XGSP-Floor: Floor Control for Synchronous and Ubiquitous Collaboration

Kangseok Kim[1, 2], Wenjun Wu[5], Geoffrey C. Fox[1, 2, 3, 4]

1 Community Grids Laboratory, Indiana University, Bloomington, IN 47404, USA

2 Department of Computer Science, Indiana University, Bloomington, IN 47404, USA

3 School of Informatics, Indiana University, Bloomington, IN 47404, USA

4 Physics Department, College of Arts and Sciences, Indiana University, IN 47404, USA

5 Computation Institute, University of Chicago, Chicago, IL 60637, USA

kakim@indiana.edu, wwj@uchicago.edu, gcf@indiana.edu

Abstract— **With the advances in 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, mechanisms for dealing with consistency in an application shared among collaborators will have to be considered in an unambiguous manner. In this paper we address 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 briefly present a framework built on heterogeneous (wire, wireless) computing environment and a set of session protocols defined in XML to provide a generic solution for controlling sessions and participants' presences in collaboration, developed by Global-MMCS (Global Multimedia Collaboration System) project in Community Grids Lab (CGL) at Indiana University. Second, 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.**

1. INTRODUCTION

Collaboration is about interaction among people and between people and resources. With the advances in 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 [46] made ubiquitous access possible with the integration of wireless communication technology in anytime and in anywhere. With grid computing paradigm [15, 25, 26] which is about sharing resources, resources are distributed into workspaces and shared among geographically dispersed collaborators. With pervasive computing paradigm [13, 41], 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 have seen the improvement of computing performance, the increase of network bandwidth, and the advance of wireless networking. We believe from Moore's law [39] 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 [41] more promising as well in the future.

The following scenario illustrates the needs of ubiquitous collaboration and access, and motivates the research issues described in this paper. Researchers in Community Grids Lab (CGL) [7] 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) [17, 20]. 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. 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.

This paper is organized as follows. Section 2 presents research issues and our solution. We discuss related works in Section 3. Section 4 discusses considerations for a selection of a floor control for shared whiteboard application in heterogeneous community collaboration, synchronous and ubiquitous collaboration domains. Section 5 briefly presents the architecture of collaboration framework and the implementation of it, and describes XML based General Session Protocol (XGSP). Section 6 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 with a brief discussion of future work.

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 [24, 42, 48, 49]. 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 [44], SIP (Session Initiation Protocol) [30, 33, 43], 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. 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 [8, 9] 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 may be 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.

3. RELATED WORKS

In this section we examine existing floor control schemes for collaboration system. Dommel [9, 10] 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 [32]). 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 [37]) and floor control protocol built on MBone seminars [34]. 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 [4] 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 [21, 22] 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 [22, 35] 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.

Dommel [11] 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 [32]). McKinlay [38] 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 [6] 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 [5] which studies the use of PDAs (Personal Digital Assistants) simultaneously with a desktop PC.

4. CONSIDERATIONS FOR A SELECTION OF A FLOOR CONTROL FOR SHARED WHITEBOARD APPLICATION IN HETEROGENEOUS COMMUNITY COLLABORATION, SYNCHRONOUS AND UBIQUITOUS COLLABORATION DOMAINS

In this section 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. 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 3, 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 [23, 45]. 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 [9]. An example is Tivoli 1.0 which is shared whiteboard system designed for supporting small group size meetings in collocated place [45]. 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 [40].
- 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 [12]. 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 [29] 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 [23]. 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 [6]. 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 [23]. 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 [23]. Also, the overheads incurred from the computational complexity of some optimistic schemes may have to be considered [23]. For example, many different floor control mechanisms for shared whiteboard application in large group size can be considered using the mechanisms recommended by Dommel [9] – 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.

4.1. 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 [29] 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 [29]. 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.

4.2. 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, 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. 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 paper 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.

5. COLLABORATION FRAMEWORK ARCHITECTURE AND XML BASED GENERAL SESSION PROTOCOL (XGSP)

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 our messaging and service middleware. A set of protocols are defined in section 5.7 for maintaining consistent collaboration state information among participants in conference collaboration.

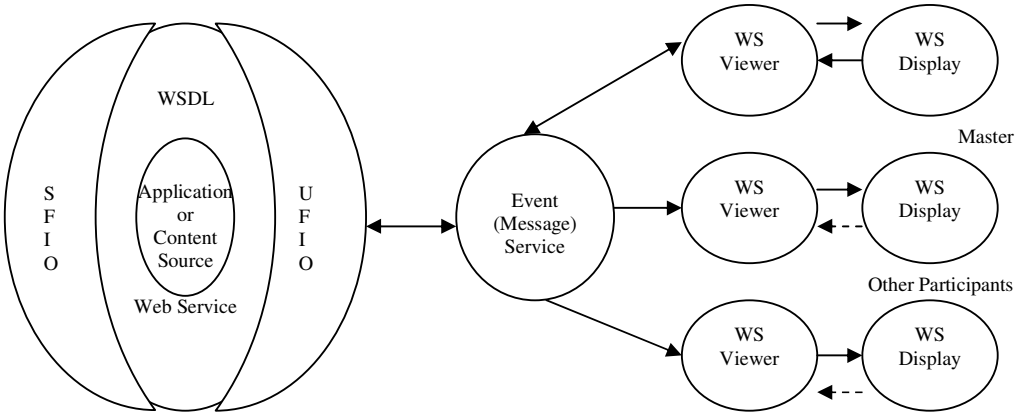

Figure 1: Shared Output Port Model (also called SMMV (Single Model Multiple View) Collaboration Model). UFIO and SFIO mean User Facing and Service Facing Input/Output Ports respectively.

Figure 1 shows SMMV (Single Model Multiple View) collaboration model (also called shared output port model) [16, 19, 56]. 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, 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.

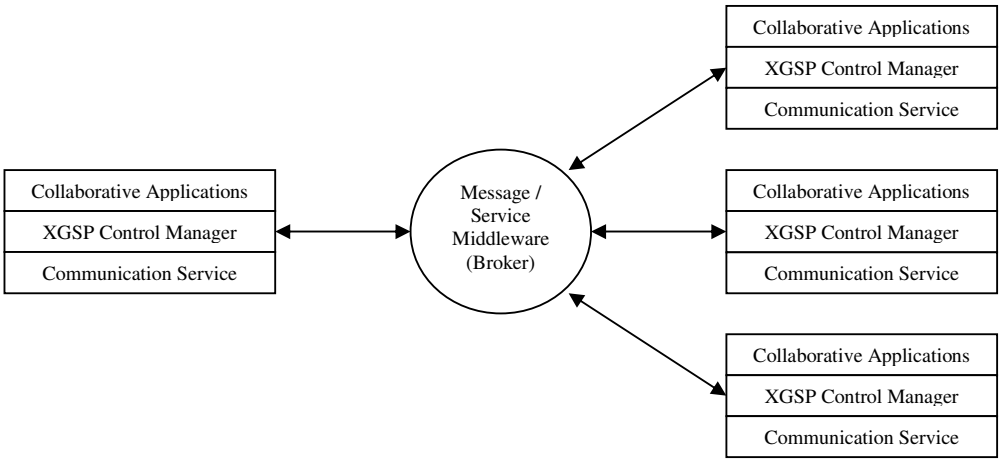

Figure 2: XGSP Service Architecture.

As shown in Figure 3, 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.

5.1. 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.

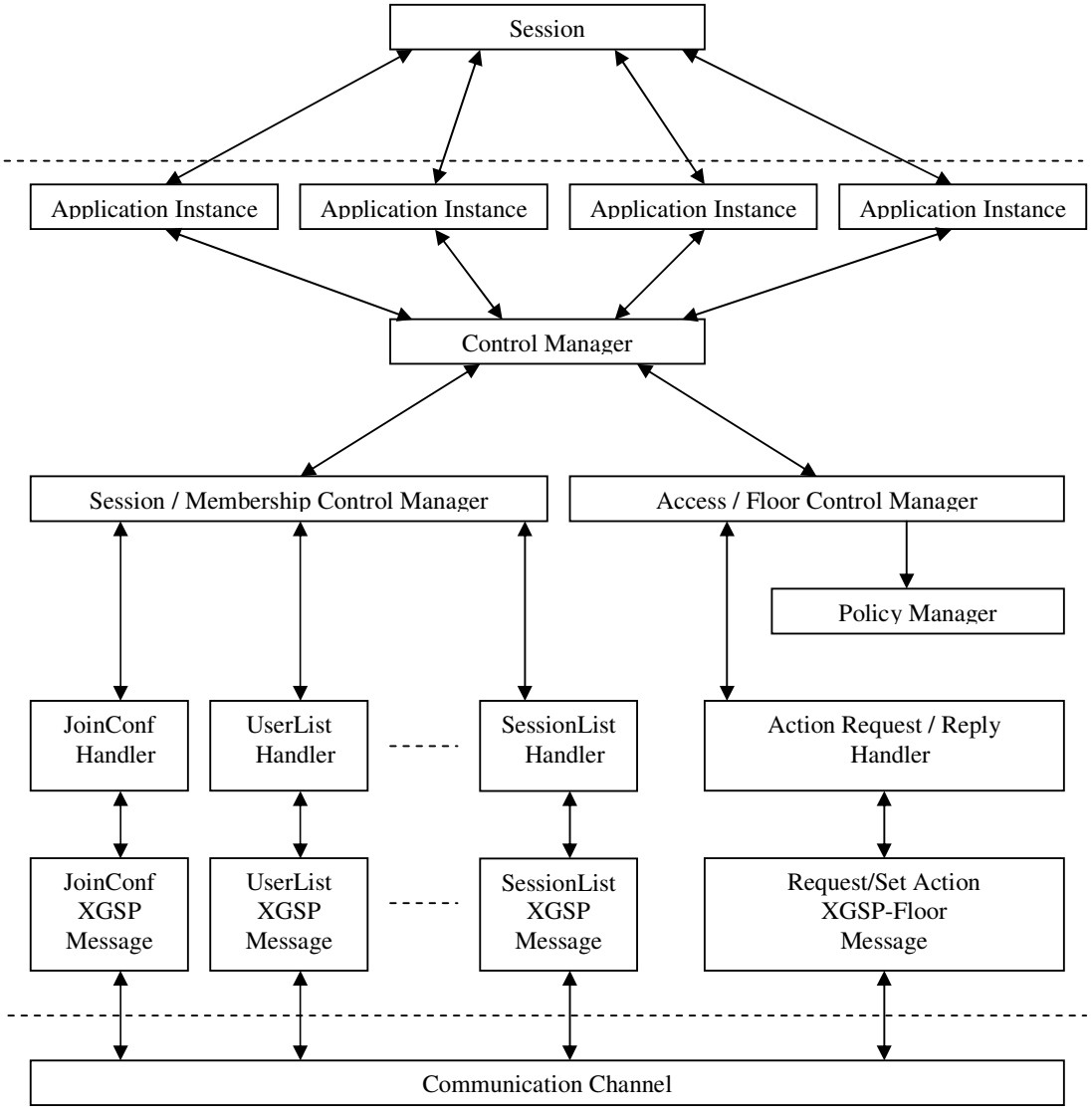

Figure 3: Collaboration framework architecture consists of three layers (collaborative applications, managers, and communication service) and six major components.

5.2. 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) [53] is described in section 5.7 in more detail.

5.3. Access and Floor Control Manager

The access and floor control manager component in the 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 Floor control mechanism (XGSP-Floor) is described in section 6 in more detail.

5.4. Policy Manager

Access and floor control policies are written in XML and put into the conference manager which resides on web server running on tomcat 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.

5.5. 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 (messaging and service middleware) via the communication channel shown in Figure 3. The broker disseminates the messages to other participants connected to the collaborating workgroup (session).

5.6. 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.

5.7. XML based General Session Protocol (XGSP)

Collaboration can be defined as interaction for cooperation on shared resources among 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) [53] 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 briefly describe conference, session, and presence management in turn with our session control protocol - XGSP.

5.7.1. XGSP Conference Management

The conference manager, which resides on web server running on tomcat, manages information related to all the conferences. 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 Global-MMCS system [17, 20] 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.

5.7.2. 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 [28]. 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 [18]. 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 4 and 5 respectively. If a chairperson creates a new application session, the session information is broadcasted through a broker as shown in Figure 4. But if a chairperson destroys a session, the old session information is broadcasted as shown in Figure 5. 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"?>
<CreateAppSession>
<ConferenceID>ourtestroom</ConferenceID>
<ApplicationID>wb</ApplicationID>
<SessionID>NewAppSession</SessionID>
<Creator>kskim</Creator>
</CreateAppSession>
```

Figure 4: XML Stream for Creating an Application Session.

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

Figure 5: 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 6. 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 7 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 6: XML Stream for Joining a session NewAppSession

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

Figure 7: XML Stream for Leaving a session NewAppSession

5.7.3. *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 8 and 9.

- 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 8 and 9 we show an example join-conference XML stream for a chairperson and mobile users with role names chairperson and mobile-user respectively.

```
<?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 8: 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>mobile-user</RoleName>
<UserID>kskim</UserID>
<UserName>kangseok-kim</UserName>
</User>
</JoinConf>
```

Figure 9: Join XML stream of Mobile 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 10.

```
<?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 10: 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 11 shows a presence request XML stream and Figure 12 shows a session request XML stream.

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

Figure 11: Presence request XML stream

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

Figure 12: Session request XML stream

- A chairperson node informs users' presence list and an available session list. Figure 13 shows a presence reply XML stream and Figure 14 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 13: 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 14: Session reply XML stream

- When a participant leaves a conference, her leave-conference XGSP event message is disseminated in the XML stream of Figure 15. 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 15: Leave Conference XML stream

6. 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.

6.1. 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. An example XGSP-Floor policy written in XML is shown in Figure 16. 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 6.3, 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 [51, 55, 56] 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 in XML and is implemented by the XGSP-Floor mechanism which is described in next section 6.2.

```
<XGSP-FloorPolicy>
 <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-FloorPolicy>
```

Figure 16: An Example of XGSP-Floor Policy with the Role Name mobile-user and Application Name whiteboard (wb).

6.1.1. Collaboration Role in XGSP-Floor

Collaboration roles in XGSP-Floor 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-Floor are represented. For the representation we use functional notion to show the relationship between roles and action privileges.

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 17. 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

6.1.2. Fine-grained action (Interactive smallest Major event)

We define fine-grained actions in our collaborative application as the smallest interactive major events (semantic events [56]). 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 [47]) browser and a collaborative chess game application with SVG [51, 55, 56]. 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.

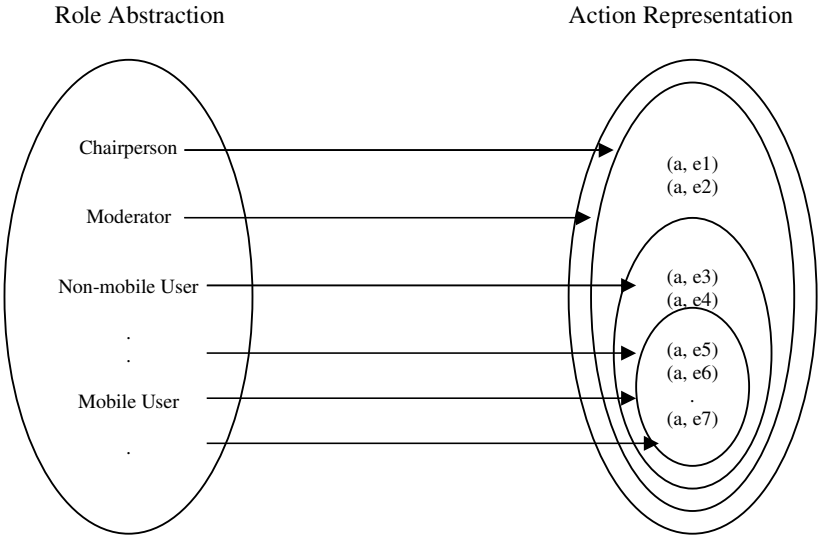

Figure 17: A collaboration action is represented as a pair (a, e) ∈ A x E, where a ∈ A is an application and e ∈ E is the authorized smallest major event defined by a, and A is a set of applications, E is a set of the smallest major events defined by an application in A.

6.2. 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 18. Also, we show a request-response interaction scheme between a moderator and a floor requester with human-computer interaction in Figure 19.

6.2.1. Decision Procedures of XGSP-Floor Mechanism

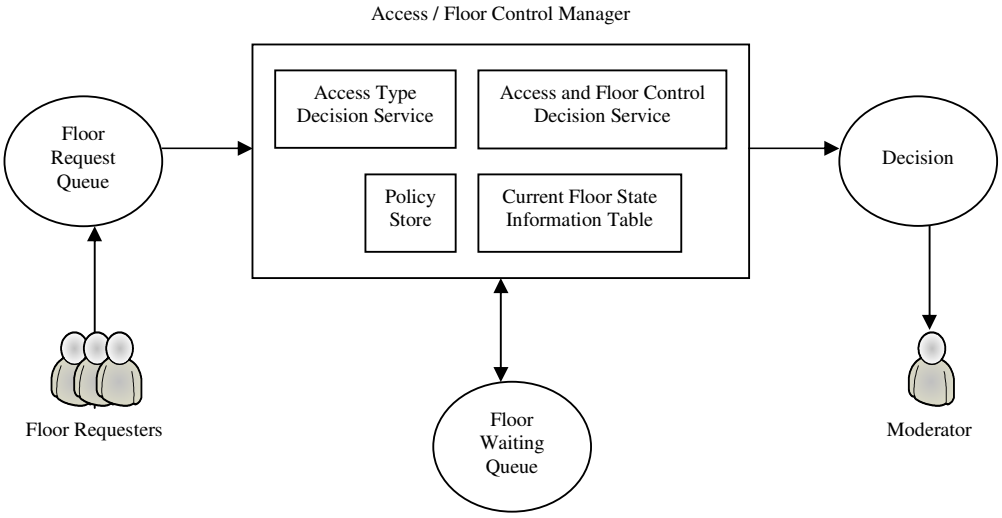

Figure 18: Decision Procedure of XGSP-Floor Mechanism

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.

6.2.2. 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 3 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 19. 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.

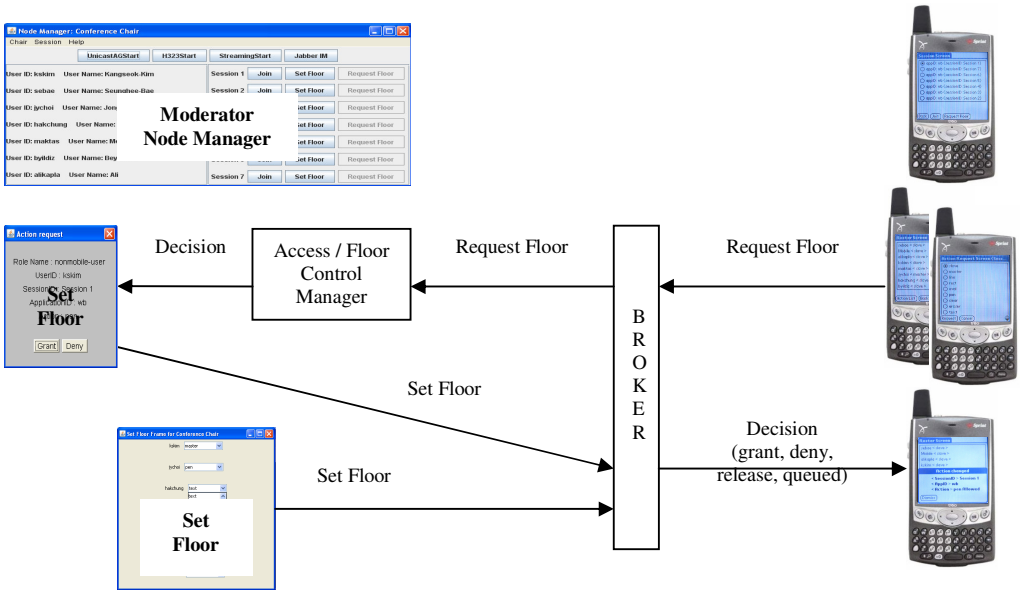

Figure 19: Request-Response Interaction Scheme between a Moderator and a Floor Requester with Human-Computer Interaction.

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 20 depicts the instance of the two-player turn-taking mechanism for the collaborative chess game application used in our collaboration.

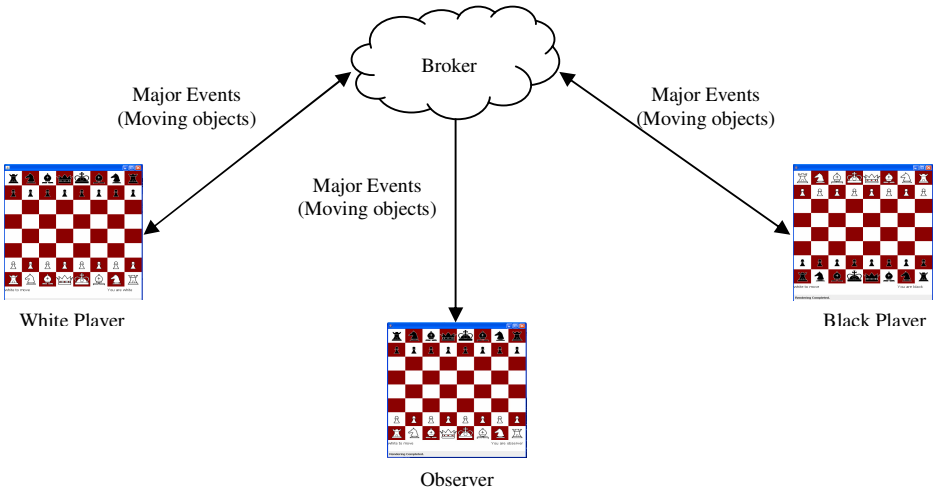

Figure 20: Two-player Turn-taking Mechanism for Chess Game Application

6.3. 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 21 shows a node manager of a moderator on desktop. Figure 22 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 21. The pop-up window frame is shown in the left figure of Figure 23. Participants can request floors in the window frame invoked by clicking on the "Request Floor" button in Figure 22. The pop-up window frame for the request floor is shown in the right figure of Figure 23. A moderator can control floors of all the participants joined in a session via the window frame shown in the left figure of Figure 23 while participants can request only their own floors via the window frame shown in the right figure of Figure 23 but see current floor states of other participants.

Figure 21: Node Manager for a Moderator on Desktop

Figure 22: Node Manager for Normal Users on Desktop

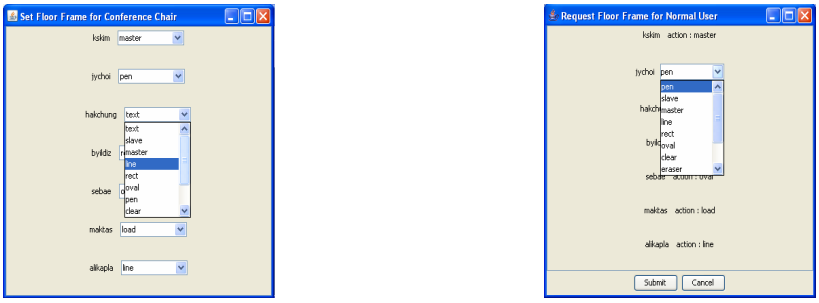

Figure 23: Set Floor Frame for a Moderator vs. Request Floor Frame for a Normal User

Figure 24: Node Manager for Normal Users on Cell Phone

Figure 24 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 24 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 [3] 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 25 shows the request screen for a floor. The screen is displayed from selecting the "Request Floor" button shown in the right figure of Figure 24. Figure 26 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 27 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.

Figure 25: Request Floor Screen on Cell Phone

Figure 26: Pop-up Window Frames (for Floor Request, Floor Grant, Floor Deny, Floor Conflict, and Floor Queued Notifications respectively) on Desktop

Figure 27: 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.

6.4. 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 18 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) [50] 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 [14] 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 [9].

6.5. Locking of XGSP-Floor Mechanism

Locking [23] is a method of gaining privileged access to shared resource for some amounts of duration. In this section we show non-optimistic locking mechanism [23] 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 [36], where the fine-grained lock means the locking of the major event described in section 6.1.2. 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 28. 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 29.

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.

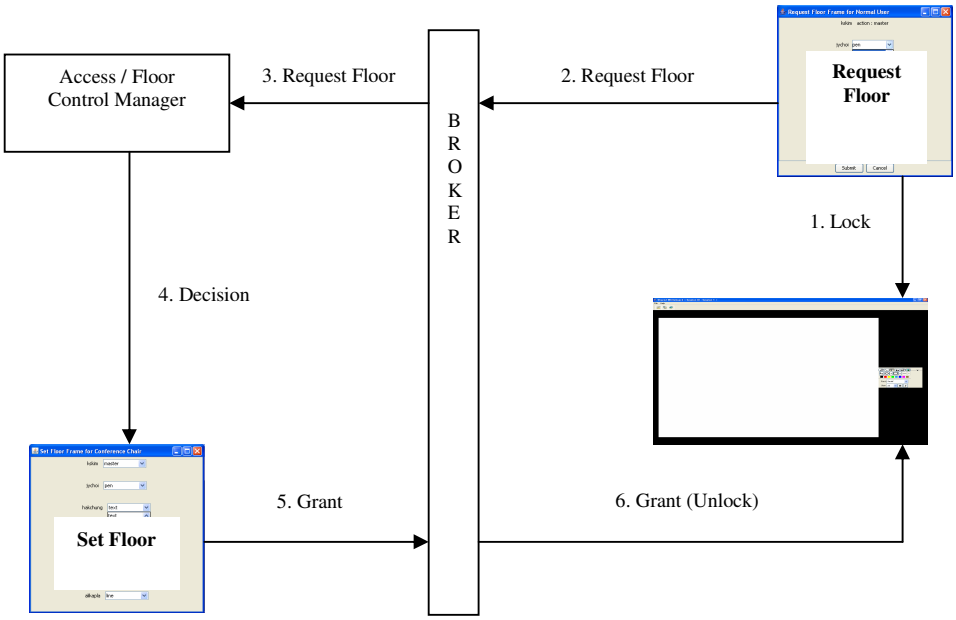

Figure 28: Locking Mechanism of Shared Whiteboard

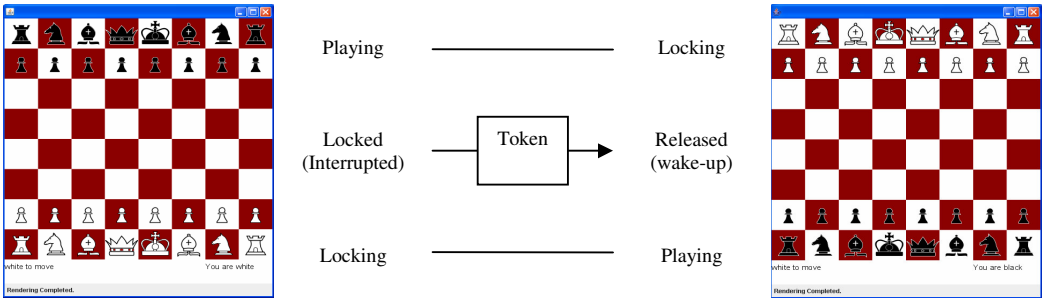

Figure 29: Locking Mechanism by Logical Token-passing in Chess Game

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, especially with cell phone device involved in collaboration since the wireless network transit time is in the range of seconds [31]. 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.

7. SUMMARY AND FUTURE WORK

In this paper we have attempted to provide a virtual workspace for not only remotely dispersed users but also roaming users with cell phone devices – Palm OS 5.2.1H Powered Treo600 [52]. 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.

In this paper, we also 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.

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) [54] 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 3G [27] which is multimedia and Internet-enabled mobile phone.

REFERENCES

[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] 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.

[4] 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.
[5] Brad A. Myers, et al. “Using Hand-Held Devices and PCs Together”. ACM Communications, 2001. To appear.
[6] 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".
[7] Community Grids Lab (CGL), http://communitygrids.iu.edu
[8] 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.
[9] 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.
[10] 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.
[11] 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.
[12] 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.
[13] 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.
[14] 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.
[15] F. Berman, G. Fox, and A. Hey, editors. Grid Computing: Making the Global Infrastructure a Reality, John Wiley & Sons, 2003.
[16] 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).
[17] 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.
[18] 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.
[19] 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.
[20] Global-MMCS (Global Multimedia Collaboration System). http://www.globalmmcs.org
[21] 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.
[22] 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.
[23] 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.
[24] 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.
[25] I. Foster and C. Kesselman, editors. The Grid: Blueprint for a New Computing Infrastructure, Morgan-Kaufman, 1998.
[26] 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.
[27] iPhone. http://en.wikipedia.org/wiki/IPhone
[28] Jabber Instant Messenger, http://jabber.org
[29] 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.
[30] J. Rosenberg et al. (2002) “SIP: Session Initiation Protocol”, RFC 3261, Internet Engineering Task Force, http://www.ietf.org/rfc/rfc3261.txt

[31] Kangseok Kim. "A Framework for Synchronous and Ubiquitous Collaboration" Indiana University PhD September 2007.
[32] Katrinis K., Parissidis G., and Plattner B., "Activity Sensing Floor Control in Multimedia Collaborative Applications", 10th International Conference on Distributed Multimedia Systems (DMS 2004).
[33] Koskelainen P., Schulzrinne H. and Wu X.(2002), A SIP-based Conference Control Framework, NOSSDAV'02, May 12-14, 2002, Miami Beach, Florida, USA.
[34] 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.
[35] 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.
[36] Matt Bishop. Introduction to Computer Security. Addison Wesley.
[37] 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.
[38] 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.
[39] Moore's Law. http://en.wikipedia.org/wiki/Moore's_Law
[40] 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.
[41] M. Weiser. "The Computer for the Twenty-First Century," Scientific American, September 1991.
[42] NaradaBrokering, http://www.naradabrokering.org
[43] NIST SIP (2001), http://snad.ncsl.nist.gov/proj/iptel
[44] OpenH323 Project (2001), http://www.openh323.org
[45] 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.
[46] R, B'Far. Mobile Computing Principles: Designing and Developing Mobile Applications with UML and XML, Cambridge University Press 2005.
[47] Scalable Vector Graphics (SVG). http://www.w3.org/Graphics/SVG/
[48] 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.
[49] 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.
[50] 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.
[51] SVGArena. http://www.svgarena.org/
[52] Treo 600. http://en.wikipedia.org/wiki/Treo_600
[53] 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).
[54] Wi-Fi (wireless fidelity). http://en.wikipedia.org/wiki/Wi-Fi
[55] Xiaohong Qiu, Bryan Carpenter, Geoffrey Fox, Collaborative SVG as a Web Service, SVG Open 2003 Conference and Exhibition, Vancouver, Canada, July 2003.
[56] Xiaohong Qiu. "Message-based MVC Architecture for Distributed and Desktop Applications" Syracuse University PhD March 2 2005.