ appID: wb (sessionID: Session 6)
○ appID: wb (sessionID: Session 5)
○ appID: wb (sessionID: Session 4)
○ appID: wb (sessionID: Session 3)
○ appID: wb (sessionID: Session 2)

Figure 4.8: Node Manager for Normal Users on Cell Phone

Figure 4.8 shows a node manager of normal users (nomadic users) on cell phone. The functionality of the node manager on cell phone is similar to that of the node manager on desktop except that the node manager on cell phone uses two different screens for the presence membership of participants and a list of sessions existing in a conference. The left figure in Figure 4.8 shows a list of participants joined in the conference. The right figure shows a list of sessions available in the conference. Note that the cell phone uses the term screen instead of the term window or frame used in desktop. The application model for pervasive computing [5] requires creating a task-based model and a navigation model for program structure at design time. This means the task-based structure needs to generate device specific "presentation units" – screen and to specify the flow of the presentation units. Therefore, an application has to be depicted into tasks, subtasks of the tasks, and subtasks of the subtasks, and so on. On the modest-size window like desktop, the tasks (displays of participants' presence panel, session panel, floor set window frame, floor request window frame, and so on) in node manager can be presented on the same screen, whereas on the small-size screen like cell phone, the displays of the tasks have to be presented on separate screens including easy-to-use interfaces and a set of well-defined navigation to subtasks from the tasks.

Figure 4.9 shows the request screen for a floor. The screen is displayed from selecting the "Request Floor" button shown in the right figure of Figure 4.8. Figure 4.10 shows pop-up window frames (floor request, floor grant, floor deny, floor conflict, and floor queued notifications respectively) occurred as the request-response interaction mechanism between a moderator node and a requester node on desktop is used.

Figure 4.11 shows the screens (floor grant, floor deny, and floor queued notifications respectively) occurred as the request-response interaction mechanism between a moderator node and a requester node on cell phone is used. Then the screen in cell phone is often called an alert screen that shows a message to the user for a certain period of time.



Roster Screen

sebae < slave >
Mobile < slave >
alikapla < slave >
kskim < slave >
maktas < slave >
jychoi < master >
hakchung < slave >
byildiz < slave >

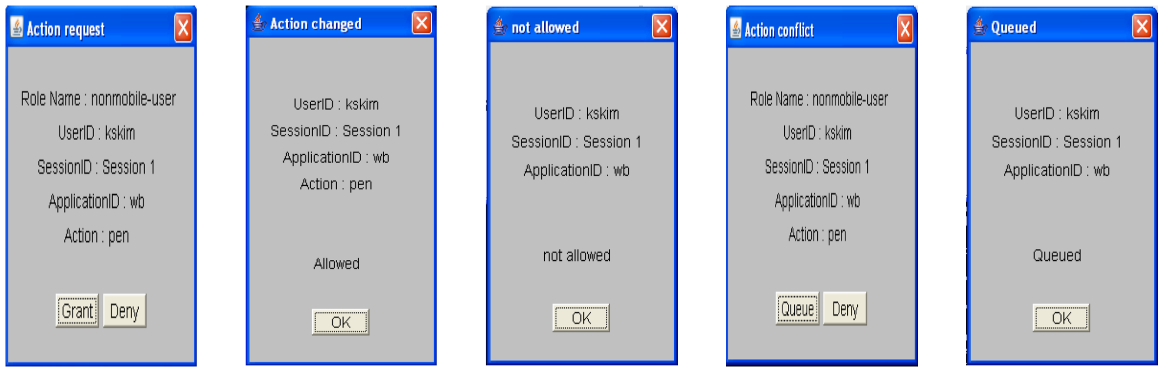
Action List | Back

Action Request Screen (Sess...

- slave
- master
- line
- rect
- oval
- pen
- clear
- eraser
- text

Request | Cancel

Figure 4.9: Request Floor Screen on Cell Phone



Action request

Role Name : nonmobile-user
UserID : kskim
SessionID : Session 1
ApplicationID : wb
Action : pen

Grant | Deny

Action changed

UserID : kskim
SessionID : Session 1
ApplicationID : wb
Action : pen

Allowed

OK

not allowed

UserID : kskim
SessionID : Session 1
ApplicationID : wb

not allowed

OK

Action conflict

Role Name : nonmobile-user
UserID : kskim
SessionID : Session 1
ApplicationID : wb
Action : pen

Queue | Deny

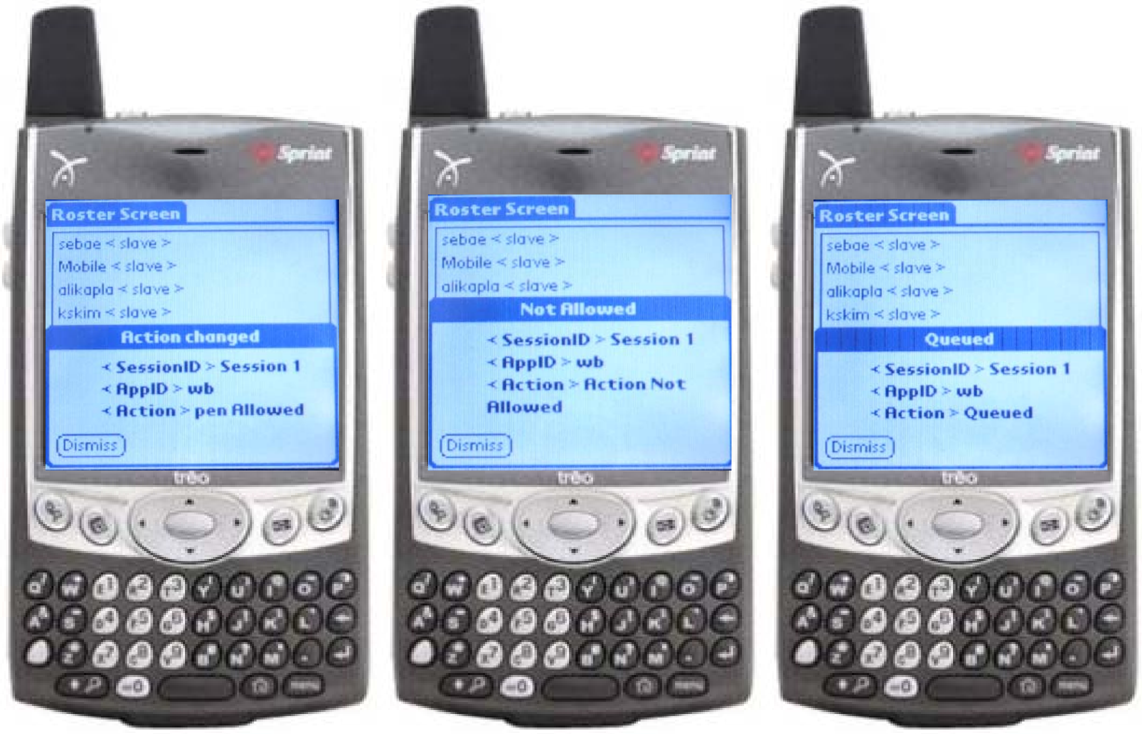
Queued

UserID : kskim
SessionID : Session 1
ApplicationID : wb

Queued

OK

Figure 4.10: Pop-up Window Frames (for Floor Request, Floor Grant, Floor Deny, Floor Conflict, and Floor Queued Notifications respectively) on Desktop



sebae < slave >
Mobile < slave >
alikapla < slave >
kskim < slave >
Action changed
< SessionID > Session 1
< AppID > wb
< Action > pen Allowed

sebae < slave >
Mobile < slave >
alikapla < slave >
Not Allowed
< SessionID > Session 1
< AppID > wb
< Action > Action Not Allowed

sebae < slave >
Mobile < slave >
alikapla < slave >
kskim < slave >
Queued
< SessionID > Session 1
< AppID > wb
< Action > Queued

Figure 4.11: Screens (for Floor Grant, Floor Deny, and Floor Queued Notifications respectively) on Cell Phone

In this section we showed the current implementation of an XGSP-Floor tool. It shows a simple interface between participants and collaboration environment for floor request, response, release (or revoke), and queued interaction with the synchronous collaborative application – shared whiteboard used in our collaboration. We need to further implement the XGSP-Floor tool with more detailed functionalities for synchronous collaborative media applications such as audio and video applications in future work.

4.4.4 A Major Event Conflict Detection Function of XGSP-Floor Mechanism

This section describes a major event conflict detection function that determines whether an action in a floor request conflicts with the action of current floor holder. When a floor is requested at the application being shared among participants, it consults with the current floor state information table in the access and floor control manager shown in Figure 4.2 to avoid the collision with current floor holder. If a request action exists in the current floor state information table, then the request is queued. Otherwise, the request is granted. The floor state information is updated to reflect an action in a floor request whenever the request is granted.

Participants maintained in the floor state information table have to assume at least one action but can not assume both actions at the same time even though the participants can assume different actions at the different time in our current floor mechanism. Participants in passive state may assume the action "slave" in our collaboration. The action "slave" means participants are in state joined in a session and in view-sharing state of the application being shared with other participants for WYSIWIS (What You See Is What I See) [98] which is an inclusion of collaboration. Also, the action can be used as a major event for releasing a floor which a participant is currently holding. The strict conflict avoidance [28] like our floor control mechanism allows all participants to have the same views and data at all times. Pessimistic (or non-optimistic) floor control follows the strict conflict avoidance strategy whereas optimistic floor control strategy allows conflicts and resolves them [18].

4.4.5 Locking of XGSP-Floor Mechanism

Locking [41] is a method of gaining privileged access to shared resource for some amounts of duration. In this section we show non-optimistic locking mechanism [41] used in our synchronous collaboration from two different viewpoints, moderator's point of view (system's point of view) and participant's point of view (application's point of view), where the non-optimistic locking mechanism means a request node (or a requesting participant) has to wait until the floor request gets granted from the moderator. This non-optimistic locking mechanism ensures that all the participants always have consistent views and data.

From moderator's point of view or system's point of view, the floor request queue in a moderator node is locked until the moderator node (or moderator) makes a decision on a floor request and dispatches the decision to the request node. During the lock, the floor state information table is updated. After the lock of the floor request queue is released, the next request in the queue is serviced if the queue is not empty. This locking mechanism guarantees the mitigation of race conditions of floor requests to shared application and thus enforces mutual exclusion in the shared application. Therefore, participants can access a shared application with a granted fine-grained action one participant at a time.

From participant's point of view or application's point of view, we used an application specific locking mechanism. According to shared applications, different fine-grained locks are used to allow more concurrent activity among participants and to follow the principle of least privilege [75], where the fine-grained lock means the locking of the major event described in Chapter 3. Also, a coarse-grained lock can be used to allow a participant to make more activities at a time. In our whiteboard application example, as a fine-grained request action (major event) is granted from a moderator and the granted message arrives at the requester node, the lock for the use of the requesting action is released as depicted in Figure 4.12. The coarse-grained action "master" in the application can be used to allow a participant to assume many different fine-grained actions at a time. This locking mechanism guarantees that the consistent state at application level is maintained among participants. In our chess game application example, if an action (a major event moving a object) of a player is a legal move validated by the chess game rule as the user input event (releasing the object – mouse release) of the player is occurred, then the user input event (mouse click) of the player is locked and the lock (mouse click) of another player is released in turn by passing a logical token as depicted in Figure 4.13.

In our first implementation, we used a mechanism: it does not use a reply message from a requester node for a lock release of the floor request queue in a moderator node (no acknowledgement) and the action, which the requester is currently holding, is not blocked (non-blocking). This means the requester can manipulate shared application with a holding action until a grant message for new floor arrives at the requester node. Then, we identified the mechanism results in inconsistency. The inconsistency comes from: before the floor of current floor holder for shared application is revoked, the floor can be reassigned to another participant.

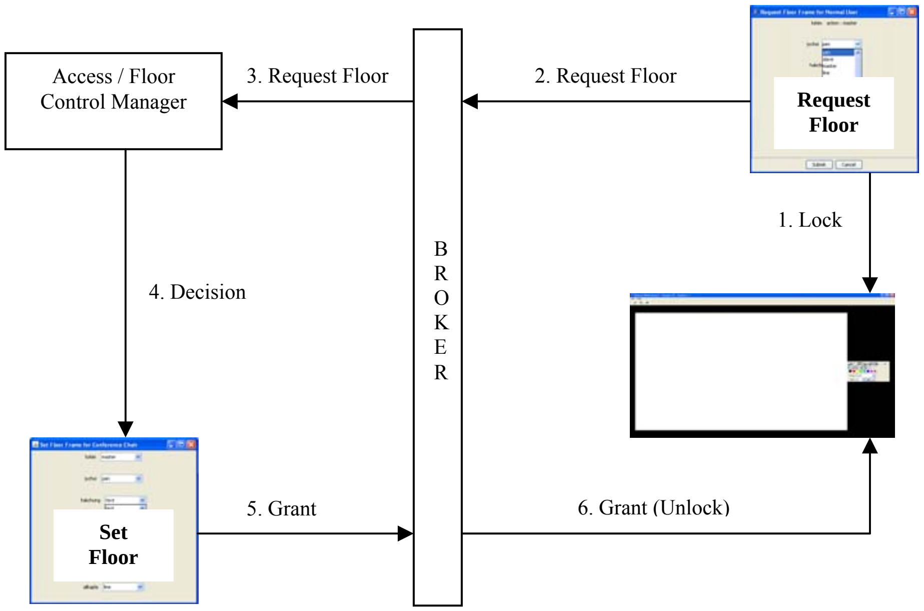


Figure 4.12: Locking Mechanism of Shared Whiteboard

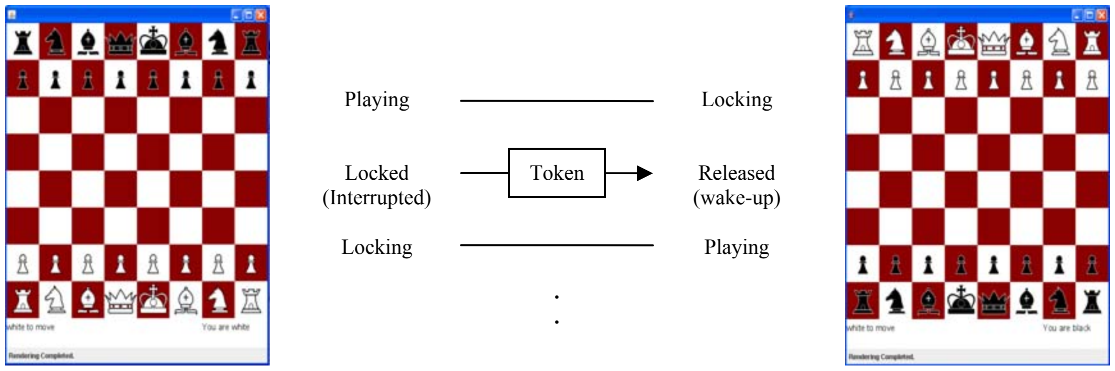


Figure 4.13: Locking Mechanism by Logical Token-passing in Chess Game

In our second implementation, we used an acknowledgement (reply message) from a requester node after a request-response interaction for a floor between a request node and a moderator node. The implementation with the acknowledgement from the requester node ensures that the floor of current floor holder for shared application is revoked before the floor is reassigned to another participant and thus previous floor holder no longer holds the floor of newly reassigned current floor holder. Also, a floor requester is assigned a floor after current floor state information of the requester is updated. The acknowledgement enforces mutual exclusion in shared application and thus ensures consistency. But it resulted in response overhead as shown in Figure 4.14, especially with cell phone device involved in collaboration since the wireless network transit time is in the range of seconds as shown in Figure 2.24 and Figure 2.25. Another encountered problem was deadlock occurred from the loss of the acknowledgement (lock release of a moderator node), especially with cell phone disconnected. We could resolve the problem by a moderator directed-floor-assignment or communication among participants through text chat using no floor control mechanism. Then the lock for mitigating floor requests in a moderator node is released and the next request can be served.

Our current floor control mechanism implements no acknowledgement and blocking mechanism where blocking means the action, which a floor requester is currently holding when she makes a request for a floor, is blocked if she holds a floor. But, the mechanism uses the acknowledgement (reply message) from revoked node to prevent the floor of current floor holder from assigned to another participant before the floor of the floor holder is revoked.

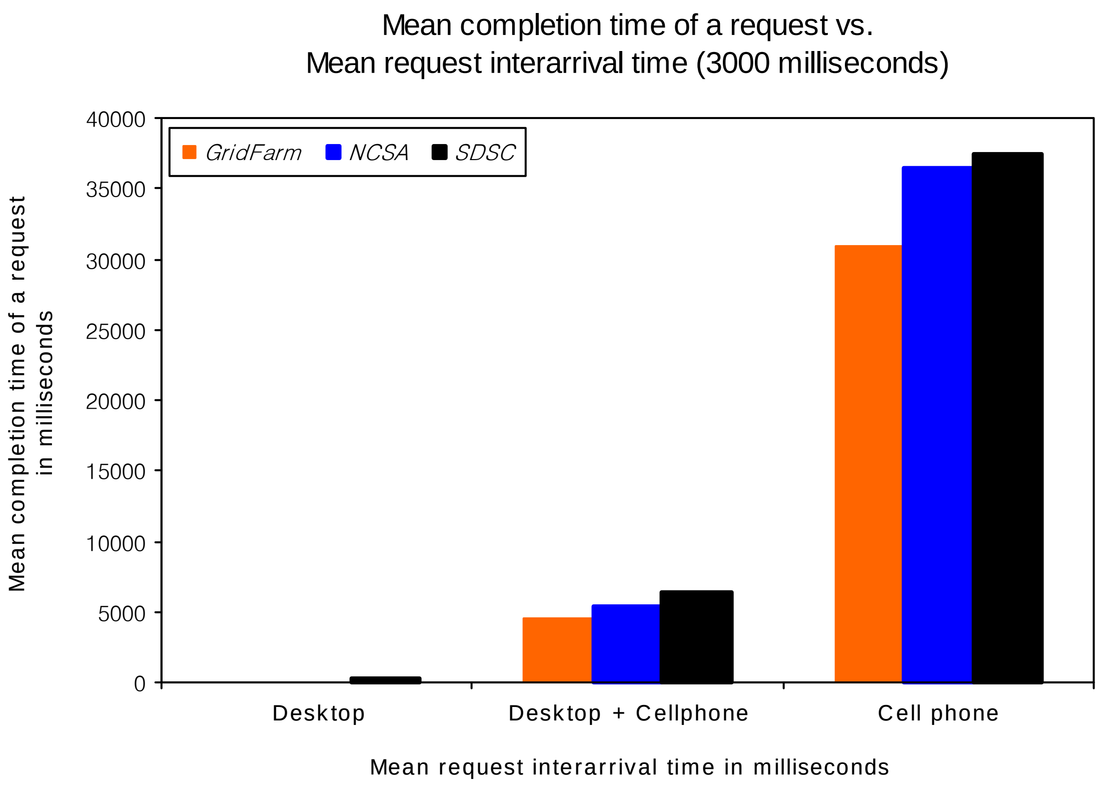


Figure 4.14: Mean completion time of a request vs. Mean request interarrival time (3000 milliseconds)

4.5 Summary

In this chapter, we presented a policy and a mechanism implementing it – XGSP-Floor (XGSP Floor control) for coordinating concurrent activities to synchronous collaborative applications and maintaining shared state consistency at application level. The XGSP-Floor mechanism uses moderator-mediated interaction with a major event conflict detection function and non-optimistic locking mechanism.

The XGSP-Floor integrated into our collaboration framework provides significant flexibility, ranging from free-for-all (no floor) to application specific floor control mechanism for avoiding uncoordinated activities to shared collaboration applications. A moderator (moderator node) is responsible for maintaining the consistent state of applications in our collaboration system. Even though our underlying floor control scheme is a moderator-mediated interaction mechanism, a floor can automatically be assigned to a floor requester without the mediation of the moderator according to the policy.

We showed with example applications – shared whiteboard and collaborative chess game that social protocols used in a face-to-face offline session can be mapped to mechanisms able to be used in an online session with user interfaces between participants and CSCW environment. The XGSP-Floor control tool provides human-computer interaction for control of floor for roaming participants with cell phone as well as desktop participants in collaboration.

We left support for floor control of synchronous collaborative media applications such as audio and video applications in future work. In the future work, we also need to further implement XGSP-Floor tool with more detailed functionalities for the synchronous collaborative media applications.

Chapter 5

Formal Verification of Control Mechanisms by Colored Petri Net

This chapter shows modeling of XGSP based control mechanisms, in this thesis referred to as XGSP-RBAC (XGSP Role Based Access Control) and XGSP-Floor (XGSP Floor control) mechanisms which are integrated into our collaboration framework for building collaborative applications in heterogeneous (wire and wireless) computing environment. We also show formal verification to prove the correctness of the modeled XGSP based control mechanisms in terms of mutual exclusion, dead lock, and starvation. The key part for the modeling and formal verification of the modeled control mechanisms is to show consistent shared state at application level to collaborating users by mitigating race conditions for shared resources and thus to attain mutual exclusion among resources. For the abstract modeling representation of the control mechanisms, we used Colored Petri Nets (CP-nets or CPNs) with time [64]. One of the main reasons for using the CP-nets is the fact that it allows each activity in modeling to have a number of different types of occurrences by using different types of token colors, and thus reduces a large number of places (states), transitions (activities), and arcs (expressions), and makes modeling and verification of a system much easier than classical Petri net (place/transition net) [60]. The CP-nets provides a formal simulation tool [14] to model a system. In our modeling, the simulation involves programming the model of the control mechanisms. Data structures in the simulation represent the major components of the control mechanisms. The CP-nets also provides analysis functions using state spaces (also called occurrence graphs) to prove the correctness of a system based on mathematical methods. As the simulation executes, the simulation tool (simulator) updates the simulation state to reflect the activities of the modeled control mechanisms and a set of statistical data are gathered and used to prove the correctness of the modeled mechanisms by the analysis functions provided by the CP-nets. This chapter is organized as follows. Section 5.1 provides a general modeling description of our XGSP based control mechanisms and presents the procedural behaviors of the mechanisms in terms of decision procedures, predicate rules for subsequent procedures and current status for actions of which collaborating users in a session are holding to access resources. Translating terms used in section 5.1 into terms of CP-nets, Section 5.2 provides a modeling of the XGSP based control mechanisms built by the CP-nets and informal introduction for the modeled control mechanisms. Section 5.3 presents a formal definition of static structure and dynamic behavioral properties which CP-nets has in a modeling, and an adaptation of the formal definition to the XGSP based control mechanisms. In section 5.4, we verify the correctness of the modeled control mechanisms based on state space analysis provided by CP-nets. Finally, we conclude by summarizing main points drawn from proving the correctness of the modeled XGSP based control mechanisms.

5.1 Modeling of Control Mechanisms (XGSP-RBAC and XGSP-Floor)

In this section we present decision procedures behaved in a moderator node (which is a decision node to control accesses for resources in our collaboration system) to determine grant or deny of collaborating users' requests to access resources in our XGSP based control mechanisms. The decision procedures of the moderator node (or by a moderator in our collaboration system) are modeled in terms of the following five different types of stages. The broad view of the modeling for the XGSP based control mechanisms is shown in Figure 5.1.

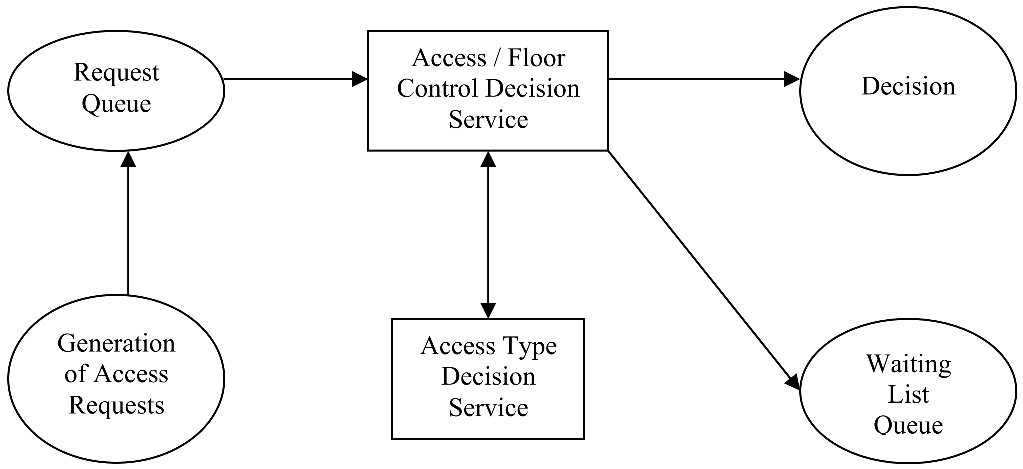
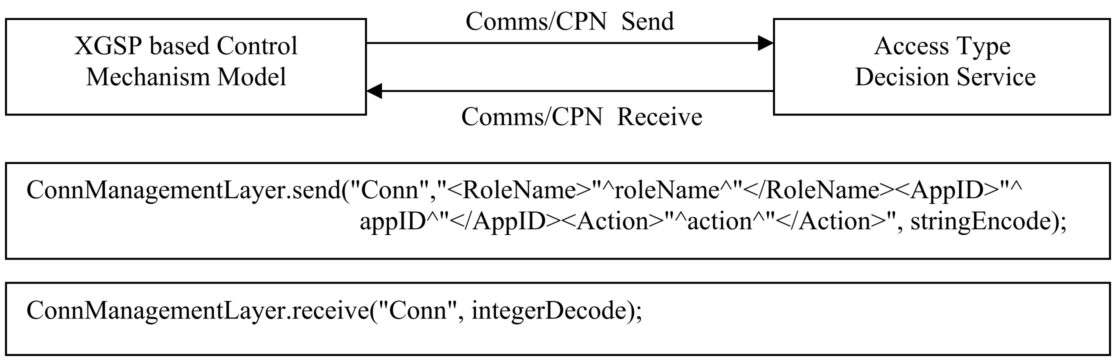

Request Queue
Access / Floor Control Decision Service
Decision
Generation of Access Requests
Access Type Decision Service
Waiting List Queue

Figure 5.1: Broad View of the Modeling for Control Mechanisms

First, the modeling randomly generates access requests to resources by the simulation on behalf of collaborating users in a collaboration session and has a single queue (request queue) for storing access requests from the simulated users. The queue is implemented in FIFO (First-In, First-Out) order for mitigating race conditions of access requests to resources and thus enforces mutual exclusion among resources. The first request in the queue is sent to the access type decision service located in external process module for parsing the requests written in XML and returning a type value into the modeling. Then, the first request is removed from the queue. During the activity, new access requests are generated and stored in the request queue waiting for next service.


XGSP based Control Mechanism Model
Comms/CPN Send
Comms/CPN Receive
Access Type Decision Service

```
ConnManagementLayer.send("Conn","<RoleName>"^roleName^"</RoleName><AppID>"^
                               appID^"</AppID><Action>"^action^"</Action>", stringEncode);
```

```
ConnManagementLayer.receive("Conn", integerDecode);
```

Figure 5.2: Comms/CPN for Communication between Control Mechanism Model and Access Type Decision Service which is located outside the modeling as external process.

Second stage is a communication activity for accessing the access type decision service, mentioned above, which is located in external process module outside the XGSP based control mechanism model. For the communication activity, CP-nets provides Comms/CPN [35, 36] which is a library for communication service between

CP-nets models and external processes. The Comms/CPN library allows CP-nets models to interact with the external environments via TCP/IP. Using the library, the control mechanism model connects to the access type decision service which is a module written in Java 1.5 and which is practically used in XGSP based control mechanisms (XGSP-RBAC and XGSP-Floor) integrated into our collaboration framework. The service parses XML requests sent from the control mechanism model and returns a type value among Invalid, Implicit, Exclusive, Shared, or Released into the model as practically does the access type decision service module in our collaboration framework. Figure 5.2 shows the communication between the control mechanism model and the access type decision service, Figure 5.3 shows a communication page used in the modeling and Figure 5.4 shows SML (Standard Meta Language) functions [58] for the communication used in our modeling. In Figure 5.2, the Comms/CPN send function is used to send access requests written in XML to the external type decision service and the Comms/CPN receive function is used to receive the response from the external service.

Third, decision activities in the modeling are behaved with a type value returned from the access type decision service. The decision activities are also classified (or branched) into the same access type activities with a returned value as mentioned in the second stage. Each decision activity simulates decision behaviors (grant, deny, released or queued) of a moderator in collaboration by randomly generating access decisions for requests in the modeling.

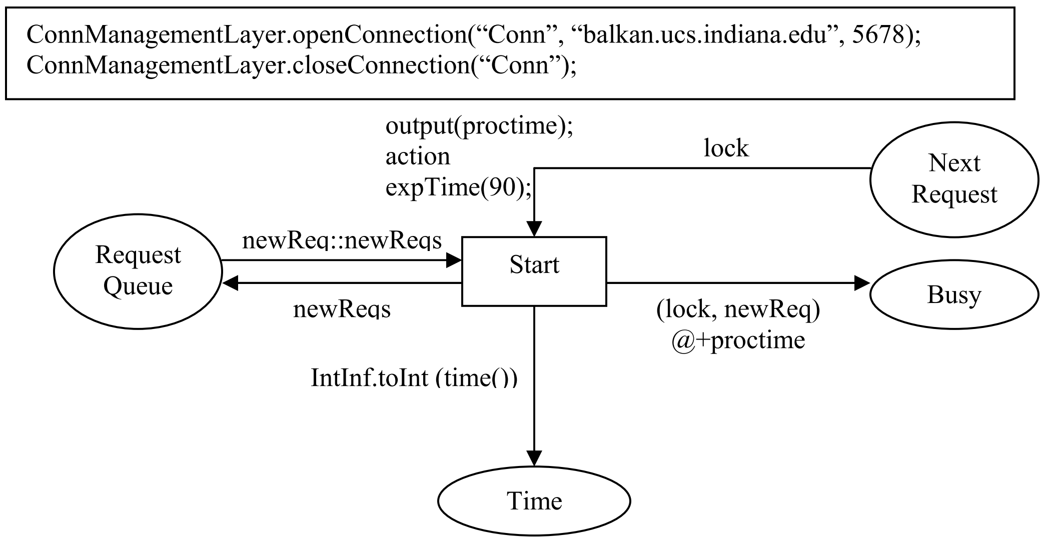

ConnManagementLayer.openConnection("Conn", "balkan.ucs.indiana.edu", 5678);
ConnManagementLayer.closeConnection("Conn");
output(proctime);
action
expTime(90);
lock
Next Request
Request Queue
newReq::newReqs
Start
newReqs
Busy
(lock, newReq) @+proctime
IntInf.toInt (time())
Time

Figure 5.3: Communication Page in Control Mechanism Model

Next stage is to update a state information table of a requesting user with simulated decision behaviors. Some requests are denied without need for updating current action state information of the requesting users, some requests are granted with need for updating the current action state of users, or others are stored in a queue (waiting list queue for access to shared resources) different from the request queue storing access requests to resources.

Finally, all the requests stored in a queue for the use of resources are serviced in prefixed amount of time to avoid starvation. The access requests to shared resources are stored in a single queue which is implemented in FIFO (First-In, First-Out) order which has a fair characteristic. The first request in the queue is serviced when a floor holding a shared resource is released or after an appropriate amount of time. Then, the request is removed from the queue and the current state information table is updated with the removed request action.

```
fun init () =
  if !connected = true
  then (ConnManagementLayer.closeConnection("Conn"); connected := false)
  else ()

fun pred (bindelem) =
  let
    fun predBindElem (ModeratorNode'Start (1, {newReq,newReqs,proctime})) = true
        | predBindElem _ = false
  in
      predBindElem bindelem
  end

fun obs (bindelem) =
  let
     fun obsBindElem(ModeratorNode'Start(1,{newReq,newReqs,proctime})) =
       (RoleName.mkstr(#roleName newReq), AppID.mkstr(#appID newReq),
        Action.mkstr(#actions newReq)) | obsBindElem _ = ("", "", "")
  in
       obsBindElem bindelem
  end

fun action (s1, s2, s3) =
  (if not(!connected)
   then (ConnManagementLayer.openConnection("Conn", "balkan.ucs.indiana.edu",
          5678); connected:=true)
   else ();
   send_to_decisionService(s1, s2, s3);  response:=get_from_externalProcess(); ())

fun stop () =
    if !connected = true
    then (ConnManagementLayer.closeConnection("Conn"); connected := false)
    else ()

fun send_to_decisionService(roleName, appID, action) =
   ConnManagementLayer.send("Conn",
    "<RoleName>"^roleName^"</RoleName><AppID>"^appID^"</AppID><Action>"
    ^
      action^"</Action>", stringEncode);

fun get_from_externalProcess() =
  ConnManagementLayer.receive("Conn", integerDecode);
```

Figure 5.4: SML functions for communication between Control Mechanism Model and External Process (Access Type Decision Service)

Also, we define a set of predicate rules used as determinants in decision procedures of a moderator node in terms of the following two types of predicate statements:

1 Determination of types classified to access resources.

A. If each element of an access request which is composed of 4-tuple (userID, roleName, applicationID, action) exists in the policy which is accessed by the access type decision service located outside modeling, then an access type value allowed for the resource based on the policy is returned into the model. In the 4-tuple, userID means an identifier of a user joined in a session, roleName means the name of roles assigned to users in our collaboration system, applicationID means an application identifier existing in application registries of our collaboration system, and action means the name of a means which users access resources. Note that 2-tuple (userID, action) for requesting the use of a resource is used in our practical mechanism.

I. If the return type is "Implicit", then the request is granted.

II. If the return type is "Exclusive", then the request is granted or queued.

III. If the return type is "Shared", then the request is granted or denied.

IV. If the return type is "Released", then the request is granted.

B. If one of elements in the 4-tuple does not exist, then a type "Invalid" is returned into the modeling and the request is denied.

2 Determination of whether an action in a request exists in state information table, in other words, a request action conflicts with the action of current floor holder.

A. If the return type from the access type decision service is “Exclusive” and action name in the fourth element of the 4-tuple exists in the state information table of a moderator node, then the request is queued. Otherwise, the request is granted.

B. If the return type is “Released” and a floor waiting queue is not empty, then the request is granted and the first request in the queue is granted and removed from the queue.

C. If the return type is “Released” and a floor waiting queue is empty, then the request is granted.

To maintain consistent shared state at application level among collaborating users, we need to maintain current state information table. This action state information in the table is updated to reflect an action in a request whenever the request is granted. The state is represented in a list having request records as elements in the modeling.

5.2 Informal Introduction of Control Mechanisms Modeled by Colored Petri Nets

In this section, we informally show how CP-nets with time is able to be used to model our XGSP based control mechanisms. The basic idea for modeling of the XGSP based control mechanisms using CP-nets with time is to describe a method for mitigating race conditions of access requests to resources, and thus ensuring shared state consistency at application level by attaining mutual exclusion among resources. The CP-nets model with time of the XGSP based control mechanisms is depicted in Figure 5.5. The XGSP based control mechanisms are modeled by means of states (often referred to as places in CP-nets), actions (often referred to as transitions), and expressions (often referred to as arc expressions which is inscribed on the arc) between the states and the actions. We present an informal introduction of the modeled mechanisms in terms of places, transitions, and arc expressions. Also we show the correctness of the modeled XGSP based control mechanisms with the informal definition in terms of mutual exclusion and starvation.

Each place in the model has a color set (interchangeably often referred to as a data type). The color set determines a kind of data type which places can have. A value of a color set is called a token color in CP-nets as an element of a data type in high-level programming language is called a value of the type. From Figure 5.5, it can be seen that the places Simulation-Start and Request-Nodes have a color set COUNT and the places Time and Waiting-Time have a color set INT. The place Request-Queue has a color set NewReqs, the place State-Information-Table has a color set OldReqs, and the place Waiting-List-Queue has a color set SharedReqs. Also, the places Busy, Invalid, Implicit, Shared, Exclusive, and Released have a color set LockxNewReq. The places Next-Request, L, and GiveFloor have a color set Lock. The place U has a color set LockxGrantDeny. The place TimeOver has a color set BOOL. Other places have a color set LockxGrantDenyxNewReq. For example, the place Z is used to send a decision on an access request to a request node. Then the place Z in our modeling can have values which are composed of a value from Lock color set, a value from GrantDeny color set, and a value from NewReq color set where the color set Lock is a data type used to unlock decision procedures, the color set GrantDeny is a data type which can have a value from {grant, deny, queued, released, give}, and the color set NewReq is a data type of new request which the place can have. The informal descriptive definitions of the color sets in the modeling of the XGSP based control mechanisms are as follows.

COUNT = {0@time, 1@time, 2@time...} where @time means some model time.
INT = {0, 1, 2, 3 ...}
smallINT = {0, 1, 2, 3, 4}
BOOL = {true, false}
UserID = {A, B… J}
RoleName = {nonmobile_user, mobile_user}
AppID = {wb}
Action = {master, slave, line, rect, oval, pen, eraser, move, load, clear}
NewReq = {(i, j, k, l, m) | i ∈ UserID and j ∈ RoleName and k ∈ AppID and l ∈ Action and m ∈ BOOL}
OldReq and **SharedReq** = same as NewReq
NewReqs = [{(i, j, k, l, m) | i ∈ UserID and j ∈ RoleName and k ∈ AppID and l ∈ Action and m ∈ BOOL}, {…..}, {…..}, ...]
OldReqs and **SharedReqs** = same as NewReqs
Lock = {lock@time where lock is a variable used to lock decision procedure and @time means some model time}
LockxNewReq = {(i, j) | i ∈ Lock and j ∈ NewReq}
GrantDeny = {grant, deny, queued, released, give}
GrantDeny2 = {grant, deny}
LockxGrantDenyxNewReq = {(i, j, k) | i ∈ Lock and j ∈ GrantDeny and k ∈ NewReq}
LockxGrantDeny = {(i, j) | i ∈ Lock and j ∈ GrantDeny}

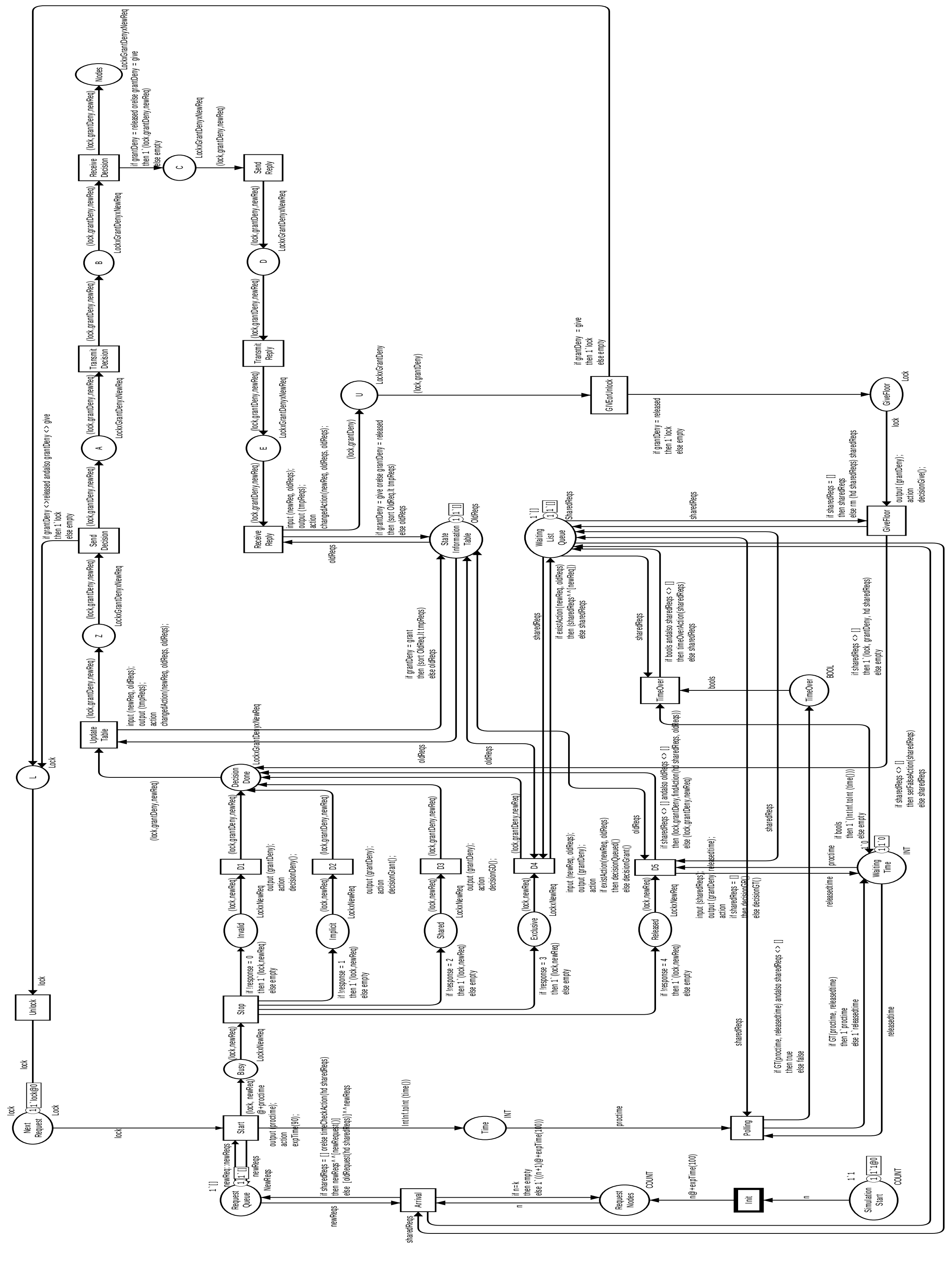


Figure 5.5: Control Mechanisms Modeled by CP-nets

A state (often referred to as marking in CP-nets) of a place represents current state of token colors of color sets which the place has. For example, the current and initial marking of the place Simulation-Start in Figure 5.5 is COUNT. This means the place can have token colors from arbitrary natural numbers beginning with 1. The current value of the place is used as initial value (or token color) to count the number of requests for accessing resources from simulated users. The initial markings of places Request-Queue, Next-Request, Waiting-Time, Waiting-List-Queue, and State-Information-Table from Figure 5.5 represent initial token value of the color sets which the places have. The place Next-Request has an initial token color lock with timed type and the place Waiting-Time has an initial token color 0 (zero) with integer type. The initial markings of the places Waiting-List-Queue and State-Information-Table have a token color with empty list which means the number of elements in the list is zero respectively. All other places initially have empty which means the places have no token colors.

Token removed from incoming places are transferred to outgoing places by evaluating arc expressions occurred by the transition connected to the places. For example, in Figure 5.5, the transition Init has one incoming arc and one outgoing arc. The arc expression of incoming arc into the transition is n where n is a variable of a color set COUNT in the place Simulation-Start. The value of the CP-nets variable n is 1 since the token value which the place Simulation-Start has is 1. And the arc expression of outgoing arc from the transition Init is a variable n and function expTime (100) where n is also a CP-nets variable and has the same value 1 as the value of the color set COUNT transferred into the transition, and expTime (100) is a function to exponentially calculate some delay time in the interarrival requests and is used to simulate the requests as if collaborating users practically behave to request accesses to resources. The interarrival requests' times are exponentially distributed with a mean of 100 time units between two successive requests issued by the simulation tool. So the delay time has no any meaning and just is used to randomly generate independent requests. The enabled transitions are usually occurred by evaluating outgoing arcs as a previous instance. Then, a binding element, which is composed of an enabled transition and a binding of outgoing arcs, has to be considered to evaluate the outgoing arcs connected from the transition. Also, as another instance in Figure 5.5, each of the transitions D1, D2, and D3 has one incoming arc and one outgoing arc. The arc expression of the incoming arc is (lock, newReq) where lock is a variable of color set Lock and newReq is a variable of color set NewReq. And the arc expression of outgoing arc is (lock, grantDeny, newReq) where lock is a variable of color set Lock, grantDeny is a variable of color set GrantDeny and newReq is a variable of color set NewReq. Assume that a global variable - response has now a value 2. Then the place Shared has a token value (lock, newReq) and the transition D3 is enabled. With a decision value returned from the function decisionGD(), the transition D3 binds the expression (lock, grantDeny, newReq) of the outgoing arc to (lock, grant or deny, newReq). Therefore, the place has a token value which is composed of a value of lock variable, a value of grantDeny variable, and a value of newReq variable. Thus, in such sequences of occurrences, the occurrence of a transition simulates decision procedures of the control mechanisms.

All other transitions in the model of the control mechanisms are enabled and occurred in such a similar way.

To show the correctness of the modeled mechanisms in terms of mutual exclusion, we consider the places Request-Queue and Next-Request, and the transition Start. A request token from the place Request-Queue and a lock token from the place Next-Request enable the transition Start. When the transition is occurred, the two tokens are added to the place Busy by evaluating the arc expression between the transition Start and the place Busy, and the tokens (a request token and a lock token) are removed from the places Request-Queue and Next-Request. The transition Start then will be not enabled until the place Next-Request has a new token value lock, where the token value lock is generated after the state information table is updated and a decision on a request is sent to a request node. Therefore, the following requests are not able to enter the decision procedures which are regarded as a critical section in the modeling until the place Next-Request has a new token. This means at most one request is processed in the decision procedural stage and thus indicates the modeled control mechanisms are ensuring mutual exclusion for the decision procedural stage to avoid race condition among requests issued by the simulation tool on behalf of users.

Next, we consider the transitions Start, Polling, and TimeOver to show that there is no starvation among requests issued in the modeling. When the transition Start is occurred, the transitions Polling and TimeOver check the time duration of the requests waiting for floors in the place Waiting-List-Queue. If the waiting time duration of a request is over the prefixed amount of time, then the request is serviced. Thus, the requests waiting for floors in the place Waiting-List-Queue will never be starved.

5.3 Formal Definitions and Notations of Control Mechanisms Modeled by Colored Petri Nets

In this section we present a formal definition of static structure and dynamic behavioral properties that CP-nets has, and the representation of the static properties and the example representation of the dynamic behavioral properties for the CP-nets model of our XGSP based control mechanisms.

5.3.1 Static Structure Properties of CP-nets and Representation of Control Mechanisms by the Properties

The static structure is basically composed of building blocks (places, transitions, and arcs), the connection points through which data flow into and out of the building blocks, and the connection paths along which data flow between the building blocks.

The static properties of the CP-nets [64] are represented as a 9-tuple ($\sum$, P, T, A, N, C, G, E, I) where

1. $\sum$ is a finite set of types (also, called color sets).
2. P is a finite set of places.
3. T is a finite set of transitions.
4. A is a finite set of arcs such that $P \cap T = P \cap A = T \cap A = \emptyset$.

5. N is a node function: $A \rightarrow (P \square T) \square (T \square P)$.
6. C is a color function: $P \rightarrow \sum$.
7. G is a guard function such that $\square t \in T$ $[\mathrm{Type}(G(t)) = \mathrm{Bool} \wedge \mathrm{Type}(\mathrm{Var}(G(t))) \subseteq \sum]$ where Type(G(t)) denotes the type of G(t), Bool denotes {true, false} and Var(G(t)) denotes the set of variables in G(t).
8. E is an arc expression function such that

   $\square a \in A$ $[\mathrm{Type}(E(a)) = C(p(a))_{MS} \wedge \mathrm{Type}(\mathrm{Var}(E(a))) \subseteq \sum]$

   where p(a) is the place of N(a), Type(E(a)) denotes the type of E(a), $C(p(a))_{MS}$ denotes the set of multisets over a set C(p(a)), and Var(E(a)) denotes the set of variables in E(a).
9. I is an initialization function such that $\square p \in P$ $[\mathrm{Type}(I(p)) = C(p)_{MS}]$

   where Type(I(p)) denotes the type of I(p) and $C(p)_{MS}$ denotes the set of multisets over a set C(p).

From the set of color sets expressed $\sum$ in above 9-tuple, the XGSP based control mechanisms have the set of color sets as follows.

$\sum$ = {COUNT, INT, UNIT, BOOL, smallINT, Lock, GrantDeny, GrantDeny2, UserID, RoleName, AppID, Action, NewReq, OldReq, SharedReq, NewReqs, OldReqs, SharedReqs, LockxNewReq, LockxGrantDenyxNewReq, LockxGrantDeny}

The elements P, T, and A in the 9-tuple are a set of places, transitions, and arcs respectively. The N means no arc may connect twp places or two transitions. In the CP-nets model of XGSP based control mechanisms, the color function C maps the places Simulation-Start and Request-Nodes into COUNT, the place Request-Queue into

NewReqs, the places Time and Waiting-Time into INT, the place State-Information-Table into OldReqs, the place Waiting-List-Queue into SharedReqs, the places Busy, Invalid, Implicit, Shared, Exclusive, and Released into LockxNewReq, the places Next-Request, L, and GiveFloor into Lock, the place U into LockxGrantDeny, the place TimeOver into BOOL, and all other places into LockxGrantDenyxNewReq. Item7, the guard function is an expression which evaluate to Boolean (true or false). The arc expression function and initialization function are also an expression which evaluate to valid type value. The declarations for the CP-nets model of XGSP based control mechanisms are represented using the CP-nets ML (an extension of the functional programming language SML (Standard Meta Language)) [58] in Figure 5.6.

Declaration of Variables
val k = 1000;
val queuedTime = 100;
var bools : BOOL;
var n : COUNT;
var decisionNum : smallINT;
var proctime : INT;
var releasedtime : INT;
var newReq : NewReq;
var oldReq : OldReq;
var sharedReq : SharedReq;
var newReqs : NewReqs;
var oldReqs : OldReqs;
var tmpReqs : OldReqs;
var sharedReqs : SharedReqs;
var grantDeny : GrantDeny;

var grantDeny2 : GrantDeny;
globref connected = false;
globref response = 0 : smallINT;
Declaration of Color sets
colset BOOL= bool;
colset COUNT = int timed;
colset UNIT = unit timed;
colset INT = int;
colset smallINT = int with 0..4;
colset Lock = with lock timed;
colset UserID = with A | B | C | D | E | F | G | H | I | J;
colset RoleName = with nonmobile_user | mobile_user;
colset AppID = with wb;
colset Action = with master | slave | line| rect | oval | pen | eraser | move | load | clear;
colset NewReq = record userID : UserID * roleName : RoleName * appID : AppID * actions : Action * AT : BOOL;
colset OldReq = record userID : UserID * roleName : RoleName * appID : AppID * actions : Action * AT : BOOL;
colset SharedReq = record userID : UserID * roleName : RoleName * appID : AppID * actions : Action * AT : BOOL;
colset NewReqs = list NewReq;
colset OldReqs= list OldReq;
colset SharedReqs = list SharedReq;
colset LockxNewReq = product Lock * NewReq timed;
colset GrantDeny = with grant | deny | queued | released | give;
colset GrantDeny2 = subset GrantDeny with [grant , deny];
colset LockxGrantDenyxNewReq = product Lock * GrantDeny * NewReq;
colset LockxGrantDeny = product Lock * GrantDeny;

Functions

```
fun decisionGrant() = grant;
fun decisionDeny() = deny;
fun decisionQueued() = queued;
fun decisionGD() = GrantDeny2.ran();
fun decisionGT() =(released, IntInf.toInt (time()));
fun decisionGR() = (grant, IntInf.toInt (time()));
fun decisionGive() = give;
fun intTime() = IntInf.toInt (time());
fun GT(a:int, b:int) = ((a-b) > queuedTime);
fun changedAction(newReq:NewReq, tmpReqs:OldReqs, oldReqs:OldReqs) =
( if length oldReqs = 0 orelse length tmpReqs = 0
  then oldReqs^^[newReq]
  else (if (#userID newReq) = (#userID (hd tmpReqs))
          then (rm (hd tmpReqs) oldReqs)^^[newReq]
          else  changedAction(newReq, tl tmpReqs, oldReqs)))
fun existAction(newReq:NewReq, oldReqs:OldReqs) =
( if length oldReqs = 0
  then false
  else (if (#actions newReq) = (#actions (hd oldReqs))
          then true
          else  existAction(newReq, tl oldReqs)))
fun findAction(sharedReq:SharedReq, tmpReqs:OldReqs) =
( if length tmpReqs = 0
  then sharedReq
  else (if (#actions sharedReq) = (#actions (hd tmpReqs))
          then {userID = #userID (hd tmpReqs), roleName = #roleName (hd tmpReqs),
                    appID = #appID (hd tmpReqs), actions = slave, AT = #AT  (hd tmpReqs)}
          else  findAction(sharedReq, tl tmpReqs)))
fun timeCheckAction(sharedReq:SharedReq) =
( if (#AT sharedReq) then false else true)
fun timeOverAction(sharedReqs:SharedReqs) =
( [{userID = #userID (hd sharedReqs), roleName = #roleName (hd sharedReqs),
   appID = #appID (hd sharedReqs), actions = #actions (hd sharedReqs),
   AT = true}]^^rm (hd sharedReqs) sharedReqs)
fun setFalseAction(sharedReqs:SharedReqs) =
( [{userID = #userID (hd sharedReqs), roleName = #roleName (hd sharedReqs),
   appID = #appID (hd sharedReqs), actions = #actions (hd sharedReqs),
   AT = false}]^^rm (hd sharedReqs) sharedReqs)
fun expTime (mean: int) =
  let
   val realMean = Real.fromInt mean
   val rv = exponential((1.0/realMean))
  in
    floor (rv+0.5)
  end;
fun newRequest() =
  {userID = UserID.ran(), roleName = RoleName.ran(),
    appID   = AppID.ran(), actions = Action.ran(), AT = false}
fun oldRequest(sharedReq:SharedReq) =
  {userID = #userID sharedReq, roleName = #roleName sharedReq,
    appID = #appID sharedReq, actions = slave, AT = #AT sharedReq}
  fun get_from_externalProcess() =
    ConnManagementLayer.receive("Conn", integerDecode);
  fun send_to_decisionService(roleName, appID, action) =
  ConnManagementLayer.send("Conn",
"<RoleName>"^roleName^"</RoleName><AppID>"^appID^"</AppID>
  <Action>"^action^"</Action>", stringEncode);
```

Figure 5.6: Declarations for the CP-nets model of Control Mechanism

5.3.2 Dynamic Behavioral Properties of CP-nets and Representation of Control Mechanisms by the Properties

The dynamic behavior of CP-nets is a data transformation between the occurring transition and the occurred transition with a time delay of some small magnitude. In this section, we present the dynamic behavioral properties of CP-nets [64] about binding, marking, enabling, and occurrence, and the example representation of them in the modeled XGSP based control mechanisms.

- Binding – this means to bind correct token values to the variable of the token type. For example, in our modeling, the transition Start may have the binding element such as <Start, (lock, {userID = kakim, roleName = mobile-user, appID = wb, action = pen, AT = false})@777> which is composed of a transition and a binding.
- Marking – a set of multisets over the set of tokens positioned on the individual places. For example, the places Request-Queue, Waiting-List-Queue, and State-Information-Table have an empty list token as an initial marking respectively.
- Enabling – when tokens from all the input places of a transition are evaluated by the arc expressions between the input places and the transition and before the tokens are added to the output places of the transition, the transition is called enabled with a set of binding elements.
- Occurrence – after the transition is enabled and by removing token from the input places and adding the tokens to the output places of the transition, the transition is called occurred and then the occurrence sequence is composed of a

sequence of reachable markings and occurring steps. For example, when the transition Start occurs, one specified token will be removed from the input places Request-Queue and Next-Request. At the same time, three tokens will be added to the output places. The place Request-Queue will get a token with a list type as a token color set, the place Busy will get a token with a value (lock, {userID = kakim, roleName = mobile-user, appID = wb, action = pen, AT = false})@777, and the place Time will get a token with the value of current modeling time.

5.4 Verification for Correctness of Control Mechanisms based on State Space Analysis

In this section, we verify the correctness of the XGSP based control mechanism modeled by the CP-nets from the previous sections with a means of simulations and state spaces (which are also called occurrence graphs) [64]. The CP-nets provides a simulation tool [14] that simulates a system by nondeterministic distributing color tokens into a model, and a state space generation tool [14] that generates a report for a sequence of occurrence states. To construct state spaces means to generate all the possible occurrence graphs that are composed of nodes and arcs. Nodes in state spaces are generated for each reachable markings, and arcs in the state spaces are generated for each occurring binding elements. The report generated from the state space generation tool is used for verifying the correctness of a model. In the next five subsections, we analyze the simulation behaviors and the occurrence state information generated from the state space generation tool.

5.4.1 Statistical Information of State Spaces and SCC Graph

The state spaces report has the following five parts. The first part, statistical information report of state spaces and SCC (strongly connected components) graph [64], is shown in Table 5.1. The statistical report contains information about the size of nodes and arcs, and time and calculation status took for generating state spaces and SCC graphs. In the case of 4133 which the number of requests is, the state space has 52643 nodes and 79809 arcs in partially calculated graphs, and this took 300 seconds for generating the state space which is composed of the nodes and the arcs. Also, the report shows information of SCC graph that is identical to the information of the state space except for time taken for generating the components. The strongly connected components are a maximal subgraph to find a path from any one node to any other node. In the report, the number of strongly connected components in the modeled control mechanisms is equal to the number of state space nodes. This implies that the modeled control mechanisms have strongly connected components with just one node. Therefore, the modeled mechanisms have no infinite occurrence sequences. This means the simulation of the modeled mechanisms terminates after processing some number of requests if we put the stop criteria such as limiting the number of requests into the model. In other words it shows the modeled mechanisms are working as we expect to achieve termination normally in forcibly limiting the number of requests. This report shows statistical information about sizes of state spaces and strongly connected components of the state spaces generated from the tools of CP-nets. Therefore, diverse properties generated from state spaces and the strongly connected graphs of the state spaces can be used to get important and useful information such as mutual exclusion, deadlock, and starvation about the modeled XGSP control mechanisms.

Statistics		
# of Requests = 4133		
	State Space	**SCC Graph**
Nodes	52643	52643
Arcs	79809	79809
Seconds	300	16
Status	Partial	

Table 5.1: Statistical Information of State Space and Scc Graph

5.4.2 Boundedness Properties and Mutual Exclusion of Modeled Control Mechanisms

The second part shows boundedness properties of the state spaces report in Table 5.2. The properties express the upper integer bounds which is the maximal number of tokens and the lower integer bounds which is the minimal number of tokens that the places in a modeling may have. In the integer bounded information of the modeled mechanisms, the places Busy, Decision-Done, Exclusive, Implicit, Invalid, Next-Request, Released, Shared, L, and Z have one token in upper bound and zero token in lower bound. These are places in decision procedural stage of a moderator node that is a critical section for mutual exclusion. Note that the upper integer bound of the places Busy and Next-Request is 1. This implies if 1 is upper integer bound for the places, then at most one request is processed in the decision procedural stage (critical section) at any time. As a contradiction, if the upper integer bound of the place Busy is more than two, then the upper bound of the place Next-Request has to be at least more than two at any time in any reachable markings. But we see the upper bound of the place Next-Request is 1 and thus the place Busy is not able to contain more than two token at any time in each reachable marking. This indicates the modeled control mechanisms are ensuring mutual exclusion for the decision procedural stage to avoid race condition among requests.

Also, we need to show all the requests that wish to enter a critical section have to be serviced and only one request must enter the critical section. In other words, all the requests have to be serviced and thus not starved for getting the service. To show this property, we consider the transitions Arrival and Start, and the integer bounds of the places Request-Queue and Next-Request in Figure 5.5. The occurring transition Arrival puts new requests into the place Request-Queue and then the first request in the queue enters the decision procedural stage (critical section) through the occurrence of the transition Start. As shown in Table 5.5, the place Request-Queue has exactly one token at any time during the execution of the modeling, and the place Next-Request has one token as upper integer bound. This means the transition Start will be enabled at any time during the execution of the modeling, and hence one request will enter the decision procedural stage. Thus, the requests will be serviced one by one as we expect. This report also tells us that the places State-Information-Table, and Waiting-List-Queue as well as the place Request-Queue in each reachable marking have exactly one token in upper and lower integer bounds. This means the places have one list token each used as a queue and the modeled mechanism in Figure 5.5 is also working as expected.

Boundedness Properties		
Best Integer Bounds	**Upper**	**Lower**
A	2	0
B	3	0
Busy	1	0
C	1	0
D	1	0
Decision_Done	1	0
E	1	0
Exclusive	1	0
GiveFloor	1	0
Implicit	1	0
Invalid	1	0
L	1	0
Next_Request	1	0
Nodes	14	0
Released	1	0
Request_Nodes	1	0
Request_Queue	1	1
Shared	1	0
Simulation_Start	1	0
State_Information_Table	1	1
Time	3	0
TimeOver	3	0
U	1	0
Waiting_List_Queue	1	1
Waiting_Time	1	1
Z	1	0

Table 5.2: Boundedness Properties

5.4.3 Home Properties

The third part of the report provides information about home properties. The home markings in the home properties mean markings which can always be reached from all reachable markings [64]. The property in Table 5.3 shows that the initial marking of the modeled control mechanisms in Figure 5.5 is not a home marking because the initial marking is a marking for starting just the modeled mechanisms by substituting a integer value one for the transition Init as a binding element of the transition, and hence any subsequent markings never return to the initial marking. Table 5.7 shows the fairness property of the state spaces report. It also provides information about how often different binding elements occur in each markings of the modeled mechanisms. The property shows that the modeled mechanisms have no infinite occurrence sequences. In other words, the modeled mechanisms have no infinitely many different binding elements. These properties imply any subsequent markings are not able to reach the initial marking and the initial marking has no many different binding elements in the modeling. Therefore, these properties show that the modeled mechanisms in Figure 5.5 are working as expected.

Home Properties	
Home Markings	Initial Marking is not a home marking

Table 5.3: Home Properties

5.4.4 Liveness Properties of Modeled Control Mechanisms

The fourth part of the report provides information about liveness properties. The dead transition instances in the properties of the report mean some transition instances which are never enabled in all reachable markings. Also, the live transition instances in the properties mean transition instances which can always be enabled at least once more in all reachable markings. The report in Table 5.4 shows the modeled control mechanisms have no dead transition instances, and no live transition instances. This implies that each transition in the modeled mechanisms is enabled in at least one marking among all reachable markings. This also tells us that there are no transition instances which can always be enabled at least once more in all reachable markings of the modeled mechanisms.

To show that the modeled mechanisms are working as expected and the correctness of the external process (Access Type Decision Service), we consider two more reports about the liveness properties. Table 5.5 and 5.6 show the liveness properties of the modeled access control mechanism and floor control mechanism respectively separated from the modeled control mechanisms. In Table 5.5, the report shows there are two dead transition instances. This means the transitions D4 and D5 in Figure 5.5 are never enabled in all reachable markings. This implies the access type decision service is correctly working as expected because the two transitions are never used to process requests in the modeled XGSP-RBAC mechanism. There are two dead transition instances as shown in Table 5.6. This also means the transitions D2 and D3 are never enabled in all reachable markings. Therefore, the access type decision service is correctly working as expected because the two transitions are never used in the modeled XGSP-Floor mechanism. Thus, the reports in Table 5.4, 5.5, and 5.6 show the modeled mechanisms and the access type decision service are correctly working as expected, and from the timed CP-nets and the properties of previous sections, new requests in exponentially distributed arbitrary interval are generated and thus no dead lock situation in which all the requests may be blocked is occurred.

Liveness Properties	
Dead Transition Instances	None
Live Transition Instances	None

Table 5.4: Liveness Properties of Modeled Control Mechanisms

Liveness Properties	
Dead Transition Instances	D4 and D5
Live Transition Instances	None

Table 5.5: Liveness Properties of Modeled XGSP-RBAC Mechanism

Liveness Properties	
Dead Transition Instances	D2 and D3
Live Transition Instances	None

Table 5.6: Liveness Properties of Modeled XGSP-Floor Mechanism

5.4.5 Fairness and Starvation Properties of Modeled Control Mechanisms

The fifth part of the report provides information about fairness properties. The information tells us how often the different binding elements of each transition in a modeling can occur [64]. The report in Table 5.7 shows there are no infinite occurrence sequences in the modeled mechanisms. But the modeled mechanisms may have infinite occurrence sequences in the transition Arrival of the modeling that has some binding elements (Arrival, NewReqs) which are repeated indefinitely where NewReqs is a type of color set with list type. The CP-nets does not consider such binding elements as making sense. Thus, the report shows the modeled mechanisms have no infinite occurrence sequences. The trivial infinite occurrence sequences occurred in the transition Arrival also show if the transition ceases to occur, the simulation of the modeled mechanisms must terminate. The transition generates new requests until some prefixed number of requests. Therefore, the properties also show the modeled mechanisms are correctly working satisfying stop criteria to terminate the simulation as expected because there are no enabled transitions in the modeling after the transition Arrival generates some prefixed number of requests, i.e. there are no more enabling and occurrence in the transition Arrival. In addition to the expectation, the modeled mechanisms are fair. Any requests may never be starved since there are no infinite occurrence sequences which may starve the requests forever. Also, the requests waiting for floors in the place Waiting-List-Queue in Figure 5.5 will never be starved. Thus, it shows there is no starvation in the modeled control mechanisms.

Fairness Properties
No infinite occurrence sequences

Table 5.7: Fairness Properties

5.5 Summary

In this chapter we modeled the XGSP based control mechanisms (XGSP-RBAC and XGSP-Floor) by CP-nets with time which is practically integrated into our collaboration framework. Also we presented informal introductions and formal definitions for the modeled control mechanisms. The key reason why we modeled the mechanisms was to prove the correctness of the mechanisms by means of simulations and state spaces (also called occurrence graphs). The formal verification was done with a state space report which contains useful information about some dynamic properties of state space graphs and SCC graphs generated by CP-nets tool. The modeled control mechanisms are working as expected through the analysis of the state space report in terms of mutual exclusion, dead lock and starvation. Thus, the main contribution of this chapter is a formal verification by CP-nets for the correctness of the modeled control mechanisms to show consistent shared state at application level among collaborating users by mitigating race conditions to shared resources.

Chapter 6

Conclusion

In this thesis we have attempted to provide a virtual workspace for not only remotely dispersed users but also roaming users with cell phone devices. This attempt has been driven by building integrated collaboration system including cell phone devices. We have also attempted to provide the virtual workspace with control capabilities and collaborative applications for synchronous and ubiquitous collaboration. As ubiquitous collaboration and access becomes more prevalent in the future, it will become more important to provide a virtual workspace in which geographically dispersed users can work together. But, as the number of users with a large number of disparate access devices increases, the difficulties for protecting secured computing environments and resources from unauthorized users as well as unsecured access devices will increase since computing environments and resources can be compromised by inadequately secured entities – human, devices, software, data, and so on. The ubiquitous collaboration includes sharing applications in anytime and anywhere. Mechanisms for dealing with consistency in the use of application shared among collaborators will have to be considered in an unambiguous manner according to increasing heterogeneous collaboration applications. Also, the intrinsic latency occurred due to the increase of interactive distance, relatively to the latency occurred in collocated place, may affect on the choice of the mechanism for shared state consistency of application.

The following sections discuss the current status of collaboration framework, problems encountered during the development of the framework, and future works. These sections are categorized with control and collaborative application / component issues.

6.1 Collaboration Framework and Control Mechanisms

We built a collaboration framework on heterogeneous (wire, wireless) computing environment for synchronous and ubiquitous collaboration as well as heterogeneous community collaboration. A key function of the framework is to provide a generic solution for controlling sessions in a conference, accesses to resources, and maintaining shared state consistency at application level by defining a general protocol in XML. Also, the framework provides a structure for development and deployment of collaborative applications that can be used to support asynchronous collaboration by allowing different users of a session to access the same resource at different times, and synchronous collaboration by enabling different users of a session to share the same resource in real time (at the same time).

Future work

Fault-tolerant role delegation mechanism with role hierarchy policy: During our experiments with the collaboration framework, one of problems encountered was a failure like network disconnection of a moderator or chairperson node. If a moderator or chairperson node fails or is disconnected, and is not able to recover from the failure for some amount of time, one of participants in collaboration capable of having the role capability of the moderator or chairperson has to be elected. We tested it with an event driven message mechanism. But, when the network connection of a moderator or chairperson node was lost, it did not work since the event messages could not be disseminated in disconnected network. One approach to overcome the problem by exploring different fault-tolerant role delegation mechanism (for example, polling mechanism by heart-beat message between a moderator node and a conference manager) with role hierarchy policy will be considered in future work. We also left in future work support of the role hierarchy policy with the fault-tolerant role delegation mechanism issue.

6.1.1 Session Control

We presented a generic solution for controlling sessions in a single easy-to-use integrated collaboration system. The solution provided fundamental control logics to manage presences of and connectivity among collaborating users in an online session, and organize online sessions or a conference. To describe control logics of presences, connectivity, and sessions' states, we used XML as a protocol definition language of session control. The XML based General Session Protocol (XGSP) is a protocol for streaming control messages written in XML to provide various collaboration sessions (heterogeneous community collaboration sessions) in a conference for users. The session control protocol account for policy, presence, session creation, initiation, teardown, and so on.

6.1.2 Access Control

We presented a solution for controlling accesses to applications with the notion of role as an intermediate control entity between collaborating users and collaborative applications. The roles are based on users' privileges and devices' capabilities to allow users to manipulate protected applications in the collaboration – XGSP-RBAC (XGSP Role Based Access Control). The use of role simplifies the administrative management of access rights for applications and gives an administrator flexible adaptation to the changes of collaboration environment. To specify access control policies and exchange request-response messages of access control for applications, XML was used because it is easy to understand and use with pre-existing industry standard parsers. The XGSP-RBAC mechanism provides flexibility adapting to the state change of collaborative applications that may be occurred from cooperation among collaborators at run time in collaboration system. Also, fine-grained access control for the instance of individual resource as well as for individual user was used.

Future work

- We left in future work support of role assignment policy for assigning roles to users.

- In future work, we will design and implement the authentication service for users joining a session during roaming with cell phone devices, and the encryption service for messages sent to and from the cell phone devices.

6.1.3 Floor Control

We presented a policy and a mechanism implementing it – XGSP-Floor (XGSP Floor control) for coordinating accesses to applications and maintaining shared state consistency at application level. The XGSP-Floor provides significant flexibility, ranging from free-for-all (no floor) to application specific floor mechanisms for avoiding uncoordinated accesses to shared collaboration applications. A moderator is responsible for maintaining the consistent state of applications in our collaboration system. Even though our underlying floor control scheme is a moderator-mediated interaction mechanism, a floor can automatically be assigned to a floor requester without the mediation of the moderator according to the policy. We showed with example applications – shared whiteboard and collaborative chess game that social protocols used in a face-to-face offline session can be mapped to mechanisms able to be used in an online session with user interfaces between participants and CSCW environment. XGSP-Floor control tool provides human-computer interaction for control of floor for roaming participants with cell phone as well as desktop participants in collaboration. Also, we showed a synchronous collaboration, which means all participants in collaboration always have the same views and data in real time, with a major event conflict detection function and a non-optimistic locking mechanism used in our collaboration.

Future work

In future work we left support for floor control of synchronous collaboration media applications such as audio and video applications. We will apply the moderator-mediated floor control mechanism to the synchronous collaboration media applications and consider different floor control mechanisms with different parameters for floor control of the shared whiteboard application in our collaboration domains – heterogeneous community collaboration as well as synchronous and ubiquitous collaboration. In the future work, we also need to further implement XGSP-Floor tool with more detailed functionalities for the synchronous collaboration media applications.

6.2 Collaborative Applications and Components

6.2.1 Instant Messenger and Proxy

We designed and implemented an application proxy used for Instant Messenger (IM). The proxy has responsibility for getting responses from Jabber server and performs any necessary conversions for clients on mobile device. As an intermediary, the proxy retains communication interfaces and thus can offload some computational needs. The benefits of using Jabber server include presence management, message processing based on XML, transparent interoperability, structured information data, and open formats. With such an approach using open or commercial technology, in a modular fashion, with appropriate interface for collaborative applications, we can build a sustainable high functionality system taking advantage of the latest technologies and enable multiple collaborative applications to re-use the same basic technologies. We showed the architecture of Jabber with the proxy as a derivative of shared event model.

In the architecture, all the clients have a copy of the application before joining collaboration. Then events such as presence are sent to all participating clients.

6.2.2 Shared Whiteboard and Filter

We designed and implemented an application filter able to cooperate and to coordinate heterogeneous types of applications on heterogeneous platforms. The purpose of the application filtering is to convert one type of representation to other types of representations on heterogeneous platforms with different screen (or canvas) sizes and different representation formats. We used shared event model for drawing objects and shared display model for graphical image display. In the shared display collaboration model, clients share the graphical image display and the state is maintained (shared) among clients by transmitting the changes in the display through our message and service middleware. The supporting heterogeneous clients require that sophisticated shared display environments automatically change size and display representation formats to suit each client. The application filter is used for synchronizing views (or results) shared among heterogeneous collaborative applications and thus maintaining consistent shared state across the applications.

- **Future work**

Much of image detail losses were found in the experiment of the application filter with shared whiteboard application. Much details of image displayed on cell phone device through transcoding of a image transferred from desktop were lost because the graphical image was shrunk to accommodate the screen size of cell phone. To improve the quality of the transcoded image from desktop into cell phone, we should consider different transcoding and scaling algorithm in future work for heterogeneous collaborative applications on heterogeneous computing platforms. Another interesting issue is the magnification from cell phone to desktop. We tested this with Nokia phone [86] which allows application developers to control media like cell phone camera. The test experiment is as follows: Nokia phone user takes a picture with built-in camera. The picture is displayed on the shared whiteboard canvas in Nokia phone, and then the byte message of the picture is sent into the whiteboard application filter and the magnified byte image through transcoding in the application filter is disseminated into a desktop through a broker. Here we encountered the scaling (magnification) problem as well. Therefore we also should consider this case in future work.

6.2.3 Collaborative Chess Game Application

- **Future work**

CGL has an interactive two-player collaboration chess game for desktop devices. We did not implement the chess game on cell phone. In future work, we will consider the design and implementation of the chess game with shared event model on Wi-Fi (wireless fidelity) [107] phone, to reduce latency and to improve computing capability. Also we will consider the extension of our work (collaboration framework and other collaborative applications as well as the collaborative chess game) to new generation of cell phone such as iPhone [47] which is multimedia and Internet-enabled mobile phone.

Bibliography

[1] Access Grid (2003), http://www.accessgrid.org

[2] Ahmet Fatih Mustacoglu, Wenjun Wu, and Geoffrey Fox. Internet Calendaring and Scheduling Core Object Specification (iCalendar) Compatible Collaborative Calendar-Server (CCS) Web Services Proceedings of IEEE 2006 International Symposium on Collaborative Technologies and Systems CTS 2006 Conference Las Vegas May 14-17 2006.

[3] Andreas Schaad, Jonathan Moffett, and Jeremy Jacob. The Role-Based Access Control System of a European Bank: A Case Study and Discussion. SACMAT: 6th ACM Symposium on Access Control Models and Technologies, ACM. Chantilly, VA.

[4] Anil L. Pereira, Vineela Muppavarapu, and Soon M. Chung. Role-Based Access Control for Grid Database Services Using the Community Authorization Service. IEEE Transactions on Dependable and Secure Computing, vol. 3, no. 2, April-June 2006.

[5] Banavar, G., Beck, J., Gluzberg, E., Munson, J., Sussman, J., Zukowski, D., "Challenges: An Application Model for Pervasive Computing", To appear in the proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking, Mobicom 2000.

[6] Bernstein, P., Goodman, N. and Hadzilacos, V. (1987) Concurrency control and recovery in database systems, Addison-Wesley.

[7] B. Lampson, “Protection,” Proceedings of the Fifth Princeton Symposium of Information Science and Systems, pp. 437-443 (Mar. 1971); reprinted in Operating Systems Review 8(1), pp. 18-24 (Jan. 1974).

[8] Boyd J., ”Floor control policies in multi-user applications”, in INTERACT ’93 and CHI ’93 Conference on Human Factors in Computing Systems, ACM Press, p. 107–108, 1993.

[9] Brad A. Myers, et al. “Using Hand-Held Devices and PCs Together”. ACM Communications, 2001. To appear.

[10] Brad A. Myers, Yu Shan A. Chuang, Marsha Tjandra, Mon-chu Chen, and Chun-Kwok Lee. "Floor Control in a Highly Collaborative Co-Located Task".

[11] BufferedImage (Java 2 Platform SE 5.0). http://java.sun.com/j2se/1.5.0/docs/api/java/awt/image/BufferedImage.html

[12] Community Authorization Server (CAS). http://www.lesc.ic.ac.uk/projects/cas.html

[13] Community Grids Lab (CGL), http://communitygrids.iu.edu

[14] CPN Tools. CPN Tools Homepage. http://wiki.daimi.au.dk/cpntools/

[15] David F. Ferraiolo and Richard Kuhn. “Role-Based Access Control”, Proceedings of the 15th NIST-NSA National Computer Security Conference, Baltimore, MD, 13-16 October 1992.

[16] D.F. Ferraiolo, J. Cugini, D.R. Kuhn (1995) "Role Based Access Control: Features and Motivations", Computer Security Applications Conference - extends the 1992 model.

[17] Dommel H.P. and J.J. Garcia-Luna-Aceves, “Design issues for floor control protocols”, In Proceedings of SPIE Multimedia and Networking, (San Jose, CA, USA), pp. 305–16, Feburary 1995.

[18] Dommel H.P. and J.J. Garcia-Luna-Aceves, “Floor Control for Multimedia Conferencing and Collaboration”, ACM Multimedia Systems, Vol. 5, No. 1, January 1997.

[19] Dommel H.P. and J.J. Garcia-Luna-Aceves, “Comparison of floor control protocols for collaborative multimedia environments,” In Proceedings of SPIE Symposium on Voice, Video, and Data Communications, (Boston, MA), November 1998.

[20] Dommel H.P. and J.J. Garcia-Luna-Aceves, “Efficacy of floor control protocols in distributed multimedia collaboration”, Cluster Computing, Vol. 2, No. 1, pp. 17-33, 1999.

[21] Dommel H.P. and J.J. Garcia-Luna-Aceves, “Group Coordination Support for Synchronous Internet Collaboration.” IEEE Internet Computing, pp. 74-80, Mar/Apr. 1999.

[22] D. Saha and A. Mukherjee. Pervasive Computing: A Paradigm for the 21st Century. Published by the IEEE Computer Society, Vol. 36, No. 3. pp. 25-31 March 2003.

[23] D. W. Chadwick, and A. Otenko. RBAC policies in XML for X.509 Based Privilege Management. SEC 2002, Egypt, May 2002.

[24] D. W. Chadwick, and A. Otenko. The PERMIS X.509 Role Based Privilege Management Infrastructure. 7th ACM Symposium on Access Control Models and Technologies, 2002.

[25] D. W. Chadwick and A. Otenko. The PERMIS X.509 Role Based Privilege Management Infrastructure. Future Generation Computing Systems, 2003.

[26] D. W. Chadwick, A. Otenko, and E. Ball. Role-based Access Control with X.509 Attribute Certificates. IEEE Internet Computing, March-April 2003, pp. 62-69

[27] DOM. http://www.w3.org/DOM/

[28] Edwards, W.K. "Flexible Conflict Detection and Management in Collaborative Applications," in Proceedings UIST'97: ACM SIGGRAPH Symposium on User Interface Software and Technology. 1997. Banff, Alberta, Canada: pp. 139-148.

[29] Extensible Markup Language (XML) 1.0, W3C REC-xml, October 2000.

[30] F. Berman, G. Fox, and A. Hey, editors. Grid Computing: Making the Global Infrastructure a Reality, John Wiley & Sons, 2003.

[31] Geoffrey Fox, Hasan Bulut, Kangseok Kim, Sung-Hoon Ko, Sangmi Lee, Sangyoon Oh, Shrideep Pallickara, Xiaohong Qiu, Ahmet Uyar, Minjun Wang, Wenjun Wu. Collaborative Web Services and Peer-to-Peer Grids or in sxw presented at 2003 Collaborative Technologies Symposium (CTS'03).

[32] Geoffrey Fox, Wenjun Wu, Ahmet Uyar, Hasan Bulut, Shrideep Pallickara. Global Multimedia Collaboration System in Proceedings of the 1st International Workshop on Middleware for Grid Computing co-located with Middleware 2003, June 17, 2003 Rio de Janeiro, Brazil.

[33] Geoffrey C. Fox, Wenjun Wu, Ahmet Uyar Hasan Bulut. Design and Implementation of Audio/Video Collaboration System Based on Publish/subscribe Event Middleware Proceedings of CTS04 San Diego January 2004.

[34] Geoffrey Fox. Collaboration and Community Grids Special Session VI: Collaboration and Community Grids Proceedings of IEEE 2006 International Symposium on Collaborative Technologies and Systems CTS 2006 conference Las Vegas May 14-17 2006; IEEE Computer Society, Ed: Smari, Waleed & McQuay, William, pp 419-428. ISBN 0-9785699-0-3 DOI.

[35] G. Gallasch and L. M. Kristensen. Comms/CPN library. http://wiki.daimi.au.dk/cpntools/

[36] G. Gallasch and L. M. Kristensen. Comms/CPN: A Communication Infrastructure for External Communication with Design/CPN. Proceedings of the CPN'2001 Workshop.

[37] Global-MMCS (Global Multimedia Collaboration System). http://www.globalmmcs.org

[38] Globus Grid Security Infrastructure (GSI). http://www.globus.org/toolkit/docs/4.0/security

[39] Greenberg, S. "Sharing views and interactions with single-user applications," Proceedings of the ACM/IEEE Conference on Office Information Systems. 1990. Cambridge, MA: pp. 227-237.

[40] Greenberg, S. "Personalizable groupware: Accommodating individual roles and group differences," In Proceedings of the ECSCW `91 European Conference of

Computer Supported Cooperative Work. 1991. Amsterdam: Kluwer Academic Press. pp. 17-32.

[41] Greenberg, S. and Marwood, D. "Real time groupware as a distributed system: Concurrency control and its effect on the interface," Proceedings of the ACM CSCW Conference on Computer Supported Cooperative Work, October 22-26, 1994. North Carolina, ACM Press.

[42] GridFTP: Universal Data Transfer for the Grid http://www.globus.org/datagrid/gridftp.html

[43] GroupKit. http://www.groupkit.org

[44] Hasan Bulut, Shrideep Pallickara and Geoffrey Fox. Implementing a NTP-Based Time Service within a Distributed Brokering System ACM International Conference on the Principles and Practice of Programming in Java, June 16-18, Las Vegas, NV.

[45] I. Foster and C. Kesselman, editors. The Grid: Blueprint for a New Computing Infrastructure, Morgan-Kaufman, 1998.

[46] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Applications, 2001. 15(3): p. 200-222.

[47] iPhone. http://en.wikipedia.org/wiki/IPhone

[48] ITU. Recommendation H.225 (2000), Calling Signaling Protocols and Media Stream Packetization for Packet-based Multimedia Communication Systems.

[49] ITU. Recommendation H.243 (2000), Terminal for low bit-rate multimedia communication.

[50] ITU. Recommendation H.245 (2000), Control Protocols for Multimedia Communication.

[51] ITU-T Recommendation H.320, Narrowband Visual Telephone Systems and Terminal Equipment. 1993.

[52] ITU. Recommendation H.323 (1999), Packet-based multimedia communication systems.

[53] ITU-T Recommendation X.509 (2001). The Directory: Authentication Framework.

[54] Jabber Instant Messenger, http://jabber.org

[55] Java. http://java.sun.com/

[56] Java Message Service (JMS), http://java.sun.com/products/jms

[57] JDOM. http://www.jdom.org/

[58] J.D. Ullman, Elements of ML Programming. Prentice-Hall, 1993.

[59] J. Gray, P. Helland, P. O'Neil, and D. Shasha, "The dangers of replication and a solution," in To appear in Proc. of the 1996 SIGMOD Conference, June 1996.

[60] J. Peterson, Petri Net Theory and the Modeling System. Englewood Cliffs, NJ: Prentice-Hall, 1981

[61] J. Rosenberg et al. (2002) "SIP: Session Initiation Protocol", RFC 3261, Internet Engineering Task Force, http://www.ietf.org/rfc/rfc3261.txt

[62] J2ME. http://java.sun.com/javame/index.jsp

[63] Katrinis K., Parissidis G., and Plattner B., "Activity Sensing Floor Control in Multimedia Collaborative Applications", 10th International Conference on Distributed Multimedia Systems (DMS 2004).

[64] K. Jensen, Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use, vol. 1, Basic Concepts. Monographs in Theoretical Computer Sciences. Springer-Verlag, 1997.

[65] Koskelainen P., Schulzrinne H. and Wu X.(2002), A SIP-based Conference Control Framework, NOSSDAV'02, May 12-14, 2002, Miami Beach, Florida, USA.

[66] kXML. http://kxml.objectweb.org/

[67] Liang Fang, Dennis Gannon, and Frank Siebenlist. XPOLA: An Extensible Capability-based Authorization Infrastructure for Grids. In 4th Annual PKI R&D Workshop, April 2005.

[68] Lightweight Directory Access Protocol (LDAP). www.openldap.org

[69] L. Pearlman, et al., A Community Authorization Service for Group Collaboration. In Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks. 2002.

[70] L. Pearlman, et al., The Community Authorization Service: Status and Future. CHEP03, March 24-28, 2003, La Jolla, California.

[71] Luca Chittaro. Visualizing Information on Mobile Devices. IEEE Computer Magazine. March 2006 (Vol. 39, No. 3) pp. 40-45.

[72] Malpani Radhika and Lawrence A. Rowe, "Floor control for large-scale MBone seminars", in Proceedings of the Fifth ACM International Conference on Multimedia, ACM Press, p. 155-163, 1997.

[73] Mark Roseman, Saul Greenberg. GROUPKIT: a groupware toolkit for building real-time conferencing applications. Proceedings of the 1992 ACM conference on Computer-supported cooperative work. Toronto, Ontario, Canada. 1992.

[74] Mark Roseman, Saul Greenberg. Building real-time groupware with GroupKit, a groupware toolkit. ACM TOCHI Trans on Comp Human Internet 3(1), pp 66-106, March 1995.

[75] Matt Bishop. Introduction to Computer Security. Addison Wesley.

[76] M. Handley, I. Wakeman, and J. Crowcroft, "CCCP: Conference Control Channel Protocol: A Scalable Base for Building Conference Control Applications," in ACM Conf. SIGCOMM., August 1995.

[77] McKinlay A., R. Procter, et al., "A study of turn-taking in a computer-supported group task", appeared in People and Computers VII: Proceedings of HCI '93, 1993.

[78] Mike Macedonia and Don Brutzman. MBONE, the Multicast Backbone, to be published in IEEE Computer. http://www.best.com~prince/techinfo/mbone.html

[79] Mobile SVG Profiles: SVG Tiny and Basic. http://www.w3.org/TR/SVGMobile/

[80] Moore's Law. http://en.wikipedia.org/wiki/Moore's_Law

[81] Moran, T., McCall, K., van Melle, B., Pedersen, E. and Halasz, F. (in press) "Design principles for sharing in Tivoli, a whiteboard meeting-support tool." In Designing Groupware for Real Time Drawing, S. Greenberg, S. Hayne & R. Rada ed. McGraw Hill.

[82] M. Weiser. "The Computer for the Twenty-First Century," Scientific American, September 1991.

[83] NaradaBrokering, http://www.naradabrokering.org

[84] NetMeeting, http://www.microsoft.com/windows/netmeeting

[85] NIST SIP (2001), http://snad.ncsl.nist.gov/proj/iptel

[86] Nokia 3650. http://europe.nokia.com/A4143688

[87] OpenH323 Project (2001), http://www.openh323.org

[88] Pederson, E., et al. "Tivoli: An Electronic Whiteboard for Informal Workgroup Meetings," in Proceedings INTERCHI'93: Human Factors in Computing Systems. 1993. Amsterdam, The Netherlands: pp. 391-398.

[89] "Peer-To-Peer: Harnessing the Benefits of a Disruptive Technology", edited by Andy Oram, O'Reilly Press March 2001.

[90] Privilege and Role Management Infrastructure Standards (PERMIS) project. http://www.openpermis.org

[91] R, B'Far. Mobile Computing Principles: Designing and Developing Mobile Applications with UML and XML, Cambridge University Press 2005.

[92] Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. Role-Based Access Control Models. IEEE Computer 29, 2 (Feb. 1996), pp. 38-47.

[93] Scalable Vector Graphics (SVG). http://www.w3.org/Graphics/SVG/

[94] Shen, H, and Dewan, P. Access Control for Collaborative Environments. In ACM Conference on Computer-Supported Cooperative Work. 1992, p. 51-58.

[95] Shrideep Pallickara and Geoffrey Fox. NaradaBrokering: A Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids. Proceedings of the ACM/IFIP/USENIX Middleware Conference. 2003. pp 41-61.

[96] Shrideep Pallickara, Harshawardhan Gadgil and Geoffrey Fox. On the Discovery of Topics in Distributed Publish/Subscribe systems Proceedings of the IEEE/ACM GRID 2005 Workshop, pp 25-32. November 13-14 2005 Seattle, WA.

[97] Shrideep Pallickara, Marlon Pierce, Harshawardhan Gadgil, Geoffrey Fox, Yan Yan, Yi Huang. A Framework for Secure End-to-End Delivery of Messages in Publish/Subscribe Systems. Proceedings of the 7th IEEE/ACM International Conference on Grid Computing (GRID 2006). Barcelona, Spain, 28-29 September 2006.

[98] Stefik, M., Bobrow, D.G., Foster, G., Lanning, S. and Tatar, D. (1987) "WYSIWIS revised: Early experiences with multiuser interfaces." ACM Transactions on Office Information Systems, 5(2), pp. 147-167, April.

[99] Sun Microsystems JXTA Peer to Peer Technology. http://www.jxta.org

[100] SVGArena. http://www.svgarena.org/

[101] Tcl/Tk (Tool Command Language / Graphical User Interface Toolkit). http://www.tcl.tk

[102] Treo 600. http://en.wikipedia.org/wiki/Treo_600

[103] T.120 Standards for Audiographic Teleconferencing.

[104] Virtual Rooms Video Conferencing systems (2003), http://www.vrvs.org

[105] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, and S. Tuecke, GridFTP: Protocol Extensions to FTP for the Grid, Argonne National Laboratory, April 2002.

[106] Wenjun Wu, Geoffrey Fox, Hasan Bulut, Ahmet Uyar, Harun Altay. "Design and Implementation of a collaboration Web-services system", Special issue on Grid

computing in Journal of Neural Parallel and Scientific Computations (NPSC), Volume 12, pages 391-408 (2004).

[107] Wi-Fi (wireless fidelity). http://en.wikipedia.org/wiki/Wi-Fi

[108] Window Meeting Space. http://en.wikipedia.org/wiki/Windows_Meeting_Space

[109] Xiaohong Qiu, Bryan Carpenter, Geoffrey Fox, Collaborative SVG as a Web Service, SVG Open 2003 Conference and Exhibition, Vancouver, Canada, July 2003.

[110] Xiaohong Qiu. "Message-based MVC Architecture for Distributed and Desktop Applications" Syracuse University PhD March 2 2005.

[111] XMPP (Extensible Messaging and Presence Protocol). http://www.xmpp.org

[112] Yao, W., Moody, K., and Bacon, J. A Model of Oasis Role-Based Access Control and Its Support for Active Security. In ACM Symposium on Access Control Model and Technology, ACM. Chantilly, VA.

Vita

Name of Author:	Kangseok Kim
Place of Birth:	Seoul, KOREA
Degrees Awarded:	
November 2007	Ph.D. in Computer Science
	Indiana University,
	Bloomington, IN, U.S.A.
June 1998	M.S. in Computer Engineering
	Syracuse University,
	Syracuse, NY, U.S.A.
February 1989	B.S. in Mathematics
	Kyungwon University,
	Sungnam, KOREA
Experience:	
AUGUST 2001 – September 2007	Graduate Research Assistant
	Community Grids Lab (CGL),
	Indiana University,
	Bloomington, IN, U.S.A.
May 2000 – AUGUST 2001	Graduate Research Assistant
	School of Computational Science & Information
	Technology (CSIT),

	Florida State University, Tallahassee, FL, U.S.A.
AUGUST 1998 – May 2000	Graduate Research Assistant Northeast Parallel Architectures Center (NPAC), Syracuse University, Syracuse, NY, U.S.A.