A Conference Control Protocol for Highly Interactive Video-conferencing

Ruibiao Qiu Fred Kuhns Jerome R. Cox

Applied Research Laboratory
Department of Computer Science
Washington University
Saint Louis, MO 63130, USA

Abstract— **Video-conferencing is an efficient tool for distributed collaboration. With the wider availability of broadband wide area networks such as the Internet2, we can expect an increasing demand for video-conferences conducting over broadband wide area networks. In this paper, we present a conference control protocol for a highly interactive conference paradigm and its collaboration environment. In such collaboration environment, a fixed number of channels (three) for multimedia traffic as well as a common control channel are used. We propose a conference control protocol that uses a three-channel rotation floor control scheme to coordinate access to the shared media channels and avoid race conditions. Experimental results as well as the implementation in a research video-conferencing system for wide area networks show that the proposed protocol can effectively eliminate race conditions while maintaining the scalability and reliability .**

I. Introduction

Video-conferencing is an efficient means for distributed collaboration especially for people separated by substantial distance. With the increasingly pervasiveness of broadband wide area networks such as the Internet2, we can expect an increasing demand for video-conferencing over these wide area networks in the future.

We can identify various paradigms of distributed multimedia collaboration. These paradigms range from small scale video phone to highly interactive multi-party video-conferences. They differ in their degree of interactivity and scalability. Among them, the interactive video-conference paradigm requires the highest degree of interactivity and scalability. A desirable paradigm for interactive video-conferences requires a number of media channels for video and audio streams from participants. Yet, to achieve good scalability, the number of channels should be limited. Therefore, mechanisms to coordinate the access to the shared channels are required. These tasks are carried out by the conference control protocols.

The major functions of a conference control protocol are floor control and session control. Dommel and Garcia-Luna-Aceves [1] identified floor control as the crucial part of interactive multimedia collaboration, and gave a comprehensive discussion about the issues with floor control. A floor control framework was described for reference [1]. They also outlined the design issues of floor control protocols in [2]. They also compared various floor control protocols for collaborative multimedia environment, and found out that floor control protocols that are based on multicast offer the best efficiency and scalability [3]. As a generic guideline, no details were presented for specific collaboration paradigms though. In the Multi-Flow Conversion Protocol [4] designed for distributed collaboration applications, Yavatkar and Lakshman devised a token-based floor control scheme. Such scheme can be used in various collaboration paradigms by different allocation of tokens. However, for interactive video-conference, it can only apply a "strict concurrency control" with the use of a single token. This implies that only one speaker can transmit his media streams at a time, which limits the interactivity. The Conference Control Channel Protocol developed by Handley, Wakeman and Crowcroft [5] uses a shared control channel for management of conferences ranging from small and tightly-coupled to large and loosely coupled ones. Such shared control channel scheme is adopted in our proposed conference control protocol.

In this paper, we present a conference control protocol intended for an interactive and scalable video-conferencing paradigm. In a collaboration environment that supports such paradigm, three media channels are used for two interactive speakers. Contention for shared media channels is resolved with a three-channel rotation floor control scheme. Such a floor control scheme avoids conflicts on shared channels while still maintains the interactivity and scalability. In addition, a dedicated channel is used for out-of-band conference control traffic. The proposed conference control protocol is implemented in a research video-conferencing system (ALX project [6]) for high quality video-conferencing over the Internet2. The ALX video-conferencing system implements the interactive collaboration paradigm, and the proposed protocol has been proved to be able to implement such collaboration paradigm.

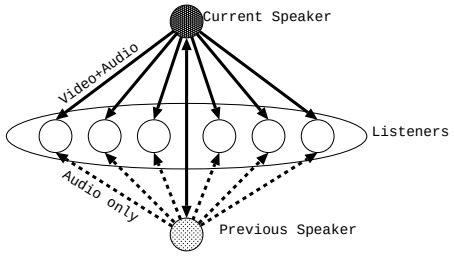
The rest of the paper is organized as follows: in Section II, we describe a collaboration paradigm for a highly interactive and scalable video-conference and the collaboration environment that supports such a paradigm. The proposed conference control protocol is presented in Section III. In Section IV, we show how the protocol behaves in a real video-conferencing session, and then we conclude the paper in Section V.

II. Collaboration Paradigm for Interactive and Scalable Video-conferencing

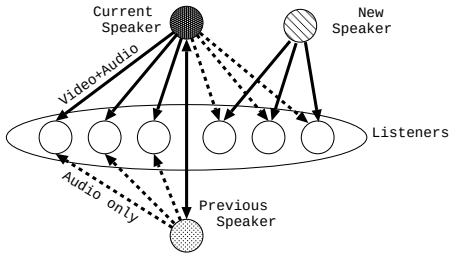
A. Collaboration Paradigms

There are various paradigms for distributed multimedia collaborations. Some examples are interactive video-conferences, video phone, group meetings, and classroom sessions. All these different paradigms exhibit different degree of interactivity and scalability. The paradigm of video phone just involves two participants with no need to scale. The group meeting paradigm involves multimedia streams from participant groups at distributed locations. These streams are presented simultaneously to participants at each location. There is no need for switching although there are multiple media streams. AccessGrid [7] is a good example of such paradigm. A classroom session is a collaboration paradigm that limits the number of media streams to one at a time. The media stream from a particular participant (the *instructor*) is transmitted to all participants during the whole session. Occasionally, the media stream from another participant is allowed to transmit back to the *instructor* for brief interaction (e.g. making comments, or answering questions). In such a paradigm, there is limited number of media streams, but no need for switching among media streams.

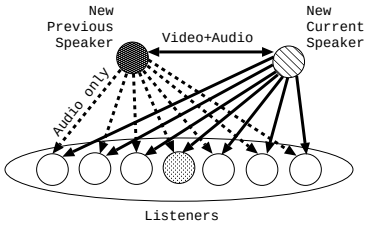
In contrast, in an interactive conference paradigm, every participant has an equal opportunity to speak to the other participants, and is able to see the speaker and to hear the ongoing discussion. We



(a) Conference paradigm with two speakers.



(b) Transition when a new speaker starts.



(c) After all participants switch to the new speaker.

Fig. 1. The transition of conference during a speaker change. The new speaker starts immediately on the third channel without interrupting the other streams.

can identify three roles a conference participant can act as, namely, the *current speaker*, the *previous speaker* and the *listener*. The *current speaker* speaks to and is seen and heard by all participants. The *previous speaker* can only be seen by the *current speaker* but heard by all participants. A *listener* can only listen to the speakers and see the *current speaker*. When a participant wants to become a speaker, he signals his request. If his request is acknowledged, he becomes the *current speaker*, and can start speaking to the other participants. Meanwhile, the other participants realize the change, and adjust accordingly to the new *current speaker*. If there is already a *current speaker*, the original *current speaker* becomes the *previous speaker*. If there is a *previous speaker* prior to the change, this original *previous speaker* terminates, and becomes a *listener*. In such paradigm, media streams are limited to two, and proper switching among media streams is required. Fig. 1 shows the transition during a speaker change in an interactive conference paradigm.

B. Collaboration Environment

To implement such an interactive conference collaborative paradigm, the collaboration environment must be setup properly.

First, multimedia streams from all participants must be transmitted in such a fashion that all participants can access equally. Although unicast and multicast can both be used to implement share media channels for this purpose, multicast is a preferred choice because of its efficiency, scalability, and growing support in broadband wide area networks such as the Internet2[8], [4], [9].

Second, these shared media channels are valuable and potentially expensive resources. The allocation and assignment of such channel resources affect the scalability, interactivity and efficiency of the conference. When there is a change of speakers, the participants should switch to the media streams from the new speaker immediately. However, the media stream from the new speaker could potentially conflict with the existing streams on the shared media channels. To avoid such a conflict, the new speaker can either wait until the original speaker stops, or inform all participants to tune to a new channel he will transmit on. The first option is not feasible because this can interrupt the interactivity by the introduction of unpredictable delay and disturbing video and audio during the switch. Because the interactive conference paradigm allows for at most two concurrent speakers, the number of media channels is at least two: one for the *current speaker* and one for the *previous speaker*. However, if we use only the minimum of two shared media channels, the media stream conflicts are inevitable during speaker switches. Alternatively, we could assign a distinct channel to each participant. This multi-channel scheme allows a participant to start his media transmission at any occasion without conflicts, and no delay during speaker changes. However, this scheme does not scale well. The number of channels increase with the number of participants, which consumes an increasing number of multicast addresses making the configuration complex. To resolve the channel conflicts and maintain scalability, we can use an additional media channel. The introduction of this channel allows the new speaker to transmit without concerning about conflicts with existing streams, reducing the switching delay and improving the interactivity. Note that proper access control to these media channels must be carried out by the conference control mechanisms.

In addition to the media channels, a common control channel is also required. Control can be centralized, or distributed among all participants. Although some video-conferencing systems on local networks use distributed control schemes, we elect to use the centralized controller because we want to avoid the multiple round trips over wide area networks for convergence in distributed schemes.

C. Conference Control Protocol Functions

The major functions of a conference control protocol are floor control and session control.

C.1 Floor Control

Floor refers to a mutually exclusive permission dynamically granted while resolving race conditions and guaranteeing fair and deadlock-free resource access [1]. Floor control allows users of networked multimedia applications to utilize and share resources without conflicts [1]. Floor control protocols add an access discipline to such environments that allows the resolution of race conditions on shared resources. For interactive video-conferences, the conference control protocol must provide mutual exclusion for concurrent access to the shared media channels to avoid conflicts. The *floor* maps to the shared media channels. Requesting and granting a floor correspond to the same actions on a shared media channel. This is the major function of a conference control protocol and the focus of this paper.

C.2 Session Control

Session control manages membership, maintains connectivity, and provides session state information. The *group membership problem* is a hard problem in distributed systems because of the difficulty to maintain a consistent view in such systems [10]. Strict consistency

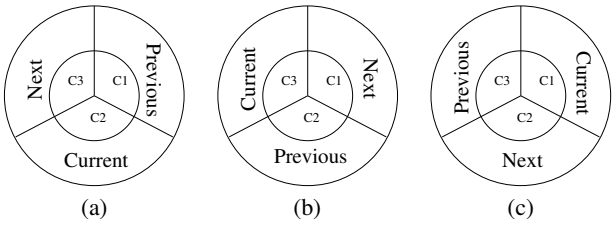


Fig. 2. Channel rotation. Fig. (a) shows the channel assignment at one point in time; (b) shows that the assignment is changed after one change in speakership; (c) shows the assignment change after another channel change. C_i corresponds to a shared media channel.

of membership is helpful but not necessary for a video-conference over wide area networks because non-speaker participants can join and leave at any time without disturbing other participants. Session control is also responsible for locating available sessions and joining the ones by the participant's choice as well as keeping track of the status of the current ongoing sessions.

III. Conference Control Protocol

Based on the previous discussion, we developed a conference control protocol that can maintain the interactivity and scalability. The details are presented below:

A. Protocol Functions

A.1 Floor Control

We devise a three-channel rotation scheme for floor control. Specifically, we assign a distinct *role* to each shared media channel. These roles (Fig. 2(a)) are *current*, *previous* and *next* which correspond to the channels for the *current speaker*, the *previous speaker* and the *next* speaker.

The new speaker is allowed to starts media transmission on the *next* channel immediately without waiting for the streams on the other channels to stop. Other participants can switch asynchronously to the new speaker when they learn of the change. After all listeners have made the change, the roles of the channels rotate 120° clockwise (Fig. 2(b)), making a new assignment of the roles to the channels. Then, a request from a new speaker will cause the process to repeat (Fig. 2(c)). Successive speaker changes cause the channel-role mapping to cycle through Fig. 2(a), 2(b) and 2(c) continuously. Notice that this scheme requires that the no speaker change can be made before all participant switch to the new speaker.

A.2 Session Control

We use a simple approach to manage membership by using a centralized controller to gather and distribute the necessary information. As we discussed earlier, such a simple management scheme is sufficient for our interactive video-conferences over wide area networks. A strict membership synchronization takes more time, and can affect the interactivity of an ongoing conference session. With the use of the centralized controller, we can keep track of the active participants without a full scale synchronization among all participants. We use pre-configured static well-known session ID (such as well-known multicast group addresses) to locate existing sessions.

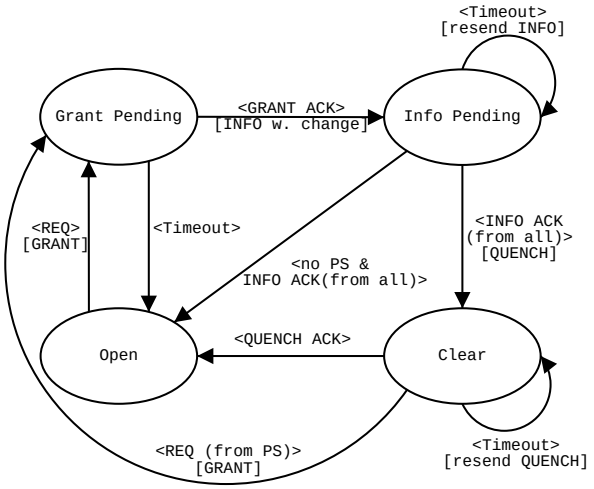


Fig. 3. State transition diagram of the controller. The labels on transition arcs are in ⟨Event⟩[Action] format, where *Event* refers to the event that triggers the state transition, and *Action* refers to the action taken when the state transition occurs. *PS* stands for the *previous speaker*.

B. Protocol Entities: Controller and Participant

There are two entities that implement the control functions in our conference control protocol: the controller and the participant. Fig. 3 and 4 show the state transition diagrams of the controller and the participant, respectively.

B.1 Controller

The controller acts as an arbitrator who takes requests from the participants, decides the speaker according to an access policy, and coordinates all participants to have a consistent view of the conference. The access policy can be simply first-come-first-serve (FCFS) or priority-based. The controller periodically sends conference control information (the *INFO* message) on the control channel, and tracks the membership information based on the participants acknowledgements. Each control messages is associated with a sequence number as a means for coarse synchronization of events, with one sequence number increment after a new *INFO* message. The controller uses the sequence numbers to check the consistency of the participants' view of the conference session as discussed below.

The controller keeps track of the conference membership by checking the acknowledgments to the *INFO* messages. If a participant does not respond to a fixed number of *INFO* messages, the controller then assumes the participant either is down or leaves the conference, and takes appropriate actions to account for the absence of the participant.

There are four states in the controller state transition diagram (Fig. 3): Open, Grant Pending, Info Pending, and Clear.

- The controller only takes requests when it is the Open state, and ignores requests in other states. This allows all participants to adapt to the current conference status before any change occurs. When the controller receives and approves a request (*REQ*) from a participant in the Open state, the controller sends a *GRANT* message to the requester, and changes to Grant Pending state.
- In the Grant Pending state, the controller waits for the requester to acknowledges the *GRANT* message. If a timeout occurs before an acknowledgement comes from the requester, the controller changes

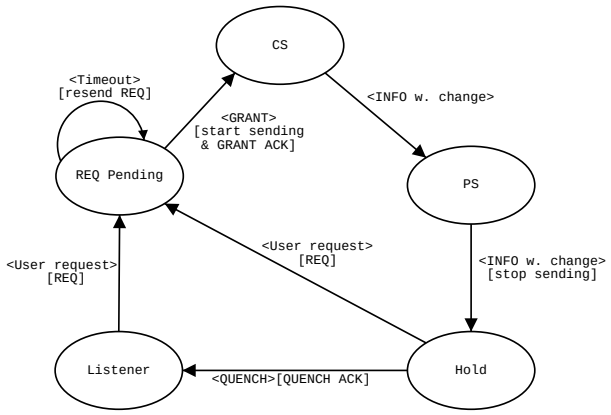


Fig. 4. State transition diagram of a participant. The labels are in ⟨Event⟩[Action] format, where *Event* refers to the event that triggers the state transition, and *Action* refers to the action taken when the state transition occurs.

back to the OPEN state, and is ready to take other requests. If the acknowledgement is received, the controller updates the conference information with the speaker change, sends out a multicast *INFO* message indicating this change, and changes to INFO PENDING state.

- In the INFO PENDING state, the controller is expecting all participants to acknowledge that they have the updated conference information after the speaker change. The controller counts the acknowledgments to his *INFO* message after the speaker change. If acknowledgments are received from all participants, every participant is informed of the change. At this moment, if there is already a *previous speaker*, the controller sends a *QUENCH* message to the *previous speaker*, and changes to CLEAR state. This *QUENCH* message explicitly informs the *previous speaker* to stop his transmission to make the *next* channel ready for future speakers. If there is no *previous speaker*, the controller simply changes back to OPEN state. If a timeout occurs before the controller can get acknowledgement of the recent speaker change indicated in the *INFO* message, the controller sends the same *INFO* message again, and waits for the participants yet to respond. If a participant has not responded for a fixed number of *INFO* messages, and thus presumed down, the conference membership information is updated, and the expected number of acknowledgements is reduced accordingly. Thus, a "down " participant will not hang the whole conference session.
- In the CLEAR state, the controller expects the *previous speaker* to acknowledge the *QUENCH* message. When the acknowledgement is received, the controller changes to OPEN state. If at this moment, the *previous speaker* decides to become a speaker again, the controller, sends a *GRANT* message to him, and changes to GRANT PENDING state. If a timeout occurs before either a *REQ* or an acknowledgement to the *QUENCH* message is received, the controller sends the *QUENCH* message again without a state change. If the controller determines that the *previous speaker* is down because it does not respond to a fixed number of *INFO* messages, the controller changes to OPEN state.

B.2 Participant

The participant part of the conference control protocol acts on behalf of the participants to interacts with the controller to implement floor and session control functions.

There are five states in the participant state transition diagram (Fig. 4): LISTENER, REQ PENDING, CS, PS, and HOLD.

- When a participant is in LISTENER state, he just passively receives video and audio streams from the *current speaker* and the *previous speaker*. If the participant decides to become a speaker, he sends a *REQ* message to the controller, and changes to REQ PENDING state.
- In the REQ PENDING state, the participant expects a *GRANT* message from the controller. If such a *GRANT* message arrives, the participant starts his video and audio transmission, sends a *GRANT ACK* message to the controller, and changes his state to CS. If a timeout occurs before a *GRANT* message, he sends the *REQ* message again while staying in the REQ PENDING state.
- In the CS state, the participant acts as the *current speaker*, and transmits his video and audio to all participants on the *current* channel. However, when the participant receives an *INFO* message which indicates that there is a change of speaker in the conference session, the participant changes his state to PS while still transmitting his video and audio streams on the same channels. While the participant is in the PS state, he is the *previous speaker* sending on the *previous* channel. When he receives an *INFO* message indicating another speaker change in the conference session, he changes to the HOLD state while still continuing with his transmission.
- In the HOLD state, the participant expects a *QUENCH* message from the controller. If such a message is received, he responds with an acknowledgement message, stops his transmission, and changes his state to LISTENER. In addition, the participant may choose to become the speaker again. In this case, the participant sends a *REQ* message to the controller, and changes to the REQ PENDING state.

Fig. 5 shows the timing diagram of the conference control protocol. As shown, a listener sends a *REQ* to the controller in OPEN state, and waits for reply in REQ PENDING state. Without any other contending *REQ*, the controller replies with *GRANT*, and changes to GRANT PENDING state. Upon receipt of the *GRANT*, the requester sends back a *GRANT ACK*, changes to CS state, and starts to transmit on the *next* channel. When the controller receives the *GRANT ACK*, a new *INFO* is generated corresponding to the change and transmitted onto the control channel. The controller changes to wait in the INFO PENDING state. As the new *INFO* propagates to all participants, they change their states accordingly (e.g. from CS to PS or from PS to LISTENER), switch to the new speaker, and reply with an *INFO ACK*. Once the controller collects all *INFO ACK* for the *INFO*, it changes back to OPEN state, and is ready to take new requests.

IV. EXPERIMENTAL RESULTS

Our conference control protocol is implemented in the ALX video-conferencing system for high quality video-conferencing over wide area networks [6]. The system uses a hardware device (ALX) to make adaptation layer translation between audio/video streams in ATM cells (obtained from the ATM-based end stations) and IP packets for proper transmission through the IP wide area networks [6]. Fig. 6 shows the bandwidth utilization in a real video-conference session. We measured the bandwidth utilization in a three-participant conference session. The three participants are A, B and C, where A and B are at one site, and C is at another site. The two sites are connected through the Internet2. Because of the instrumentation difficulties, we can only measure the aggregate bandwidth utilization at the instrumentation point (the ATM switch). Therefore, the bandwidth of C reflects what C really transmits and receives (Fig. 6c), while the transmission bandwidth of A and B are the actual bandwidth they send, but the receiving bandwidth is the aggregate bandwidth. These are sufficient to show how the control protocol coordinates the accesses to the

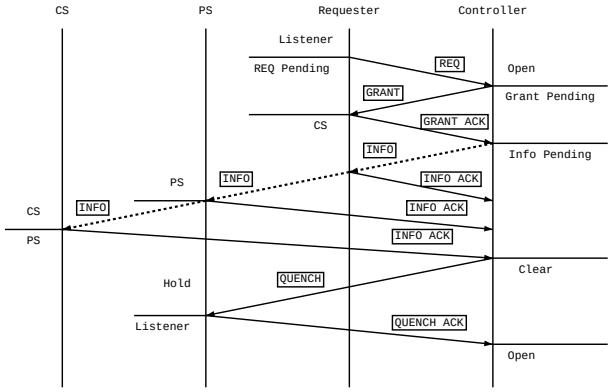


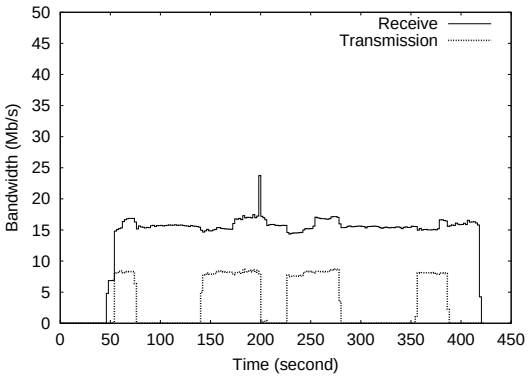
Fig. 5. Timing diagram of the conference control protocol. The text boxes are the control messages, and the text next to the vertical lines are the states. The solid line means the message is transmitted as unicast, and the dashed line means the message is transmitted as multicast.

share media channels.

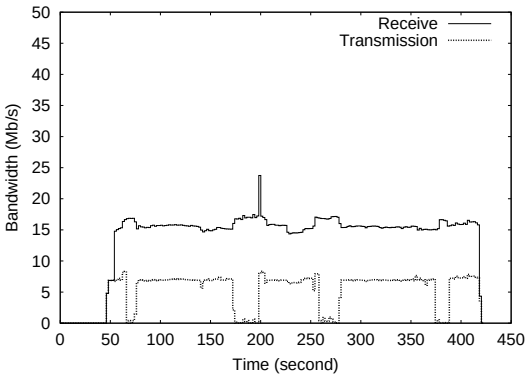
As we can see in Fig. 6, the conference started at about the 50th second when A started transmitting. Next, B requested and became the speaker. Subsequently, C, A, B, C, A, C, B, A, B, C, A became the speaker in this order. Notice that at some points, all three were transmitting (around the 200th second). Our control scheme allows this to happen because the newly granted speaker transmits on the third vacant channel which avoids any traffic collision on other channels. These periods of time are short, less than one round of *INFO* update interval (two seconds in this case.) Note that when channel conflict happens, traffic from two sources colliding on the same virtual circuit (which corresponds to a shared media channel) causes the bandwidth to become zero because the IP router drops the ill-formated packets as invalid packets. This is avoided by the control protocol.

V. Conclusions

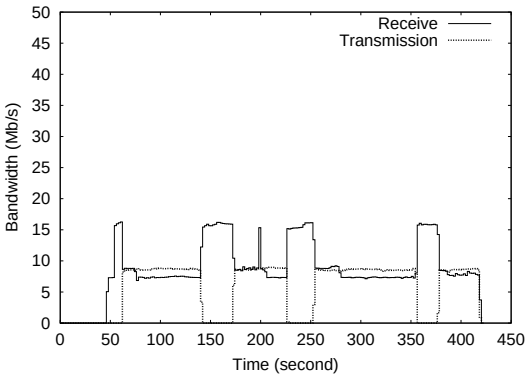
In this paper, we presented a conference control protocol for a highly interactive and scalable video-conferencing paradigm suitable for video-conferences conducted over wide area networks. We defined the collaboration environment of such paradigm. Our proposed conference control protocol uses a channel rotation scheme for floor control to avoid race conditions and coordinates the access to the shared media channels. The conference control protocol is successfully implemented in a research video-conferencing system tested over the Internet2.



(a) Participant A.



(b) Participant B.



(c) Participant C.

Fig. 6. Bandwidth utilization in a typical conference session.

Acknowledgment

This work is part of the ALX project, which is supported by the National Science Foundation (NSF) under the Grant Numbers 9729618 and ASI-9619020.

References

[1] H.-P. Dommel and J. J. Garcia-Luna-Aceves, "Floor Control for Multimedia Conferencing and Collaboration," *Multimedia Systems*, vol. 5, no. 1, pp. 23–38, 1997.

[2] H.-P. Dommel 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.

[3] H.-P. Dommel and J. Garcia-Luna-Aceves, "Comparison of floor control protocols for collaborative multimedia environments," in *Proceedings of SPIE Symposium on Voice, Video, and Data Communicaions*, (Boston, MA), November 1998.

[4] R. Yavatkar and K. Lakshman, "Communication Support for Distributed Collaborative Applications," *Multimedia Systems*, vol. 2, no. 2, pp. 74–88, 1994.

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

[6] R. Qiu, F. Kuhns, J. Cox, and C. Horn, "Bringing Studio Quality Video-conferencing to Wide Area IP Networks with an Adaptation Layer Translator (ALX)," in *Proceedings of SPIE ITCom 2002*, (Boston, MA, USA), August 2002.

[7] Argonne National Laboratories, "Access Grid Project." URL `http://www.accessgrid.org/`.

[8] O. Hermanns, "Performance Evaluation of Connectionless Multicast Protocals for Cooperative Multimedia Applications," in *Messung, Modellierung und Bewertung von*, pp. 372–384, 1995.

[9] H. Smith, M. Mutka, and D. Rover, "Controlling Video Conferencing via a Feedback Based Rate Control Algorithm," *High Speed Networks, Special Issue on Multimedia Networking*, 1997.

[10] M. Franceschetti and J. Bruck, "A Possible Solution to the Impossible Membership Problem," Tech. Rep. ETR032, Paradise, California Institute of Technology, Paradise, California Institute of Technology, October 1999.