Floor Control Alternatives for Distributed Videoconferencing over IP Networks

J. J. Garcia-Luna-Aceves *†
* Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304 USA
jj@soe.ucsc.edu

Patrick E. Mantey and Sireesh N. Potireddy
† Baskin School of Engineering
University of California Santa Cruz
Santa Cruz, CA 95064 USA
{mantey, sireesh}@soe.ucsc.edu

Abstract— **Applications that require the communication of multiple video streams can consume considerable bandwidth and computing resources, which poses a challenge for the widespread use of videoconferencing over the IP Internet. On the one hand, the bandwidth of the link connecting a given participant to a videoconferencing session may not be enough to support many video streams at bit rates of 500 kbps or more, especially when the participant is connecting to the rest of the Internet through a wireless link. On the other hand, the processing capacity of a participating site may not be enough to decode several video streams in real time.**

This paper explores the use of floor control over videoconferencing applications as a means to support videoconferences with many participating sites, but with a processing and communication overhead per site that is equivalent to a two-party videoconference. The main tradeoff we explore is the scalability attained with floor control versus the latencies incurred with floor transitions, which can be much too disruptive to the videoconference participants. We present a viable compromise in which only the video stream of the "floor holder" is sent to all sites, but the floor-passing protocol is such that it supports a brief overlap of the transmissions from the old and the new floor holder, such that the participants in the videoconference can instantaneously switch over to the media streams of the next speaker in an apparently seamless transition. Experimental results and implementation in a research video-conferencing system show that the proposed protocol can run effectively, eliminating race conditions, while maintaining scalability and reliability.

I. INTRODUCTION

The advent of networked multimedia applications and improved network, processor and compression technologies have made telecollaboration increasingly attractive. Synchronous telecollaboration allows a geographically dispersed group of users to communicate and collaborate effectively in real-time. Video conferencing is becoming economical with the increase in available bandwidth and processor speed and reduced equipment and network cost.

In face-to-face meetings, conference participants implement *floor control* (i.e., controlling the right to address any subset of the audience in the conference) by means of many verbal and non-verbal cues communicated without delay, such as selective eye gazing, speaking more loudly, or interjecting a few words in the discourse. Furthermore, the effective communication bandwidth available for communication among participants is enormous. By contrast, in videoconferencing over the Internet, there are limitations on the amount of bandwidth that each participant's computer or conference server can access, as well as the processing power of each participating site. Furthermore, access to verbal and non-verbal cues is severely restricted, and there are non-negligible latencies among conference participants.

Whether videoconference services are based on a centralized or distributed architecture, a key challenge in their offering over the IP Internet consists of ensuring that, regardless of the number of video sources, each participating site has enough processing and bandwidth resources to consume the video streams transmitted in a videoconference session, without having the collaboration among participants disrupted by the same mechanisms used to manage the processing and communication resources. This is the focus of this paper, which addresses the use of floor control mechanisms for videoconferencing over the Internet that are not very intrusive to the end users, in that the floor control schemes used to manage system resources do not interfere too much with the social protocols to which people are accustomed in face-to-face meetings.

Floor control allows users in a collaborative environment to utilize and share resources such as continuous media (audio and video), remote devices, or tele-pointers in an electronic whiteboard application, without any access conflicts. "Floors" are temporary permissions that are granted to collaborating users in order to mitigate race conditions and ensure mutually exclusive access. Dommel and Garcia-Luna-Aceves [1], [2], [3] presented a general framework for floor control and analyzed the performance of various types of floor control protocols. Broadly speaking, there are three ways in which bandwidth and processing resources can be managed with respect to floor control. If no floor control is exercised, any participant can transmit at will and all participants receive and must process the incoming video streams. If the available bandwidth is plentiful, participants can be allowed to transmit freely, and the receivers decode the video streams of only that participant that has the floor. We refer to this approach as *receiver-based floor control*, which is only meant to avoid having a participating site experiencing too much processing load. If both bandwidth and processing resources must be managed, only the video stream of the participant with the floor is transmitted; we refer to this approach as *sender-based oor control*. For the case of videoconferencing among several parties over the Internet, sender-based floor control is needed, because both bandwidth and processing resources must be managed.

Implementing sender-based floor control requires that the floor hand offs among floor holders be such that the conference participants do not experience any gaps in the rendering of the video, even if there are latencies in the floor hand off, and the receivers introduce additional delays decoding different video streams. Section II describes our floor-control protocol, which supports sender-based floor control and reduces the negative effects of floor hand offs by means of hysteresis. When a participant is granted the floor, the current floor holder continues transmitting for a limited period of time, which is larger than the floor hand-off latencies in the system. During this time period, all the participants in the conference continue to process and play the streams sent by the current floor holder. The new floor holder starts transmitting the video and audio and when the remaining participants receive the video (and audio) sent by the new floor holder, they start decompressing but do not display the video (or play the audio). When the time period expires, all the participants can instantaneously switch over to the video (and audio) from the previous floor holder to the new floor holder. The protocol is scalable in terms of the number of participants in the conference and, with the hysteresis effect, the transition appears smooth and unobtrusive. The protocol is reliable and can recover from link and node failures.

Section III discusses an existing video collaboration system and the implementation of the floor control protocol. Because IP multicast is not supported widely by Internet service providers (ISP), commercial support of videoconferencing among multiple sites over the Internet is done by means of conference servers that communicate with each conferencing site over unicast connections, given that IP multicasting is not deployed widely. However, as the size of the videoconferencing service over an internetwork grows in terms of the number of concurrent videoconferences and the number of participants per videoconference session, multicast support for videoconferencing sessions is preferable over the use of a centralized conference servers, because it renders smaller latencies and much better resource utilization. Fortunately, multicasting can be supported over network overlays on top of the network layer. An early example of such overlays is the Internet Multicast Backbone or MBone. The early MBone tools vat and rat [4], [5] for audio conferencing and vic [6] for video conferencing, the Xerox PARC video tool nv [7] and the INRIA videoconferencing system ivs [8] were the first efforts to deploy real-time multimedia collaboration services over the Internet. Furthermore, the Internet2 [9], which consists of vBNS vBNS [10] and Abilene [11], is an example of an internetwork that supports IP multicasting among research and educational institutions. Accordingly, our implementation of floor control for videoconferencing was done using multicast support; however, our floor control protocol applies equally well to server-based videoconferencing.

Section IV presents experimental results that show how the protocol performs in real video-collaboration sessions. Section V discusses our future work and conclusions.

II. Floor Control

Our floor control protocol is based on sender-based floor control. We present a modified version with the '*hysteresis*' effect to improve the smoothness of transition of the floor. The protocol can be adapted to various session types and collaborative models, such as highly interactive group meetings or distance learning. In the distance learning collaboration model, the media streams (audio, video, presentation) from a single participant (e.g., the instructor) is transmitted to all the participants during the session. The media streams of other participants are allowed to transmit back to the instructor (e.g. giving suggestions or answering questions). The instructor acts as the moderator of the conference, arbitrarily granting permission to other participants for transmitting multimedia streams. The instructor selects one of the pending requests and grants the floor. The instructor has the authority to revoke the floor at any given time or assign it to some other participant.

In a group meeting, there is no moderator and each participant has an equal opportunity to get the floor. Due to the absence of an arbitrator, the current speaker or floor holder assumes the role of the floor arbitrator and grants the floor. The order in which requests are granted depends upon the service policy adopted for the particular session. The protocol presented here is independent of the service policy used for servicing requests. The service policy could be a First Come First Served (FCFS) policy, where the participant sending the earliest floor request is granted the floor or a priority based policy where the participant with the highest priority is granted the floor or some other form of service policy.

The protocol runs independently at each node without any specific assumptions about the topology of the underlying network. Communication between two nodes in the network is solely through message passing and not through shared memory. The protocol does not assume a reliable delivery of packets. There may be packet loss or reordering of packets. Processes may fail, and the protocol is able to detect such failures and recover from them. The protocol is implemented as a middleware component below the application layer and acts as an intermediate process between the user and the underlying application. The protocol does not require a multicast enabled network, however the transmission overhead is reduced considerably if multicast is enabled.

A. System Roles

There are three system roles which refer to the control function that are used in the floor control protocol - the floor holder (*FH*), the floor controller (*FC*) and participant. The floor holder is the temporary user of the resource which is managed by the floor control protocol. In a multimedia conference the floor holder is the user who is allowed to send media streams (audio and video) to the other participants in

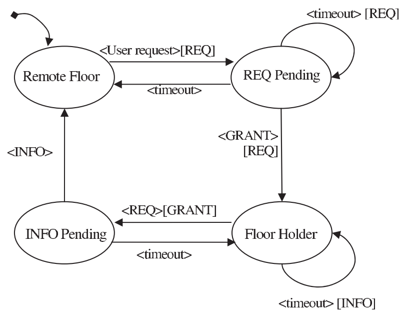

Fig. 1. State transition diagram of floor control protocol. The labels in arcs are in ⟨Event⟩[Action] format where Event refers to the event that triggers the state transition, and Action refers to the action taken the state transition occurs.

the conference. The floor controller determines the next user to be granted the floor.

In the distant learning collaboration model, the instructor is the floor controller who can arbitrarily assign and revoke the floor at any time during the conference session. Due to the absence of a moderator, the roles of the floor controller and floor holder are merged in the group meeting collaboration model. The current floor holder also acts as the floor controller and determines the next floor holder. In the remainder of this paper, the group meeting collaboration model is assumed, unless stated otherwise.

When the roles of *FH* and *FC* are merged, control for a specific floor is always centralized to the node currently holding the floor. Once a turn is completed, control shifts to the next floor holder. This model implements a hybrid approach between centralized and a fully distributed model. All users apart from the floor holder and floor controller are participants. In a multimedia conference, the participant receives and decodes the media streams transmitted by the floor holder.

B. Protocol Description

Figure 1 shows a simplified specification of the floor control protocol in the form of a state machine. The state diagram shows the necessary states required for transition of the floor from one participant to another. The state diagram does not contain transitions for exceptions such as a link failure or node failure which necessitates the election of a new floor holder. This specification is used for our implementation of receiver-based floor control and sender-based floor control, which we compare with each other in Section 4. The floor control protocol we advocate is presented in Section II-E.

The floor holder acts as an arbitrator who accepts requests for other participants in the conference, decides the next floor holder according to a session specific service policy and coordinates all participants to have a consistent view of the conference. The access policy can be a First Come First Served (FCFS) or a priority based policy. The floor holder periodically transmits control messages (*INFO*) broadcasting its state information. The control messages reveal the identity of the floor holder to all other participants at any instant of time. The control messages serve to inform the participants that have just entered the conference the current state of the protocol. The absence of *INFO* messages indicates failure of the floor holder and an election process is triggered to elect a new floor holder.

If a participant decides to request the floor, a floor request message is sent to the floor holder (*FH*). In order to ensure a more reliable transmission, the participant periodically sends floor requests (*REQ*) for a certain period of time until a grant is received or the request expires.

The floor request may be transmitted either by unicast or multicast to the floor holder. Even though sending floor requests through multicast increases the control message traffic, (as they are sent to all the participants in the conference) multicast enables all participants to maintain a global request queue which is more or less consistent throughout all the nodes. If a participant who is about to send a floor request receives a floor request from another participant that has a higher priority or earlier time stamp depending upon the service policy, the request can be postponed until the earlier request has been granted.

The floor holder receives requests from participants who wish to acquire the floor. Participants with a higher privilege (an instructor teaching a class) may retain the floor for an indefinite period of time and grant the floor only when ready to relinquish the floor. Alternatively, the floor is automatically released upon receiving a floor request. Restrictions may be applied to the length of time that a participant can hold the floor. Participants may be guaranteed to hold the floor for a certain minimum amount of time.

The floor holder stores incoming floor requests in a request queue. The floor holder waits for a small period of time to receive more floor requests in order to sort out simultaneous requests from different participants before granting the floor. The size of the request queue is limited relative to the number of participants in the session. The request queue is cleared each time a participant acquires the floor. This allows the floor holder to store only the recent requests it has received since the last floor transition.

The control can be passed on to a participant according to a service policy such as First Come First Served (FCFS), or a priority based policy, or a participant can be arbitrarily selected to receive the floor (by the instructor in a classroom session).

When the floor holder receives and approves a request from a participant, the floor holder sends a floor grant message (*GRANT*) to the requester. The floor grant message may be sent by unicast to the requester or it may be sent by multicast to all the participants in the conference. By sending floor grants using multicast, all participants are informed of the transition from the current floor holder to the next. The grant message also serves the purpose of a floor deny message to the other participants who have requested the floor.

The requester expects to receive a floor grant from the floor holder. If such a grant message is received, the participant becomes the new floor holder. The new floor holder informs the remaining participants in the conference by broadcasting floor updates to all the participants in the conference. The floor updates also serve to acknowledge to the previous floor holder that the floor grant has been received.

C. Receiver-based Floor Control

The receiver-based floor control assumes an abundant supply of bandwidth as all the participants in the conference send video (and audio) whether or not they have the floor. The receiver-based floor control is a passive control concept that enables the receiver to "filter" specific streams and ignore the remaining streams received ("*What I See Is What I Want*"). Only the media streams from the floor holder are decoded and played. The receiver-based floor control avoids the problem of saturating the end receiver.

There is a small delay involved in transitioning the floor from one participant to another. This delay occurs due to the time taken by the underlying software to switch decoding between two streams. The delay also depends upon the standard used for multimedia compression and decompression. For example, in an MPEG video stream that consists of I-frames, P-frames and B-frames, decompression can start only when the next I-frame has been received as the previous I-frame was not processed.

The receiver-based floor control results in a significant network load due to the video streams. Hence, it is more suitable for audio conferences, which do not consume much bandwidth.

D. Sender-based Floor Control

In the sender-based floor control only the floor holder is allowed to send video and audio. ("*What You See Is What I Share*") The participants in the conference automatically decode the incoming audio and video streams. As there is only a single user (*oor holder*) sending and the bandwidth consumption and CPU utilization is minimal. The time taken to switch to the new floor holder includes the time taken to start sending the video stream (includes the time taken for compression), the time taken to transmit the video stream across the network and the time taken by the receiver to start decoding the video and display it on the screen. The switching time is highly dependent upon the software complexity in initializing, compressing and decoding the video which in turn depends upon the multimedia standards that are employed.

In order to decrease the delay involved in floor transition, we use an *optimistic* version of the sender-based floor control protocol. A participant is allowed to send video (and audio) when a floor request is issued in anticipation that the request will be granted. If the participant is granted the floor, the other participants in the conference switch to the video of the new floor holder. The participant stops sending when the request expires or when the floor is granted to another participant. By the time the participant receives the floor grant, the participant gains a head start by sending the video. Switching to the video of the new floor holder will take less time compared to the original sender-based floor control.

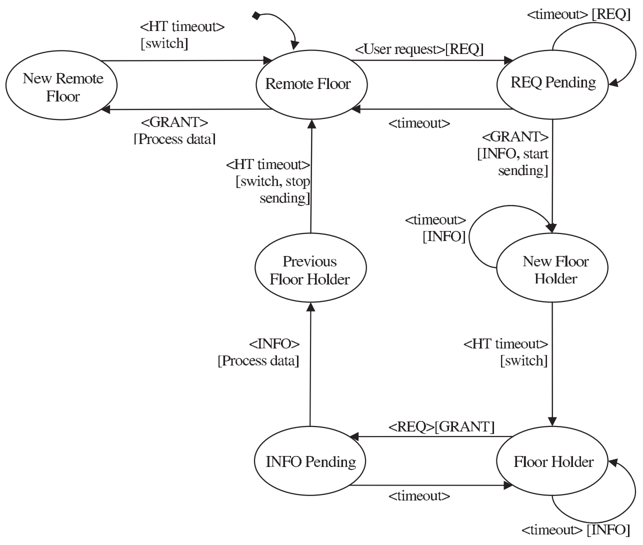

Fig. 2. State transition diagram of floor control protocol. The labels in arcs are in ⟨Event⟩[Action] format where Event refers to the event that triggers the state transition, and Action refers to the action taken the state transition occurs.

The only drawback with the optimistic version of the sender-based floor control is that when multiple participants simultaneously request the floor, there will be a sudden increase in the network load which may exceed the available bandwidth. If a well defined service policy such as FCFS or a priority based policy is used and floor requests are sent by multicast, participants may refrain from requesting the floor if floor requests sent by other participants with an earlier timestamp (as in the case of FCFS policy) or a higher priority (priority based policy) are detected.

E. Floor Control with Hysteresis

In section II-B, we presented a basic outline of the floor control protocol. In this section, we will expand on the protocol presented above to include the"*hysteresis effect*" in order to make the switch between two participants unobtrusive. Figure 2 shows the state machine for the protocol with hysteresis. Three new states have been added in order to achieve the hysteresis effect. Each of these three states serves as wait-states for a fixed period of time. This time should be sufficient enough for a participant who has just received the floor to start transmitting the audio/video streams. We name the timer which controls these three states as the '*hysteresis timer*' (*HT*).

When the floor holder grants the floor to one of the requested participants, the floor holder waits to receive an acknowledgment in the form of an *INFO* message which the new floor holder would broadcast upon receiving the *GRANT*. After sending the *GRANT*, the floor holder activates the hysteresis timer and continues transmitting its audio and video. During this time before the timer expires, the current floor holder start receiving and decoding the incoming media streams sent by the new floor holder. When the timer expires, the current floor holder immediately switches to the audio/video from the new floor holder and stops transmitting its own video (and audio).

When a participant who has requested the floor is granted a floor by the floor holder, the hysteresis timer is activated. When the timer expires, the new floor holder would have started processing and transmitting the audio and video streams. Until the timer expires, the new floor holder continues to decode and display the media streams coming from the previous floor holder. When the timer expires, the new floor holder and all participants stop decoding the data from the previous floor holder and switches over to the data of the new floor holder.

The floor holder broadcasts the *GRANT* messages to all the participants in the conference. When a passive participant receives a *GRANT* message, he is informed that a switch is about to take place. The hysteresis timer (*HT*) is activated, and within the time the timer takes to expire, the participant receives the incoming data from the new floor holder. The participant starts decompressing the audio/video streams, but does not display the video (or play the audio) until the timer expires. When the timer expires, since the video streams have already been processed, the switch takes place instantly. In particular for MPEG video streams which consist of I, B and P frames, decompression can start only from an I-frame. In this case, since the video streams of the new speaker are already being decoded, the switch can take place instantaneously.

The *hysteresis effect* also helps to improve the "*naturalness*" of the conversation by informing all the participants in the conference that a switch is about to take place, so that the switch does not take place unexpectedly. This can be reflected in the user interface by a flashing signal which indicates that a switch is about to take place. During the hysteresis time period, all the participants in the conference are still tuned into the previous floor holder. The floor holder can use this time to finish his conversation and can give up the floor in a more seamless conversational manner.

F. Reliability

Control messages are sent repeatedly to increase the reliability of message transmission. When a participant wants to request the floor, the participant sends floor requests periodically until the request expires or a floor grant is received. The floor requests may be sent at regular intervals of time or increasing intervals of time.

When the floor holder decides to grant the floor to one of the requested participants, the floor holder sends floor grants repeatedly for a period of time until an acknowledgment is received in the form of an *INFO* message from the requested participant. If an *INFO* message is not received within this period of time, the floor holder grants the floor to another participant from the remaining participants who have requested the floor. If the current floor holder does not receive an *INFO* message due to a large delay in the network, and grants the floor to another participant, then there arises the case when two participants assume the role of the floor holder. Each floor holder broadcasts *INFO* messages. When each of these two floor holders receives *INFO* messages from the other, one of the floor holders backs down and releases the floor. The floor holder which backs down can be the floor holder with a lower priority level or the floor holder which has acquired the floor earlier or has had the floor more recently.

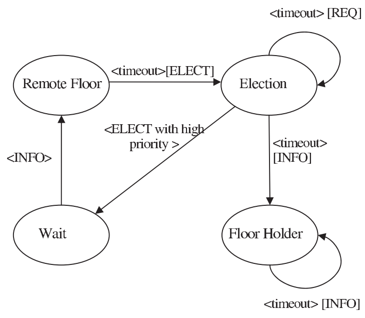

Fig. 3. State transition diagram of floor control protocol. The labels in arcs are in ⟨Event⟩[Action] format where Event refers to the event that triggers the state transition, and Action refers to the action taken the state transition occurs.

Messages from a single source may be reordered or duplicated and, due to differing delays, messages from different sources may arrive in an incorrect order. Each control message bears a sequence number and a timestamp. Using the sequence number, packets that arrive with a smaller sequence number (or earlier timestamp) than the latest seen are discarded. Messages from different sources may arrive in a different order than they are sent. The reordering of messages can be solved if it is guaranteed that the clocks are synchronized at each site within a small period of time.

Because our floor control protocol used a hybrid approach between a centralized version and a fully distributed version, the protocol has a roving critical point of failure. The protocol detects the failure of the floor holder and takes measures to recover. The floor holder continuously broadcasts floor updates at specified intervals of time. Failure of the floor holder can be detected by the nonappearance of *INFO* messages. In the classroom collaboration model, if the floor holder has crashed (and is not the instructor) then the instructor can be the point of recovery and assume the role of the floor holder. In the group meeting collaboration model, and in the classroom collaboration model when the instructor crashes while holding the floor, absence of *INFO* messages triggers an election process within the remaining participants, and a new floor holder is elected. The time which a participant waits before it initiates the election process should be multiple times larger than the interval at which the floor holder sends floor updates. This ensures that the election process is not unnecessarily triggered due to the loss or delay of a few packets.

When the election process is triggered, each participant broadcasts a floor elect (*ELECT*) message. ELECT messages are sent at regular intervals. The *ELECT* messages carry the participant id and a priority level. The priority level may be assigned beforehand or a random priority level may be generated. Each participant must have a unique participant id and priority level. Random priority levels can be generated by a hashing algorithm on the participant id.

When a participant with a lower priority level receives a floor elect message from a participant with a higher priority level, the participant stops sending floor elect messages and enters a wait state. After some time, the participant with the highest priority level remains as the only participant sending *ELECT* messages. When the participant does not receive any other *ELECT* messages, the participant assumes the role of the floor holder and starts sending *INFO* messages. The *INFO* messages inform the other participants the identity of the newly elected floor holder.

The election process is also used to elect a floor holder during the initialization phase of a session when a floor holder has not yet been assigned and several participants enter the session at the same time.

Another point of failure is the failure of the node which has just been granted the floor. If a participant that has been granted the floor crashes before broadcasting *INFO* messages to all the participants in the session, the current floor holder becomes the point of recovery. The floor holder waits for a specific period of time for the *INFO* message from the newly granted floor holder. If it does not receive an *INFO* message within this period of time, the floor holder grants the floor to another participant which has requested the floor. If no other participant has requested the floor, the floor holder retains the floor.

G. Correctness and Fairness

We can prove the correctness of the protocol if we can prove that within a finite period of time, only a single floor holder remains. Fairness is inherently given with the established floor policy.

A summary proof that there is only one floor holder within a finite period of time is outline below.

Suppose at any given time there happen to be more than one Floor Holders say $FH_1, FH_2, FH_K \; K > 1$. Each Floor holder periodically sends *INFO* messages. Even though packets may be dropped, we assume that at least some packets eventually get delivered. When a floor holder with a lower priority receives an *INFO* message from a node with higher priority, it releases the floor. Within a finite amount of time, the floor holder with the highest priority remains to be the only Floor Holder.

In case at some moment there is no floor holder, due to the absence of *INFO* messages, each participant eventually enters the election process after a timeout. Each host participating in the election broadcasts *ELECT* messages to all the other participants in the conference.

When a node with a lower priority receives an *ELECT* message from a node with a higher priority, the node enters a wait state and waits for the election process to end and a floor holder to be elected. If it doesn't receive a Floor Update before the timeout occurs, it reenters the Election state.

Whenever a node receives an *ELECT* with a lower priority it resets a timer. If a node doesn't receive any other *ELECT* message within this time (which is sufficient enough to receive packets from all participants), it assumes the role of the floor holder and broadcasts an *INFO* packet. In case the host receives an *ELECT* message from a host with a higher priority after assuming the role of the floor holder, the protocol enters the wait state and waits for the *INFO* message from the node with the higher priority. At the end of the election process, a new floor holder is elected.

Thus, within a finite period of time, there is only one floor holder.

III. Implementation

ConferenceXP is a research and development initiative of Microsoft Research's Learning Sciences and Technology Group [12]. *ConferenceXP* provides an extensible foundation for interactive collaborative environments, and it serves as a research platform for designing and implementing distance conferencing and learning applications.

ConferenceXP requires connectivity to the *Internet2 Abilene* network [9]. The *Abilene* Network is an Internet2 high-performance backbone network that enables the development of advanced Internet applications and the deployment of leading-edge network services to Internet2 universities and research labs across the country. The high speed of the Internet2 *Abilene* network, enabled for multicast, is a significant emerging technology in support of high-end collaboration solutions that provides both high quality and low latency delivery of audio and video. *ConferenceXP* utilizes *Microsoft DirectShow* and *Windows Media* audio and video codecs. *ConferenceXP* supports full screen video at 30 fps with 250 ms latency.

The *ConferenceXP* architecture [13] is divided into four logical layers: i) *ConferenceXP Application*, ii) *ConferenceXP Capability*, iii) *Conference API*, and iv) *Network Transport*. The *ConferenceXP Application* and *Capability* layers provide the user interface for *ConferenceXP*. Capabilities are add-in components that add functionality to a *ConferenceXP* application. The *ConferenceXP Capability* layer includes the Audio/Video and Presentation capabilities included with *ConferenceXP*. The *RTDocuments API* provides applications and capabilities with a standard protocol to transfer documents and ink strokes. The *DirectShow* and *Windows Media APIs* provide access to audio and video features in Windows. The Network Transport layer ensures that the audio, video, and data streams are transmitted with minimum data loss. *ConferenceXP* sends audio, video, and data streams over the network by using an implementation of the Real-time Transport Protocol (RTP). [14]

ConfernceXP employs a Venue Server which provides the services necessary to create and manage virtual venues (virtual meeting rooms) where users can participate in synchronous learning and collaboration activities. Each virtual venue is statically bound to a specific multicast IP address. The Venue server interfaces are exposed as web services.

Each participant uses his/her Windows Passport account to create a unique identifier for the conference session. A participant wishing to join a venue sends a request to the Venue Server using standard Internet protocols such as SOAP and HTTP. The Venue Server responds with the multicast IP address associated with the venue. The participant then enters the venue by joining the multicast group. *ConferenceXP* audio and video streams are transmitted using the Real-time Transport Protocol. (RTP) [14]

Our floor control described earlier is implemented as a separate layer within the *ConferenceXP* architecture, in between the user interface and the *ConferenceXP API* layer. The graphical user interface has been modified to incorporate functions related to floor control. A floor indicator indicates the current state of the floor. The indicator can exhibit three colors red, yellow and green. Red indicates that the floor is held by a remote user, yellow indicates a transition in the floor (either in the process of acquiring the floor or releasing the floor) and green indicates that the floor is held locally. A flashing yellow light indicates that a floor transition is about to take place within a small period of time (due to hysteresis) so that the speaker can give up the floor in a more *conversational* manner.

Floor requests are sent by clicking a button. Except for users with a higher permission level, (instructor in a classroom session) the floor is automatically released upon receiving a floor request. A First Come First Served policy is used to select the participant to be granted the floor. The floor control protocol controls the audio and video streams of each participant.

There are a total of four types of control messages that are used for the floor control protocol. i) A floor request (*REQ*) used for sending a request for the floor, ii) a floor info (*INFO*) used for transmitting control state information, iii) a floor grant (*GRANT*) used for granting the floor to another participant and iv) a floor elect (*ELECT*) message used for electing a new floor holder when the floor holder fails or during the initial phase of a session when there is no floor holder.

TABLE I
FLOOR CONTROL PACKET STRUCTURE.

Field	Description
Type	Packet type (*REQ*, *INFO*, *GRANT*, *INFO*)
Id	Host Id
SeqNo	Sequence Number
Ts	Time Stamp
Pr	Priority Level
ToId	Destination Id
St	Start Time

The control message structure is shown in Table I. Each control message contains information such as the type of floor control packet (*REQ, INFO, GRANT, ELECT*), the host id which has sent the message, a sequence number for detecting out of order messages, a time stamp and a priority level. The **ToId** field is used to indicate the destination id of the packet. *INFO* and *ELECT* messages which are broadcast to all the participants in the session have the field blank. *REQ* messages are addressed to the floor holder and *GRANT* messages are addressed to the participant who is granted the floor. In the case of *REQ* messages, the **St** field is used for determining the relative ordering of floor requests coming from different participants. For *INFO* messages, it indicates the time at which the floor holder received the floor. The *ELECT* and *GRANT* messages do not use this field.

IV. Experimental Results

In this section we describe the experiments that were performed and discuss the results obtained. For comparison purposes we have also run experiments using the original *ConferenceXP* platform without any floor control, and with a receiver-based floor control and a sender-based floor control. We ran the experiments with up to eight machines that are connected to the SOE local area network in the Baskin School of Engineering at the University of California, Santa Cruz. We compare the performance in terms of bandwidth consumption and processing power utilization.

A. Comparative Analysis

We measured the CPU utilization and bandwidth utilization in a eight-participant conference session for the three different floor control protocols. The session is run for a time of approximately twelve minutes. Each session is run with the same scenario, and in each scenario, all the eight hosts acquire the floor once during the session. All video streams transmitted by all the hosts are transmitted at a bit rate of 500 kbps.

Initially when the conference has started, A has the floor. B requests the floor after about 90 seconds, and acquires the floor. Subsequently C, D, E, F, G and H acquire the floor in that order. The CPU utilization and bandwidth utilization are measured at site B. The machine at site B running the application has a dual Intel Xeon 2.8 Ghz processor and 1 GB RAM.

Figure 4 and Figure 5 shows the CPU utilization and bandwidth utilization of an eight participant session using the receiver-based floor control. We can observe that the CPU utilization is between 20 and 40% throughout the session with a few regular peaks in between. The CPU utilization is quite high compared as each site is constantly compressing and transmitting video. The peaks in the CPU utilization graph correspond to the time when the speaker changes. The speaker changes roughly every 90 seconds (shown by the double headed arrows in Figure 4). When the speaker changes, the application has to stop decoding the current stream and start processing the speaker's stream. The bandwidth consumption is quite high as all the participants in the conference are transmitting at the same time.

Figure 6 and Figure 7 shows the CPU utilization and bandwidth utilization of the conference using the sender based floor control. The CPU utilization falls under 20% when B is just a receiver and the CPU utilization increases and falls in the range 20-40% when B becomes the speaker and starts transmitting. The additional CPU time is due to the video compression. There are sudden peaks when B acquires the floor and releases

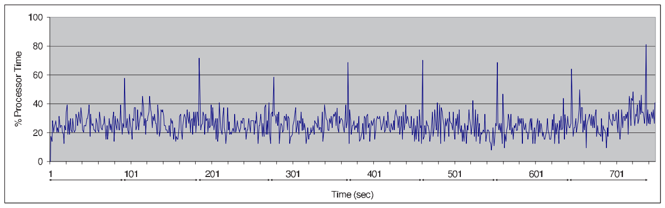

Fig. 4. CPU Utilization using a receiver-based floor control with eight hosts participating in the conference.

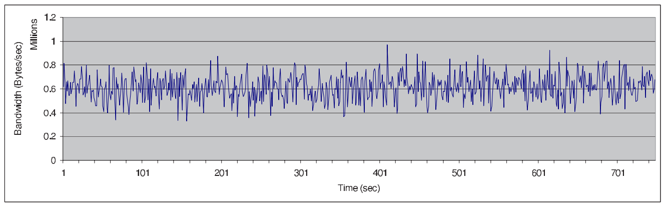

Fig. 5. Bandwidth consumption using the receiver-based floor control with eight hosts participating in the conference.

the floor. This is due to the software complexity in starting and halting the video capture and compression. The bandwidth is limited to a single video stream as only the speaker is allowed to send video.

Figure 8 and Figure 9 shows the CPU utilization and bandwidth consumption for an individual participant in the conference using floor control with *hysteresis*. Similar to the sender based floor control, the CPU utilization falls under 20% when B is just a receiver and the CPU utilization increases to 20-40% when B becomes the speaker and starts transmitting. We can observe slightly more prominent peaks in the CPU utilization when the speaker changes. This is due to the hysteresis effect. During floor transition, each participant decodes two video streams simultaneously - one which is still

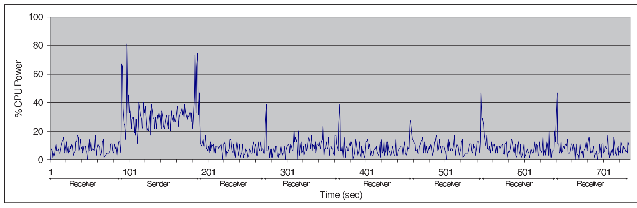

Fig. 6. CPU Utilization using a sender-based floor control with eight hosts participating in the conference.

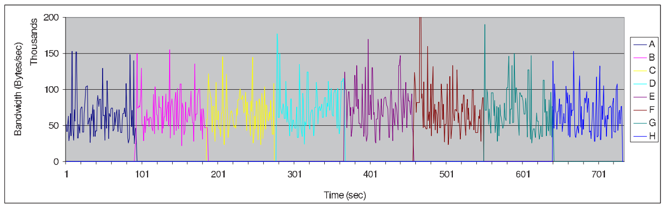

Fig. 7. Bandwidth consumption using a sender-based floor control with eight hosts participating in the conference.

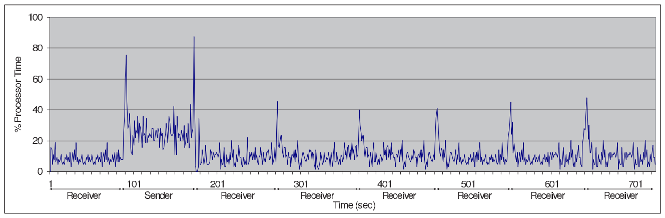

Fig. 8. CPU Utilization using floor control with hysteresis with eight hosts participating in the conference.

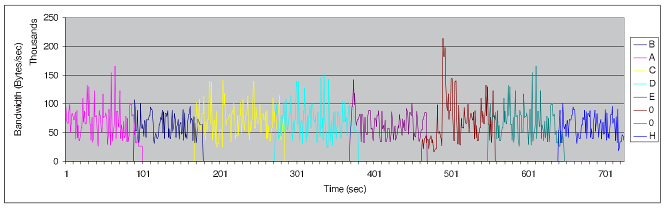

Fig. 9. Bandwidth Utilization using floor control with hysteresis with eight hosts participating in the conference.

being sent by the previous speaker and one sent by the new speaker. In Figure 11 we can observe that when the speaker changes, for a small period of time both the previous floor holder and the new floor holder are transmitting video.

B. Scalability

We measured the average CPU utilization and average bandwidth utilization with the number of participants in the conference session varying from 1 to 8. The conference sessions were using the original conferencing system with no floor control, with receiver-based floor control, with sender-based floor control, and with the floor control with *hysteresis*. Each session was run for approximately twelve minutes. As in the previous experiments, all video streams transmitted by any of the hosts at any time are transmitted at a bit rate of 500 kbps.

In the original conferencing system with no floor control, each participant was transmitting and processing all the incoming video streams. While using the receiver-based floor control, each participant transmitted video and received all the incoming video. Only the speaker's video stream is decoded. The turn to speak rotates among the participants in the conference in a round robin fashion and each participant would get an equal chance to speak.

Figure 10 shows the average processor utilization of each protocol with respect to the number of participants. As we can see, the original conferencing system cannot be used for conferences with more then eight participants unless more powerful workstations are employed. The receiver-based floor control avoids the problem of saturating the hosts. Since only a single stream is decoded, the average CPU power remains constant.

Both the sender based floor control and the floor control with hysteresis show a decrease in CPU utilization with increasing number of participants. This is due to the fact

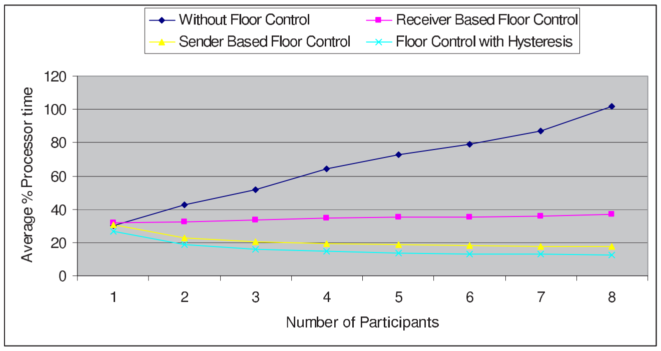

Fig. 10. Average CPU utilization with respect to the number of participants in the conference with i) without any floor control, ii) receiver based floor control, iii) sender based floor control, iv) floor control with hysteresis.

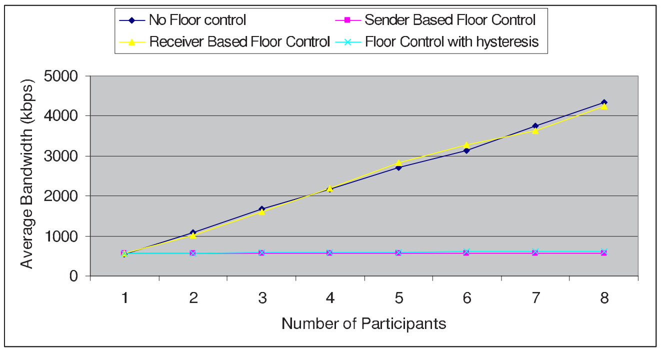

Fig. 11. Average Bandwidth Utilization with respect to the number of participants in the conference with i) without any floor control, ii) receiver based floor control, iii) sender based floor control, iv) floor control with hysteresis.

that a major chunk of the processing power is spent on video compression. In both of these protocols, a participant compresses video only upon becoming the floor holder. The amount of time a participant holds the floor is inversely proportional to the number of participants in the conference. (Assuming that each participant gets the same amount of time to hold the floor) However, even if a particular participant holds the floor throughout the entire conference session, the average CPU power will never exceed the power consumed using the receiver-based floor control.

Figure 11 shows the average bandwidth utilization with respect to the number of participants in the conference. As we can see the bandwidth consumption increases proportional to the number of participants in the conference in the case of no floor control and the receiver-based floor control. While using the sender-based floor control and floor control with hysteresis, the bandwidth is limited to a single video stream.

V. Conclusion

Videoconferencing provides an effective means for distributed remote real-time collaboration. The rapid growth of broadband multicast-enabled wide area networks such as the Internet2, makes it easier to deploy high quality video conferencing applications over the Internet. The limitations on bandwidth and processing capacity experienced by the participants of videoconferences can be overcome by floor control mechanisms, provided that they do not introduce significant disruption to the interaction among participants.

We presented a floor control protocol that can operate over the Internet and ensures that the processing and bandwidth overhead incurred by videoconferences with many sources is equivalent to that of a two-part videoconference. Our protocol uses a "*hysteresis effect*" to mask the latencies incurred by the system when floor hand offs occur, such that the smoothness of transition and the naturalness of the conversation are preserved, almost as if all participants were receiving the videostreams from all sources continuously.

A fertile area of research consists of exploring the use of different types of floors for audio and text, for example, as a means to further improve on the smoothness with which the floor for video is transitioned from one participant to another.

1) Receiver-based Floor Control: To minimize the utilization of bandwidth, video streams are compressed before being transmitted. In a typical conference, each participant transmits a video stream and receives the video streams from all other participants in the conference. The decompression at the receiver side is the most demanding for processing power, especially in a multi-party conference where each host has to process several incoming video streams. A fast machine may send audio and video streams that can overwhelm a slow machine, and as more and more participants join the conference, even the faster computers may not be able to handle the increasing workload. The higher the demand for quality, the higher the minimum processing speed required of each host.

A receiver-based floor control can be used to avoid saturating the end receivers. The receiver based approach assumes that the network bandwidth is not a bottleneck, and all participants transmit video all the time. The receiver based floor control serves as a means for the hosts to *filter* the incoming video (and audio) streams, depending upon the number of incoming streams and the current load ("*What I See Is What I Want*"). The streams from the current speaker are played out with the highest resolution and quality. Depending upon the current load, other incoming streams may be selectively decoded either at the same quality or at lower qualities and resolutions. In case of an overload, only a still image of the non-speakers may be shown.

2) Sender-based Floor Control: The receiver-based floor control can avoid the problem of saturation of the end receiver, but it still does not address the problem of saturating the network. High quality video streams consume significant bandwidth. Unless network bandwidth is very large, requirements for each stream limits the scalability of the conference. The sender-based floor control can be used to regulate bandwidth consumption. The current speaker is allowed to consume more bandwidth by transmitting video (and audio) at a higher quality and resolution. Depending upon the number of participants in the conference and the network load, passive participants (non-speakers) can transmit video at a lower quality and resolution.

In case of an acute shortage of bandwidth, passive members may just transmit still images at regular intervals or may just stop broadcasting any more video.

REFERENCES

[1] H. P. Dommel and J. J Garcia-Luna-Aceves, "Floor control for multimedia conferencing and collaboration," *Multimedia Systems (ACM/Springer)*, vol. 5, no. 1, pp. 23–28, 1997.

[2] H. P. Dommel and J. J Garcia-Luna-Aceves, "Comparison of floor control protocols for collaborative multimedia environments," *Proceedings of SPIE Symposium on Voice, Video and Data Communication*, November 1998.

[3] H. P. Dommel and J. J Garcia-Luna-Aceves, "A novel group coordination protocol for collaborative multimedia systems," *Proc. IEEE International Conference on Systems, Man, and Cybernetics 1998*, October 1998.

[4] V. Jacobson and S. McCanne, "Visual audio tool," *Lawrence Berkeley Laboratory*.

[5] V. Hardman, M. Sasse, and M. Handley, "A reliable audio for use over the internet," *Proceedings of INET*, June 1995.

[6] S. McCanne and V. Jacobson, "vic: A flexible framework for packet video," *Proceedings of ACM Multimedia 95*, November 1995.

[7] R. Frederick, "Experiences with real-time software video compression," *Proceedings of the Sixth International Workshop on Packet Video*, 1994.

[8] T. Turletti, "The inria videoconferencing system (ivs)," *ConneXions - The Interoperability Report Journal*, vol. 8, no. 10, pp. 20–24, October 1994.

[9] Internet2, ," *URL: http://www.internet2.org/*.

[10] MCI Worldcom Inc., "very-high-performance backbone network service (vbns).," *URL: http://www.vbns.net*.

[11] University Corporation for Advanced Internet Development, "Abilene," *URL: http://www.internet2.edu/abilene*.

[12] Microsoft Research Learning Sciences and Technology, "Conferencexp : Conferencing experience project," .

[13] J. Beavers, T. Chou, R. Hinrichs, C. Moffatt, M. Pahud, L. Powers, and J. V. Eaton, "The learning experience project: Enabling collaborative learning with conferencexp," Tech. Rep. MSR-TR-2004-42, Microsoft Research, Microsoft Corporation, April 2004.

[14] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "Rtp: A transport protocol for real-time applications.," *RFC 1889, Internet Engineering Task Force (IETF)*, January 1996.