Comparison of floor control protocols for collaborative multimedia environments

Hans-Peter Dommel and J. J. Garcia-Luna-Aceves

Baskin School of Engineering
Computer Engineering Department
University of California, Santa Cruz, CA 95064, USA

ABSTRACT

Improvements in networking allow for increasingly complex collaboration environments with regard to session scale, range of shared tasks, and distance between remote parties. Floor control protocols add an access discipline to such environments that allows to mitigate race conditions on shared resources and throttle media transmission. Primary causes for resource competition among users may be the lack of mutual awareness and formal session orchestration, or network and host limitations.

Various, often proprietary and unscalable solutions for floor control have been implemented for telemedicine, video conferencing, or distributed interactive simulation. To this date, an analytic comparison of the efficacy of these solutions is lacking. With efficacy, we mean the proportion of time that a protocol takes to allocate a resource, accounting for social and technical overhead from user behavior, protocol cost, and network conditions. We present a novel taxonomy and comparative performance analysis of known classes of floor control protocols, including socially driven protocols, collision sensing on shared resources, floor token passing in fully-connected and ring topologies, and, innovatively, across shared control trees. Accordingly, aggregated and selective transmission of control information over a multicast control tree offers the best scalability and efficacy. A novel hierarchical floor control protocol correlating in its operation with tree-based reliable multicast is outlined.

Keywords: Floor Control, Multimedia Collaboration, Reliable Multicast

1. INTRODUCTION

With IP-multicasting[1] more powerful collaborative multimedia applications (CMA) gradually enter mainstream computing. Users of such applications can overcome their separation in time and space and share work efforts in real-time, with the goal of approximating the quality of face-to-face interactions. While earlier CMA were proprietary, monolithic, and limited in media modality and session size, new applications support multi-party and multimodal collaboration in large sessions, such as distributed interactive simulations, distance learning seminars, or special-purpose MBone sessions.[2] However, compared to advances in reliable multicasting and multicast routing, there has been little progress with regard to group coordination support for such systems.

Floor control[3] is an access discipline for CMA, which may solve flawed telepresence and coordination problems as reported by Isaacs and Tang[4] in studies on video conferencing. Typically deployed in the session or application layer, floor control lets users attain exclusive control over a shared resource by attaining a *floor*, which is a short-lived synchronization primitive for multimedia objects. The floor semantics is generalized to multimedia from its traditional notion as the "right to speak".[5] Deployment of floor control in CMA may be complex due to system capabilities, in terms of the network, host, and communication subsystem, as well as user behavior. Early work on floor controlled systems has many different faces, including text-based remote collaboration,[6] electronic meeting support,[7] distributed teleconferencing,[8] moderation of MBone seminars,[2] or web-centric groupwork.[9] Users may profit from floor control, because it defines turn-taking rules, fosters interactivity, and prohibits unfairness. From a system perspective, it can provide quality-of-service input regarding admission control and bandwidth allocation for bulky media streams,[10] or serve as concurrency control for synchronization of multiple media flows.[11]

1. REPORT DATE **1998**	2. REPORT TYPE	3. DATES COVERED **00-00-1998 to 00-00-1998**
4. TITLE AND SUBTITLE **Comparison of floor control protocols for collaborative multimedia environments**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Santa Cruz,Department of Computer Engineering,Santa Cruz,CA,95064**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES **12**	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**			

A floor control methodology is characterized by several dichotomies, centralized vs. distributed implementation,[12] mechanism vs. policy,[13,14] and explicit vs. implicit floor hand-over.[13,15] In centralized systems, one host controls floor assignment for an entire session, which is easier to deploy, but suffers from overload and resilience problems. In distributed systems, each host in a session contributes to the global control process as much as its own resources are concerned. Consistency management and efficiency of communication are major concerns with this approach. Hybrid solutions take the best of both worlds and centralize tracking of single floors, while distributing the administrative load over the hosts in a collaborative session. Hybrid solutions are also well-suited for network computers, where hosts have limited capabilities and communicate with each other via session servers. A mechanism, e.g., activity sensing or token passing, is the physical propagation and synchronization apparatus for control information. It is instantiated with policies regarding service order, priorities, or fairness, such as "free-for-all", "moderated", "prioritized", "voice-activated", or "high-resolution". Policies allow to adjust floor control to the session style, e.g., a lecture, panel, or laboratory. The service order of queued floor requests, or the signaling on how the floor is attained has also been considered a policy.[13] In explicit floor passing, a user must submit a signal specifically to attain the floor, e.g., by raising the hand or pressing a button, in contrast to implicit passing, as it is the case with voice activation.

Despite this background of work on floor control, a detailed analysis on the operational principles and performance of floor control protocols is still missing. This paper presents a novel taxonomy and efficacy analysis of floor control protocols, based on their operational principles. Hierarchical floor control is described, operating on a control tree, which matches the backbone reliable multicast tree and multicast routing tree. The goal is to show how mechanisms for group coordination, that is floor control, fit in with the emerging large-scale Internet conferencing.

In Section 2 we present our system model and taxonomy, as the foundation for our comparative efficacy analysis presented in Section 3. Section 4 outlines the operation of a tree-based floor control protocol. The paper is concluded in Section 5.

2. SYSTEM MODEL AND TAXONOMY

We define important terms to lay the foundation for a common methodology and taxonomy of floor control protocols. A *collaboration environment* is a tuple $CE = (\mathcal{S}, \mathcal{U}, \mathcal{R}, \mathcal{F})$ consisting of hosts V connected by network links $E \subset V \times V$, users $\mathcal{U}$, shared resources $\mathcal{R}$ and floors $\mathcal{F}$. Communication among hosts is solely via message exchange, and not in shared memory. Links are assumed to be reliable for floor control messages. A *collaborative session* is a tuple $CS = (sid, \Delta, U, R, F)$, instantiating CE, and is characterized by a unique session identifier sid, session duration Δ, users U, resources R and their corresponding floors F. The session size $m = |U|$ can vary, depending on whether the session is open or closed, and in our terms $m > 100$ denotes a "large" session. Sessions may have different organizational styles, such as lecture, business meeting, panel discussion, or hearing. Users $u \in U$, which may be humans or system agents, can assume the control roles of *floor originator* (FO), owning a resource, *floor coordinator* (FC), moderating the floor of a resource, or *floor holder* (FH), having exclusive access to a resource. Depending on the session style, FH and FO may be identical. Shared resources $r \in R$, e.g., a robotic device, surgical instrument, video and audio channel, or graphical data object in the user interface, can be replicated or located at a specific host. Finally, we define a floor as a tuple

$$f(r, T, FC, FH, ta, \delta, QoS) = st \in F,$$

where r denotes the resource, T is the media type of r (video, audio, graphics, text), FC and FH identify the floor coordinator and holder, ta is a timestamp marking the start of the floor holding time δ, and QoS is a quality-of-service directive indicating delivery properties for r. The floor state $st = [free \mid busy \mid idle \mid requested \mid nil]$ records the current state of floor f at a host. For simplicity, we assume that there is one user per host, and one floor f per resource r, without finer granularity.

Floor control protocols rest on three operational pillars: (1) the random or scheduled access to resources, which is characterized by the mechanism and host topology, relegating directives among hosts; (2) the centralized or distributed control over floors in the session; and (3) the floor policies established among hosts. These properties are reflected in the taxonomy in Figure 1 (only paradigms indicated with solid lines are analytically compared.) We divide known floor control paradigms in two classes:

Random-access Group Coordination (RGC) lets users contend for a shared resource, either mediated by social protocols, or by remotely sensing its status before or during resource access. Sensing is accomplished either by users, tracking each other's activities through the user interface, or by the system, sensing the state of an application, host, or network for local and remote activities. The global floor state is marked with assertions on local variables and no token entity is explicitly exchanged as a placeholder. RGC schemes are inherently contention-based, because
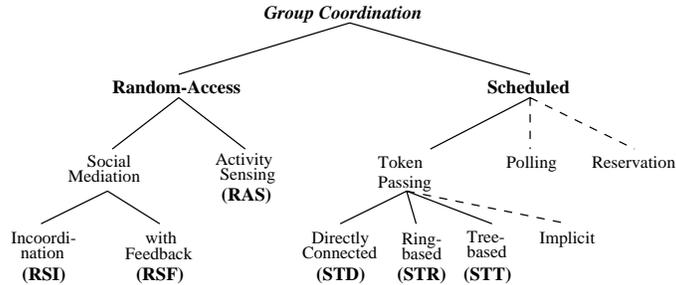
Figure 1. Taxonomy of group coordination.

hosts must actively compete for a floor. Remote sensing can be costly and inefficient. In RGC, floor acquisition is characterized by "sensing", "busy", "active", and "idle" states.

Scheduled Group Coordination (SGC) uses floor token passing, reservation or polling for pending control directives as resource access mechanism. The token is a unique placeholder, which is used to request, deny, reserve, or grant a floor. Regulated floor token capture disperses race conditions on resources. Hosts can "ask" for the floor, or a token circulates among hosts and is "offered" to hosts. Token tracking and ensuring authenticity and consistency can be costly. SGC schemes typically operate on a logical host topology such as a ring or tree. In SGC, floor acquisition is characterized by "request", "deny", "grant", and "release" control messages. Predominant session geometries are shown in Figure 2. Resource r_i in the shared workspace is hosted by user u_i.
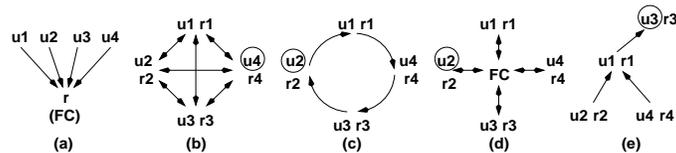
Figure 2. Session geometries.

In Fig. 2(a), users u_i attempt to gain control of a centralized resource r with limited or no coordination. In Fig. 2(b), participants u_i communicate directly with one another to attain the floor in the most direct manner. This scheme is also suited for achieving distributed consensus, e.g., with voting. Fig. 2(c) depicts ring-based control, based on the idea of passing a floor token in a linear fashion along the hosts in the ring. In Fig. 2(d), a star-topology is used to concentrate floor requests onto a dedicated FC host, which is a subcase of Fig. 2(e). In this tree structure, control directives are propagated along the branches of a multi-level control tree towards the current FH.

3. EFFICACY OF FLOOR CONTROL PROTOCOLS

The following comparative analysis of known classes of floor control protocols is a first attempt to characterize the efficacy of interactive behavior of people and processes from a resource contention perspective. The intention is not to predict exactly how a protocol performs for a given *CE*; accomplishing this would require far more host- and network-specific details and statistics on user behavior. Our goal is simply to assess the overhead of various protocols with regard to control state management. The basic methodology is derived from multiaccess communication,[16] based on the analogy between access mitigation for the data channel and shared resources.

We state the following assumptions to make the analysis tractable: the individual processing cost for control packets, including protocol overhead and user-interface specifics, is the same for all hosts; control message delivery between hosts is reliable and no failures in hosts or the network occur; information from a host reaches any other host with the same average system-wide propagation delay; and the interarrival rate of floor requests is Poisson, given that there is no indication for cross-correlations between subsequent floor requests. We take the interarrival rate of floor requests, the task length, and the network propagation delay into account. An important aspect of our

comparative analysis is the impact that multicasting at the network layer has on the efficacy of the floor control protocols. When network multicasting is not available, the user hosts are forced to contact one another explicitly, which substantially increases the processing overhead at hosts and the possibility of disagreement on which host has the floor. We model the existence of multicasting at the network layer by assuming that a user host needs only to pass information once to the network (during an activity period and to gain the floor) for the information to reach all other hosts with the same average delay. The notation used in our analysis is summarized in Table 1.

β_1	average "think" time before floor token arrival
β_2	average "think" time at floor token presence
γ	average processing time for a floor directive
δ	duration of average activity period
$n\epsilon$	processing and unicasting overhead for n hosts
η	efficacy of a floor control protocol
G	average offered floor request load
ι	average duration of idle time
λ	floor request interarrival rate
m	average number of hosts in session
n	average number of active hosts in session
ν	average vulnerability period
τ	average propagation delay

Table 1. Analysis parameters.

The "think time", denoted with β_1 (the time until the expected floor arrives at the local host) and β_2 (the time once the floor token is present at the local host and offered to the user), reflects user choices to grab the floor in token rings; γ is the average time to process and communicate a floor directive, including packetization delay, generic queueing transmission times, and local processing overhead; δ is the duration of an activity period; we use $n\epsilon > \tau$ as the additional delay to account for the communication and processing overhead in n actively collaborating hosts; $G = \delta \times \lambda$ is the normalized offered request load on floors, including new and previously denied and resubmitted floor requests; ι sums up the expected idle time for a resource during and after floor holding time; λ represents the relative frequency of demand for resource access and thus indicates the contention level; $m < \infty$ denotes all hosts in a collaborative session; $n < m$ is the number of active hosts in the multicast group for the current floor and transmission; ν is the time during which a host's attempt to access a resource can be intercepted by another host; and τ denotes the average end-to-end delay between hosts, coalescing multiple routing hops into one (a packet must hence traverse on the average the same number of hosts on the path from the sender to a group of receivers.) The activity period and time to process and communicate floor directives are assumed to include the time incurred in providing feedback among hosts.
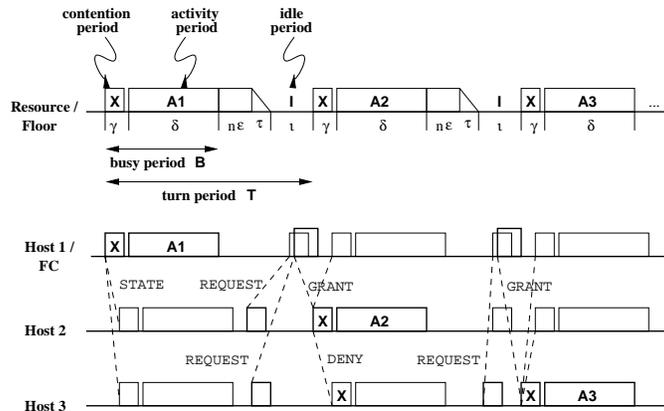
Figure 3. Turn taking periods for a resource and three hosts.

A turn taking model reflecting the switching of control over a resource by users serves as the blueprint for our analysis. A typical communication pattern between three users is depicted in Figure 3. Host 2 and 3 contend here

for the floor from host 1, and first host 2 acquires the floor, then host 3. A turn consists accordingly of a resource contention time X, an activity (floor holding) time A, and an optional idle time I. Based on this abstraction, we define the *efficacy* of a floor control protocol, denoted by η, as the proportion of time that a protocol needs to allocate a resource, including overhead from the protocol itself, the network, and user behavior. Formally, the efficacy is the ratio of the average floor usage time $\bar{U}$ vs. the overall average turn length $\bar{T} = \bar{X} + \bar{A} + \bar{I}$, given by Eq. 1. The average contention period $\bar{X}$ and activity period $\bar{A}$ together are called the average busy period, $\bar{B} = \bar{X} + \bar{A}$. These turn periods serve as the building blocks for our efficacy analysis, considering both point-to-point and broadcast style communication.

$$\eta = \frac{\bar{U}}{\bar{T}} = \frac{\bar{U}}{\bar{B} + \bar{I}} \tag{1}$$

3.1. Random-Access Group Coordination Schemes

Incoordinated Social Mediation (RSI) reflects purely random resource access in self-moderating sessions. There is no system support to mediate conflicts and ensure system-wide resource consistency. Assuming that all information is sent reliably to all user hosts, the latency introduced by the system can lead to inconsistent views of the floors at different hosts and corrupt user cooperation. When this occurs, we assume that all users involved in the conflict must restart their activities.

THEOREM 1. *The efficacy of RSI without multicast support is*

$$\eta_{RSI} = \lambda(\delta + n\epsilon)e^{-2\lambda(\delta + n\epsilon)} \tag{2}$$
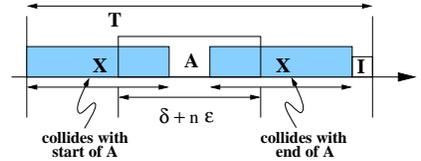
Figure 4. Typical RSI timeline.

Proof: Figure 4 shows a prototypical RSI turn. The proof is the same as for medium access in an ALOHA channel.[16] In the point-to-point model, we assume that n hosts are actively monitoring each other, and it takes an additional time $n\epsilon$ for all hosts to perceive the activity from a given host. All the information is exchanged reliably, and the average vulnerability interval is twice the total of the task length and the added overhead incurred in serial communication of the task information to all hosts (i.e., $\delta + n\epsilon$), because messages can intercept activities any time. Because request arrivals are Poisson, the probability that a task is successful is $e^{-2\lambda(\delta + n\epsilon)}$. The success probability times the number of arrivals in one activity period results in Eq. 2. □

COROLLARY 1. *The efficacy of RSI with multicast support is*

$$\eta_{RSI}^{MC} = \lambda\delta e^{-2\lambda\delta} \tag{3}$$

This follows under the assumption that every update to and from a host requires only one transmission.

Social Mediation with Feedback (RSF) assumes that users cooperate based on social protocols. Feedback on remote activities is gathered through the user interface. If a user contends for a floor and perceives remote activity, she would back off for a random period and attempt to reclaim the floor, after remote activity subsided. RSF in video conferencing is often realized as "voluntary distributed control",[15] where cooperative users switch video streams manually on and off, depending on whether they prefer to receive or send specific video transmissions. POTS conferencing also relies on RSF, which works well for very small groups.

THEOREM 2. *The efficacy of RSF without multicast support is*

$$\eta_{RSF} = \frac{\delta}{\delta + e^{2\lambda(\gamma' + n\epsilon)}\left[\frac{e^{\lambda(\gamma' + n\epsilon)} - 1 - \lambda(\gamma' + n\epsilon)}{\lambda(\gamma' + n\epsilon)(1 - e^{-\lambda(\gamma' + n\epsilon)})} + \gamma' + n\epsilon + \tau + \iota + \frac{1}{\lambda}\right]} \tag{4}$$
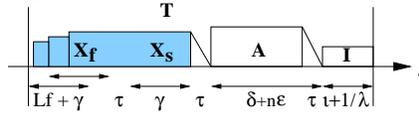
Figure 5. Typical RSF timeline.

Proof: A prototypical timeline of RSF is depicted in Figure 5. n active hosts need to sense and process remote activity through the network interface. γ' is the user time to remotely sense the resource state. The success probability, denoted as P_s, equals the probability that no activity packet arrives in an average vulnerability period ν of $2(\gamma' + n\epsilon)$ sec., i.e., $P_s = P[0 \text{ packets in } \nu] = e^{-2\lambda(\gamma' + n\epsilon)}$. The average utilization period lasts $\bar{U} = \delta P_s$, and the length of the average busy period $\bar{B} = \bar{X} + \bar{A}$ is determined by the time needed to handle unsuccessful floor requests in the failed contention period X_f and successful requests in X_s, with $\bar{B} = (1 - P_s)X_f + P_s X_s$. An average failed turn attempt consists of a geometrically-distributed indefinite number $(\bar{L})$ of interarrival times of floor requests with duration $\bar{f}$ sec (average time between failed floor-request arrivals), plus the duration observing a request (γ'). The values for $\bar{L}$ and $\bar{f}$ have been derived by Takagi and Kleinrock.[17] Substituting our notation in these results, we obtain $\bar{L} = e^{\lambda(\gamma' + n\epsilon)}$ and $\bar{f} = (\lambda(\gamma' + n\epsilon))^{-1} - e^{-\lambda(\gamma' + n\epsilon)}/(1 - e^{-\lambda(\gamma' + n\epsilon)})$, respectively. Accordingly, the average time of a failed turn attempt equals $X_f = \left[\frac{e^{\lambda(\gamma' + n\epsilon)} - 1 - \lambda(\gamma' + n\epsilon)}{\lambda(\gamma' + n\epsilon)(1 - e^{-\lambda(\gamma' + n\epsilon)})} \right] + \gamma' + n\epsilon + \tau$. An average successful turn lasts $X_s = \delta + \gamma' + n\epsilon + \tau$. Finally, based on the Poisson assumption, an idle period consists of an average idle time interval plus the time until the next floor request arrives on the average, $\bar{I} = \iota + \frac{1}{\lambda}$. Substituting into Eq. 1, we obtain Eq. 4. $\square$

COROLLARY 2. *The efficacy of RSF with multicast support is*

$$\eta_{RSF}^{MC} = \frac{\delta}{\delta + e^{2\lambda\gamma'} \left[\frac{e^{\lambda\gamma'} - 1 - \lambda\gamma'}{\lambda\gamma'(1 - e^{-\lambda\gamma'})} + \gamma' + \tau + \iota + \frac{1}{\lambda} \right]} \tag{5}$$

With multicast support, the vulnerability period reduces to $2\gamma'$, and ϵ becomes negligible.

In *Activity Sensing* (RAS),[14,18] activities on shared resources are monitored by a background process at the session layer (without the user having to do so), in order to sense which host currently operates on the resource. The RAS concept is related to collision sensing on a multiaccess channel.[16] In principle, no changes to a collaboration-unaware application are required, because a modified X-server would intercept and filter calls to shared applications. The RSF system agent would back off for the user, when perceiving remote activity, or allow immediate access to the resource otherwise. Distributed activity sensing agents collectively monitor resource states more accurately than humans could.

THEOREM 3. *The efficacy of RAS without multicast support is*

$$\eta_{RAS} = \frac{\delta}{\delta + e^{2\lambda(\gamma + n\epsilon)} \left[\frac{e^{\lambda(\gamma + n\epsilon)} - 1 - \lambda(\gamma + n\epsilon)}{\lambda(\gamma + n\epsilon)(1 - e^{-\lambda(\gamma + n\epsilon)})} + \gamma + n\epsilon + \tau + \iota + \frac{1}{\lambda} \right]} \tag{6}$$
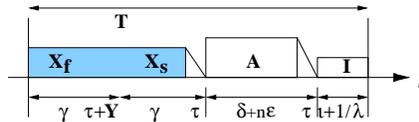
Figure 6. Typical RAS timeline.

Proof: The timeline is shown in Figure 6. γ is the system time to sense the resource state. Without multicast support, a host has to send floor directives individually to every other host, which according to our assumption takes $\gamma + n\epsilon$. Because multiple unicast messages are used by each host, floor-state inconsistencies can arise during the entire time. The host is exchanging floor directives, i.e., the vulnerability period of the protocol is $\nu = 2(\gamma + n\epsilon)$. Using this vulnerability period, Eq. 7 follows using the same approach as described in the proof of Theorem 4.

THEOREM 4. *The efficacy of RAS with multicast support is*

$$\eta_{RAS}^{MC} = \frac{\delta}{\delta + \tau + \frac{1}{\lambda} + e^{\lambda\tau}(\gamma + 2\tau + \iota)} \tag{7}$$

Proof: The access strategy is assumed as nonpersistent, i.e., a host backs off immediately from attempting to access a resource and claims the resource once it appears free again. The vulnerability period for accessing an unused resource is one propagation delay, $\nu = \tau$, within which other hosts can cause a conflict (opposite to RSF, where twice the length of the contention interval is the average vulnerability period); therefore, $P_s = P[0 \text{ packets in } \nu] = e^{-\lambda\tau}$. The average utilization period is $\bar{U} = P_s\delta$. The average length of a successful busy period is simply $\gamma + \delta + 2\tau$, which accounts for the delivery and processing of floor directives, the activity period, and associated network latencies. The length of an average unsuccessful activity period consists of one truncated activity lasting γ sec, followed by one or more similarly truncated activities sent within time Y sec, where $0 \le Y \le \tau$. The expected value of Y is[17] $\bar{Y} = \tau - \frac{1}{\lambda}(1 - e^{-\lambda\tau})$; therefore, the average duration of a failed contention period is $\gamma + 2\tau - \frac{1}{\lambda}(1 - e^{\lambda\tau})$. The length of the average busy period is then $\bar{B} = \gamma + 2\tau - \frac{1}{\lambda} + e^{-\lambda\tau}(\delta + \tau + \frac{1}{\lambda})$. The average idle interval is again $\bar{I} = \iota + \frac{1}{\lambda}$. Substitution into Eq. 1 yields Eq. 7. □

3.2. Scheduled Group Coordination Schemes

In contrast to the contention time in RGC, γ in SGC denotes the time to transmit a floor token to the next host. Two SGC approaches, *polling*, and *reservation*, have previously not been used in telecollaboration. Polling involves long wait times, and reservation schedules can quickly become obsolete. We focus on *token passing*, where the floor is being asked for, or offered to hosts in a predefined service order. We discuss three main cases of control topologies for hosts.

In *Direct Coordination* (STD), each host in a group is fully connected to every other host. STD improves the response time, however, the number of links is $\frac{n(n-1)}{2}$ and grows as the square of the number of hosts in the session. Many small-scale commercial video conferencing systems follow this unscalable model.

THEOREM 5. *The efficacy of STD without multicast support is*

$$\eta_{STD} = \frac{\delta}{\delta + n(\gamma + \tau + \epsilon) + \iota + \frac{1}{\lambda}} \tag{8}$$
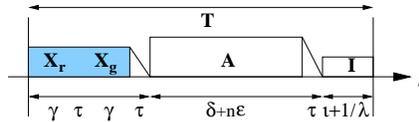
Figure 7. Typical STD timeline.

Proof: The timeline for STD is shown in Figure 7. The average utilization period is $\bar{U} = n\delta P_s$, because floor capture is perfect and any one of the n active hosts can acquire a floor of holding time δ with success probability $P_s = \frac{1}{n}$. A floor may not be released at FH, until successor FH' received it. The control packet overhead is γ and the propagation delay is τ. Unicasting a floor request to the $n - 1$ active hosts amounts to $(n-1)(\gamma + \tau + \epsilon)$, plus $(\gamma + \tau + \epsilon)$ for a reply. The average activity duration is $\bar{A} = \delta$ and may be trailed by an idle interval consisting of a period ι and an average interarrival time for all hosts, $\bar{I} = \iota + \frac{1}{\lambda}$. Substitution into Eq. 1 results in Eq. 8. □

COROLLARY 3. *The efficacy of STD with multicast support is*

$$\eta_{STD}^{MC} = \frac{\delta}{\delta + 3(\gamma + \tau) + (n-1)\epsilon + \iota + \frac{1}{\lambda}} \tag{9}$$

With multicast support, the request-reply-release exchange of control packets takes a time of $3(\gamma + \tau) + (n-1)\epsilon$, if we assume that every host incurs host processing overhead from its $n - 1$ neighbors.

In *Ring-based Coordination* (STR), a floor token cycles through a logical ring arranging hosts. Various systems[19,20] based on this idea have been discussed. A host that is ready to start an activity, captures the passing token, inserts

a command sequence with address and control information, sends the activity packets within this turn period and transfers the token after completion to the successor host. A host without pending floor requests passes on the offered token. Tokens held in excess time may expire and incite automatic transfer. The predefined token passing schedule may not reflect spontaneous interactivity. The floor can be granted ahead of its token position to a successor host, or it may only be acquired by a host when the token passes through that host. Likewise, a token can be immediately released after transmission (RAT), or released after one more reception (RAR) at the sending host. For efficiency reasons we focus on RAT. In our model, the pre-arrival think time β_1 plus the token-presence time $\beta_2 < \beta_1$ must be smaller than the ring cycle time. If the floor is taken, then $\beta_2 \approx (\delta + \tau + \gamma)$.

THEOREM 6. *The efficacy of STR without multicast support is*

$$\eta_{STR} = \frac{\delta(1 - e^{-\lambda(\beta_1+\beta_2)})}{\frac{n}{2}(\tau + \gamma + \epsilon + \beta_2) + \delta(1 - e^{-\lambda(\beta_1+\beta_2)}) + \iota + \frac{1}{\lambda}} \tag{10}$$
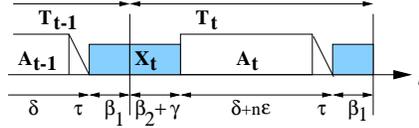
Figure 8. Typical STR timeline.

Proof: The timeline for STR is depicted in Figure 8. We assume perfect floor capture and only one host is active at any time. The average utilization is $\bar{U} = \delta P_s$, with P_s as the success probability that the token is available in the period $\bar{X} = \beta_1 + \beta_2$. The probability that a floor can be claimed by a user is $P_s = 1 - e^{-\lambda(\beta_1+\beta_2)}$. The token cycle time is a function of the active host set n, and with growing session size it is less likely that all hosts will engage in floor contention. The cost to transfer a floor token in a cycle involves on the average $\frac{n}{2}$ hosts, amounting to $\frac{n}{2}(\gamma + \tau + \epsilon + \beta_2)$ including processing overhead $\frac{n}{2}\epsilon$. The average turn lasts hence $\bar{T} = \frac{n}{2}(\gamma + \tau + \epsilon + \beta_2) + \bar{A} + \bar{I}$. The idle time is again $\bar{I} = \iota + \frac{1}{\lambda}$. Substituting $\bar{U}$ and $\bar{T}$ into Eq. 1 yields Eq. 10. The overhead to maintain the token is not included in this result. □

COROLLARY 4. *The efficacy of STR with multicast support is*

$$\eta_{STR}^{MC} = \frac{\delta(1 - e^{-\lambda(\beta_1+\beta_2)})}{\frac{n}{2}(\tau + \gamma + \beta_2) + \delta(1 - e^{-\lambda(\beta_1+\beta_2)}) + \iota + \frac{1}{\lambda}} \tag{11}$$

With multicasting, a token is sent only once to the network interface, but it takes on the average $\frac{n}{2}$ hops to cycle back for another turn option, however, the processing overhead ϵ vanishes.

Tree-based Coordination (STT) allows for more efficient inter-group collaboration and hierarchical mixing of media sources.[21] Its control infrastructure can work in symbiosis with reliable multicasting over a single shared acknowledgment tree,[22] allowing more scalable and economic transmission of session data. STT control messages traverse branches of the tree in a parent-child relation reflecting multicast group membership. Each host must only deal with messages from immediate neighbor hosts. Floor directives can be coalesced into single messages, and can be forwarded through the tree in an aggregated fashion. Single hosts will hence not suffer from message implosion, a problem known from reliable multicast with regard to acknowledging received or lost messages. The communication delay depends on the height of the control tree, rather than the session size. The star topology is a special case of a tree, which works for small sessions.

THEOREM 7. *The efficacy of STT without multicast support is*

$$\eta_{STT} = \frac{\delta}{(K+1)\bar{P}(\gamma + \tau + \epsilon) + (\delta + \bar{P}\tau) + \iota + \frac{1}{\lambda}} \tag{12}$$

Proof: The timeline for STT is depicted in Figure 9. We have again perfect floor capture due to explicit token exchange, with $\bar{U} = n\delta P_s$, and $P_s = \frac{1}{n}$. With the normalized average path length $\bar{P}$, the average duration of the floor capture period amounts to $2\bar{P}(\gamma + \tau + \epsilon)$. Assuming that a host does not know the location of FH or FC,
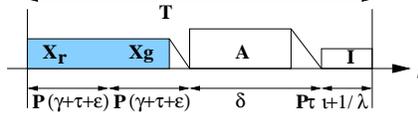
Figure 9. Typical STT timeline.

exploratory unicast of a control message from one host to its parent and to each of its children would amount to $\bar{X} \leq (K+1)\bar{P}(\gamma + \tau + \epsilon)$ plus a targeted reply costing $\bar{P}(\gamma + \tau + \epsilon)$. However, as outlined in Section 4, hosts can be tagged with unique labels, which allow for efficient absolute or relative routing of control directives toward the FC or FH. Consequently, a control directive must be sent only to one neighbor node as the gateway on the path to FH, hence $K = 1$. Signaling the conclusion of the activity period adds another propagation delay, $\bar{A} = \delta + \bar{P}\tau$. The activity period may be trailed by another idle period of average length $\bar{I} = \iota + \frac{1}{\lambda}$. Substituting into Eq. 1 gives Eq. 12. $\square$

COROLLARY 5. *The efficacy of STT with multicast support is*

$$\eta_{STT}^{MC} = \frac{\delta}{\delta + 2\gamma + 3\tau + \iota + \frac{1}{\lambda}} \tag{13}$$

With multicasting, a request-reply pair takes two γ and τ, plus another τ to signal completion of the turn. A close correlation between the control tree of the STT protocol and the end-to-end multicast tree is assumed. A host sends only one message to the network interface, i.e., $\bar{P} = 1$, $K = 1$, and ϵ becomes negligible.

3.3. Results

We compare η^{MC} of the discussed paradigms in four cases: (1) a small group in a network with low link latency; (2) a small group in a high-latency network; (3) a large group in a low-latency network; and (4) a large group in a high-latency network. Group sizes are $n = 5$ (small) and $n = 300$ (large), corresponding to PC-conferencing systems, or average MBone session size. Latency is indicated by $\tau = 0.005$ s (low), and $\tau = 0.4$ s (high). The time to sense floor information is $\gamma' = 0.25$ s. A control packet of 25 bytes length is measured with $\gamma = \epsilon = 0.02$ s. The normalized activity time is $\delta = 1$ and token-ring "think times" are set to $\beta_1 = \frac{\delta}{2}$ s and $\beta_2 = \frac{\delta}{10}$ s. The typical idle time is chosen as $\iota = \frac{\delta}{5}$ s. Figures 11 and 10 plots the resulting efficacy.
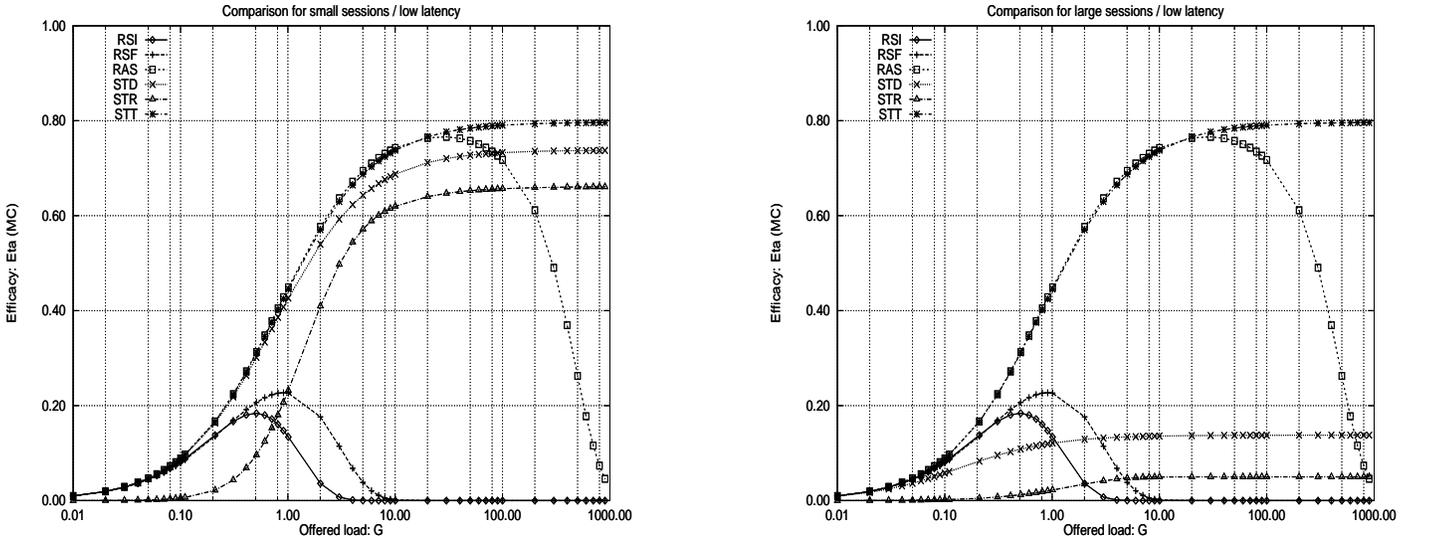
Figure 10. Efficacy with multicast support for low network latency.

The efficacy of RSI is below 20% in all four scenarios. Systems employing RSF benefit from coordination attempts and improve slightly over RSI, approximating 25% efficacy in networks with faster links. Both schemes are instable
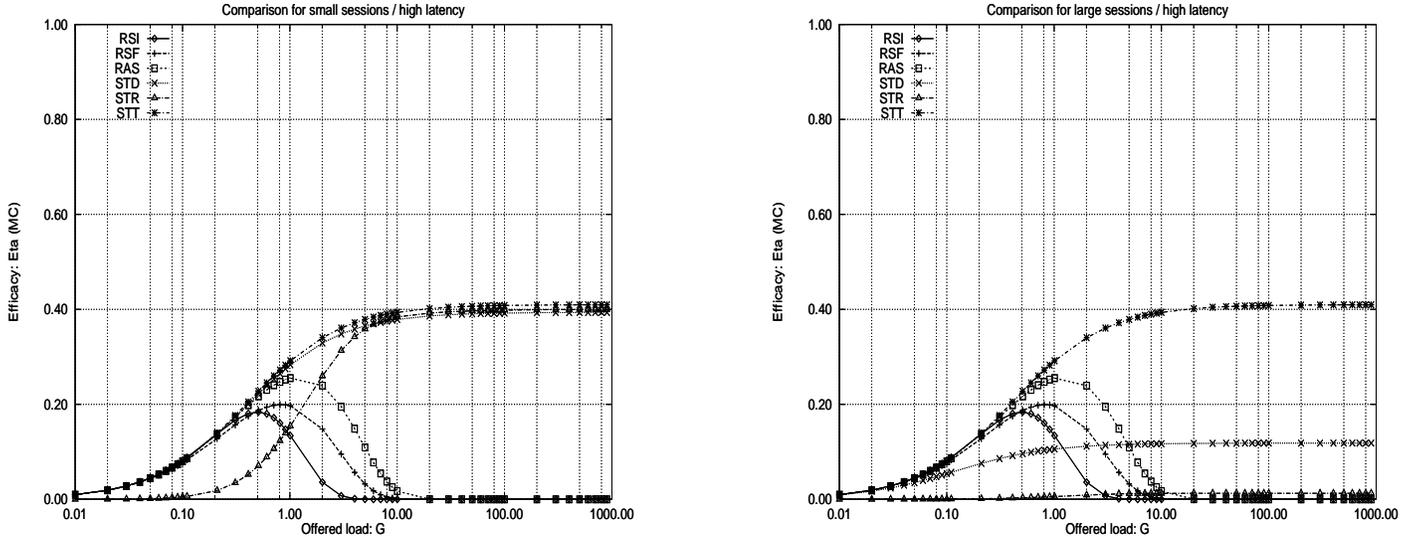
Figure 11. Efficacy with with multicast support for high network latency.

even at low request rates and collaboratory efficacy falls off quickly and the average delay becomes unbounded. This models the phenomenon that users avoid frequent turn-taking in telecollaboration.[4] RAS was a first attempt toward unintrusive machine-assisted floor control and performs very well for local area networks, where the system's responsiveness warrants quick updates and steady resource tapping. For high link latencies it provides only slightly higher loads than RSF, however, it also degenerates quickly and rises barely above 20%. STD performs well for small groups, where a host needs only to send few packets to the session remainder, but despite stability for higher loads it barely exceeds 15% efficacy in large sessions. In its best case of small sessions and low network latency, STR approximates 65% efficacy and it is generally stable, but degrades with rising scale and delay. In particular, it collapses for large sessions with high latency. STT shows the best overall behavior, both in terms of scalability and stability. It reaches up to 80% efficacy in low latency networks, and about 40% efficacy for high link latencies, independent of session size. According to these results, STT fills a gap and is particularly well-suited for Internet collaboration in large groups.

4. A TREE-BASED FLOOR CONTROL PROTOCOL

The protocol outlined in this section presents two innovations: floor control is inherently hierarchical, and the control tree for group coordination is correlated in its operation to an underlying tree-based multicast service, as outlined with the Lorax protocol.[22]   In this protocol, a single shared acknowledgment (ack) tree per session is used to disseminate information reliably among hosts on the tree. Hierarchical acknowledgments (hacks) are used to avoid the situation that the source is contacted by all hosts requesting retransmissions of lost packets. Lorax introduces recursive top-down labeling of tree nodes such that a label $l(x)$ of node x at level h in the tree is a prefix of its children's labels at level $h + 1$. These address labels can be used for absolute or relative self-routing of packets within a session and its multicast groups. Adding a node in the tree involves only the new node as a child and its parent, while deletions require relabeling of the subtree of the deleted node. The label cardinality depends on the tree branching factor and the session size.

The floor control tree mirrors the Lorax hack tree, sparing the floor control protocol from managing its own dissemination infrastructure, however, the propagation mechanisms is geared toward cascaded processing of control directives (CDs) for resource sharing, rather than recovery of lost transmissions. A CD contains label information on senders, receivers, a timestamp, time-to-live field TTL indicating scope and persistence of the CD, a privacy level indicator, and floor descriptor including FH, priority, resource and current state. Standard CDs are REQUEST, GRANT, DENY, RELEASE, or STATE UPDATE. The current floor state is tracked distributedly, and hosts on the tree store the label of FH for each floor locally for efficient addressing. Communication about the floor state is retained in local groups, and CDs of a node x are aggregated and responded to by its immediate neighbors, if these nodes can provide an up-to-date response to a query. Call setup, late joining and withdrawal from a session are handled by a membership

protocol interfacing with the protocol. Hosts in a session can assume one or more of the following control roles in the tree: a *coordinator node* hosts the FH for a resource r; *relay nodes* collect CDs from their children, forward them in the tree towards the FH, and relay replies back to their children; and *leaf nodes* delimit tree branches, comparing on the average more bits in the control routing procedure, than nodes closer to the root. Address labels allow to maintain only one logical control tree for all floors in a session, instead having to maintain separate trees, one per floor.

The protocol operates in a setup, active and teardown phase. *Setup* is initiated at session start or when a node x joins a session. Floor state information relevant for locally shared resources is retrieved by x from its neighbor nodes and new resources are advertised to the session. The *active phase* concerns aggregation and self-routing of floor information in local groups. A request is sent to FH by comparing prefix(l(x)) with prefix(l(FH)), and transmitted by relay nodes on the path to FH. When the floor is granted to x, all hosts in the session receive a multicast update on $l(FH) = l(x)$ and submit their directives to x for a new turn. A floor transfer between FH and its successor FH' must be confirmed, or will not be enacted. In *teardown*, which is initiated when a session terminates, a host withdraws or fails, floor state information is retained in a log file, but deleted from records in active session hosts.
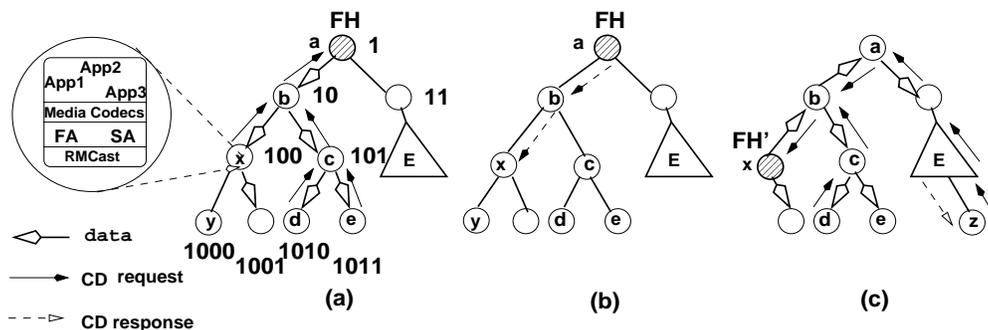
Figure 12. Sample HGCP scenario.

Figure 12 illustrates the operation of the protocol, indicating collocation of floor control with session control, and interfacing with reliable multicast below and applications above. Hollow arrowheads indicate data multicasting, and the other arrows indicate CDs. In scenario 12(a), initially $l(FH) = l(a) = 1$, which operates on a resource r and transmits updates to selected session members. Nodes x and e contend for the floor of a shared resource r. Looking up the FH entry for the resource in question, they send a REQUEST CD to their parent, because $l(FH) = prefix(l(e))$, and $l(FH) = prefix(l(x))$. For the parent node c of e, $l(FH) = prefix(l(c))$, and similarly, $l(FH) = prefix(l(b))$. Hence, both requests cascade upward in the tree toward the root. Node b is relay node for x and e and either already forwarded the request from x to a, if CD(e) arrives after CD(x), or it aggregates both CDs into one request, and propagates it to a. Assume that a satisfies the request from x first, sending a GRANT CD across b to x. Once x confirms reception of the floor, node a multicasts an update with label information $l(FH') = l(x) = 100$ to the remainder of the session, indicating the start of a new turn. In scenario 12(b), x is the new FH', starts using r and multicasts updates to its children. All hosts in the multicast group propagate their CDs towards the location of FH', as indicated in scenario 12(c), where node y dropped out of the session and all its shared resources r_i, with $FO(r_i) = y$, are withdrawn and floor states tables across the session are marked up accordingly. A new node z in subtree E also joins the session and sends a STATE CD to its parent, retrieving the current state table for its local resources shared with the session. A more detailed efficacy analysis and protocol description can be found in Ref. 23.

5. CONCLUSION

System support for improved telepresence and group coordination in collaborative multimedia systems is still in its infancy. This paper focused on a comparative analysis of floor control protocols, merging time aspects of protocol operations with end-user behavior and thus accounting for both internal and external factors regarding control of shared resources. This is to our knowledge the first attempt to quantify the effectiveness of various floor control mechanisms. To keep our bare-bones analysis tractable, we needed to make several strong assumptions to unify various strategies of floor management in one framework. A basic mechanism for tree-based floor control has been outlined, which operates in close correlation to a reliable multicasting protocol. Our conjecture is that hierarchical floor control is more scalable and efficient than previous paradigms of group coordination.

ACKNOWLEDGMENTS

REFERENCES

1. S. Deering, "Host extensions for IP multicasting." RFC-1112, August 1989.

2. R. Malpani and L. Rowe, "Floor control for large MBone seminars," in *Proc. ACM Multimedia*, pp. 155–163, (Seattle, WA), Nov. 1997.

3. H.-P. Dommel and J. J. Garcia-Luna-Aceves, "Floor control for multimedia conferencing and collaboration," *Multimedia Systems J. (ACM/Springer)* **5**, pp. 23–38, Jan. 1997.

4. E. A. Isaacs and J. C. Tang, "What video can and cannot do for collaboration: a case study," *Multimedia Systems J.* **2**, pp. 63–73, Aug. 1994.

5. H. M. Robert, *Robert's rules of order*, Bantam Books, Toronto; New York, 1986.

6. S. Sarin and I. Greif, "Computer-based real-time conferencing systems," in *Computer-Supported Cooperative Work: A Book of Readings*, pp. 397–420, Morgan-Kaufman, 1988.

7. M. A. Stefik, G. Foster, D. Brobrow, K. Kahn, S. Lanning, and L. Suchman, "Beyond the chalkboard: Computer support for collaboration and problem solving in meetings," *Comm. ACM* **30**, pp. 32–47, Jan. 1987.

8. L. Aguilar, J. J. Garcia-Luna-Aceves, D. Moran, E. Craighill, and R. Brungardt, "Architecture for a multimedia teleconferencing system," in *Proc. ACM SIGCOMM*, pp. 126–136, Aug. 1986.

9. S. Shirmohammadi, J. C. D. Oliveira, and N. D. Georganas, "Applet-based telecollaboration: a network-centric approach," *IEEE Multimedia J.* **5**, pp. 64–73, April-June 1998.

10. E. Amir, S. McCanne, and R. Katz, "Receiver-driven bandwidth adaptation for light-weight sessions," in *Proc. ACM Multimedia*, (Seattle, WA), Nov. 1997.

11. R. Yavatkar, "MCP: A protocol for coordination and temporal synchronization in multimedia collaborative applications," in *Proc. of the 12th IEEE Int. Conf. on Distributed Computing Systems*, June 1991.

12. K. A. Lantz, "An experiment in integrated multimedia conferencing," in *Computer Supported Cooperative Work: A Book of Readings*, pp. 533–556, Morgan-Kaufman, 1988.

13. T. Crowley, P. Milazzo, E. Baker, H. Forsdick, and R. Tomlinson, "MMConf: An infrastructure for building shared multimedia applications," in *Proc. ACM CSCW*, pp. 637–650, (Los Angeles, CA), Oct. 1990.

14. J. J. Garcia-Luna-Aceves, E. Craighill, and R. Lang, "Floor management and control for multimedia conferencing," in *Proc. IEEE Multimedia, 2nd COMSOC Int. Multim. Comm. Worksh.*, (Ottawa, Can.), Apr. 1989.

15. F. Fluckiger, *Understanding Networked Multimedia*, Prentice Hall, Englewood Cliffs, NJ, 1995.

16. D. Bertsekas and R. Gallager, *Data networks*, Prentice Hall, Englewood Cliffs, N.J., 2nd ed., 1992.

17. H. Takagi and L. Kleinrock, "Output processes in contention packet broadcasting systems," *IEEE Trans. Commun.* **COM 33**(11), pp. 1191–1199, 1985.

18. E. Craighill, R. Lang, M. Fong, and K. Skinner, "CECED: A system for informal multimedia collaboration," in *Proc. ACM Multimedia*, (Anaheim, CA), Aug. 1993.

19. M. O. Pendergast, "Multicast channels for collaborative applications: design and performance evaluation," *Computer Communication Review* **23**, pp. 25–38, April 1993.

20. C. Ziegler, G. Weiss, and E. Friedman, "Implementation mechanisms for packet switched voice conferencing," *IEEE J. on Sel. Areas in Comm.* **7**, pp. 698–706, June 1989.

21. H. M. Vin, P. V. Rangan, and S. Ramanathan, "Hierarchical conferencing architectures for inter-group multimedia collaboration," in *ACM SIGOIS Bull., Proc. Org. Comp. Sys.*, pp. 43–54, (Atlanta, GA), Nov 1991.

22. B. N. Levine, D. Lavo, and J. J. Garcia-Luna-Aceves, "The case for concurrent reliable multicasting using shared ack trees," in *Proc. ACM Multimedia*, (Boston, MA), Nov. 1996.

23. H.-P. Dommel and J. J. Garcia-Luna-Aceves, "Efficacy of floor control protocols in distributed multimedia collaboration," *Cluster Computing* , submitted for publication 1999.