

# Efficacy of Floor Control Protocols in Distributed Multimedia Collaboration

H.-Peter Dommel and J.J. Garcia-Luna-Aceves \*

*Computer Engineering Department  
Baskin School of Engineering  
University of California  
Santa Cruz, CA 95064, USA  
E-mail: {peter, jj}@cse.ucsc.edu*

Distributed multipoint applications for group interaction across wide-area networks, such as for simulation and telecollaboration, are becoming increasingly popular. While reliable multicasting has made significant advances in recent years, effective mechanisms to synchronize and coordinate work within large multicast groups and across long distances are still lacking. Synchronous sharing of resources, whose operational semantics prohibits parallel usage, typically creates race conditions among users, which can be resolved through an access discipline called floor control. Existing solutions on floor control, implemented either at the session or application layer, are mostly proprietary, limited in scope and not scalable. Furthermore, no performance comparison of floor control protocols has been attempted to date. We present a novel taxonomy and comparative performance analysis of known classes of floor control protocols, ranging from socially mediated control to protocols operating on ring and tree topologies. We find that aggregation and selective transmission of control information in a tree structure is the most promising solution with regard to scalability, efficacy, and robustness. The principal operation of such a tree protocol is outlined, which dynamically organizes participants in a multi-level control tree and aggregates resource sharing directives on the paths between interacting stations.

**Keywords:** multimedia collaboration, floor control, reliable multicast

## 1. INTRODUCTION

The increasing deployment of IP-multicast [9] has brought a new generation of collaborative multimedia applications (CMA) to mainstream computing. Such applications allow users to overcome their separation in time and space and share information and work efforts in real-time, with the goal to at least approximate the quality of face-to-face interactions. While previous CMA were proprietary, monolithic, limited in the supported media types, and designed for small-scale collaboration in local area networks, newer applications are designed for multi-

\* This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) under Grants F19628-96-C-0038 and DAAB07-97-C-D607.

party interactivity with multimedia support in a wide geographic range and for larger sessions involving hundreds of participants. Sample applications are distributed interactive simulations, distance learning seminars, or special-purpose MBone sessions [14]. Shared resources in the workspace can be for example motorized cameras, robotic devices, surgical instruments, video and audio channels, or graphical objects at various levels of granularity. Collaborative applications often require the dynamic identification of a subset of session participants, which may contribute to the collective data flow to and from shared resources. Although reliable multicasting and multicast routing technology have advanced considerably, support for such controlled group interaction, particularly for applications geared towards synchronous and wide-area groupwork, is still lacking.

Sharing of virtual workspaces poses fundamental coordination problems and is limited by several factors, including local screen real-estate and visual feedback at a computer workstation, input and output devices to the shared workspace, the complex sharing semantics of applications, and limited network and host capabilities. The consequences of these limitations are flawed telepresence and coordination problems concerning mutual awareness, speech initiation conflicts, and diminished interaction, as reported by Isaacs and Tang [18] in studies on video conferencing. However, telepresence can be improved both from a networking and user interface perspective by supplying an effective mediation protocol, which directs users on the fine line between exclusive, yet shared resource usage.

Group coordination in distributed systems and multimedia systems has many faces that have been manifested in user interfaces and network protocols. To this date, however, no commonly accepted recipe on designing group coordination algorithms exists. Ellis *et al.* [13] discuss groupware issues and coordination solutions from a user interface standpoint, geared towards tasks such as shared editing. Networked multimedia systems, operating with continuous media and high user interactivity, differ in their requirements significantly from traditional distributed systems, which handle discrete data and long-lived access privileges. Many solutions to mutual exclusion [33] or concurrency control are hence not applicable for CMA in a straightforward manner.

*Floor control* [10] is a higher-level communication abstraction, which slowly gains acceptance as a building block in group coordination services. Typically deployed in the session layer or above, floor control lets users attain exclusive control over a shared resource by being granted the floor, extending the traditional notion as the “right to speak” [29] to the multimodality of data formats in networked multimedia systems. We understand floor control as a technology to implement group coordination, but use both terms synonymously in this paper. We define the *floor* as a generic short-lived synchronization primitive for multimedia objects. Several factors contribute to the complexity of deploying floor control for CMA: the system capabilities, in terms of the network, host, and communication subsystem, as well as users’ capabilities to process concurrent information sources of varying media types through user interfaces. Ideally, floor control is as unintrusive and adaptive to as many session types [36] as possible, and as

application-specific as needed. Control should reflect characteristics of tasks and interaction styles, such as user roles, usage quotas, and resource contention manifested in deteriorated quality of service. Sarin and Greif discussed floor control for text-based real-time conferencing and point out a potential efficacy gain by employing multicasting [30]. Schooler touches on floor control issues with regard to video and audio in surveying multimedia conferencing [31]. ITU standardization efforts on multipoint multimedia conferencing [38] are more geared towards circuit-switched networks and do not include multicast provisions. In contrast to LAN-oriented CMA, wide-area conferencing tools for the Mbone [14] are dominated by light-weight session management without stringent membership control, catering to the large-scale group dynamics, however, a bulletin-board floor control mechanism has also been introduced [23] to facilitate moderated feedback in large seminars.

The following benefits of floor control from a user and system perspective can be identified. On the user level, floor control establishes a resource-sharing discipline and defines clear rules for turn-taking. Misbehavior such as indefinite resource holding or other unfairness patterns are avoided by shifting the risk of collisions from data to floor signaling. Floor control either prevents or mediates race conditions and may foster a higher degree of user interactivity than possible in physical presence. The semantic integrity of shared applications is better preserved, in particular for data that are ill-suited for transaction and consistency management. Floor control deployment may involve social or cultural issues of communication among users, which are beyond the scope of this paper. On the system level, floor control can orchestrate intelligent allocation of scarce shared resources based on user input. For example, the SCUBA protocol [4] intertwines floor control with a rate-adaptation mechanism throttling media stream transmission according to the interest or capabilities of a heterogeneous receiver set. Only sources being granted the floor are allowed to consume bandwidth, and floor control directives can be used to reserve future bandwidth shares. Yavatkar [41] embeds floor control into a multiframe synchronization protocol for media streams, supporting causally ordered flow coordination by floor tokens on top of a reliable multicast protocol.

A floor control methodology distinguishes between “mechanisms” as a physical propagation and synchronization apparatus for control information, instantiated with “policies” [8,16], by which users agree on service order, priorities, fairness, and other fine-tuning parameters affecting high-level collaboration flow. Mechanisms such as activity sensing, polling, or token passing across rings or trees are discussed in Section 4. Policies can reflect local user preferences or global processing regulations on floor information, such as “free-for-all”, “voice-triggered”, “prioritized”, “pause-detecting”, “moderated”, or “low-bandwidth-first”. Policy enforcement can range from lenient to strict, reflecting the collaborative session model and incorporating quality-of-service conditions, such as delay bounds and the streaming quality permissible by the host and network. Queuing of requests adds the global and resource-local service order as another policy facet, e.g., first-

come-first-served, or least-recently-served. Fluckiger [15] distinguishes between implicit vs. explicit signaling to acquire a floor, e.g., voice-activated speaker switching vs. hand-raising through the user interface.

As another important dichotomy, floor control can be enforced in a centralized or distributed fashion [8,16]. Centralized control has the advantage of simpler implementation and floor tracking, but a single controlling node may be easily overloaded, obstruct parallel, but independent work efforts, and spoil an entire session in the case of failure. Several systems [1,34] used this paradigm with floor control implemented as a chalk passing mechanism between a session server and conferees.

In distributed control, floor states for shared resources are calculated based on tracking of local and remote events. The control load is shared among hosts, which can autonomously control their media streams, benefiting stability and flexibility in session management. However, because hosts are typically heterogeneous, duplicated computations and consistent replication may be costly, and both implementation and run-time monitoring of a floor control system can become very complex. Distributed floor control has been proposed for the MERMAID system [40], and in the form of a distributed voice-activated collision-sensing algorithm in the EMCE teleconferencing system [2]. EMCE supports a “free-for-all” floor policy in a local area network with ample bandwidth and low delay, allowing users to give brief feedback to a speaker without the need to take the floor. Based on this design, Craighill *et al.* [7] implemented the task-activated COMET algorithm for a collaborative engineering environment, where a floor control agent intercepts X-server events and allows for detailed capture of the collective process history. The COMET description also entailed preliminary discussions on floor granularity, limitation of input capabilities based on user roles, and combining access control with floor control policies. The CSpray system is another replicated collaborative visualization application, in which floor control over visualization primitives is coordinated by a roving floor holder [25]. Current web-centered applications facilitate asynchronous and transaction-oriented collaboration, and synchronous groupwork is limited mostly to text and chatting. The JETS system [32] is a recent example for a Java-based synchronous collaboration environment, with floor control provided by a simple locking mechanisms to prevent event collisions in the shared workspace.

To enjoy the best of both worlds, a large number of applications have been designed in a hybrid mix of centralized and distributed control, cf. [6,19]. Hybrid solutions are also well-suited for network computers, where hosts have limited capabilities and communicate with each other via session servers. Floor control is often viewed as an add-on to existing session services, such as bridging in call control [3] or large-scale conference management [17]. However, session and floor control, while coexisting, are independent and floor control can be centralized, while session control is distributed, or vice versa.

In summary, much previous work has focused on system-level mechanisms to facilitate point-to-point conferencing, often with a limited range of media, lack-

ing scalability, and without taking into account multicasting and the underlying communication infrastructure. The floor control solutions existing to date lack detailed presentation and analysis. In this paper, we offer a fresh perspective on group coordination by proposing a taxonomy of floor control protocols based on their operational principles, and by comparing their performance taking into account user behavior and generic system properties. Based on our results, we make the case for hierarchical floor control, which operates on a control tree matching the backbone reliable multicast tree and multicast routing tree. The goal is to show how floor control fits in with large-scale Internet conferencing.

The paper is organized as follows: Section 2 clarifies terminology and defines our system model for group coordination. Section 3 presents a taxonomy of existing classes of floor control protocols, encompassing user- and system-driven floor control mechanisms. The comparative analysis presented in Section 4 shows that floor control over a shared propagation tree, corresponding to an underlying end-to-end reliable multicast tree, is a scalable and efficient way to store and forward control information. Section 5 outlines the Hierarchical Group Coordination Protocol (HGCP), exemplifying such a tree-based floor control protocol. A summary and conclusion are offered in Section 6.

## 2. FLOOR CONTROL MODEL

A *collaboration environment*  $CE = (\mathcal{S}, \mathcal{U}, \mathcal{R}, \mathcal{F})$  consists of a network (session graph)  $\mathcal{S} = (V, E)$  of nodes (stations, hosts)  $V$  connected by links (channels)  $E \subset V \times V$ , users (participants)  $\mathcal{U}$ , shared resources (media)  $\mathcal{R}$  and floors  $\mathcal{F}$ . Communication among nodes is solely via message exchange, and not in shared memory. Links may be unreliable for data transmissions, but are assumed to be reliable for control message dissemination.

A *collaborative session*  $CS = (sid, \Delta, U, R, F)$  is an instantiation of  $CE$ , with a unique session identifier  $sid$ , session duration  $\Delta$ , a set of geographically distributed users  $U$ , a set of resources  $R$  and their corresponding floors  $F$ .  $CS$  can be open, allowing to choose resources, endpoints in interaction and to add and remove parties and media, or closed with a predetermined list of invited participants. If  $CS$  is open, the node set  $V$  is dynamic and the session size  $m = |U|$  is an upper bound on the number of participants. In our terms,  $m > 100$  denotes a “large” session. Collaborative sessions may vary strongly with regard to style, such as lecture, business meeting, panel discussion, or hearing [36], impacting size, organization, data flow, and the interaction model.

A *collaborative snapshot*  $CS_{sid}^j$ ,  $j = 1, \dots, \Delta$ , represents a discrete moment in the timeline of session  $CS_{sid}$ . It depicts the distributed computational state of a floor control protocol in terms of active users, resources, and floors, and is the result of querying this state locally or remotely.

Users  $u \in U$  can be humans or system agents [12], and occupy in our model one or more of the following control roles: the *floor originator* (FO) of a resource  $r \in R$  is the node that injects  $r$  into a session and initiates floor control for  $r$ . If

FO withdraws from *CS*,  $r$  will disappear unless it is replicated; the *floor coordinator* (FC) is an arbiter for a resource  $r$ , or a session moderator who grants or denies the floor for  $r$  during session time to the *floor holder* (FH), who exclusively has the floor on  $r$  for a finite time period. FC and FH may be identical for reasons of messaging efficiency, which affects the messaging pattern of the control protocol. These roles, according to the session model and purpose, may be statically assigned at startup, or rove among the user stations during session time. The active receiver set  $U' \subset U$  in any collaborative work step may be anonymous, although we assume that a global session directory service provides user registration. Resources  $r \in R$  can be located at one particular node, be distributed in their components across the node set, or be replicated over all nodes in *CS*. A resource, having spatial or temporal characteristics, can be an application, host object, or network entity shared in collaboration, such as a component of the user interface (telepointer), a hardware device, or a video stream. Any user or resource can function as an information source in the collaborative work act.

We define a floor as a tuple

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

denoting control over a resource  $r$ .  $T$  is the resource type (video, audio, text etc.), that specifies access control and the object semantics for  $r$  (for example, “receive”, “move”, or “write”.) FC and FH denote the coordinator and holder, which can be a session-wide unique identifier, or mark the location of the node in the control infrastructure. A timestamp  $ta$  shows, when the floor was allocated. The *floor holding time*  $\delta$  is the time, during which the adjunct resource is consumed by FH and inaccessible for others. If  $\delta = \infty$ , there is no expiration time for the floor. The *QoS* field may contain a directive on the service quality for using the resource, such as the image quality for video streams. The current protocol state  $st$  for  $f$  is a value from [*free* | *busy* | *idle* | *requested* | *nil*], which represents typical control states found in floor control protocols. The *nil* entry marks floors with pending state updates. Floors in this model are valid across session boundaries and carry hence no session identifier *sid*. For certain resources, such as audio channels, the FC can maintain a request queue, which records floor requests and serves them according to the floor policy.

For simplicity, we assume that there is one user per node, and one floor  $f$  is allocated per resource  $r$ , covering all functionality of a specific resource. However, it is possible to refine the floor granularity by introducing subfloors for components of resources that need to be controlled individually, such as specific operations of a device. Variations, such as a single floor controlling multiple resources, concurrent holdership of one floor by multiple users, or multiple floors controlling one particular resource will not be considered. The goal of this model is to lay the foundation for a common working ground and taxonomy of CMA.

### 3. TAXONOMY OF FLOOR CONTROL

Three properties forming the operational pillars for a floor control protocol are: (1) the random or scheduled access to resources, characterized by the mechanism and node topology, by which floor information is probed or relegated via directives between nodes; (2) the centralized or distributed establishment of control among the nodes in the session; and (3) the policy, by which floors are commonly processed among the nodes. Property (1) is the major design decision for a group coordination protocol, and determines, which control and logical roles as well as policies are established in a session. This property lays the foundation for the taxonomy shown in Figure 1. Known paradigms of floor control protocols are divided here into two classes: *Random-access Floor Control* (RFC) and *Scheduled Floor Control* (SFC) protocols. RFC protocols lets users contend for the shared resource, by sensing its status remotely before or during resource access. SFC protocols use token passing, reservations or polling as access mechanism in a session. A dashed line shows a protocol, whose performance will not be discussed in this paper. This taxonomy is not exhaustive, but serves as a first attempt to characterize predominant solutions on floor control.

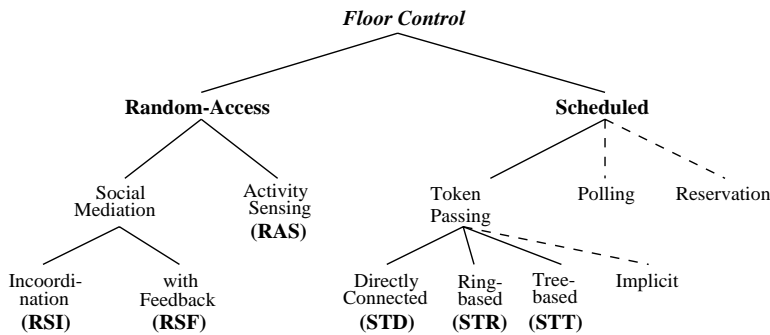


Figure 1. Taxonomy of floor control.

RFC protocols are based on the concept of sensing the status of a remote resource, which can be 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 placeholder. RFC schemes are inherently contention-based, because nodes must actively compete for a floor. The collective state of local assertions must be kept globally consistent. Continuous sensing of many remote resources' states, either by man or machine, can be costly, and there is a higher likelihood of collisions due to network latency or lack of coordination.

In contrast, SFC protocols rely on the exchange of a unique and explicit floor token, with nodes either being polled by a coordinator node for pending control messages, or with the token being passed in the order prescribed by the

logical geometry of the session. The uniqueness of the token guarantees exclusive usage of the floor. Resource contention is hence dispersed by regulated floor capture. Control directives are used to request, deny, reserve, or grant a floor. Using explicit floor tokens does not mean that users have to make a conscious effort to signal reservation and exchange of such tokens. Rather, the floor control subsystem orchestrates these electronic placeholders as a form of concurrency control, although users can influence or modify system decisions. SFC comes in two flavors: nodes can proactively send control messages to “ask” for the floor, or they can wait passively, until the floor is being “offered” by polling or passing through. A shortcoming of SFC schemes is the cost incurred in tracking floor tokens and ensuring their uniqueness and authenticity. SFC schemes generally operate on a specific infrastructure such as a ring or tree, which logically organizes the session.

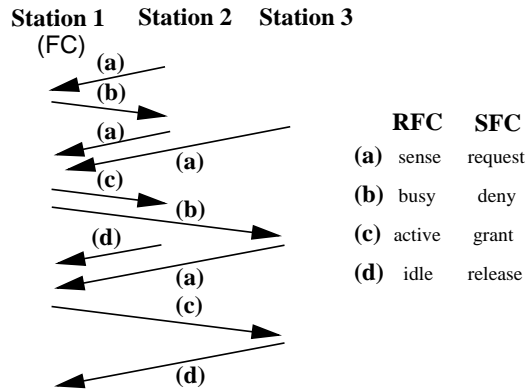


Figure 2. Generic message exchange in RFC and SFC.

Figure 2 depicts a typical dialogue between three nodes competing or cooperating for access to a shared resource. Station 1 is FC, and holds the floor by default in the beginning. Stations 2 and 3 contend for the floor, by remotely checking the status of the resource governed by station 1, or sending requests to it. Station 1 denies the first request, because it still uses the resource. Because station 1 is FC, it sends no release signal after completion. Stations 2 and 3 resubmit requests subsequently and station 2, being first, receives the floor, and releases it after completion. A repeated request from station 3 is finally being answered with a grant message. Station 3 frees the floor after completion. This sample dialogue assumes separation of FC and FH, no queueing of requests, and stations in SFC asking for the floor. The dialogue between the nodes is depicted here as one-to-one, however, directives concerning more than one node are assumed to be multicast to all group members.

Figure 3 depicts coordination geometries, as they are prevalent in existing commercial or experimental CMA, between four users  $u$  and resources  $r$ . Each resource  $r_i$  in the shared workspace belongs to user  $u_i$ . Nodes are either FH

(circled) or FC, assuming a separation of these roles, or they are regular participants trying to attain the floor by probing or sending messages to the node in charge. For simplicity, each node contributes here at most one resource to the session. The logical topology is crucial insofar, as it determines the efficiency by which control is relegated among nodes. Most topologies allow to implement both centralized or distributed session and floor management.

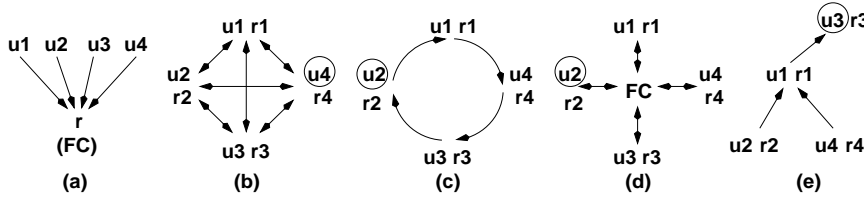


Figure 3. Coordination geometries.

In Fig. 3(a), users  $u_i$  attempt to gain control of a centralized resource  $r$  with limited or no coordination. In Fig. 3(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. 3(c) depicts ring-based control, based on the idea of passing a floor token in a linear fashion along the nodes in the ring. In Fig. 3(d), a star-topology is used to concentrate floor requests onto a dedicated FC node, which is a subcase of Fig. 3(e). In this tree structure, control directives are propagated along the branches of a multi-level control tree towards the current FH.

#### 4. COMPARATIVE ANALYSIS

Our analytic comparison 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 and to highlight performance differences among coordination schemes. 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, based on the analogy between access mitigation for the data channel and shared resources.

To make the analysis tractable, we state the following assumptions: the individual host processing cost for control packets, including protocol overhead and user-interface specifics, is the same for all hosts; message delivery between hosts is reliable and no failures in hosts or the network occur; we only account for the processing effort in sending floor control messages; information from a station reaches any other station with the same average system-wide propagation delay; the interarrival rate of floor requests is Poisson, given that there is no indication

for cross-correlations between subsequent floor requests; depending on the session model, subsequent requests are either being discarded, or queued and served in FIFO order; finally, the task length, i.e., the floor holding time, is normalized.

For the analysis, we take into consideration the interarrival rate for floor requests, the task length, and the network propagation delay. We consider a point-to-point model of message dissemination, as well as a broadcast model, because a number of systems have been implemented for one model, but not for the other. The broadcast model uses IP-multicast, i.e., a hosts needs to send a message only once to the network interface, where it is multicast to all the receivers.

$\beta_1$	average “think” time before floor token arrival
$\beta_2$	average “think” time at floor token presence
$\gamma$	average processing time for a floor directive
$\delta$	duration of average activity period
$n\epsilon$	processing and unicasting overhead to n receivers
$\eta$	efficacy of a floor control protocol
$G$	average offered floor request load
$\iota$	average duration of idle time
$\lambda$	floor request interarrival rate
$m$	average number of stations in session
$n$	average number of active stations in session
$\nu$	average vulnerability period
$\tau$	average propagation delay

Table 1  
Analysis parameters.

The notation used in our analysis is summarized in Table 1: to model acquisition of a floor token ahead of time, we introduce the idea of “think time”, denoted with  $\beta_1$  (the time until the expected floor arrives at the local station) and  $\beta_2$  (the time once the floor token is present at the local station and offered to the user);  $\gamma$  is the average time to process or communicate a floor directive;  $\delta$  is the duration of an activity period, which is normalized to 1; we use  $n\epsilon > \tau$  as the additional delay to account for the dependency of the communication and processing overhead on the number of stations collaborating;  $G = \delta \times \lambda$  is the normalized offered request load on floors, including new and previously denied and resubmitted floor requests;  $\iota$  sums up the expected idle time for a resource during floor holding time;  $\lambda$  represents the relative frequency of demand for resource access and thus indicates the contention level;  $m < \infty$  stations denotes all stations being a member in a session;  $n < m$  is the number of active stations in the multicast group for the current floor and transmission;  $\nu$  is the time during which a station’s attempt to access a resource can be intercepted by another station; and  $\tau$  denotes the average end-to-end delay between stations, including packetization delay, generic queuing transmission times, and local processing overhead. The activity period and time to process and communicate floor directives are assumed to include the time incurred in providing feedback among stations.

User behavior in terms of the switching of control over a particular resource is incorporated into this analysis with a turn taking model [11], which serves as

the conceptual blueprint for our analysis. A prototypical turn taking period is depicted in Figure 4, which shows the call pattern between stations and the effect on the particular controlled resource. The calls shown are explained in Section 5. A turn has accordingly three stages, the contention time  $X$ , accounting for request, contention and granting or denial periods; an activity time  $A$ , accounting for floor holding and releasing; and an idle time  $I$  (which may or may not occur) accounting for the time period when the resource is not actively being used.

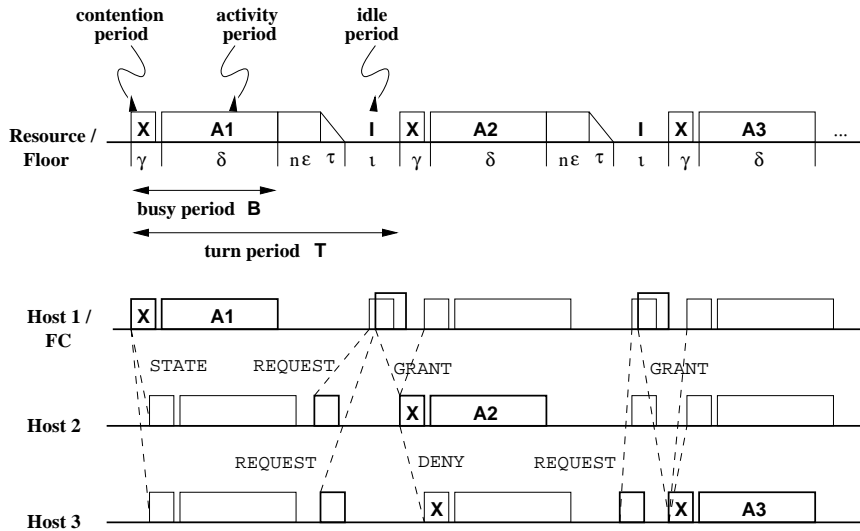


Figure 4. Turn taking periods for a resource and three stations.

Figure 4 depicts a conversation between three stations trying to access a resource. First, station 1, which is also FC, holds the floor and its activity outcome is transmitted to the other stations. Second, stations 2 and 3 try to acquire the floor, and station 2 wins, because its request was first received at FC. Station 2 starts accessing the resource, until the floor holding time expires. Third, station 3 acquires the floor without collisions. This diagram represents the case when FC is separate from FH, and is applicable to both implicit and explicit floor passing. It is the variations in allocation patterns and duration of floor periods that make certain control mechanisms more efficient than others. Based on this abstraction, we define the *efficacy* of a floor control protocol, denoted by  $\eta$ , as the proportion of time that a protocol needs to allocate a resource, including overhead from the protocol itself, the network, and user behavior. In other words, we want to assess the reactivity of a protocol in a specific system architecture to signal for attention, submit a request, receive a reply, and select a user to be in charge of a resource. 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.

$$\eta = \frac{\bar{U}}{\bar{T}} = \frac{\bar{U}}{\bar{B} + \bar{I}} \quad (1)$$

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 stations are forced to contact one another explicitly, which substantially increases the processing overhead at stations and the possibility of disagreement on which station has the floor. We model the existence of multicasting at the network layer by assuming that a user station 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 stations with the same average delay. In the following paragraphs, we discuss the efficacy of prevalent solutions with and without multicast support, and summarize the multicast efficacy  $\eta^{MC}$  in Table 2.

#### 4.1. Random-Access Group Coordination Schemes

*Incoordinated Social Mediation* (RSI) refers to a purely random scheme of resource access in a self-moderating session. Shared access becomes guesswork and data inconsistencies are expected. Possible causes are insufficient feedback through the user interface about remote activities, delay caused by host or network limitations, or uncooperative users on a self-moderating channel. As a consequence, contending stations, being unaware of each others' immediate actions, may experience floor acquisition conflicts by trying to access a resource at the same time. Although all information is distributed reliably to all user stations, the latency introduced by the system can lead to inconsistent views of the floors at different stations. When this occurs, we assume that all users involved in the conflict must restart their activities. There is no system support to mediate conflicts and ensure system-wide consistency of the state of the shared resource, and all messages must hence percolate through the entire protocol stack to be reflected in the user interface, which makes the conflict detection process long and inefficient.

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

$$\eta_{RSI} = (\delta + n\epsilon)\lambda e^{-2\lambda(\delta+n\epsilon)} \quad (2)$$

*Proof:* Figure 5 shows a prototypical RSI turn. The proof is the same as for medium access in an ALOHA channel [5]. In the point-to-point model, we assume that  $n$  stations are actively monitoring each other, and it takes an additional time  $n\epsilon$  for all stations to perceive the activity from a given station. 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

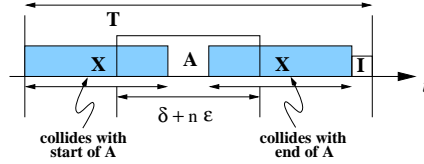


Figure 5. Typical RSI timeline.

task information to all stations (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  $e^{-2(\delta+n\epsilon)\lambda}$ . The success probability times the number of arrivals in one activity period results in Eq. 2.  $\square$

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

$$\eta_{RSI}^{MC} = \lambda\delta e^{-2\lambda\delta} \quad (3)$$

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

*Social Mediation with Feedback* (RSF) assumes cooperative users following agreed-upon social protocols, relying on feedback support on remote activities from the network and user interface. If a user contends for a floor and perceives remote activity, she would back off for a random short period and attempt to reclaim the floor, after remote activity subsided. RSF in video conferencing, as with the MBone tool *vic* [24], 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. Analog POTS conferencing also relies on RSF, which works well for very small groups. Except for user interface updates, both RSF and RSI incur little implementation cost regarding user coordination.

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

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

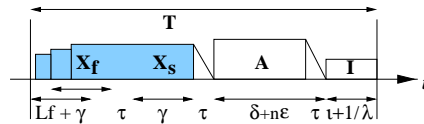


Figure 6. Typical RSF timeline.

*Proof:* A prototypical timeline of RSF is depicted in Figure 6.  $n$  active stations need to sense and process remote activity through the network inter-

face. The success probability, denoted as  $P_s$ , equals the probability that no activity packet arrives in an average vulnerability period  $\nu$  of  $2(\gamma' + n\epsilon)$  s, 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  $\bar{f}$  s (average time between failed floor-request arrivals), plus the duration observing a request ( $\gamma'$ ). The values for  $\bar{L}$  and  $\bar{f}$  have been derived by Takagi and Kleinrock [37]. 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

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

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

In *Activity Sensing* (RAS) [7,16], 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 node currently operates on the resource. The RAS concept is related to collision sensing on a multiaccess channel [5]. In principle, no changes to a collaboration-unaware application are required to integrate floor control by modifying the X-server to intercept and filter calls to the application signifying access to a shared resource. Similar to the socially protocolled RSF, a RAS system agent would back off when detecting remote activity, deny the local user the floor until remote activity subsides, and signal a free floor afterwards. By that, a distributed collective of activity sensing agents enacts more accurate monitoring about resource states, than humans could deliver. The disadvantage of this scheme is its high implementation cost and its reliance on short link latencies, as shown below, which makes it only suitable for LANs.

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

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

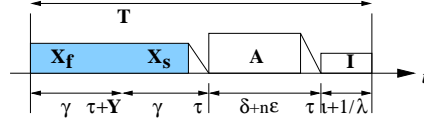


Figure 7. Typical RAS timeline.

*Proof:* The timeline is shown in Figure 7. Without multicast support, a station has to send floor directives individually to every other station, which according to our assumption takes  $\gamma' + n\epsilon$ . Because multiple unicast messages are used by each station, floor-state inconsistencies can arise during the entire time. The station 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.

**Corollary 3.** The efficacy of RAS with multicast support is

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

*Proof:* The access strategy is assumed as nonpersistent, i.e., a station 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 stations 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  $\gamma'$  s, followed by one or more similarly truncated activities sent within time  $Y$  s, where  $0 \leq Y \leq \tau$ . The expected value of  $Y$  is [37]  $\bar{Y} = \tau - \frac{1}{\lambda}(1 - e^{-\lambda\tau})$ ; 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.  $\square$

#### 4.2. Scheduled Group Coordination Schemes

In SFC protocols, floor token transmission must be reliable to ensure that floors are not duplicated, lost or forged. We exclude such situations from our

analysis. In contrast to the contention time  $\gamma'$  in random-access schemes,  $\gamma$  denotes here the time to transmit a floor token to the next station. We assume for all protocols that  $\tau$  signifies the average propagation delay for multiple routing hops between stations coalesced into one hop. A packet must hence traverse on the average the same number of hosts on the path from the sender to a group of receivers.

We can identify three major SFC solutions to floor control: In *token passing* the floor is being asked for or offered to the stations in the session in a predefined service order. Stations, active or inactive, are either directly connected to each other, or they are logically arranged in a multihop ring or tree geometry. *Implicit token passing* [5] is a special case used to reduce token size, in which a station relinquishing the floor simply goes idle. The next node in sequence detecting resource idleness takes the floor if a local request has been recorded, or remains quiet. Similarly, the other nodes wait for increasingly longer timeout periods to detect idleness and take the floor if requested. However, such collision avoidance lacks fairness in accessing tokens, because stations further away from the current FH may never attain the floor.

In *Direct Coordination* (STD), each station in a group is fully connected to every other station. 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 nodes in the session, which makes this solution unscalable, in particular for unicasting. Messages must be ordered or a voting mechanism among nodes must determine a FH successor. Furthermore, cognitive abilities of human users to handle flows from all participants are limited. Many small-scale commercial video conferencing systems follow this model.

**Theorem 4.** The efficacy of STD without multicast support is

$$\eta_{STD} = \frac{\delta}{n(\gamma + \tau + \epsilon) + \delta + \iota + \frac{1}{\lambda}} \quad (8)$$

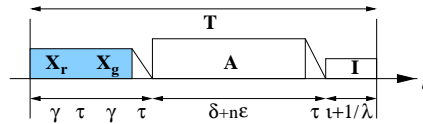


Figure 8. Typical STD timeline.

*Proof:* The timeline for STD is shown in Figure 8. The average utilization period is  $\bar{U} = n\delta P_s$ , because floor capture is perfect and any one of the  $n$  active stations can acquire a floor of holding time  $\delta$  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  $\gamma$  and the propagation delay is  $\tau$ . Unicasting a floor request to the  $n-1$  active nodes 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  $\iota$  and an average interarrival time for all nodes,  $\bar{I} = \iota + \frac{1}{\lambda}$ . Substitution into Eq. 1 results in Eq. 8.  $\square$

**Corollary 4.** The efficacy of STD with multicast support is

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

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 station incurs host processing overhead from its  $n - 1$  neighbors.

In *Ring-based Coordination* (STR), stations in a session form a logical ring, and a floor token cycles through the ring in a round robin order. STR is particularly suited to ensure totally ordered and atomic delivery. Pendergast [27] discusses a group support system operating on a token ring. Ziegler *et al.* [43] analyze packet-switched voice conferencing mechanisms in a logical ring. A station 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 station. A station without pending floor requests passes on the offered token. If a token is held for excess time, it can expire, or a busy-token for this floor is sent across the ring to indicate that the floor is taken and the token “alive”. A station waiting for the floor can insert a reservation tag into a busy token to mark that it is next in line for holdership. The circulating token hence replaces request and grant messages as a form of explicit signaling, and contention among nodes in the ring is based on claiming the roving token.

A related case is floor control over a token bus, where a node passes the floor token along the list of stations attached to the bus. The delay in a token bus is inherently larger compared to a token ring [5] and will not be discussed. A shortcoming of STR is that a predefined token passing schedule does not reflect spontaneous interactivity. There are many variations on the detailed operation of floor control in a ring structure, concerning acquisition and release of the floor token. The floor can be granted ahead of its position to a successor station, or it may only be acquired by a station when it passes through that station. Likewise, a token can be immediately released after transmission (RAT) or released after one more reception (RAR) at the sending station. For efficiency reasons we focus on RAT. In our model, the pre-arrival think time  $\beta_1$  plus the token-presence time  $\beta_2 < \beta_1$  must be smaller than the ring cycle time. In case that the floor is taken,  $\beta_2 \approx (\delta + \tau + \gamma)$ .

**Theorem 5.** The efficacy of STR without multicast support is

$$\eta_{STR} = \frac{\delta(1 - e^{-\lambda(\beta_1 + \beta_2)})}{\frac{n}{2}(\tau + \gamma + \epsilon + \beta_2) + \delta(1 - e^{-\lambda(\beta_1 + \beta_2)}) + \iota + \frac{1}{\lambda}} \quad (10)$$

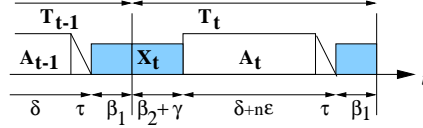


Figure 9. Typical STR timeline.

*Proof:* The timeline for STR is depicted in Figure 9. We assume perfect floor capture and only one node 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 node set  $n$ , and with growing session size it is less likely that all nodes will engage in floor contention. The cost to transfer a floor token in a cycle involves on the average  $\frac{n}{2}$  nodes, amounting to  $\frac{n}{2}(\gamma + \tau + \epsilon + \beta_2)$  including processing overhead  $n\epsilon$ . 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.  $\square$

**Corollary 5.** The efficacy of STR with multicast support is

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

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  $\epsilon$  vanishes.

*Tree-based Coordination* (STT) is a hybrid control solution based on the idea to perform floor control across a logical tree structure, which allows for more efficient mixing of individual media sources [39] and close correlation of the control geometry with the actual underlying multicast routing tree, or the end-to-end reliable multicast tree. Hierarchical organization of stations also supports inter-group collaboration, subgroup addressing, and allows more scalable and economic transmission of session data [21]. A tree-based group coordination protocol representing STT is discussed in Section 5.

Control messages in a STT protocol are passed along branches of the tree in a parent-child relation reflecting multicast group membership. No specific node alone is hence burdened with the obligation to make floor allocation decisions, and tokens can wander freely across the tree branches, without being cast into a specific traversal order other than what multicast group membership expresses. Control messages are aggregated across the tree, by coalescing multiple messages of the same type, such as floor requests, into single directives on their path to receivers. Such aggregated management of control information liberates the FH from *control implosion*, where handling of control messages is concentrated in a single node, and allows instead for message exchange in local groups. A control

tree is assumed to have a branching factor of  $K$ , indicating the number of children for each node. The communication delay depends on the height of the control tree, not the session size. The average messaging cost is  $O(\log_{K-1}(n))$ . A special case of STT, which equals a moderated session model, is a radiating star-topology with the FC at the core, surrounded by  $n - 1$  neighbor nodes. This model is efficient for small sessions, however, the FC node may become overloaded or fail for large sessions.

**Theorem 6.** The efficacy of STT without multicast support is

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

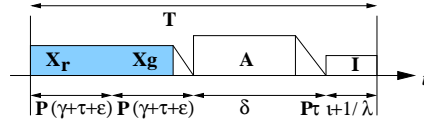


Figure 10. Typical STT timeline.

*Proof:* The timeline for STT is depicted in Figure 10. 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 node does not know the location of FH or FC, exploratory unicast of a control message from one node 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 5, nodes can be tagged with unique labels, which allow for efficient absolute or relative routing of control directives toward the FC or FH. Consequently, control directive must be sent only to one neighbor node as the gateway on the path to FH, hence  $K = 0$ . 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 6.** The efficacy of STT with multicast support is

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

With multicasting, a request-grant pair takes two  $\gamma$  and  $\tau$ , plus another  $\tau$  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 station sends only one message to the network interface, i.e.,  $\bar{P} = 1$ ,  $K = 0$ , and  $\epsilon$  becomes negligible.

Two other SFC approaches are part of the taxonomy. In *polling*, stations are systematically or randomly probed for pending floor requests by a central station, with polls acting as floor tokens. Delay between polling a secondary node, waiting for its response and shifting to the next node can be large. Another disadvantage is the reliance on a central coordinator node. In a *reservation* system, floor allocation is divided into a reservation interval and a data interval. The reservation period is finite, but allows for variance in holding duration. However, the natural flow of interaction can hardly be accommodated by preset reservation periods, which may quickly become obsolete. Polling and reservation were excluded from our analysis, because, to our knowledge, no telecollaboration systems have been built based on these principles.

For convenience, the efficacy  $\eta^{MC}$  for the discussed floor control paradigms with multicast support is summarized in Table 2.

Protocol	$\eta^{MC}$
RSI	$\lambda\delta e^{-2\lambda\delta}$
RSF	$\frac{\delta}{\delta + e^{2\lambda\gamma'} \left[ \frac{e^{\lambda\gamma'} - 1 - \lambda\gamma'}{\lambda\gamma'(1 - e^{-\lambda\gamma'})} + \gamma' + \tau + \iota + \frac{1}{\lambda} \right]}$
RAS	$\frac{\delta}{\delta + \tau + \frac{1}{\lambda} + e^{\lambda\tau}(\gamma' + 2\tau + \iota)}$
STD	$\frac{\delta}{\delta + 3(\gamma + \tau) + (n-1)\epsilon + \iota + \frac{1}{\lambda}}$
STR	$\frac{\delta(1 - e^{-\lambda(\beta_1 + \beta_2)})}{\frac{n}{2}(\tau + \gamma + \beta_2) + \delta(1 - e^{-\lambda(\beta_1 + \beta_2)}) + \iota + \frac{1}{\lambda}}$
STT	$\frac{\delta}{\delta + 2\gamma + 3\tau + \iota + \frac{1}{\lambda}}$

Table 2  
Efficacy of floor control protocols with multicast support.

### 4.3. Results

We contrast the efficacy of the discussed schemes with 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. We set  $\tau = 0.005$  s to characterize a short propagation delay as it is typical for local area networks, and  $\tau = 0.4$  s for wide area networks and Internet collaboration. The latter value is also an upper bound for acceptable delay in applications dependent on timing relations among separate

media streams [35]. Small sessions are set to  $n = 5$ , which is representative for PC-conferencing systems, and the value  $n = 300$  for very large sessions corresponds to traces from MBone sessions [22], sampled over a period of several hundred hours and indicating a group size ranging from 150 to over 550 participants. The time to sense and react to floor information in RSF is much slower than for machine driven sensing, hence we assume a lower bound of  $\gamma^l = 0.25$  s. For automatic detection or processing of a control packet of 25 bytes length we assume  $\gamma = \epsilon = 0.02$  s. The token-ring “think times” for arriving and offered tokens are relative to the activity time, and set to  $\beta_1 = \frac{\delta}{2}$  s and  $\beta_2 = \frac{\delta}{10}$  s. The typical idle time is chosen to be  $\iota = \frac{\delta}{5}$  s. The normalized activity time is  $\delta = 1$ . Figures 12 and 11 plots the resulting efficacy for these scenarios.

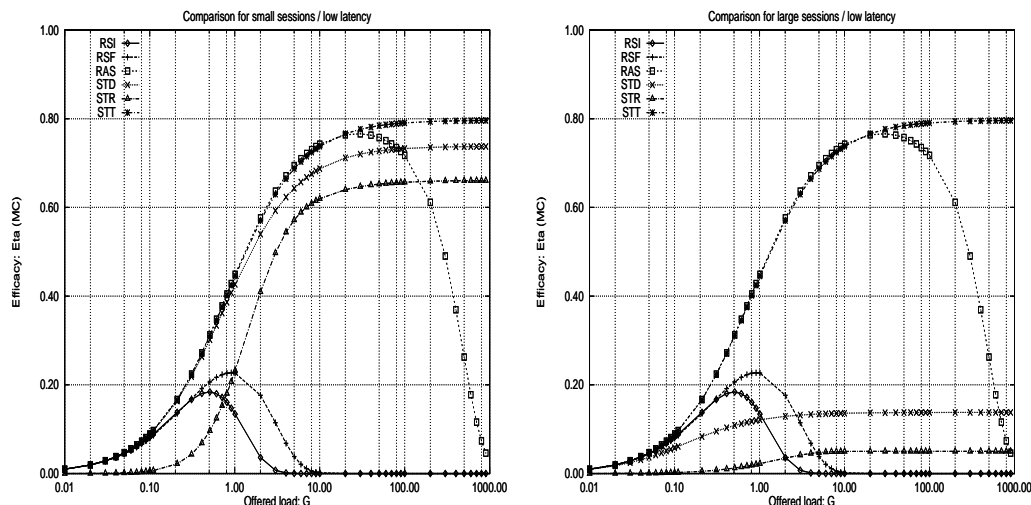


Figure 11. Efficacy with multicast support for low network latency.

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 unstable even at low request rates and collaborative efficacy falls off quickly and the average delay becomes unbounded. This models the phenomenon that users avoid frequent turn-taking in telecollaboration [18]. 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 allows for slightly higher loads than RSF, however, it also degenerates quickly and rises barely above 20%. STD performs well for small groups, where a station needs only to send few packets to the session remainder, but despite stability for higher loads it barely exceeds 15% efficacy in larger sessions. In its best case of small sessions and low network latency, STR approximates 65% efficacy and it is generally stable, but degrades

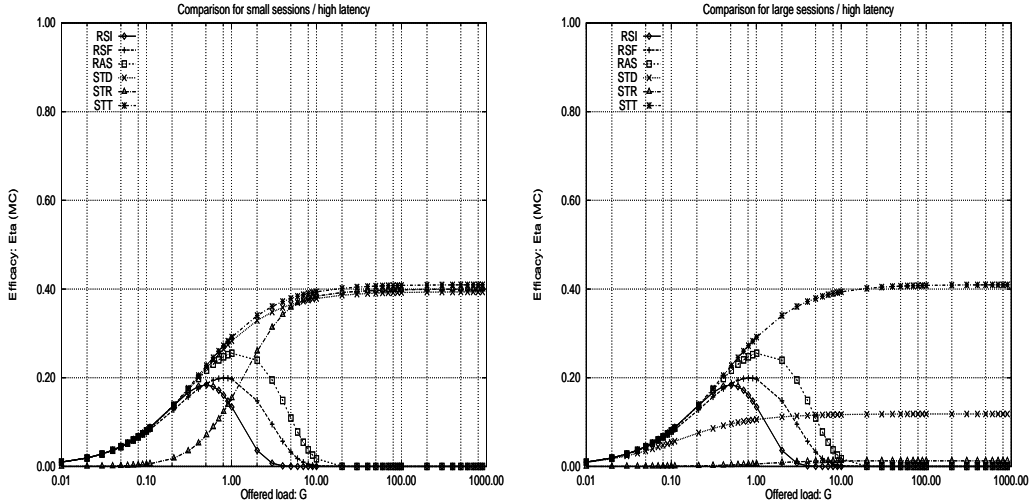


Figure 12. Efficacy with with multicast support for high network latency.

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.

## 5. AN EXAMPLE OF TREE-BASED FLOOR CONTROL

### 5.1. HGCP Description

We outline the operation of the Hierarchical Group Coordination Protocol (HGCP) as an example for STT protocols. Although tree protocols have been the subject of related research, e.g., on mutual exclusion [28], channel access [5], or reliable multicast [21,26,42], to our knowledge no tree-based group coordination protocol has been proposed in the literature. HGCP represents 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.

The reliable multicast protocol Lorax [21] uses a single shared acknowledgment (ack) tree per session for multiple sources, instead of creating a separate tree per transmission. The Lorax ack tree is characterized by labeling of nodes from a finite alphabet with the property that the label  $l(x)$  of a node  $x$  is the prefix of its children. The purpose of these labels is to provide relative addressability of multicast groups and subgroups. The maximal length of labels in a tree reflects its depth, and labels in the tree remain constant for the lifetime of the session, except in cases when nodes withdraw or join. 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. A tree with  $n$  session participants and branching factor  $K$  has  $\log_K n$  levels, with  $\log_2 n$  bits needed for node labels. The ack tree does not require modification to the router-internal IP multicast infrastructure. However, Levine and Garcia-Luna-Aceves have generalized labeling to the router level [20], exploiting novel mechanisms of streaming and addressing. The ack tree structure enforces cascaded acks and negative acknowledgments across tree branches, and prevents receivers from contacting the source directly to confirm reception of packets or ask for retransmission of lost packets. Instead, recovery is contained in local groups of tree subbranches by aggregating ack information.

In HGCP, the ack tree becomes the control tree of a *CE* session and shrinks or expands dynamically during session lifetime. Stations are prevented by HGCP to contact the FH directly and instead submit control directives for aggregation in their immediate tree neighborhood, achieving better scalability in controlling the floor for large sessions and many resources. HGCP derives label information from the end-to-end multicast protocol to administer floor information and route control directives between hosts contending for a resource. HGCP is hybrid in that FH poses a centralized, but roving point of control for an individual floor, however, the protocol runs in every node starting or joining a session and the union of distributed floor states in every active node yields the global control state.

HGCP can implement nonpersistent floor control, with any request except the first accepted one being discarded, or persistent floor control, where a finite request queue is maintained to keep track of consecutive requests. It regulates messages exchanges between peer user agents, which are either users or applications acting on behalf of users, with the goal to intercept data and commands between users and ensure that only one user can access a particular resource at any given time. Each user agent tracks one or more floors, one per resource. The floor state is controlled in a distributed fashion. Call setup, late joining and withdrawal from a session are handled by a membership protocol interfacing with HGCP.

Control responsibilities are distributed over the entire session tree, dividing the session into local groups with three kinds of nodes: a control node, relay nodes, and leaf nodes. The *control node* hosts the FH (or FC), regulating access to a resource  $r$  and transmitting updates concerning  $r$ . *Relay nodes* collect CDs from their children and forward them in the tree towards the FH. Likewise, they relay replies back to their children. A relay node, which is not a member of the destination multicast group for a directive is called *extra node*. An extra node can be viewed as a proxy for the destination node. *Leaf nodes* delimit tree branches and communicate solely with parent relay nodes.

For messaging efficiency, HGCP unifies the roles of FH and FC and collocates them in one node. If both roles would be separated, three nodes are involved in a triangle communication: a node  $x$  would first have to contact the FC, who would await release of the floor from the current FH, and then grant

the floor to  $x$  by asking FH to transfer the floor to  $x$ . With unification of FH and FC, a moderated session style can still be supported, by sending the floor back to the moderator node after turn completion for assignment to the next floor holder. Role unification is also practical for loosely coupled sessions, which need no distinct leader and are destined to make progress in self-moderation. FH does not have to know the identity of session members, because generic labels provide sufficient routing information, i.e., HGCP supports anonymous collaboration.

HGCP scales well because nodes only maintain a local picture of their immediate session neighborhood, knowing the direction toward the current FH for a specific floor, and communicate only within the local group defined by parent and children nodes. FHs are positioned at the root of the tree, and the tree is virtually rotated at floor hand-over towards the new FH, whereby balancing properties of the tree may change. Less interactive stations are placed more likely on leaf positions. On the average, leaf nodes must compare more bits in the control routing procedure, than nodes close to the root. Compared to a geometry, in which nodes are fully connected, trees significantly reduce the amount of messaging. Labels also allow for communication between distinct coteries of hosts, without involving the entire multicast group.

A control directive CD contains a list of source labels  $\{l_s\}$  for the aggregated delivery of directives from multiple nodes, a list of destination labels  $\{l_d\}$ , a timestamp (or sequence number)  $\#$ , a time-to-live field TTL, an identity descriptor ID for public, private, or anonymous submission of directives, and a floor descriptor  $f$ , whose fields have been described in Section 2:

$$\text{CD}(\{l_s\}, \{l_d\}, \#, \text{TTL}, \text{ID}, f) = \langle \text{directive} \rangle$$

The TTL field sets an expiration date on persistent CDs. Setting TTL = 0 configures nonpersistent floor allocation. Values for  $\langle \text{directive} \rangle$  are: **REQUEST** to ask for access to a resource; **GRANT** to grant permission to access the resource, provided that the floor is free; **DENY** to signal that the floor is taken or reserved; **RELEASE** to relinquish the floor; and **STATE** to retrieve updated floor state information.

The *setup phase* of HGCP can be initiated along with or after session advertisement via a session directory service. If the session is open, new nodes can join, otherwise invited stations are only allowed to withdraw. Each station maintains a floor state table containing floor information about the local and remote resources being part of the shared workspace. The state table is compared in regular intervals with the tables of parent and children nodes in the tree, and is incrementally updated, if a younger table is detected. In this sense, floor agents are a cooperative collective, building a consistent distributed control state for the entire session. Every station advertises its shared resources and floors to neighbor stations in the reliable multicast tree and integrates updates from these stations into its local floor state table. The station introducing a resource  $r$  to a session  $s$  creates and injects the floor token for  $r$  into  $s$ .

The *active phase* of HGCP concerns the aggregation and forwarding mech-

anism between local groups in the control tree for the various node types. A node can access and use a shared resource  $r$  only if it becomes FH. The floor for  $r$  is idle if no station sends messages to the network concerning  $r$ . To acquire the floor, a station sends a **REQUEST** CD to its neighbor station in the control tree, which is closer on the path to FH. The direction is computed by taking the prefix of the location entry for the floor marked in the floor state table. Each request is compared against requests pending from other nodes. According to the aggregation semantics, if a near-by node is able to provide a CD response, the CD request is served by this node and not propagated to the FH. Compounded CDs received at FH are ordered and serviced first based on priority, second with regard to queuing, timestamp and reception order. If the FH is static, eligibility for the floor may also be decided based on statistics, indicating frequency and holding time per floor for each node to induce fairness into the floor acquisition process. When FH completes its resource access, it sends a **GRANT** CD to its successor, based on the source label information. After receiving confirmation, FH officially relinquishes the floor by multicasting the position of the new FH' to the session. A new turn cycle begins, with nodes contending for the floor now submitting **REQUEST** CDs to FH', which may also receive a copy of the request queue and hand the floor to the next station in the queue when finished.

In the *termination phase*, floor state information is deleted from the records of the local station, but retained in a log file for turn taking history. Single nodes can withdraw intentionally or by failure from a session. In normal operation, any node not interacting and affecting the global floor state can withdraw. Consider the case that a station that submitted a request for the floor, withdraws before a **GRANT** CD reaches it. A successor station FH' must confirm reception of the floor, before the remainder session updates its state table on the location of FH'. Hence, pending but unconfirmed floor hand-overs are never registered. If FH wants to withdraw from the session, HGCP revokes the floor from this node and assigns it to the node with the oldest pending request. If a relay node withdraws from the session, its children and parent link together and reinitiate any operation pending on the missing relay node.

Several cases must be considered when nodes or links fail: (1) a node fails, either (a) as FH, or (b) requesting a floor, or (c) being uninvolved in any floor allocation process; and (2) a link fails (a) with control directives in transition, or (b) otherwise. In case (1a), if FH fails holding the floor, a timeout and election protocol among neighbor nodes ensures that the lost floor will eventually resurface at one of these neighbor nodes. In cases (1b) and (2a), a pending request may be fulfilled, after a station disappeared from the session. Since this station will not confirm the floor transfer within a timeout period, the floor will be reassigned to the next contending node in line. In cases (1c) and (2b), any other node may withdraw from the session any time, however, if this node is an extra node for a floor transfer, the parent and child node embracing this node must first reestablish a connection to be able to conclude the transfer as proxy nodes for the failed node. Every floor state alteration is hence treated as a transaction,

and interrupted floor changes are reset to a consistent state before alteration, or committed when succeeding. In the nonpersistent model, nodes must resubmit their failed requests after a timeout. Further details on these reliability issues for tree-based floor control, incorporating dynamic changes to the multicast tree, as well as measures against forging floor information and securing collaborative sessions, are beyond the scope of this paper.

The basic example in Figure 13 illustrates the operation of the protocol in a binary control tree and with a binary labeling alphabet. The protocol stack indicates that HGCP is collocated with a session management protocol, interfacing with reliable multicast below and applications above.

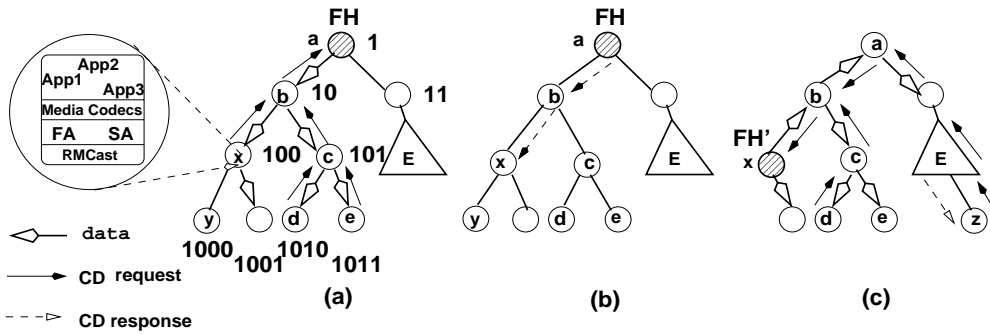


Figure 13. Sample HGCP scenario.

In scenario 13(a), FH for some resource  $r$  is initially placed at the root  $a$ , with  $l(a) = 1$ . All nodes have a consistent update on the current FH for various resources in their floor state table. FH multicasts data updates of its work on  $r$ , indicated by hollow arrowheads, to all the sites in the multicast group, which may process the update (or discard it.) Switching a remote resource on or off, such as a video stream, compares to a local “receiver floor”, by which a station can individually configure its stream and data reception. Floor control exerted by HGCP concerns only “sender floors”. Consider the case that 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 the request from  $x$  arrived earlier or that it is prioritized. Once FH finished and released the floor for  $r$ , or when its floor expires, it multicasts a **GRANT** CD across  $b$  to  $x$ . Once  $x$  confirms reception of the floor,  $a$  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 13(b),  $x$  is the new FH', starts using  $r$  and multicasts updates to

its children. All nodes in the multicast group propagate their CDs towards the location of FH', as indicated in scenario 13(c). 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  $E$  joins the session and sends a STATE CD to its parent after integration into the control tree, retrieving the current state table for its local resources shared with the session. Assume that the parent node of local multicast group  $E$  has the most recent state table (all nodes are steadily querying their neighbors and eventually converge towards the most current state table.) Then the update request from  $z$  remains in the right subbranch of the tree and does not need to be forwarded to FH', i.e., target nodes such as FH' are relieved from the obligation to handle every request from within the session. After this update, node  $z$  can start contending for the floor with the other active nodes by propagating CDs toward FH' based on label information.

It shows that labels are valuable both as absolute and relative pathfinding markers in the control process. Reliable multicasting and aggregation of CDs and floor states between a node and at most  $K$  children (instead of the entire session) allows hence for more economic storing and forwarding of control traffic. In the following paragraph, we show that HGCP is safe and live for the case of a static FH.

## 5.2. HGCP Correctness

**Theorem 7.** HGCP is safe, i.e., at most one node receives the floor for a specific resource at any time.

*Proof:* A floor control deadlock exists if node  $x$  holds the floor and at least one or more nodes request the floor, but are unable to acquire it in finite time. There are several potential reasons: (1) A node operates on the resource without being really granted the floor; if there is no legitimate FH, the floor privilege cannot be passed on; (2) the GRANT CD is transmitted but does not reach any requesting node; and (3) FH is unaware of requests. The proof is based on the idea that the control tree is acyclic and propagation of request messages between nodes towards the FH prevents circular stalls of control messages. Using induction on the height of the  $K$ -ary control tree, the tree reduces for  $h = 1$  to a non-hierarchical star-based scheme with  $n = K + 1$  nodes.

In case (1), the floor semantics states that a floor is at all times assigned to one node, that is either FH, or one of the other  $K$  nodes in the session. If the node currently holding the FH privilege is different from the node legitimately holding the floor, this node will acquire the FH tag. In case (2), a FH candidate node is either nonexistent or withdraws from a session, while the floor is in transit. As stated earlier, floor transfers must be confirmed by the receiver to the sender, and stalled transfers are nullified. In case (3), if FH is unaware of requests, this node is eventually timed out, and the FH privilege is shifted and assigned to one of the  $K$  active neighbor nodes. If a node  $x$ , as one of the active nodes, sends a REQUEST

CD towards the FH, it is received by the FH assuming reliable transmission. Upon reception, FH sets a requested flag and, if queueing is supported, it inserts the label information of the requesting node in its FIFO request queue. Based on the queue semantics, the floor will eventually be shifted to node  $x$ .

For  $h > 1$ , we assume that the theorem holds for any height  $l$  with  $1 \leq l < h$  and we need only show that the theorem also holds for  $l = h$ . The mechanism of transferring floor holdership, as discussed in the simpler case, can only be bypassed if a REQUEST CD is continuously outrun by a GRANT CD. However, the implicit routing scheme based on labels defines a forwarding order on nodes, that defies indefinite stalls of REQUEST. No node can indefinitely hold on to a floor, because floors expire or peer nodes will revoke the floor. The FH must eventually become legitimate, or floor transfers must be confirmed, or FH must become aware of other nodes requesting the floor based on the principle that REQUEST CDs are propagated steadily towards the FH. Thus, directives toward and from FH must eventually arrive at their destination node, and thus deadlock is impossible.  $\square$

**Theorem 8.** HGCP is live, i.e., a request by any node  $x$  must be served in finite time, and no node suffers from “floor starvation”.

*Proof:* HGCP is live because the FH node either services the first request and discards all follow-up requests, or it maintains a request queue with nodes marked by prefix labels. This queue cannot be longer than  $n$ , given that no node is allowed double entry until other nodes have been served. Each of the possible actions of FH will reduce the request queue ultimately to length 1, which allows the remaining node to eventually acquire the floor. Hence, every node requesting the floor will be served in finite time.  $\square$

## 6. CONCLUSION

Telecollaboration is characterized by multimedia multiparty interactivity and slowly enters mainstream computing, due to advances in internetworking and web technology. However, protocol support to achieve better telepresence and support group coordination in a “virtual LAN” of Internet dimensions is still lacking. This paper focused on contention resolution with floor control protocols and their analytic comparison, making three contributions.

First, a novel taxonomy for floor control protocols has been proposed based on operational principles. Second, a performance analysis in accordance with the taxonomy has been presented, subsuming rather different protocol classes in one coherent framework. It merges time aspects of protocol operations with end-user behavior and thus accounts for both internal and external factors regarding control of resource sharing. Third, a novel floor control protocol operating in a logical shared control tree has been outlined, whose operation is correlated to a tree-based end-to-end reliable multicast protocol. Future work must address issues of session stability, authentication, message ordering, and fairness.

Although strong assumptions were made to keep our bare-bones analysis tractable, this is to our knowledge the first attempt to quantitatively characterize the effectiveness of various strategies of floor management in distributed group collaboration. We found that an approach based on a logical tree structure outperforms other control schemes in terms of scalability, efficacy and practicality, and also introduces novel features such as selective subgroup addressability. We conclude that tree-based floor control, embedded with reliable multicast, represents a promising approach to support large-group interactivity across local-area and wide-area networks.

## References

- [1] H. M. Abdel-Wahab, S.-U. Guan, and J. Nievergelt. Shared workspaces for group collaboration: an experiment using internet and unix interprocess communications. *IEEE Comm. Mag.*, 26(11):10–16, Nov. 1988.
- [2] L. Aguilar, J. J. Garcia-Luna-Aceves, D. Moran, E.J. Craighill, and R. Brungardt. Architecture for a multimedia teleconferencing system. In *Proc. ACM SIGCOMM*, pg. 126–136, Aug. 1986.
- [3] S. R. Ahuja and J. R. Ensor. Coordination and control of multimedia conferencing. *IEEE Comm. Mag.*, 30(5):38–43, May 1992.
- [4] E. Amir, S. McCanne, and R. Katz. Receiver-driven bandwidth adaptation for light-weight sessions. In *Proc. ACM Multimedia*, Seattle, WA, Nov. 1997.
- [5] D. Bertsekas and R. Gallager. *Data networks*. Prentice Hall, Englewood Cliffs, N.J., 2nd ed., 1992.
- [6] J.F. Buford. *Multimedia Systems*. Addison-Wesley/ACM Press, New York, NY, 1994.
- [7] E. Craighill, R. Lang, M. Fong, and K. Skinner. CECED: A system for informal multimedia collaboration. In *Proc. ACM Multimedia*, Anaheim, CA, Aug. 1993.
- [8] T. Crowley, P. Milazzo, E. Baker, H. Forsdick, and R. Tomlinson. MMConf: An infrastructure for building shared multimedia applications. In *Proc. ACM CSCW*, pg. 637–650, Los Angeles, CA, Oct. 1990.
- [9] S. Deering. Host extensions for IP multicasting. RFC-1112, August 1989.
- [10] H.-P. Dommel and J. J. Garcia-Luna-Aceves. Floor control for multimedia conferencing and collaboration. *Multimedia Systems (ACM/Springer)*, 5(1):23–38, Jan. 1997.
- [11] H.-P. Dommel and J. J. Garcia-Luna-Aceves. Network support for turn-taking in multimedia collaboration. In *Proc. IS&T SPIE Multimedia Computing and Networking*, pg. 304–315, San Jose, CA, Feb. 1997.
- [12] E. A. Edmonds, L. Candy, R. Jones, and B. Soufi. Support for collaborative design: Agents and emergence. *Comm. of the ACM*, 37(7):41–47, July 1994.
- [13] C. A. Ellis, S. J. Gibbs, and G. L. Rein. Groupware - some issues and experiences. *Comm. of the ACM*, 34(1):38–58, Jan. 1991.
- [14] S. Floyd, V. Jacobson, S. McCanne, C.-G. Liu, and L. Zhang. A reliable multicast framework for light-weight sessions and application level-framing. In *Proc. ACM SIGCOMM*, pg. 342–356, Cambridge, MA, Aug./Sep. 1995.
- [15] F. Fluckiger. *Understanding Networked Multimedia: Applications and Technology*. Prentice Hall, Englewood Cliffs, NJ, 1995.
- [16] J. J. Garcia-Luna-Aceves, E. Craighill, and R. Lang. Floor management and control for multimedia conferencing. In *Proc. IEEE Multimedia, 2nd COMSOC Int. Multimedia Comm. Workshop*, Ottawa, Canada, Apr. 1989.
- [17] M. Handley, I. Wakeman, and J. Crowcroft. The conference control channel protocol (CCCP): A scalable base for building conference control applications. In *Proc. ACM SIGCOMM*, pg. 275–287, Cambridge, MA, Aug./Sep. 1995.
- [18] E. A. Isaacs and J. C. Tang. What video can and cannot do for collaboration: a case study. *Multimedia Systems J.*, 2(2):63–73, Aug. 1994.

- [19] K. A. Lantz. An experiment in integrated multimedia conferencing. In *Computer Supported Cooperative Work: A Book of Readings*, pg. 533–556. Morgan-Kaufman, 1988.
- [20] B. N. Levine and J. J. Garcia-Luna-Aceves. Improving internet multicast with routing labels. In *Proc. IEEE ICNP*, pg. 241–250, Atlanta, GA, Oct. 1997.
- [21] 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.
- [22] J. Liebeherr and B. S. Sethi. A scalable control topology for multicast communication. In *Proc. IEEE Infocom*, San Francisco, Mar./Apr. 1998.
- [23] R. Malpani and L. Rowe. Floor control for large MBone seminars. In *Proc. ACM Multimedia*, pg. 155–163, Seattle, WA, Nov. 1997.
- [24] S. McCanne and V. Jacobson. vic: A flexible framework for packet video. In *Proc. ACM Multimedia*, pg. 511–522, San Francisco, Nov. 1995.
- [25] A. Pang, C. Wittenbrink, and T. Goodman. CSpray: A collaborative scientific visualization application. In *Proc. IS&T SPIE Multimedia Computing and Networking*, San Jose, CA, Feb. 1995.
- [26] S. Paul, K. K. Sabnani, J. C.-H. Lin, and S. Bhattacharyya. Reliable multicast transport protocol (RMTP). *IEEE J. on Sel. Areas in Comm.*, 15(3):407–421, April 1997.
- [27] M. O. Pendergast. Multicast channels for collaborative applications: design and performance evaluation. *Comp. Comm. Review*, 23(2):25–38, April 1993.
- [28] K. Raymond. A tree-based algorithm for distributed mutual exclusion. *ACM Trans. on Comp. Sys.*, 7(1):61–77, Feb. 1989.
- [29] H. M. Robert. *Robert's rules of order*. Bantam Books, Toronto; New York, 1986.
- [30] S. Sarin and I. Greif. Computer-based real-time conferencing systems. In *Computer-Supported Cooperative Work: A Book of Readings*, pg. 397–420. Morgan-Kaufman, 1988.
- [31] E. M. Schooler. Conferencing and collaborative computing. *Multimedia Systems J.*, 4(5):210–225, Oct. 1996.
- [32] S. Shirmohammadi, J. C. De Oliveira, and N. D. Georganas. Applet-based telecollaboration: a network-centric approach. *IEEE Multimedia J.*, 5(2):64–73, April-June 1998.
- [33] M. Singhal. A taxonomy of distributed mutual exclusion. *J. of Parallel and Distributed Systems*, 18(1):94–101, May 1993.
- [34] K. Srinivas, R. Reddy, et al. MONET: A multimedia system for conferencing and application sharing in distributed systems. Concurrent Engineering Research Center, West Virginia University, Morgantown, WV, Feb. 1992. TR CERC-TN-RN-91-009.
- [35] R. Steinmetz. Synchronization properties in multimedia systems. *IEEE J. on Sel. Areas in Comm.*, 8(3):401–411, April 1990.
- [36] C. Szyperski and G. Ventre. Efficient group communication with guaranteed quality-of-service. In *Proc. IEEE 4th Workshop on Future Trends of Distributed Comp. Sys.*, pg. 150–156, Lisbon, Portugal, Sept. 1993.
- [37] H. Takagi and L. Kleinrock. Output processes in contention packet broadcasting systems. *IEEE Trans. Commun.*, COM 33(11):1191–1199, 1985.
- [38] International Telecommunication Union. Recommendation T.120 on data protocols for multimedia conferencing. <http://www.itu.int>, July 1996.
- [39] H. M. Vin, P. V. Rangan, and S. Ramanathan. Hierarchical conferencing architectures for inter-group multimedia collaboration. In *ACM SIGOIS Bulletin, Proc. Org. Comput. Sys.*, pg. 43–54, Atlanta, Georgia, Nov 1991.
- [40] K. Watabe, S. Sakata, K. Maeno, H. Fukuoka, and K. Marbara. Distributed multiparty desktop conferencing system: MERMAID. In *Proc. ACM CSCW*, pg. 27–38, Los Angeles, CA, Oct. 1990.
- [41] R. Yavatkar. MCP: A protocol for coordination and temporal synchronization in multimedia collaborative applications. In *Proc. of the 12th IEEE Int. Conf. on Distributed Comp. Sys.*, June 1991.
- [42] R. Yavatkar, J. Griffioen, and M. Sudan. A reliable dissemination protocol for interactive collaborative applications. In *Proc. ACM Multimedia*, pg. 333–344, San Francisco, Nov. 1995.
- [43] C. Ziegler, G. Weiss, and E. Friedman. Implementation mechanisms for packet switched voice conferencing. *IEEE J. on Sel. Areas in Comm.*, 7(5):698–706, June 1989.