A Novel Group Coordination Protocol for Collaborative Multimedia Systems

H.-P. Dommel and J. J. Garcia-Luna-Aceves

Computer Communication Research Group
Baskin School of Engineering
University of California
Santa Cruz, CA 95064, USA

ABSTRACT

Group collaboration in distributed multimedia environments extends gradually to larger groups and wide-area networks. While reliable multicasting has made significant advancements in recent years, effective mechanisms to synchronize and coordinate work within large multicast groups and across long distances are still lacking. Group coordination is here understood as the mediated access to shared resources in synchronous work, for example in telecollaboration and distributed simulation environments, complementing protocols for group membership, media synchronization and reliable ordered multicast. A comparative analytic model for known classes of group coordination mechanisms, ranging from socially mediated control to floor control in ring and tree topologies, is presented. It is shown that hierarchical group coordination is the most efficient and scalable approach to date. Based on these findings, a novel protocol is described, which dynamically organizes participants in a multi-level control tree and aggregates resource sharing directives on the paths between interacting stations.

1 INTRODUCTION

Increasing deployment of IP-multicast [6] has brought a new generation of collaborative multimedia applications to mainstream computing. While previous collaboration tools were proprietary, monolithic, and designed for small-scale collaboration in local area networks, newer applications are geared towards larger sessions with wide geographic range. For example, distributed interactive simulations and distance learning sessions can easily involve hundreds of participants. Although reliable multicasting and multicast routing technology have advanced considerably, efficient group coordination support for applications characterized by synchronous and wide-area groupwork is still lacking.

Our goal is to extend support for dynamic group coordination to wide-area networks and large multicast groups, characterized by intermittent connectivity and heterogeneous multimedia information. The central idea behind group coordination protocols is to establish a resource-sharing discipline, counter end-application misbehavior, and incorporate end-user demand for a specific Quality-of-Service, allowing to throttle bandwidth utilization and impacting informed resource allocation. It is still a subject of controversy, whether group coordination services are to be tailored to the application-level, or whether they should be offered as a middleware component, which a variety of applications demanding for resource cooperation can tap into. To this date, a variety of group coordination protocols have been proposed under different names, but have not been specified in more detail, compared, or placed in a consistent methodological framework.

We regard coordination of group activities on shared resources as a distributed middleware service tailored toward the semantics of the media involved, rather than specific applications. We focus in this paper on a particular form of group coordination, known as floor control. By being granted a "floor", a user attains a permission for exclusive usage of a resource. Floor allocation establishes clear rules for turn-taking, and prevents race conditions, indefinite resource holding, and unfairness in resource access patterns. Participating stations agree on a specific floor policy, concerning service order and priorities. The spectrum of control can range from lenient to strict, reflecting characteristics of tasks and interaction styles, such as user roles, usage quotas, or resource contention periods. As a component within a general coordination architecture for many-to-many groupwork, floor control coexists with protocols for reliable ordered multicast and media synchronization at a sub-application level. Orchestration of multiparty groupwork with fine-grained and fair floor control is an open research problem. In this paper, we offer a fresh look on this topic by making the case for hierarchical floor control.

The remainder of the paper is organized as follows: Section 2 discusses cornerstones of related work. Section 3 presents a brief overview and comparison of existing classes of group coordination protocols. We find that floor control over a shared propagation tree, corresponding to the underlying end-to-end reliable multicast tree, represents the most scalable and efficient way to store and forward control information. In particular, it allows to exploit the hierarchical nature of multicast groups, supports selective control packet dissemination to multicast subgroups, and distributes load about floor state keeping across the multicast tree, without compromising ease of implementation. Section 4 describes the operation of the Hierarchical Group Coordination Protocol (HGCP), a novel approach for floor control in shared propagation trees. Section 5 concludes the paper.

2 RELATED WORK

Coordination problems such as initiation conflicts and resource competition in multimedia conferencing have been reported in [9]. Early efforts on coordination of group activities led to the conception of floor control for networked databases and telecollaboration systems [20]. Various precursors of collaborative applications [2, 5, 21, 23] contained some form of proprietary floor control for small sessions with broadcast of control information and centralized coordination. The centralized method is comparable to a sender-initiated multicast solution, which severely limits the capacity of a group coordination system. First, one station must maintain and process a large amount of state information associated with each resource and floor contender. Second, the sender as the floor controller must process many floor requests in a short time and may corrupt the entire session, if failing. An alternate mechanism has been proposed in the teleconferencing architecture by Aguilar et al. [1], in which a distributed, task-activated floor control for teleconferencing is used, as a high-level analogy to collision-sensing in channel access. In Yavatkar and Lakshman's approach [25], floor control is understood as a transport level service, realized as a token-based concurrency control scheme for media flows, which has been implemented on top of the reliable multicast protocol XTP. Lately, floor control has appeared in a variety of commercial and experimental systems. For example, the MBone application suite has been enriched by a moderated bulletin board [15], which allows users in a larger scale conference to post questions and receive attention from specific participants. Amir et al. [3] suggest to intertwine floor control with a rate-adaptation mechanism, which throttles 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.
A Novel Group Coordination Protocol for Collaborative Multimedia Systems

University of California at Santa Cruz, Department of Computer Engineering, Santa Cruz, CA, 95064

Approved for public release; distribution unlimited

The original document contains color images.

Report Documentation Page

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE 1998

2. REPORT TYPE

3. DATES COVERED 00-00-1998 to 00-00-1998

4. TITLE AND SUBTITLE

A Novel Group Coordination Protocol for Collaborative Multimedia Systems

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

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:

a. REPORT unclassified

b. ABSTRACT unclassified
c. THIS PAGE unclassified

17. LIMITATION OF ABSTRACT

18. NUMBER OF PAGES 6

19a. NAME OF RESPONSIBLE PERSON

Standard Form 298 (Rev. 8-98)
Prepared by ANSI Std Z39-18
3 GROUP COORDINATION PARADIGMS

We propose a basic taxonomy which divides known group coordination algorithms into two classes: implicit group coordination (IGC) and explicit group coordination (EGC) algorithms.

Taxonomy

IGC algorithms mark the state of resource usage with assertions on local variables determined through one or more rounds of messages or probing of remote resources’ states. Since no token entity is explicitly exchanged, such algorithms are also referred to as “permission-based”. The challenge for this class lies in the globally consistent state keeping of local assertions. IGC schemes are inherently contention-based, since sites must actively compete for a floor. Disadvantages of IGC are the need for continuous sensing and the failure-prone observation of remote resource states, either by man or machine, with a strong likelihood of collisions due to network latency, or lack of coordination. As a consequence, contending stations, being unaware of each other’s immediate actions, may experience collisions by trying to access a resource at the same time.

In contrast, EGC algorithms explicitly exchange a unique token among sites. Obtaining this token symbolizes the right to access a resource, thereby dispersing floor contention by regulated passing of permission place-holders. Hence, this class is also referred to as “token-based”. In addition to the token being passed among sites, control messages are exchanged to request, deny, reserve, or grant the token, depending on the control scheme used.

A shortcoming of EGC schemes is the higher cost of token tracking, in order to ensure token uniqueness and authenticity. EGC schemes generally operate on a defined infrastructure, which logically organizes the receiver set.

Both classes operate based on the dichotomy between a floor holder (FH), which designates the current legitimate user of a resource, and a floor coordinator (FC) for that resource, which determines the legitimate floor holder at a given time. Often, the FH is also the FC. We assume that a generic group coordination protocol uses the following basis set of control messages: a station sends a FRQ (floor request) to the FC to demand access to a resource; the FC either replies with FGT (floor grant) to grant permission to access the resource (when the floor is free), or FDY (floor deny) to signal that the floor is taken or reserved; a station transmits a FRL (floor release) message to relinquish the floor.

The crucial difference between algorithms lies in the assignment of these roles to various nodes during a session, which impacts the routing of control messages. Figure 1 depicts typical coordination scenarios, where a circled entity marks the FC or FH: (a) several participants P_i try to gain control of a centralized resource R with limited or no coordination; (b) P_i communicate directly with each other in order to obtain a consensus about floor holdership; (c) a ring structure linearizes interaction and control passing between pairs of nodes; (d) one centralized station serves as FC in a star-topology, which is a one-level subcase of (e); (e) control messages are propagated along branches of a multi-level tree topology towards the current FH.

The goal for all algorithms is the correct and efficient determination of the FC and FH per resource at every turn in a collaborative session. The taxonomy in Figure 2 corresponds to the constellations in Figure 1.

Comparative Analysis

Our analytic comparison of known classes of group coordination protocols is a first attempt to characterize the efficacy of interactive behavior of people and processes from a resource contention perspective. The goal is to assess how much overhead is intrinsic to the various protocols with regard to control state management, based on topology and signaling. We take into consideration an average floor request arrival rate, task length, and network propagation delay. In our model, a collaboration session C = (S, L) in a computer network consists of a set of stations S (sites, nodes) and a set of links L ⊆ S x S. Each station, in an intranetwork or internetwork, hosts a session member (user, process), serves remote stations with local resources, and is client for remote resources. Links are either reliable (loss, but possibly delayed delivery), resilient (timely delivery, but likely some loss), or without service guarantees.

For reasons of tractability, we make the following assumptions: the underlying dissemination model is broadcast, i.e., a host sends each message only once to the network interface, where it is IP-multicast to the receivers; message delivery between hosts is FIFO, and no station failures occur; user interface and host processing overhead are negligible for each station; the interarrival rate of floor requests is Poisson, given that there is no indication for cross-correlations between subsequent floor requests; floor allocation is greedy, that is, the first registered request is fulfilled, and others are discarded and may be resubmitted later; finally, the floor holding time to execute a task is on the average the same and normalized to unity.

Figure 1: Group coordination geometries.

The average think time at floor token presence

T = average “think” time before floor token arrival

T = average think time at floor token presence

T = average processing time for a floor directive

T = duration of average activity period

T = processing and unicasting overhead to n receivers

T = efficacy of a floor control protocol

T = average offered floor request load

T = average duration of idle time

T = floor request interarrival rate

T = average number of stations in session

T = average number of active stations in session

T = average propagation delay

Table 1: Analysis parameters.

For reasons of tractability, we make the following assumptions: the underlying dissemination model is broadcast, i.e., a host sends each message only once to the network interface, where it is IP-multicast to the receivers; message delivery between hosts is FIFO, and no station failures occur; user interface and host processing overhead are negligible for each station; the interarrival rate of floor requests is Poisson, given that there is no indication for cross-correlations between subsequent floor requests; floor allocation is greedy, that is, the first registered request is fulfilled, and others are discarded and may be resubmitted later; finally, the floor holding time to execute a task is on the average the same and normalized to unity.

Figure 2: Group coordination mechanisms.
for reasons mentioned above, if many receivers demand control, then a single host and user become moderator and floor controller for the duration of any given floor request \( \gamma' \). The values for \( \overline{L} \) and \( \overline{T} \) have been derived in [22]. Substituting our notation in these results, we obtain \( \overline{L} = e^{-\lambda \gamma} \) and \( \overline{T} = (\lambda \gamma)^{-1} - e^{-\lambda \gamma} / (1 - e^{-\lambda \gamma}) \), respectively. Accordingly, the average time of a failed turn attempt equals \( X_f = \left( \frac{\lambda \gamma}{\gamma'+\tau} + \frac{\tau}{\gamma'} \right) \). The probability that no activity packet arrives in a vulnerability period \( \nu \) of \( 2 \gamma' \) sec, i.e., \( P_\nu = P[0 \text{ packets in } \nu] = e^{-2 \lambda \gamma} \). A successful turn is \( X_s = \gamma' + \delta + \tau \). Finally, the expected idle time is \( I = \overline{t} + \frac{1}{\lambda} \).

Activity Sensing (ICAS) is a special class of floor control proposed in [8], where shared activities on resources are monitored by a background process at the session layer, in order to sense which site currently operates on the resource. This mechanism is comparable to collision sensing in medium access control. If an ICAS process at a local site detects remote activity on a resource, it denies the local user the floor, until remote activity subsides. In that way, a distributed collective of activity sensing agents ensures better tracking of resource states, and helps to prevent conflicts. The disadvantage of this scheme is its high implementation cost and its reliance on short link latencies, which makes it usable only in LANs.

**Theorem 3** The efficacy of ICAS is

\[
\eta_{ICAS} = \frac{\delta}{\delta + \gamma + \tau + 1 + e^{\lambda \gamma} (\gamma' + 2 \tau + 1)}
\]  

![Figure 6: ICAS timeline.](image)

**Proof:** The timeline is shown in Figure 6. The average utilization period is \( \overline{U} = \delta P_a \), with \( P_a = P[0 \text{ packets in } \nu] = e^{-\lambda \gamma} \). The average length of a successful busy period is simply \( \delta + \gamma + 2 \tau \). The length of an average unsuccessful activity period consists of one truncated activity lasting \( \gamma \) sec, followed by one or more similarly truncated activities sent within time \( Y \) sec, where \( 0 \leq Y \leq \tau \). The expected value of \( Y \) is \( Y = \tau - \frac{1}{\lambda}(1 - e^{-\lambda \gamma}) \) [22]. The length of the average busy period is then \( B = \gamma + 2 \tau - \frac{1}{\lambda} + e^{\lambda \gamma}(\delta + \gamma + \tau + \frac{1}{\lambda}) \). The average idle interval is again \( I = \overline{t} + \frac{1}{\lambda} \). Substitution into Eq. 1 yields Eq. 4. \( \square \)

The class of explicit group coordination comprises three protocol families, as well. Common to all families is the exchange of a floor token symbolizing exclusive access to the shared resource. It must be assured that token transmission is reliable and that all sites execute the same protocol correctly to avoid conflicts based on duplicated, lost or forged tokens.

**Diagram:** The complexity of ECFC grows quadratically with the number of hosts. For each floor transaction, \( 2(n - 1) \) messages are required, following the Ricart-Agrawala mutual exclusion solution [19]. Newer quorum or tree-based [18] algorithms for obtaining an exclusive token can reduce the number of messages to \( O(\log n) \), where \( n \) is the session size. However, these schemes do
not scale well, since the network can easily be flooded with too many control messages, and significant delay variations between hosts can eventually deadlock collaborative turn-taking.

**Theorem 4** The efficacy of ECFC is

\[ \eta_{ECFC} = \frac{\delta}{\delta + \eta(\gamma + \tau) + \eta(n - 1) + \frac{\eta}{2}} \]  

**Proof:** The timeline for ECFC is shown in Figure 7. The time per station to send, receive, and update floor information with every other station is \(3(\gamma + \tau)\). An extra processing time \(e\) from \(n - 1\) stations must be added. Substitution into Eq. 1 results in Eq. 5. \(\square\)

In **Ring-based Coordination** (ECRB), a token rotates in a well-defined sequence among sites and holding the token determines the current floor holder, with two subcases: (a) a token claim can be decided upon before arrival, or during the arrival hold-time, or only while it visits the local station, and (b) a token can be transferred immediately after a forward from a station to the next one in the ring, or only after the releasing station received an acknowledgment. ECRB is particularly suited to ensure totally ordered frames among stations and holding the token determines 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 prescribes. While various tree-based reliable multicast protocols have been proposed for highly scalable conferencing [12, 14, 24], no group coordination mechanism or applications have been developed yet. The goal for such a protocol is to keep the tree depth minimal, in order to keep the number of hops for control transmissions low. A special case of ECTB is a star-topology, which is a tree of depth one with a root and \(n - 1\) descendants. The moderator-driven question board [15] exemplifies this scheme. Such a centralized scheme suffers as a communication bottleneck primarily from overload when the request rate increases. In our analysis, focused on multicast support for turn-taking, we did not account for this additional individual host processing cost.

**Theorem 5** The efficacy of ECRB is given by

\[ \eta_{ECRB} = \frac{\delta}{\delta + \eta(\gamma + \tau) + \eta(n - 1) + \frac{\eta}{2}} \]  

**Proof:** The timeline for ECRB is depicted in Figure 8. The average utilization is again \(\bar{U} = \delta P_e\), with \(P_e\) as the probability that the token is available in the period \(X = \beta_1 + \beta_2\). This period includes the time \(\gamma\) needed to transfer the token to the neighbor station. The token cycle time is a function of the session size. On the average, a floor token has to cycle through half the ring to be transferred. The probability that a floor can be claimed by a user is \(P_e = 1 - e^{-\lambda(\beta_1 + \beta_2)}\). Accordingly, the average turn lasts \(\bar{T} = \frac{3}{2}(\tau + \gamma + \beta_2) + \bar{U}\). The idle time is again \(\tau + \frac{1}{\lambda}\). Substituting \(\bar{T}, \bar{U}\) and \(\bar{I}\) into Eq. 1 yields Eq. 6. \(\square\)

**Tree-based Coordination** (ECTB) algorithms conduct floor management in a logical tree structure. Such a tree may reflect the actual multicast routing tree, or mirror the end-to-end reliable multicast tree. Control messages are passed along branches of the tree in a parent-child relation, which reflects multicast group membership. No specific site 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 prescribes. While various tree-based reliable multicast protocols have been proposed for highly scalable conferencing [12, 14, 24], no group coordination mechanism or applications have been developed yet. The goal for such a protocol is to keep the tree depth

\[ \text{Figure 7: ECFC timeline.} \]

\[ \text{Figure 8: ECRB timeline.} \]

\[ \text{Figure 9: ECTB timeline.} \]

\[ \text{Figure 10: Efficacy for large groups and high link latency.} \]
4 PROTOCOL DESCRIPTION

We outline the basic operation of the Hierarchical Group Coordination Protocol (HGCP). For a more detailed description and proof of correctness for HGCP we refer to [7]. Our goal is to exploit the best-effort delivery mechanism of IP multicast in a scalable and efficient tree protocol for reliable group coordination. The protocol consists of two stages: (1) control tree construction, and (2) control message dissemination. In order to consistently account for floor control messaging, we assume ordered message delivery between hosts. Ordering in multicast trees has been tackled for example by Jia [10].

We distinguish between four types of nodes in a propagation tree: the source node is the current FH and transmits information to the receiver set; hop nodes are positioned on the path from the source to the receivers, and are called extra nodes, if they are not contained in the receiver set; destination nodes are members of the receiver set. Each of these nodes can contend for any floor at any time and become FH. In a multicast collaboration session with multiple floors and floor holders, it is not practical to manage separate control trees, one per floor. Similar to the concurrent reliable multicast scheme proposed in [12], we hence assume a single shared control tree with a branching factor $B$, which can be re-hung as a control tree with any other node as the root and FH, while preserving the property that all nodes have $B$ children. Ideally, the control tree correlates closely to the multicast routing tree as built by multicast routing protocols such as DVMRP, CBT or PIM [16].

We focus on an approach comparable to receiver-initiated multicast [12], which lets containing stations in the group-coordinated session maintain state information about a particular floor. Floor state information can hence be passed along between containing stations, without requiring the controller station to contact each station. For this purpose, each station maintains two timers. Timer one is employed to detect whether a floor control message has been lost. Timer two is used to delay repetitive control messaging to the controller, in the hope that a near-by station is able to report an update from the controller station. The collective state information about all floors is hence distributed across the control tree. Note that HGCP is independent of a particular reliable multicast implementation, but relies on the failure-free transmission of control packets concerning floor states is based on the abstract notion of control routing procedure, than nodes close to the root. The label format allows for self-routing of control directives and to discern media types, priorities, and the operational semantics of a resource. This format allows to send and receive composite floor directives and reduce the amount of control traffic, which is particularly critical for large and highly interactive sessions with many floors and resources.

Assume that a node $a$ with $l(a) = 1001$ wants to submit a FRQ CD, and finds $FH = d$, with $l(FH) = 110$, in its local record. The request percolates up the tree across the root node and down on the right branch according to the label prefix semantics. The label format allows for self-routing of control directives and addressing of specific nodes in the multicast group, simply by knowing their logical address, and without nodes having to identify themselves otherwise. Until FH receives a FRQ, it multicasts resource state updates to its immediate neighbors, which forward the information to the session remainder. CDs are aggregated at hop or extra nodes, by coalescing requests or responses from neighbor nodes into single CDs and forwarding them accordingly. Concurrent submissions are resolved by attaching a timestamp to each control packet, which gives preference to the earliest submitted request. Node $d$, after completion of its resource operation, multicasts a FGt message to node $a$, which also informs all other nodes in the session via flooding about the new position of $FH = a$. Each node maintains an entry for each floor on the relative path direction (next hop) in the tree, where FHs are currently to be found. An FH also may keep a record that indicates the number and type of floors granted to each receiver, and establish a fairness policy based on service priorities. Furthermore, denied requests may be queued at FH and held pending, until they can be fulfilled, or are timed out. Implicit routing allows also to address coterie members in a floor-controlled multicast group by listing individual labels as destinations, which is also an elegant solution to the anycasting problem. No updates regarding the position of the FC and FH are disseminated, if these role assignments are static.

On the average, leaf nodes must compare more bits in the control routing procedure, than nodes close to the root. The label cardinality depends on the tree branching factor and the session size. A tree with $n$ session participants and branching factor $B$ has $\log_B n$ levels, with $\log_B n$ bits needed for node labels. The virtual re-hanging procedure on the tree may cause imbalances with regard to the average pathlength and create unfairness in the time-based allocation of floors. However, since spatio-temporal information regarding control message origin is available to the respective FH, floor allocation can include also criteria such as
distance and promote uniform treatment of stations. A basic algorithm for labeling and routing of floor control messages, including only FRQ and FRL cases and separation of FC and FH, is shown in Figure 12.

set session graph S ← C;
set floor f ← state(resource r);
/* initialize floors, f natural, for each resource */
protocol HGCP();
start session C;
LABEL(CONTROL_TREE (S));
while C is running
begin
  foreach f do
    ROUTE_CONTROL_MESSAGE (f);
end

LABEL(CONTROL_TREE (graph G))
begin
  construct tree T ≤ E from G with root r;
  set( r ) = 1;
  LABEL_SUBTREE (r, T, l(r));
end

LABEL_SUBTREE (node s, tree T, integer β)
begin
  set α = 0;
  foreach child c of s do
    set( c ) = α + β;
    increment α;
  LABEL_SUBTREE (c, T, l(c));
end

ROUTE_CONTROL_MESSAGE (tree T, node x, node c, directive p)
begin
  /* multicast to the children of x with matching prefix */
  HANDLE_FLOOR_DIRECTIVE (node x, node c, directive p);
end

HANDLE_FLOOR_DIRECTIVE (node x, node c, directive p)
begin
  case p
  FRL: if c == FC
    then FH = nil;
    FRQ: if c == FC and FH == nil
    then c sends FRQ to x;
    else if c == FH
      or if FH != nil
      then FH sends FDL to coterie;
    /* percolate floor busy message back along tree */
  end case
end

Figure 12: Basic HGCP Specification

5 CONCLUSION

Software to support multiparty interactivity and coordination becomes more important with improvements in networking technology, allowing more sophisticated ways of telepresence and real-time collaboration. The current state-of-the-art of reliable multicast uses single shared acknowledgment trees to warrant scalable and failure free message dissemination within large multicast groups. A primary application area for such protocols is telecollaboration, however, protocols to coordinate large groups with regard to resource access are lacking. This paper compared known solutions for permission- and token-based floor control and described the first tree-based floor control protocol embedded with reliable hierarchical multicast. An implementation and further performance assessment are under way.

6 ACKNOWLEDGMENTS

This work was supported by the Defense Advanced Research Projects Agency (DARPA) under grant F19628-96-C-0038.

7 REFERENCES


