2004

Design and Implementation of a Collaboration Web-Services System

Wenjun Wu
Indiana University, Community Grid Labs

Geoffrey C. Fox
Indiana University, Community Grid Labs

Hasan Bulut
Indiana University, Community Grid Labs

Ahmet Uyar
Syracuse University, Department of Electrical Engineering and Computer Science

Harun Altay
Indiana University, Community Grid Labs

Design and Implementation of a collaboration Web-services system

Wenjun Wu[1] Geoffrey Fox[1] Hasan Bulut[1] Ahmet Uyar[2] Harun Altay[1]
[1]Community Grid Computing Laboratory, Indiana University
Indiana Univ Research Park, 501 North Morton Street, Suite 222, Bloomington, IN47404
[2]Department of Electrical Engineering and Computer Science, Syracuse University

Abstract
Conference control has been studied for years but most researches focus on homogenous collaboration. There is no conference control framework for integration of multiple collaboration systems such as H.323, SIP and AccessGrid. In this paper we present a web-services based scalable conference control framework for such a heterogeneous collaboration system. Based on this framework, we implemented a prototype system to verify and refine our framework. This system can support various conferencing endpoints and communities.

Keywords
Web-Services, Collaboration, Global-MMCS, XGSP, NaradaBrokering

1. Introduction
Collaboration and videoconferencing systems have become a very important application in the Internet. There are various solutions to such multimedia communication applications, among which H.323 [1], SIP [2], and Access Grid [3] are well-known. It will bring substantial benefits to Internet users if we can build an integrated collaboration environment, which combines these systems into a single easy-to-use, intuitive environment. However, at present they have features that sometimes can be compared but often they make implicit architecture and implementation assumptions that hamper interoperability and functionality. Therefore it is very important to create a more general framework to cover the wide range of collaboration solutions and enable users from different communities to collaborate. In this paper, we present such a common, interoperable framework based on Web services technology for creating and controlling multipoint video & audio collaborations. And we also introduce the implementation of the collaboration system based on this framework.

The paper is organized in the following way: Section 2 introduces related work and our research issues. Section 3 describes the XGSP (XML based General Session Protocol) conference control framework. Section 4 presents the implementation of Global Multimedia Collaboration System (Global MMCS). And we give the conclusion and future work in section 5.

2. Related Work and Problem Statement
The multimedia collaboration framework has been studied over years. The well-known solutions have H.323, SIP and IETF MMUSIC [4]. The IETF's Multi-Party Multimedia (MMUSIC) working group proposed its own solution SCCP (Simple Conference Control Protocol) [5]. However, its main target was lightweight conference management for multicast instead of tightly controlled models. Because multicast can't be deployed widely in the Internet in near future, in the year 2000 MMUSIC WG gave up and removed conference control from the WG charter. The project Access Grid started from the MBONE tools: VIC and RAT, and is also trying to define its own conference control framework rather than SCCP.

H.323 is a communication standard produced by the ITU, initiated in late 1996, and aimed at the emerging area of multimedia communication over LANs. It is an outgrowth of the traditional H.320 technology but optimized instead for the Internet. H.323 is widely supported by many commercial vendors and used throughout the world in commercial and educational markets.

H.323 is defined as an umbrella standard specifying the components to be used within an H.323-based environment. It provides conference management functionality for audio/video conferences using the call signaling functionality of H.225 [6], H.245 [7]. These protocols provide call set-up and call transfer of real-time connections to support small-scale multipoint conferences. The protocol H.243 [8] concerns the system operation for a conference call between three or more audiovisual terminals. It defines some commands between the MCU and H.320 terminals to implement audio mixing, video switch and cascading MCU. H.243 commands have been included in H.245.

For the data part of a conference, the conference management of the T.120 recommendation [9] is used. This standard contains a series of communication and application protocols and services that provide support for real-time, multi-point data communications. The multi-point facilities are important building blocks for a whole new range of collaborative applications including desktop data conferencing, multi-user applications, and multi-player gaming.

The Session Initiation Protocol (SIP) defines how to establish, maintain and terminate Internet sessions including multimedia conferences. Initially SIP was designed to solve problems for IP telephony. To this end, SIP provides basic functions including: user location resolution, capability negotiation, and call management. All the capabilities are basically equivalent to the service H.225 and H.245 in H.323 protocol. The major difference is that SIP was designed in a text format and took request-response protocol style like HTTP. But H.225 and H.245 were defined in a binary format and kept a style of OSI (Open System Interconnection). Therefore SIP has some advantages of interaction with web protocols like HTTP in VoIP industry.

More importantly, SIP doesn't define the conference control procedure like H.243 and T.120. Additional conference control mechanisms have to be implemented on the base of SIP to support the A/V and data collaboration. Recently SIP research group begun to develop their framework and produced a few drafts [9,10]. But SIP work is still in the beginning phase and has not been widely accepted.

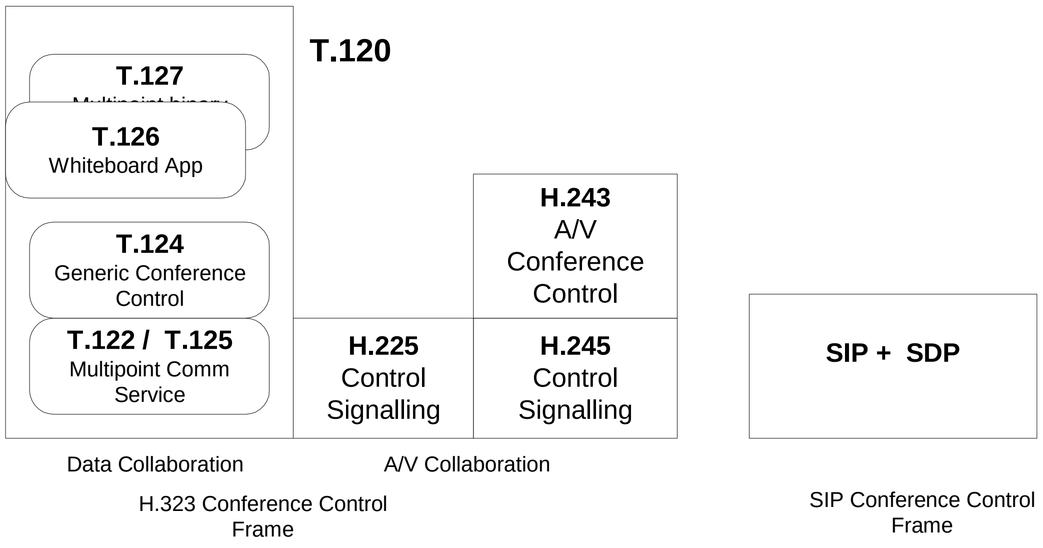


Figure 1 H.323, T.120 and SIP protocol stack

Figure 1 compares the H.323, T.120 and SIP protocol stacks. T.120 defines very complete framework for general data conferencing. At the bottom of the T.120 protocol stack, T.122 [11] defines the multi-point services available to the developer, while T.125 [12] specifies the data transmission protocol. Together they form MCS (Multipoint Communication Service) which become an "overlay" network for any multi-point application.

Based on this "overlay" multipoint communication network, T.124 [13] (Generic Conference Control) provides a comprehensive set of facilities for establishing and managing the multi-point conference. It maintains the state information about the nodes and applications that are in a conference. Using mechanisms in T.124, applications create conferences, join conferences, and invite others to conferences. T.126 [14] and T.127 [15] are whiteboard application and multipoint file transfer application based on T.124.

For the tightly coupled conferencing, T.120 provides a good general architecture which clearly defines the functions and services of each layer. And it should be able to support many collaboration application including audio and video. However in H.323, T.120 is completely independent of H.225 and H.245. In fact, A/V and data collaborations should be integrated in the same framework so that the architecture can be easily implemented and maintained.

To build more advanced and integrated collaboration systems, neither H.323/T.120 nor SIP is sufficient:

(1) SIP has very limited supported for conference control.
(2) In H.323 framework, A/V collaboration and T.120 are not well integrated. Moreover, the A/V communication services and T.122 overlay networks don't have very good scalability.
(3) H.323 and T.120 are designed in a relative complicated OSI model. It is not easy to understand and develop in their APIs.
(4) All these frameworks only deal with homogenous video conferencing and can't connect to other collaboration systems.

Recently, many new technologies in the Internet such as XML, SOAP, Web-Service, Publish /Subscribe messaging as well as peer-to-peer computing have emerged and started to change the Internet applications. These new technologies enable the new architecture for collaboration systems:

(1) A unified, scalable, robust "overlay" network is needed to support A/V and data group communication over heterogeneous networking environments. Centralized conferencing systems usually depend upon a single conference server for group communication purpose. Distributed conferencing systems take IP multicast. For example Access Grid uses Internet2 multicast for audio/video transmission. Both of the services have some limitations. Centralized conferencing systems don't have good scalability. And it is not easy to deploy distributed conferencing systems on current Internet because IP multicast has not become ubiquitously available. Therefore such an overlay will fundamentally solve the issue of scalability and deployment under the Internet environment.

(2) A common A/V signaling protocol has to be designed to support interactions between different A/V collaboration endpoints. For example, in order to get the H.323, SIP and MBONE endpoints to work in the same A/V session, we have to translate their signaling procedures into our common procedure and build the collaboration session. Just like the text messages in SIP, XML should be used to describe the common signaling protocol because it makes the protocol easier to be understood and to interact with other Web based components.

(3) A core conference control mechanism is required for establishing and managing the multi-point conference. The service of this part is quite like T.124 (Generic Conference Control). However this mechanism will provide more flexible facilities to describe application sessions and entities. And it can be designed in a more scalable approach based on the powerful publish/subscribe messaging services. All the description information for the applications and sessions can be kept in XML format rather than binary format, which will lead to easier development. Furthermore most control messages can be transferred through messaging middleware rather than central servers and the most session information can be distributed in all the participating nodes.

(4) Finally, we'd like to use web-services to integrate collaboration communities in different technologies. Various collaboration systems including Access Grid, H.323 and SIP should be regarded as Web-services components and provide Web-services interface of their conference control protocols to the core conference control components. They can invoke these services to build an integrated community-to-community conference across the communities.

There have been some projects like VRVS [16] that can also provide some kind of integration of different A/V endpoints. But VRVS is not an open project having few documents for their architecture and conference control framework. From the introduction in its web site, we can make some comments: VRVS builds its collaboration service on top of pure-software reflector infrastructure which is a kind of software multicast. It is capable of supporting MBONE tools, H.323 terminal as well as QuickTime player. It also supports some data sharing collaborations, like shared web browsing and shared desktop (VNC). However it doesn't seem to have floor control mechanism, support community-to-community collaboration as well as many important data collaborations.

The following table gives a comparison between XGSP and other frameworks. Although the SIP and Access Grid are trying to add the conference control mechanism, their frameworks haven't been well defined. So we make this comparison according to the current capabilities of their systems.

	H.323, T.120	**SIP**	**IETF Access Grid**	**VRVS**	**XGSP**
Conference Management	supported	No	supported	supported	supported

Overlay Network Environment	Internet / ISDN Firewall transversal under the support of VPN	No	High performance with multicast support , No firewall tunneling	Reflector Infrastructure Software Multicast	Publish/Subscribe with Firewall transversal (VPN optional)
Data Collaboration	Limited: T.120 whiteboard, File FTP	No	Limited to (PowerPoint, Chat)	Limited to (Shared browsing and VNC)	allows full integration of all tools
Floor Control Mechanism	H.243 T.120	No (Under development)	No	No	Chairman based Flexible role setting
Scalability	Not good	Not good	Good	Good	Good
Support heterogeneous clients	No	No	No	H.323, MBONE	H.323, SIP, MBONE
Community-To-Community Collaboration	No	No	No	No	Yes

Table 1 Comparison of XGSP with Competitive Framework

3. XGSP Design

3.1 XGSP Architecture

In our XGSP framework, we use the messaging middleware for the "overlay" over heterogeneous networks to support publish/subscribe communications. NaradaBrokering [17] from the Community Grid Labs is adapted as a general event brokering middleware, which supports publish-subscribe messaging models with a dynamic collection of brokers and provide services for TCP, UDP, Multicast, SSL and raw RTP clients. Also NaradaBrokering provides the capability of the communication through firewalls and proxies. It can operate either in a client-server mode like JMS or in a completely distributed JXTA-like peer-to-peer mode. JMS API is very good for developing scalable collaboration applications. The publish/subscribe interaction paradigm make it possible to build a peer-to-peer and loosely coupled distributed system. And publish/subscribe topics, which represent keywords for publisher and subscriber, can be used to describe hierarchy and complicated collaboration groups. Built upon the service, one of the important issues for XGSP is the organization of the topic name space for conference control purpose. It will be introduced in Sec 3.3.

Figure 2 shows the important components in XGSP framework:

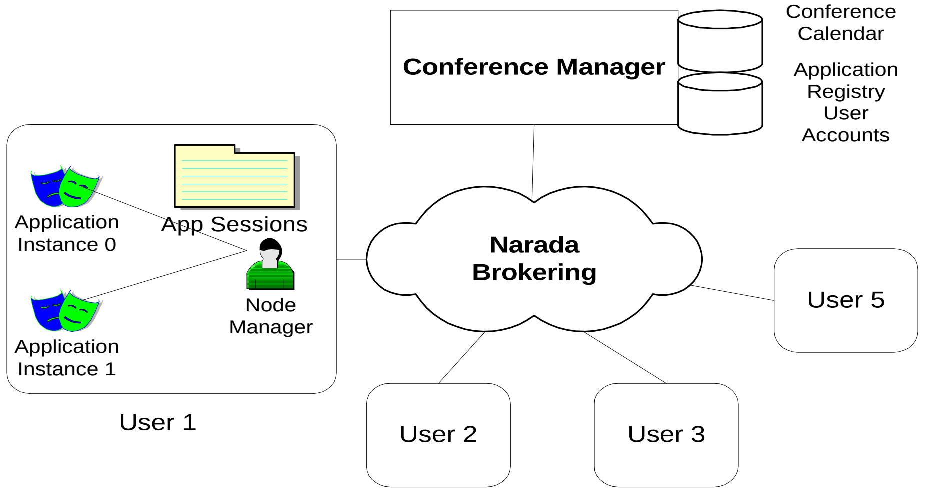


Figure 2 XGSP Conference Control Architecture

The conference manager is the server keeping the important information for all the conferences. Through the manager, users can create a new conference or terminate an old one. The meta-data repositories in the conference manager includes: a conference description set, application registry set as well as user accounts. The conference description set contains the registries of all the scheduled conferences. Each conference registry includes the fields: Conference ID, Conference Name, Conference Mode, Conference Time, and Available Application Session Templates. The application registry set has all the registries of the collaboration application such as A/V, chat and whiteboard. An application registry usually contains the entries like application identification, role systems definition as well as specific attributes.

A node manager is the user interface for the XGSP conference management service in each user. An application instance refers to a client of the collaboration applications. Because a node manager has the factories for all kinds of applications, it can create new application instances, and invoke start, stop, and set-role methods in them.

The XGSP conference control includes three services: conference management, application session management and floor control. The conference management supports user sign-in, user create/terminate/join/leave/invite-into XGSP conferences. The application session management provides users with the service for creating/terminating application sessions. And the floor control manages the access to shared collaboration resources in different application sessions. Through Section 3.2 to 3.4, we discuss them in detail. And an example is presented in Appendix A.

3.2 XGSP Conference Management

These services have the methods of Create / Modify / Terminate Conference, allowing users to make meeting schedules and look up active meetings.

(1) Conference Schedule

The conference manager offers a meeting calendar object listing all the meetings requested by users. It can be easily implemented in web pages. Users can make meeting reservations via their browsers or emails. The conference manager can approve or deny the requests of the users according to the capability of conference servers.

Each conference schedule record keeps the description of the conference, including the starting and end time. The conference manager will activate the conference at the starting time and de-activate it at the ending time. Some persistent conferences which have permanent schedule records have to be supported for the purpose of testing and management.

A user can have two ways to know how many meetings have been scheduled and active. He can either search the web conference calendar or send a query message to the conference manager.

(2) Users in a XGSP Conference

After the XGSP conference is activated, XGSP users can join this conference by starting their node managers. XGSP users can be divided into three categories: administrator, conference chairman and normal users. The administrator user is a very special user which can be regarded as a super user in the system. A chairman user usually has the power of creating application sessions and setting the roles of users in the conference.

(3) XGSP Conference membership maintenance

The session membership containing a list of the participants is shared by all the participants. Whenever there is some change in the membership, for example a new member joins in the session, the membership has to be updated and distributed to all the participants.

Users can send Join/Leave/InviteIntoConference messages to change the conference membership. Whenever there is a change in the membership, a membership event is generated and broadcasted to all the participants and the conference manager. Late-joining users have to send Request Membership messages to the manager server to get the whole view of the membership. Each user sends heart-beat messages during some period to all the peers in the conference. In this way, we implement a soft-state mechanism for the membership maintenance.

Each conference has a particular NaradaBroker topic for the control purpose. The topic name takes the form of "/xgsp/conferenceID/ctrl", in which conferenceID is the string identification for the conference.

3.3 XGSP Application Session Management

Application sessions refer to the groups of the collaboration application. Note that various collaboration applications may have quite different architectures even based on the same NaradaBroker messaging service. A shared-input port collaboration model is the peer-to-peer communication style without any centralized components, whereas a share-output port model needs some servers [18]. The XGSP application session management doesn’t make any assumption about how the applications are organized, but to support the core services for all kinds of applications.

(1) Different Application sessions in XGSP

There are three types of application sessions: public application sessions and private application session. A public session is open to all the users in the conference. All the participants in the conference can join the public application sessions. A private session is not open to all the users in the conference. And only invited users can join.

Default public application sessions refer to the sessions pre-defined in the conference description. For example, each XGSP conference usually has a default public A/V group and a public chat group. A non-default public application session has to be created by the conference chairman. The conference chairman can set the roles of the users in this public application session. He will terminate the application session when the session is over.

A private application session can be created by any member in the conference for private purpose. A user can join the private session only after he receives and accepts the invitation from the session initiator. The initiator user who is the conference chairman in the private session has the power of setting roles and closing the session.

(2) Topic name schema for application sessions

Each application session should have its own topic name space inside NaradaBrokering. We can define the naming schema: /xgsp/conferenceID/Application-Session-ID. The conferenceID field is generated by the conference manager and determined when the conference is activated. The Application-Session-ID field is generated when the application session is created. This field can have three kinds of forms: the default public session can use the application type identification like av, chat, whiteboard. The public application sessions take the format of < application type, sequence number >. The sequence number represents the last number of the application sessions. The private application sessions can be < application type, initiator-ID, sequence number>. In the following, we give a few examples: suppose the conference named ourtestroom is created. And it has two default application sessions with the topic names: /xgsp/ourtestroom/av and /xgsp/ourtestroom/chat. If the chairman in the conference creates two whiteboard sessions, their topic names should be: /xgsp/ourtestroom/whiteboard-0, /xgsp/ourtestroom/whiteboard-1. For a private whiteboard session initiated by the user with the user ID: “testuser”, its topic name should be /xgsp/ourtestroom/whiteboard-testuser-0.

(3) Procedures in creating and terminating application sessions

The procedures involve the node managers and the conference managers. Each node manager keeps a directory of the application sessions. It shows all the public sessions and visible private application sessions to users.

The node manager of the chairman user in the conference can send a “Create Application Session” message to all the node managers in the conference. Each node manager adds the application session information in this message into a local session directory. When the session is over, the chair node manager will send the message of “Terminate Application Session” to all the node managers. They will close the application instances which are active in the session and remove the session information from the local session directories.

Private sessions are created in the similar way like public sessions. Suppose a user A wants to create a private chat session. His node manager will send a “Create Application Session” message with the private flag set to all the node managers. Although it is written by each node manager into the local session directory, it will not be visible to users until the node manager receives an “Invite Into Application Session” from user A manager.

The conference manager monitors the life-cycles of the sessions. If the session has a session server, the manager commands the application session servers to work for the management of the session. Section 3.5 presents a detailed description how A/V application sessions are handled.

(4) Join/Leave Application Session

The application session directory is showed in the UI of the node manager. When a user wants to join this application session, he can select it from the directory. The application factories in the node manager will create an application instance and start it. During the initialization of the new application instance, the NaradaBroker topic name and the initial role are passed on to the application instance. It is up to the application instance to deal with the details of joining the application session.

The private application session is only showed in the UI after the “Invite Into Application Session” message for this session arrives. If the user accepts the invitation, the node manager will send the reply to this invitation and create a new application instance.

3.4 XGSP Floor Control

Conference applications often have shared resources such as the right to talk, a pointer or input focus in a shared application, access to shared lesson or game rooms. Floor control enables applications or users to gain safe and mutually exclusive or non-exclusive access to the shared object or resource. The policy of floor control usually highly depends upon the collaboration applications. Therefore it is not the job of XGSP to define the floor control policy for the applications. However, any conference can be divided into two modes: moderated and unmoderated. A moderated conference needs a chair to determine some "floors" for the collaboration applications. To support this general policy for floor control, XGSP takes a role-based approach and provides a chairman mechanism. Roles define the access right to shared collaboration resources in a very natural way. For example, in the chess game, we can have a white player, a black player and many observers.

Each collaborative application defines its role system in a XML registry. A role description includes the role name and the role capability. The conference manager keeps the database of all these definition registries. It copies the database to a user node manager when the user joins the conference.

The conference chairman has the right of setting the roles. It can send a "SetApplicationRole" message to the application instance running in other users. A "SetApplicationRole" message tells the conference participants which user should be changed to this role. All the application instances have to parse the message and take appropriate actions.

In a XGSP conference, only one chair should be present. It is important to keep the chairmanship and deal with the failure of the chair node. The following messages are introduced to implement this: RequestChair, RleaseChair, GiveChair and ChairAnnouncement. The initial chairman is defined in the conference description. Other users who want to become a chair, has to send a RequestChair message to the chair for permission. The chairman will send a "GiveChair" back to the requestors if he agrees to give the chair to the new one. "Chair Annoucement" is the heat-beat message for the chair user. If the chair node failed and can't recover, other users with the role capability of chair election have to vote for a new chair.

3.5 XGSP AV Application Session

XGSP A/V Session supports multiple kinds of clients including H.323, SIP and Access Grid, which use RTP protocol for their transportation. The NaradaBrokering event overlay provides messaging service to the A/V components. For the transportation of RTP packets over NaradaBroker overlay networks, RTP packets have to be encapsulated into a special NaradaBroker event named RTP event by RTPLinks components in border brokers before they can be routed in the overlay network. And the RTP events also have to be transformed back into normal RTP packets when they leave for their subscribers. Therefore for every legacy RTP A/V endpoints, one corresponding RTPLink needs to be set up at a broker within broker network.

In order to get better communication performance in the NaradaBroker overlay, a RTP event uses a 32-bit integer for topics rather than a string keyword. And each RTPLink occupies two NaradaBroker topics to publish its RTP and RTCP data, which means there are many topics for an A/V collaboration group. Therefore in XGSP AV application session management, we need a mapping between SSRC number of each A/V stream and integer topics generated by the NaradaBroker.

To support different A/V application endpoints having their own signaling procedures, XGSP framework provides a common XML based signaling protocol for them. H.323 (H.225, H.245) and SIP signaling protocols have to be translated into the XGSP A/V signaling protocol and vice versa.

A XGSP A/V session needs H.323, SIP Gateway Servers and an A/V Session Server to deal with the control layer problems of the A/V collaboration system. The H.323 and SIP gateway transform H323 and SIP messages into XGSP signaling messages so that H.323 and SIP A/V endpoints could communicate with the XGSP A/V session server. The session server implements session management logics including creating/destroying A/V sessions, allowing endpoints to join/leave session and make audio/video selection, managing A/V application components like audio mixers, video mixers as well as image grabbers.

Based on such a design of XGSP AV application session, when an AV session is created, the XGSP conference manager has to ask AV session server to activate the session. And when a user wants to join this AV session, his node manager will start the A/V application instance and the instance will contact the A/V session servers to join the session.

3.6 Connecting to other collaboration communities using web services

The above sections describe the core conference control for XGSP communities. In the section, we discuss about how to build an integrated collaboration system on the scale of community-to-community. Web-services seems to be the best candidate for this conference control framework since it can run across various platforms and is easy to be extended and understood. Figure 3 shows the architecture of the XGSP web-services framework. The overlay networks (or servers) in other communities are connected to the

NaradaBroker messaging overlay. And each community provides the state-aware web-services of conference control to the XGSP servers.

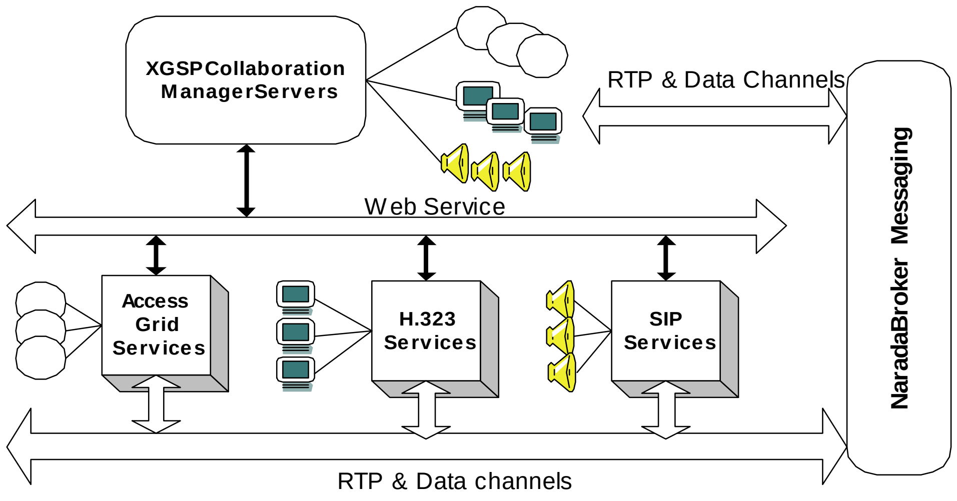


Figure 3 XGSP Web-Services Framework

In this community-to-community collaboration system, users can schedule XGSP conferences not only held in XGSP servers but also the servers from other technologies communities. Such a XGSP conference is organized in a hierarchy way: a top-XGSP-conference and multiple sub-XGSP-conferences happened in local communities. Local collaboration managers work like service factories, which can create a conference service instance to communicate with the XGSP conference manager through SOAP RPC [19]. Therefore the components including conference management, application sessions management and floor control can be implemented in this two-level control structure. The XGSP conference manager controls the top-XGSP-conference and the local manager instances only control sub-XGSP-conferences.

(1) Service Creation / Termination

The XGSP conference manager keeps the registries of the communities that would like to connect to the XGSP community. When a user schedules a conference including these communities, the XGSP conference manager has to invoke “Creation-Conference” requests in the manager factories of the local communities. If the request is successful, it will result in the creation of a conference manager instance with the initial lifetime for this conference. When the conference is over, the XGSP conference manager can invoke “Terminate-Conference” to destroy the local manager instance.

(2) Conference Membership Service

The local manager instances in sub-XGSP-conferences can play as an intermediate node to implement the distributed membership maintenance. The XGSP conference manager collects the membership report from the local conference service. And the sub-conference managers collect the local membership reports from the local users. The XGSP conference manager announces the change of the membership to all the users and the local managers.

(3) Application Session Creation / Termination

Only public sessions can be supported in XGSP conferences with multiple communities because the private application sessions are usually not implemented in other collaboration technologies. When a public application session is created in the XGSP conference, the XGSP conference manager will invoke a “create-application-session” request to the local managers. The local managers will create the session by their own mechanisms and try to link their A/V and data channels to the NaradaBroker.

Here we give an example to show how we can create an A/V application session in a multiple-community conference. When the XGSP A/V application session is activated, the XGSP conference manager will invoke “create-A/V-session” in all the local managers who use internal mechanisms to connect their A/V servers to the NaradaBroker. For H.323 and SIP communities, they connect with the NaradaBroker by dialing in the H.323 and SIP gateway. Since a MBONE community like AccessGrid, has no signaling procedure, the XGSP servers will launch an AG agent that joins in the multicast A/V groups and forwards the packets between the top XGSP session and the AG multicast groups.

(4) Floor control Mechanism and Policy

Only the conference chairman has the power to set the roles of the users in both the conference and sub-conferences. And only the users in the top conference can have chances to become a chairman. When the conference chairman wants to set the role for a user in sub-conference, the "Set Role" request will be forwarded to the XGSP conference manager. The manager will invoke "Set Role" service in the local manager. The local manager has to employ its own way to execute the role changing procedure.

4. Global MMCS Implementation

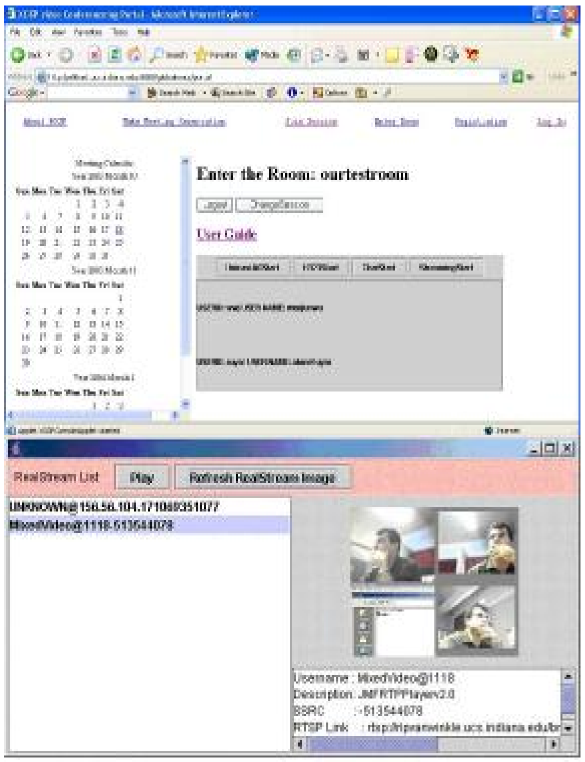
We have developed a prototype system called Global-MMCS (Global Multimedia Collaboration System) to verify and refine our XGSP conference control framework. Global-MMCS prototype is built based on the NaradaBrokering middleware. All the A/V processing components, including the video mixer, audio mixer as well as the image grabber servers are developed using Java Media Framework [20]. Java Media Framework (JMF) package, which can capture, playback, stream and transcode multiple media formats, gives multimedia developers a powerful toolkit to develop scalable, cross-platform technology.

To implement the H.323 gateway and SIP gateway, we use the protocol stacks from the open source projects, including OpenH323 [21] and NIST-SIP [22] project. The A/V Session Server is built to manage real-time A/V sessions, receiving messages from gateways and the web server through different control topics on the NaradaBrokering.

The XGSP web server based on Apache Tomcat provides an easy-to-use web interface for users to make meeting schedules, join XGSP conferences and for administrators to perform the tasks of the system management. The XGSP conference manager is implemented as an embedded server in the web container. It can create/destroy conferences, activate/deactivate A/V application sessions and generate the active conference directory to all the users. Users should log into Global-MMCS through their web browsers and select active conferences.

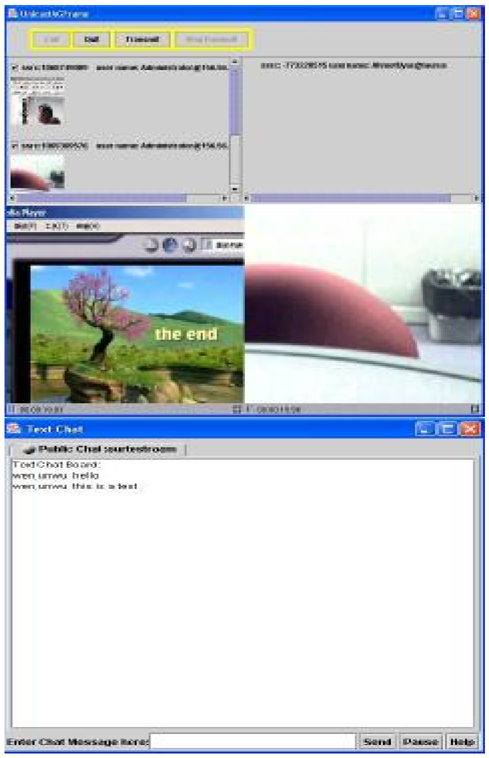
The node manager is implemented in an applet running inside the Global-MMCS portal browser. The node manager will show up when the user joins the conference. Right now the node manager can report the membership in the XGSP conference. In addition, it has a few buttons for the available application endpoints, including: a Unicast JMF, H.323, Real Streaming and chat application portlet. Depending upon XGSP AV servers, the Unicast JMF portlet can build up their A/V stream list in a videoconference and allow the user to choose any number of video streams for receiving and rendering. The H323 and Real Streaming portlets are the wrappers for H.323 terminals and RealPlayer, supporting a single video selection and rendering in their particular clients. The chat portlet provides the text chat collaboration.

The following figure shows the Global-MMCS web portal and the node manager on the left side. On the right side, the Unicast JMF and chat portlets are displayed. On the left bottom, it is the Real Streaming portlet.



UNKNOWN@156.56.104.171060351077
MixedVideo@1118-513544078

Username : MixedVideo@1118
Description: JMFRTPPlayerv2.0
SSRC : -513544078
RTSP Link : rtsp://ripvanwinkle.ucs.indiana.edu/



Text Chat Board:
wen,umwu: hello
wen,umwu: this is a test

Figure 4 Global-MMCS Web Portal and application portlets

5. Conclusion

In this paper, we have described a web-service based framework XGSP for conference control. Under the XGSP framework, we can integrate many collaboration applications into a single intuitive and easy-use environment. For the A/V collaboration, XGSP can support various audio/video endpoints including H.323, SIP and Access Grid. Furthermore XGSP uses web-services to build a community-to-community collaboration to share the global collaboration resources. The XGSP framework is not designed for replacing the frameworks of H.323, SIP as well as Access Grid, but for bridging them based on web-services technology.

In this paper we have presented our prototype collaboration system named Global-MMCS, which provides services to heterogeneous endpoints. This collaboration system is developed based on our XGSP collaboration framework and NaradaBrokering messaging middleware. Such an integrated collaboration environment greatly benefits those users that want to enter Access Grid world via H.323, SIP as well as streaming clients.

Current Global-MMCS only implemented part functions of the XGSP framework. In the next step, we will try to add the floor control service, more collaboration applications, and extend the scalability of the system to 10,000 users.

References

[1] ITU. Recommendation H.323(1999), Packet-base multimedia communications systems.
[2] J. Rosenberg et al. (2002) "SIP: Session Initiation Protocol", RFC 3261, Internet Engineering Task Force, http://www.ietf.org/rfc/rfc3261.txt.
[3] Access Grid (2003), http://www.accessgrid.org
[4] Handley, M., Crowcroft, J., Bormann, C. and J. Ott (2002) The Internet Multimedia Conferencing Architecture, Internet Draft, draft -ietf-mmusic -confarch-03.txt.
[5] Bormann, C., Kutscher, D., Ott, J., and Trossen, D. (2001). Simple conference control protocol service specification. Internet Draft, Internet Engineering Task Force, Work in progress.
[6] ITU. Recommendation H.225(2000), Calling Signaling Protocols and Media Stream Packetization for Packet-based Multimedia Communication Systems.
[7] ITU. Recommendation H.245(2000), Control Protocols for Multimedia Communication.
[8] ITU. Recommendation H.243(2000), Terminal for low bit-rate multimedia communication.
[9]Koskelainen P., Schulzrinne H. and Wu X.(2002), A SIP-based Conference Control Framework, *NOSSDAV'02*, May 12-14, 2002, Miami Beach, Florida, USA.
[10] Wu, X., Koskelainen P., Schulzrinne H., Chen C (2002). Use SIP and SOAP for conference floor control.Internet Draft, Internet Engineering Task Force, Feb. 2002. Work in progress.
[11] ITU. Recommendation T.122 (1993), Multipoint Communication Service for audiographics and audiovisual conferencing service.
[12] ITU. Recommendation T.125 (1994), Multipoint Communication Service, 1994.
[13] ITU. Recommendation T.124 (1995) Generic conference control, 1995.
[14] ITU. Recommendation T.126 (1995), Multipoint still image and annotation protocol.
[15] ITU. Recommendation T.127 (1995), Multipoint binary files transfers protocol.
[16] Virtual Rooms Video Conferencing System (2003), www.vrvs.org
[17] Geoffrey C. Fox and Shrideep Pallickara (2002). "The Narada Event Brokering System: Overview and Extensions", proceedings of the 2002 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'02)
[18] Geoffrey Fox, Hasan Bulut, Kangseok Kim, etc, (2003). Collaborative Web Services and Peer-to-Peer Grids , Keynote speech 2003 Collaborative Technologies Symposium (CTS'03).
[19] Steve Graham, Simeon Simeonov (2002) etc, Building Web Services with Java, ISBN0-672-32181-5, Sams publishing.
[20] Sun Microsystems, Java Media Framework 2.1, (2001), http://java.sun.com/products/javamedia/jmf/2.1.1/index.html.
[21] OpenH323 Project (2001) , http://www.openh323.org
[22] NIST SIP (2001), http://snad.ncsl.nist.gov/proj/iptel/.

Appendix A. A Simple Example and XML fragments

In this section, we give a simple example to illustrate the basic procedures of XGSP. Assume a user named "John" schedules a meeting named our chess game from 9:00AM to 12:00AM. The meeting has an A/V, chat, chess application facilities. At 9:00AM, the XGSP conference manager activates this meeting. The user "John", "Bob" and "Jack" join the meeting. Since John is the conference chair, John creates a chess public session. Bob and Jack join the session. John sets Bob to the white player and Jack to the black player. And John sets himself as an observer. At 11:30, they finished the chess game. John terminates the chess session. At 12:00, the conference manager deactivates the meeting.

When the meeting is scheduled, the following schedule record is kept:

```
< XGSPConference>
<ConferenceID> GameRoom </ConferenceID>
<ConferenceName> Our Chess Game </ConferenceName>
<ConferenceCreator> John </ConferenceCreator>
<ConferenceDate>
<StartTime> 9:00AM </StartTime> <EndTime> 12:00AM </EndTime>
</ConferenceDate>
<ConferenceType>  Moderated </ConferenceType>

<ApplicationList>
<ApplicationID> Audio-Video </ApplicationID>
<ApplicationID> chat      </ApplicationID>
<ApplicationID> chess    </ApplicationID>
</ApplicationList>
</XGSPConference>
```

For the chess application, the conference manager has its application registry defining three different roles: black, white and observer.

```
< ApplicationRegistry>
< ApplicationID>  chess </ApplicationID>
<roles>
   <role>
        <roleName> black </roleName>
        <capabilities> player-first </capabilities>
        <roleName> white </roleName>
        <capabilities> player-second </capabilities>
        <roleName> observer </roleName>
        <capabilities> non-player </capabilities>
   </role>
</roles>
….
</ApplicationRegistry>
```

At 9:00AM, the XGSP conference manager activates this meeting. John creates a chess public session. The following XML segment shows the message of "Create-Application-Session" indicating the new application session ID is "chess-0".

```
<CreateAppSession>
<ConferenceID> GameRoom </ConferenceID>
<ApplicationID> chess </ApplicationID>
<AppSessionID> chess-0 </AppSessionID>
<AppSession-Creator> John </AppSession-Creator>
<Private> false </Private>
</CreateAppSession>
```

After Bob and Jack join the session, John sets Bob to the white player, Jack to the black player and himself to an observer. The following XML segment shows the content of the "Set-Application-Role" messages:

```
<SetAppRole>
<AppSessionID> chess-0 </AppSessionID>
<UserID> Bob </UserID>
```

```
<RoleDescription> black </RoleDescription>
</SetAppRole>

<SetAppRole>
<AppSessionID> chess-0 </AppSessionID>
<UserID> Jack </UserID>
<RoleDescription> white </RoleDescription>
</SetAppRole>

<SetAppRole>
<AppSessionID> chess-0 </AppSessionID>
<UserID> John </UserID>
<RoleDescription> observer </RoleDescription>
</SetAppRole>
```

At 11:30, they finished the chess game. John terminates the chess session and they leave the conference. At 12:00, the conference manager deactivates the meeting. The following XML segments show the messages “Terminate-Application-Session” and “Terminate-Conference”.

```
<TerminateAppSession>
<ConerenceID> GameRoom  </ConferenceID>
<AppSessionID> chess-0 </AppSessionID>
</TerminateAppSession>

<TerminateConference>
<ConferenceID> GameRoom  </ConferenceID>
<Reason> the conference is over </Reason>
</TerminateConference>
```