

Floor control for multimedia conferencing and collaboration

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

Baskin Center for Computer Engineering & Information Sciences, University of California, Santa Cruz, CA 95064, USA

Abstract. Floor control allows users of networked multimedia applications to utilize and share resources such as remote devices, distributed data sets, telepointers, or continuous media such as video and audio without access conflicts. Floors are temporary permissions granted dynamically to collaborating users in order to mitigate race conditions and guarantee mutually exclusive resource usage.

A general framework for floor control is presented. Collaborative environments are characterized and the requirements for realization of floor control will be identified. The differences to session control, as well as concurrency control and access control are elicited. Based upon a brief taxonomy of collaboration-relevant parameters, system design issues for floor control are discussed. Floor control mechanisms are discerned from service policies and principal architectures of collaborative systems are compared. The structure of control packets and an application programmer's interface are proposed and further implementation aspects are elaborated. User-related aspects such as floor presentation, assignment, and the timely stages of floor-controlled interaction in relation to user-interface design are also presented.

Key words: Multimedia collaboration – Distributed application sharing – CSCW – Middleware for session orchestration – Floor control for shared multimedia objects

1 Introduction

Communication in face-to-face encounters is based on sharing the same space, ether, and scheduled times to convey auditive, pictorial, and gestural messages. Computer-supported cooperative work (CSCW) helps to overcome spatiotemporal limitations and allows people to share information from distributed locations at the same or different times. The collaborative environment is a virtual rendition of the “real” meeting space, composed through a network of computers running the same applications. To facilitate remote conferencing, information sharing, and interactive collaboration,

various system and application-inherent resources may be used.

Until now, interoperability, compatibility, and portability have been major concerns for hardware platforms and standalone software. Broadening the scope of software usage towards cooperation in distributed settings introduces a concept that we term *collaborability*. Beyond the established notion of (relaxed) What You See Is What I See (WYSIWIS) for groupware, the goal of such improved collaboration-enabling software is to share information during the work process with finer granularity. Users are hence enabled to share subsets of their work progress and data selectively with other people. Upscaling of groups, more sophisticated networking with increasing range, multimodality of user interfaces, the capability of working with multiple types of media, and a richer spectrum of group interaction create many new problems in group work facilitation that cannot be solved with previous CSCW approaches.

The goal of any groupware is to integrate textual, visual, and auditory communication into a coherent software environment, integrating local-versus-remote and private-versus-public information with usability metaphors based upon real-world customs. However, even though a wide range of groupware tools exists, the modalities for sharing information are often separated and quite limited. This is due to the fact that the cooperative processes among humans, involving issues of conflict, selective attention, self-interest, trust, and privacy are not understood well enough, especially when computers are used to support such processes. Paradoxically, although online cooperation is intended to enhance the productivity and communicative scope of people towards more complex task types, technology-mediated interaction creates new bottlenecks. The communicational bandwidth of people is limited by their collaborative tools, which in themselves and collectively allow only for certain interaction patterns, causing depersonalization, psychological distance, disengagement, and more formal encounters [69].

Face-to-face meetings are spatially open and allow for turn-taking [55] in conversation and activities via nonverbal cues, such as selective eye gaze, pointing (deixis), or raising of the volume. In contrast, in “virtual” meeting and cooperation spaces created by electronic means, implicit commu-

nication based upon cultural, social, and pragmatic conventions often cannot be fully conveyed. This is due to a limited contextual view on remote events and work spaces, i.e., lack of mutual awareness [13] of individual and group activities. While conversational interruptions and informal hand-overs of the “speaker floor” seem to be more common in face-to-face meetings than in virtual meetings, phenomena such as simultaneous starts occur at the same frequency in both meeting types [69]. Other studies have shown that the efficacy of video relies to a large extent on synchronized audio channels with short delay [54].

While a lack of personal presence may impose certain quality limitations in group encounter, *telepresence* can be improved by middleware mechanisms that aid in online facilitation of joint work. The particular problem of who controls the speaker floor, or in a more general metaphor, who controls the floor on activities with specific media and information, has not been addressed comprehensively enough in the past. *Floor control* is a technology that deals with conflicts within shared work spaces. It helps coordinate joint and competing activities among people and their interacting computational processes, such as regulating turn-taking in conversations or write-updates on shared files and preserving coherency of local and remote information. Protocols implementing floor control for online group work need to observe system and network performance constraints, i.e., the quality of service (QOS) for specific interactions, as well as patterns of interaction between humans.

It has been argued that systems providing floor control in the sense of an explicit action-coordination tool “do not recognize or support the need for mutual awareness”, and that people as “efficient managers” of their own would rather need mechanisms for communication breakdown and repair [35]. However, such judgment depends upon the definition of floor control and its task scope. While audio communication with floor mediation may in fact render meetings less spontaneous, general testbeds for conducting floor control research were previously rather limited. Mutual awareness, as provided in the *Portholes* experiment [14], is a matter of group size and availability of communication channels. Floor control can provide a means of avoiding communication breakdown and support management of group work. The crux of remote collaboration lies in issues of (a)synchrony, scale, and modalities for establishing a common work space. The notion of CSCW is too rich to exclude certain paradigms of group work facilitation. Certain types of systems and applications intrinsically need floor control.

The following sections present a comprehensive view on floor control from a systems and user perspective. Section 2 clarifies important terms and further motivates the idea of floor control. Section 3 characterizes collaborative environments and the requirements for realizing floor control middleware. Section 4 discerns session control and floor control as two primary components of a collaboration architecture and discusses the differences between floor control, concurrency control, and access control. Section 5 identifies parameters of collaborative systems relevant to protocol design. Section 6 presents system and user issues in designing floor control protocols by discerning implemented control mechanisms from application-level service disciplines. Various collaborative architectures are described, a basic appli-

Table 1. Collaborative activities classified according to degrees of synchrony and interactivity

Time	Asynchronous	Synchronous
Relatedness		
Noninteractive/ transactive	1 Database access	2 Simultaneous file/ resource access
Interactive/ reactive	3 WWW, email, bulletin boards, workflow	4 conferencing, telecooperation, application sharing

cation programmer’s interface (API) for floor control is proposed, and important user-related aspects, such as interface design and the timely stages of interaction, are discussed. Related work is outlined in Sect. 7, and Sect. 8 concludes the paper.

2 Terminology and principles

Any form of joint work across computer networks on behalf of centralized or distributed applications is called remote collaboration. This form of group work entails issues of coordination, competition, conflict, negotiation, and cooperation. A one-on-one encounter is termed an *interaction*. The infrastructure for remote collaboration is provided through sessions. A session is an online aggregation of session members, also termed *subjects*, co-working on shared *objects*. Subjects are either people or their controlled or autonomous system processes (agents). Objects in multimedia systems are resources (devices, files, user interface widgets, graphical objects, applets etc.) or media (voice, video) facilitating synchronous or asynchronous communication. A multimedia system allowing for various remote collaboration with multiple types of resources is called a *collaborative environment*. The sharing process revolves around contents and control of single and multiple tools, their views, and data.

Fundamental factors for classifying remote collaboration are relatedness and time. Table 1 depicts a classification of application types according to these factors.

Conflicts in data sharing arise in any of the four categories of collaborative activities shown in Table 1 with varying impact on the consistency of data, service delivery, and the users’ workflow. Conflicts in category 1 are the domain of transaction theory and concurrency control; the goal is reordering of queries and transactions to achieve the most efficient and conflict-free data access. Related problems of consistency preservation and replication in updating shared files arise in category 2. Conflicts in category 2 and 3 are resolved with security measures such as access control. Even though the principle of floor control can also be applied to the other categories, its main domain is category 4.

Floor control coordinates concurrent usage of shared resources and data among users in centralized or distributed environments. [We define the “floor” as: “the right to address an assembly” (Webster’s New World Dictionary); asking a chair person for the speaker floor (cf. Robert’s Rules of Order); a computational access metaphor for the speaker floor.] A prime example of a floor control problem is turn-taking

on the audio channel to allow for smooth multiparty conversations. If everybody “speaks” simultaneously, packets from different sites collide and messages get scrambled. Backing off and retrying, chaired control, or mediation through other communication channels are possible, but are inefficient solutions to resolve such conflicts. Even though special hardware is offered to allow for analog telephone-style conferencing and simultaneous speech, the regulation of turn-taking remains problematic, particularly for larger meetings.

Floor control hence mitigates race conditions within sessions on who is allowed to send, receive, or manipulate shared data at which time. Floor control shifts the paradigm of concurrency control for databases into the realm of cooperation among people and intelligent agents within networked multimedia systems. While session control orchestrates membership, and starts up and tears down applications, floor control acts as a supplementary service, regulating the actual application-level modalities of information sharing by enforcing *mutual exclusion* in collaborative activities. It aids session conduction by:

- Reducing nondeterminism, setbacks, redundancy, and inconsistencies in group work
- Providing coordination and decision support for loosely related task groups
- Balancing contributions among session members in a fair manner
- Regulating collaboration via a predictive and binding protocol for all session members to compensate for lack of social protocols used in face-to-face encounters
- Ensuring liveness of cooperation as a countereffect to lack of engagement
- Promoting intergroup awareness, cohesiveness, and integrity
- Performing “application-level admission control” by assigning floors based upon priorities and the likelihood of completing an activity under a given QOS.

Floors are temporary access and manipulation permissions, able to manage access for multimedia data streams that are continuous and open ended, e.g., for speech/video transmissions. Optimistic floor control is the strategy of allowing conflicts and providing means to resolve them. Pessimistic floor control follows more strictly the premise of conflict avoidance. Unlike concurrency control, which heads for solving transactional race conditions in a user-machine paradigm emanating from within the same application, floor control regulates user-machine-user interactivity in a wide variety of online-session types and between cross-linked applications.

Several notions are important to the concept of “controlling the floor”. For facilitating some sessions, a chair provides the necessary group cohesion, moderating activities and hence ensuring rapport among members. *Assistive* floor control aids a session chair in orchestrating a session, while *autonomous* floor control steers sessions without chair guidance. If the control over floors is entirely left to an agent with decision knowledge with regard to floors and sessions, we speak of *automatic* floor control. *Explicit* control is enforced on subjects with predefined turn-taking rules, whereas *implicit* control leaves freedom of choice to subjects, according to the ad hoc dynamics in group work.

We can distinguish between four paradigms to deal with race conditions in cooperative work: they block conflicts with exclusive locks, disallow conflicts with permission tokens, mitigate conflicts by detecting dependencies and re-ordering activities into nonconflicting series, and resolve inconsistencies created by conflict. The first two are restrictive and prevent conflicts; the latter two are permissive and allow for progress into conflict with pre- and postconditions. Floor control can be relaxed for concurrent activities if the chance of direct conflict is smaller (e.g., in joint editing of text paragraphs), but it must be strict in opposing activities such as speaking over the same audio channel.

Clearly, floor control is a user-centered concept, and implementations need to walk the fine line between user expectations and the constraints imposed by the system and the network. Beyond guaranteeing safe handling of shared data, the ultimate goal of collaborative work with floor control is to allow for maximum synergy in the cooperation among subjects. Ultimately, groupware with floor control should allow for smooth switching between private or selectively shared work [What You See Is What I Share (WYSIWISH)] and total sharing (WYSIWIS). The factors that influence the process of assigning the floor should be transparent in group work; however, individuals and groups in different sections of sessions should be able to parameterize the control process. Finally, human factors aspects such as usability and acceptance of control mechanisms for “telecooperation” as an aid rather than an obstruction also need to be considered.

3 Characteristics and requirements

The design choices for providing floor control are manifold. However, the usage modalities of various resources are helpful in narrowing down the possible ways of implementation. One design premise is to emulate the “real-world” task-typed environment as closely as possible in a collaborative environment. While certain “real” qualities are intangible, new advantages of virtual collaboration can be exploited, for example, the combination of various tasks within one work session. Until now, most collaborative software [46] is specialized for dedicated task types without sophisticated floor control. In many cases, applications run either in a centralized way or a replicated way with exactly the same data on all sites. Popular groupware approaches are workflow and database-oriented, such as Lotus’ *Notes*, where problems with concurrent updates on shared data objects arise. Such problems are even more acute for multithreaded replicators for fields of distributed data [79].

Collaborative systems become more polymorphic with regard to the mixture of resources that can be used at the same time. Also, certain concurrently used resources such as video and audio need additional causal or temporal synchronization. Table 2 classifies typical applications according to their generic data and task types and shows important QOS criteria for floor control. Fields are marked with \checkmark , if the corresponding criteria apply: is work with the resource transient (T) or does it result in persistent data? Is it typically used for synchronous (S) collaboration? Does the resource need real-time (RT) delivery and is it hence sensitive to delay? Does it tolerate lossiness (L) or incur jitter (J)? Is the

Table 2. Resource types and handling characteristics

Type	T	S	RT	L	J	BW	FD
Text							
Editor		✓				l	t, s, f
Chat	✓	✓				l	t
Email						l	t, f
Scheduling		✓				l	t, s
Coding		✓				l	t, s, f
BBS/usenet						m	t, s, f
Spreadsheet		✓				l	t, s, f
Audio							
Speech	✓	✓	✓	✓	✓	m	t
Sound	✓	✓	✓		✓	m	t, f
Images/video							
Still				✓		m/h	s, f
Motion	✓	✓	✓	✓	✓	h	t, f
Graphics							
Still {2D, 3D}		✓				m	t, s, f
Motion {2D, 3D}	✓	✓	✓			m/h	t, s, f
WWW							
Virtual reality	✓	✓	✓		✓	l-h	t, s, f

bandwidth requirement (BW) high (h), medium (m), or low (l)? The last column indicates the floor dimensions, i.e., the kind of conflicts in the temporal (t), spatial (s), or functional (f) domain that floor control typically needs to resolve in group work with a specific resource type. Temporal sharing indicates timely or causal conflicts, e.g., in the alternation of speakers in conversations. Spatial sharing conflicts arise by using the same presentation area or storage space of a shared work space, e.g., pixel areas in a drawing canvas, a fixed-size buffer or paragraphs in a text document. Functional sharing centers around usage of the same application functions that change the status or content of a shared resource, e.g., delete for text files.

Sharing revolves around text cursors, words, paragraphs, sections, pages, or entire files for “editor” documents; single entries, columns, or cursors for spreadsheets; audio channels for speech, etc. The class of numeric data is included in the “text” category, blackboards can be subsumed under BBS (bulletin-board systems), and graphical drawing tools such as whiteboards with telepointers, views, and graphical objects fall into the 2D-still-graphics category. Tools for scientific visualization are subsumed under 3D graphics. Many applications combine resources and media for more specialized task types, e.g., toward decision-support, calendaring, authoring, drawing, or visualization, such as for volume rendering purposes. Floor control for World-Wide Web (hypertext) documents is applicable to contention in updating of links and interactive pages embedding shared resources. Synchrony arises in web-based collaboration, if the embedded resources allow for such synchronous work. Such Java-based groupware applications for the Web are currently emerging. Finally, collaborative virtual reality (VR) tools [28] create a literal need for spatial floor control of virtual objects, next to collision control. The usage of tools is often interdependent – a video channel may assist in conveying nonverbal information, but its usability for most information relies also upon accompaniment with audio and proper temporal synchronization.

Four primary classes of multimedia traffic can be identified [2], all of which impose different QOS requirements on

the collaborative environment. Control packets are needed for floor and session control information, which are mostly of low volume, but require reliable transmission. Real-time packets are needed for time-critical media that can possibly incur some loss. Non-real-time data are needed for resources with soft delivery timing constraints, but with no toleration of loss. Bulky data transfers that typically occur in replication and resynchronization at major session epochs such as startup, recovery, or teardown. Bulky transfers require high throughput and reliable transmission, but can tolerate some delay.

Looking at the multitude of resource types, session scenarios, and network conditions, it is apparent that floor control must be provided either via an adaptive protocol or via a multiprotocol suite that covers various service types. To achieve the best possible service in a collaborative environment, the primary goals and their solutions need to be identified. The following design decisions are favored because they allow for flexibility and stability in service provision:

- *Distribution* of state information storage to yield session scalability, improved performance with respect to multipoint-connection floor assignment, and resilience. The latter allows for fault-tolerant session continuation in case of site crashes or link failures and the implied network partitions. The opposite model, a centralized floor server, would be a performance bottleneck and single point of failure.
- *Asymmetry* of interaction to allow each site to proceed independently with its local work and link into a session with its “idiosyncratic” resource mix, when desired. Resources can hence be heterogeneous, and smooth switching between asynchronous and synchronous group work during live interactions is possible. The opposite symmetric model would require full replication of applications, resources, run-time states, views, etc., at all times.
- *Hierarchy* in session management [60] reflects face-to-face group dynamics. It enables session reconfiguration into subgroups, side-chats therein, and focus on subtasks without the need to split from the embedding session. Attributes and resource usage states from the main session can be inherited from the supersession and overridden. The opposite “flat” membership model has the same drawbacks versus hierarchical sessions as a one-level file system has versus hierarchical file organization.
- *Adaptation* with regard to resources, tightness in control of the shared space, and servicing of requests based upon a desired versus feasible QOS allows applications to adjust to user expectations and the configuration of the collaborative environment. Adaptivity also accounts for scheduling constraints of various resources, for example, for hard or soft deadlines in real-time dependent media. The more interactive and synchronous group work is, the tighter the control of the session state must be. Tightness of control governs the degree of synchrony of sites and their end-to-end coordination. However, based on a “lazy consistency” principle, individual sites need only impose tight control on resources that are live in session, and can disconnect otherwise, making this model suitable for “mobile” collaboration as well. Protocols of the opposite model are fixed on a specific paradigm, ei-

ther tight or loose session control, and are limited to resource management with fixed QOS guarantees.

Protocols that deliver these premises need to be verified according to correctness (“each floor request is ultimately serviced”), promptness (“each floor request is serviced in the minimal amount of time possible”), fairness (“all sites requesting a floor are equally serviced on the average, based on a common metric, and no starvation of single sites occurs”), and stability (“control information remains consistent in the presence of failures and requests are never lost”). For the last criterion, it is assumed that single lost sites and their eventual recovery do not interrupt a session in general.

The suitability of floor control protocols with regard to specific resource types needs to be evaluated according to performance measurements, which take specific QOS constraints into account. For group work, it is unreasonable to detach the notion of what the system can deliver from what the user demands within an interaction. The current notion of QOS, quantified by throughput, burstiness, latency, and jitter from a low-level network perspective, is insufficient to account for high-level interactivity with patterns that again depend on the lower layers.

4 Relation to other control paradigms

We now describe floor and session control tasks and discern floor control from other common approaches to deal with race conditions in data sharing environments.

4.1 Floor control and session control

Providing session management without floor control is analogous to a transportation infrastructure without traffic control. Any architecture that allows entities to coordinate and cooperate needs protocols to mitigate race conditions and to either avoid or resolve conflicts and their implications.

Floor control encompasses the following tasks: provision of mutual exclusion for shared objects in concurrent access, granting or denying floor requests according to the enacted service policy, tracking the status of shared resources at all connected sites, relaying floor requests between sites, managing temporary access rights to data, authorizing or denying usage according to session control information, and broadcasting or multicasting changes of floor control states to collaborating members. The default service policy for shared resources is “free for all”, and conflicts are resolved by serialization of floor requests on a first-come-first-served basis. As mentioned earlier, a floor control protocol must inherently assure safety, timeliness, fairness, adaptivity, and stability.

Session control manages membership, maintains connectivity, and orchestrates sessions, including tool invocation, and provides session state information and teardown, based on a connection management protocol. The “session body” can have a set of designated members with well-known addresses (tight coupling), or consist of an “amorphous” set of users that visit sessions without prior notification (loose coupling). Session control mediates between upper application layers and relays requests down to end-to-end services. It comprises the following tasks: initiation, pause,

resume, and stop of sessions, which are characterized by their purpose and a set of resources. Session participants are validated and tracked via a directory based upon their group membership. Basic membership support includes creation, joining, withdrawing, inviting, excluding, etc., by single members or whole groups. A session control protocol also needs to account for concurrent membership in parallel sessions, switching, overlap, recursive establishment of subgroups, and collaboration across session boundaries.

4.2 Floor control versus concurrency control and access control

There have been various earlier efforts to account for conflicts in group work. We delineate floor control from concurrency control, access control, and its spin-offs, such as version control and dependency detection. The common ground of all control methods in the context of remote group work is *distributed mutual exclusion*. A large number of message-passing algorithms for stable and failure-prone systems exists [72]. Most solutions presume a finite and fixed number of nodes in the “session graph”, targeted at preventing update conflicts on shared and discrete data. A floor control algorithm based upon the mutual exclusion paradigm must allow for multiple entries into a critical section of interactive work, must be resilient to failure-prone sessions, and must allow for dynamic extensions of the node set.

Specific types of multimedia collaboration such as group work with shared graphical or textual documents are based on a database notion of the joint work space. For such scenarios, traditional concurrency control [7] applies. Previous studies on managing concurrent work activities have been conducted primarily in this context [1, 17]. Concurrency control is a low-level mechanism for resolving update conflicts on (intrinsically shared) database records. Standard conflicts reduce to *read-write* and *write-write* pairings and are resolved with transactions, whose operational semantics guarantees the atomicity, consistency, isolation, and durability (ACID) properties. In this classic notion, the goal is serialization of activities. “Optimistic” transactions are carried out despite conflicts, and are *committed* in case of success, if a quorum of all sites votes to accept the respective transaction, or they are aborted otherwise. Already performed operations are rolled back. Some standard techniques are two-phase locking, token passing, logical time-stamping, and majority consensus. Optimistic concurrency control is supplemented by coherency control to preserve consistency across replicated documents.

Advanced concurrency control models [5, 21], which have been mainly developed in the context of collaborative CAD, CASE and editing, employ multilevel updates, nested locking, semantic domain information, its dynamic restructuring, and other refined techniques. In opposition to standard transaction models, such cooperative transactions in advanced models try to relax the ACID premises, allowing for long-lived multiple access, unpredictable events, interaction with other concurrent activities, and user control. Other concepts such as split-level, conversational, or chopped transactions are targeted at similar ideas. However, existing models do not fulfill all of the previous premises at

the same time and have often not been developed beyond the conceptual stage. Furthermore, as with standard transaction models, cooperative transactions are geared towards finite, discrete files with predefined courses of action.

Access control methodology encounters new problems in the context of distributed multimedia collaboration as well [70]. By introducing permission hierarchies and a finer granularity of locks, including views and the coupling of rights on collaborative session objects, a wider range of shared information types can be governed. A special intermediary case between access and concurrency control is version control. In version control, a difference scheme and a locking technique are employed to allow for concurrent editing, coherency maintenance, and economic storage. Access control is used in “static” file systems, and permissions on resources are permanent. However, despite a similar problem domain and methodology, there are major differences between access control and concurrency control versus floor control.

Contrary to concurrency control, resources in multimedia collaboration can be continuous or discrete, open-ended or finite, and located within multiple concurrent applications. Rather than being handled transactively from users to a server, I/O is interactively regulated between users. Sessions provide an infrastructure for binding resources and users together that transactive models lack. User entries are multimodal, depending on the resource type, and span files, instruments, and application-intrinsic objects with finer granularity than is required for database entries. For continuous media streams, optimistic execution of conflicting activities is infeasible, since conflicts carry through to the user. Accordingly, the semantics of concepts such as “commit” and “rollback” is more limited, and a control scheme must avoid conflicts in a “pessimistic” way, rather than resolve. Undo or redo operations can stall the workflow and may be costly with bulky data transfers. Serializability constraints on resources are different from transaction-type activities, since the service order for multimedia resources often does not commute. While transactions predictively complete at commit time with a definite result, interactions follow a pattern of open-ended turn-taking, and often have no sense of completion. Also, people make mistakes and need to reassess statements and actions during their turn for repair or additions under supervision from a chair or session peers [42].

Hence, floor control must regulate repeated access to the same objects, observe synchronization dependencies among concurrently used resources, and allow for multiple floors on the same resources. Similar to file protection, users must have a choice of system control versus user control over the information flow within and across sessions. Atomicity and durability conditions, as stated for transaction models, must be relaxed for interactive work. Since multimedia group work is more tangible through the user interface than database sharing, more sophisticated presentation and manipulation choices must be given to users. The rules of mitigating contention on shared resources must be more flexible with regard to buffering, allocation, and scheduling.

Opposite to access control and its “static” permissions, floors are *ephemeral* permissions on transient cooperative events in dynamic sessions. For discrete files, floor control regulates the cooperative course of action, using the underlying access control mechanism to impose different

lock modalities on the shared object. Instead of “read/write” conflicts, the semantics of conflicting multimedia activities extends to more permission types – “transmit” for audio channels, “modify” for shared editors, “change position” for remote instruments, etc. Similar to file system modes for users and groups, a floor can carry several permission types in conjunction if the associated resource allows for this.

5 Floor control parameters

A collaboration environment C is a quadruple

$$C = (S, U, R, F)$$

of a set of sessions, $S = \bigcup_i S_i$, a set of users, $U = \bigcup_j U_j$, a set of resources, $R = \bigcup_k R_k$, and a set of floors, $F = \bigcup_l F_l$, where i, j, k, l are natural numbers. A floor is associated with a resource and is granted to users working with the resource. Sessions can aggregate into supersessions and recursively form subsessions depending on the group dynamics of the startup sessions. Such online session reformation can be permanent or temporary, and session attributes are inherited from supersessions by their children, but can be overridden. This concept of hierarchical session control [59] reflects the organizational nature of face-to-face meetings. A temporary subsession, which is also called a coterie, allows for concentrating task threads and side chats to a smaller scope of selected session members. (The term “coterie” in this context is related, yet slightly different from the concept of coterie used in distributed systems and database theory, particularly in the context of voting [26].) The activities within coterie are invisible for the wider session audience. A simple instance of this subgrouping concept is, for example, provided for textual discussion groups within the Internet Relay Chat (IRC).

Similar to the dynamic notion of sessions, sets of users also form static groups with structures that can serve as membership templates for sessions. The online session aggregation can then be altered in the course of online activities. Additional users can join open sessions as temporary guests or prospective members, whereas closed sessions allow participation by invitation only. Sessions with overlapping or diverging interests can merge or split. Such reconfiguration of sessions with regard to membership and session events linked to specific phases must be possible without session termination or restarting of applications. For specific sessions, each user can have a default set-up for collaborative tools and the possible sharing modalities via control default lists, given a task definition with the intended session purpose and floor control regime. Users who are concurrently members of several groups can also participate in multiple sessions at the same time by being active in one session and idle in others. Floors can be granted tentatively to idle members in order to rekindle them for active participation. Resources and their subcomponents can be controlled at large or fine granularity.

A sample scenario for a collaboration environment with three sessions, two subsessions, and one coterie is given in Fig. 1.

Depicting membership in set-theoretic notation, we have $C = \{S_1, S_2, S_3\}$, $S_2 = \{s_1, s_2\}$, $c \in s_2$. Figure 1 is simpli-

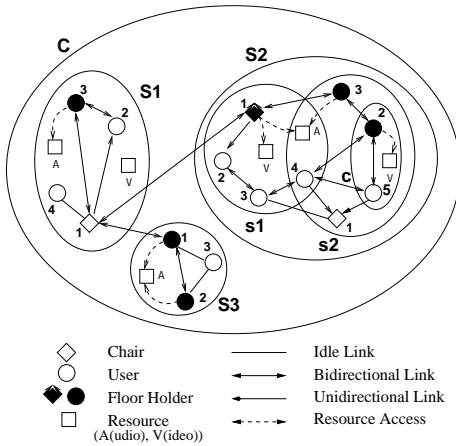


Fig. 1. Snapshot of sample hierarchical aggregation of multiple sessions

fied in that only a few members and resources are shown, with each “cluster” competing for their respective resources. For example, the coterie c between users U_2 and U_5 is utilizing video. The clustering of users, groups, and sessions is an abstraction based upon session attributes and does not necessarily reflect geographic proximity. Except for S_3 , all other sessions are chaired. Such roles can be assigned a priori, or online with the consensus of session members. The intersection between s_1 and s_2 elicits the sharing of resources across session boundaries so that users can actively be part of two (or more) sessions at the same time. Floor-controlled chair activity can preempt other member’s activities.

A taxonomy of attributes for sessions, users, floors, and resources helps in selecting appropriate control strategies for collaboration facilitation. The following discussion of session and floor characteristics is intended as a basic compilation, not as a complete account. Users should be able to customize parameter settings that influence the workflow for individuals and entire groups.

Sessions S are characterized by their size (in terms of the number of people), their structure (flat vs. hierarchical), membership dynamics (creating, inviting, joining, pausing, switching, withdrawing, and leaving users), purpose (lecture, council, panel, examination, interview, etc.), duration, agenda, organization (informal, formal), decision procedure (consensus, voting, chaired), and distribution scope (local, wide area, global). The latter implies low or high latencies, and hence waiting times for floor turn-arounds. Each combination of such session attributes creates conversational and collaborative problems of its own and alleges strategic opportunities for the participants.

Users U are characterized by their aggregation (individual vs. group), social role (chair, speaker, proctor, auditor, notetaker, interviewee, etc.), identity (anonymous, known), authority (full, restricted, privileged), their entry type (single, group consensus, recorded), the resource or medium in usage, and finally the link used for collaboration (sending only, receiving only, or bidirectional). Links can be unidirectional, for example, when somebody supplies data, but needs no feedback. The “social” roles of session members reflect the group orientation and direct the session purpose, imposing rules of formal or informal interaction on its par-

ticipants. Roles can be predefined before session start-up or designated to members at run time.

Resources R are objects within different application classes, as depicted in Table 2. The resource type characterizes whether a resource is text based, graphical, or some real-time medium and identifies the purpose it serves. Resources can be virtual, or they can represent actual remote devices, for example, a surgical device in telemedicine. Resources need not necessarily be proprietary to the session from which they are accessed, but could also be hosted from a machine “outside” the session. A security attribute denotes whether a resource is public, private, or proctored.

Floors F can be characterized by their state (held, idle, requested, free), their granting and releasing process (explicit by the chair, implicit by time limits, etc.), their function (operational, feedback), their attribution policy (queued or non-queued), the granularity (attachment to entire applications, windows or graphical widgets etc.), and permissions (read, write, execute, move, transmit, annotate etc.).

Among all four entities in C , a causal relation determines the current instantiation of the network-wide floor set F . In a nutshell, sessions comprise a varying number of users sharing (possibly session-specific) resources and media, all of which are controlled by floors in finely grained access and usage.

6 Design issues for floor control protocols

The design of floor control services [11, 12] must address system and user interface issues. System design entails architectural considerations, the control packet structure, dissemination and maintenance of control information, disciplines, and exception handling. User-related issues entail presentation, triggering, service, and manipulation of control with regard to timely progression in collaborative work. While the system perspective refers more to the implemented persistent *mechanism* of floor control, the user’s perspective is more influenced by case-based floor assignment *policies*. Certain policies are only compatible with a specific mechanism.

6.1 Mechanisms versus policies

The basic distinction between floor control mechanism and policy is due to [9]. While the floor control mechanism regulates uniformly the low-level control flow and event synchronization across all sites, the policy determines how floors can be requested and granted, allowing for reordering, preemption, and change of “prescribed” servicing. Sample mechanisms can be derived from mutual exclusion, concurrency control, and multiple access methodology. They incur trade-offs between responsiveness, fairness, and resilience:

- *Negotiation* is used for “anarchic” ad hoc floor reassignment. It is commonly performed independently of the context within which the resource contention occurs and builds upon a free-for-all policy or a “no floor” scheme.
- *Token passing* offers floors to session members in a predefined hop sequence across all sites; if the service order is relaxed, floor tokens can also be passed in the sense of

“chalk-passing” to designated successors by telepointing, etc.

- *Token asking* lets users actively bid for obtaining a floor from its current holder or the chair rather than having the floor token automatically pass by all participants according to a given topology as in token passing.
- *Time stamping* marks requests according to a globally synchronized (logical) clock mechanism or sequence number in order to achieve ordering of events. This mechanism is often used together with others to account for correctness.
- *Two-phase locking* establishes floors as locks on shared documents in a growing phase and releases them after activity completion in a shrinking phase, which prohibits acquisition of new locks.
- *Blocking* via distributed semaphores guards critical sections of group work and inherently establishes a first-come-first-served servicing order.
- *Activity sensing* [25] is a contention resolution scheme in which perceived channel activities cause colliding requests to back off until the active site has finished. New floor requests can be reissued thereafter.
- *Reservation* is performed by allocating specific resources in predetermined sequences, durations, or time slots.
- *Dependency detection* attempts reordering floor requests and activities according to the causal semantics of the underlying data.

Floor policies determine definite assignment sequences and are based on the implemented mechanism and session parameters. Crowley et al. [9] classify policies according to whether they are explicitly or implicitly requested and granted through the user interface. In more detail, we classify policies as nonqueueing or queueing, non-preemptive or preemptive, and taskdependent or taskindependent. The last characteristic presumes that a system estimation based on user input or task information is given for the amount of time required to complete the task for which a floor has been requested. This allows the scheduling of floors and hence tasks on a duration or deadline basis, similar to real-time scheduling disciplines. Furthermore, policies are either mutually exclusive or mutually selective. Either one floor is assigned to n users on one resource, or k floors are assigned to n users (with $k \leq n$) and instances of the same resource, e.g., in a whiteboard with multiple telepointers. While exclusion avoids write conflicts, selection allows for relaxed control, but does not guarantee the avoidance of conflicts. It is also possible to exert selective control within a coterie and exclusive control for the outside session, allowing for negotiation with “social” protocols.

For interactions among users, long delays created by either queueing or long floor passing times need to be avoided, since the workflow would stall. Using queue information or a given agenda, successors of floor holders could already be primed on floor take-over before a floor is relinquished by a predecessor. The notion of “request order” is relative to lower-level transport strategies and current network conditions, and it can be changed by adapting different application-level floor policies. Sample policies are:

- *Chair guidance*, in which an elected user is the arbiter over the usage of specific floors. It is also possible to

separate the chair role from the arbiter role for specific floors.

- *Agenda orientation*, in which floor assignment pertains to a given schedule or task order that is defined before the session starts or is created on the fly, depending on the session purpose.
- *Time orientation*, in which floor requests and usage have time-outs defined by events or system conditions. Each floor holder can have individual time-out constraints.
- *Predefined ordering*, in which floors are offered or requested strictly in sequence on a path in a given topology of collaborating sites. A standard example is round robin, pertaining to token passing. First-come-first-served ordering is also subsumed under this policy.
- *Ad hoc reordering* enters requests for each resource in a queue and serves them according to timeliness, priority, and QOS criteria. Sample service disciplines are demand-oriented, least-recently-served, weighted-fair-queueing, earliest-deadline-first, etc.
- *Election* lets users on involved sites vote on the next floor holder or the acceptability of an operation from a specific site. If majority consensus is achieved according to a given quorum, the new holder or resource activities can be approved; otherwise, a new election must be started.
- *Lottery scheduling* uses lottery tickets in a probabilistically fair scheme for floors assignment, cf. [75].

Further policies can be derived from buffer service disciplines in scheduling theory. A notion of “floor credit” could also be introduced for specific floors to balance the load, taking the “cost” of certain resources into account. Specific policies are appropriate for certain resources, session purposes, and mechanisms, but ill-suited for others. For example, a predefined ordering scheme is suited for instrument control, but not for unpredictable turn-taking in conversations. Performance evaluations, together with human-computer interactional studies, need to qualify and quantify these options and their compatibilities.

6.2 System-related design

To provide session management and floor control, we can identify two major paradigms: centralization or distribution together with replication. Even though a centralized scheme is perfectly suitable for management of small-scale sessions (in which, for example, a set of remotely controlled devices is steered from one site), a distributed approach is advantageous in terms of load-balancing, responsiveness, and fault tolerance. Architectures for collaborative systems can be categorized into three classes, each of which determines a different floor control paradigm:

1. Collaboration-unaware systems provide no collaboration services themselves, but utilize external processes to relay control to and from other sites. For example, X events can be tapped, filtered, and multiplexed to distributed sites by a pseudo-server outside of the application, which then grants or blocks access to the shared objects.
2. Collaboration-aware systems have collaborative services “hardwired” into the application. Session orchestration and floor control are all managed within independently running instances of such applications.

3. Collaboration-transparent [43] systems take a hybrid approach and extract collaborative services into modules (daemons/agents) outside of applications. On each site one such floor agent is always running and relays control information from and to local applications to agents on other sites that service their local applications. Their collaborability is intangibly defined inside, but actually interfaced to and regulated from the outside. Such applications can run standalone, as well as in networked mode, and are informed about distributed events via the agents.

Floor and session control can be tied together into one component, or can be provided in separate modules. A reasonable guideline is to implement session and floor control coherently within a chosen collaboration architecture. For example, in a “transparency” model, both a session and floor daemon would as separate, yet interfaced, modules keep track of the distributed progress of live collaborations. Any change in the session domain is reflected in the floor domain and vice versa. Control packets from both components must be delivered reliably and in order. In a sense, a “symbiosis” between session and floor control equips users with an infrastructure for application-level control of activities and implied data traffic. Of course, next to floor and session control, there are many more issues to be accounted for, e.g., authentication, replication, resource allocation, rate control, and temporal and causal synchronization [78] among specific media types.

Session members assume social roles with regard to the session purpose, and control roles with regard to floor assignment. While a chair facilitates a session, the floor arbiter controls floors for specific resources as a “floor superuser”. Both roles can be assumed together and exert preemptive control on some or all floors within a session. The user who creates, owns, and contributes specific shared data, is called a floor owner. In the case that such a member exits the session, the associated shared resources either disappear from the public work space, migrate to another site and owner, or the current owner can hand over ownership to a remote site by allowing remote access. The user or agent attaining a specific floor at a given moment is termed the floor holder. Session participants can also assume multiple control roles at the same time or none at all. Owning, controlling, and holding in this order are decreasingly persistent system roles. In metaphors, session guidance can follow three “political” paradigms: chair guidance through a predesignated or ad hoc elected member (“monarchy”), through a set of privileged leaders (“democracy”), or without any hierarchical control at all (“anarchy”). Certain meetings, especially smaller ones, typically do not have a designated chair and are steered “self-governed” by all members.

Possible control protocol states for floors are *idle*, *free*, *remotely used*, *locally used*, or *requested*. In the provision of floor control, many parameters within a session need to be renegotiated constantly within one site and all collaborating sites, as well as across different sessions. A control state table for all floors is kept at each site participating in a session. Session events, such as withdrawing, joining, or side-activities in coteries, are also reflected in state tables. The local and remote allocation of such tables,

Table 3. Floor control packet structure

Field	Description
SId	Unique session id
HN	Host name
GId	Unique group id
UId	User (agent) id
RP	Role-based permissions
AId	Application id
RT	Resource/media type and instance
FId	Unique floor id
F#	Number of allowed instances
FS	Current floor state

their replication, and cross-referencing are left to a detailed floor control protocol working in conjunction with the session control. The control block structure to uniquely identify and transmit actual control parameter instances within sessions is given in Table 3 and allows for session conduction and recovery.

SId and HN are unique designators for the particular session and machine. GId reflects the aggregation of users within the set of running sessions, their relation to groups and multiple activities. UId and RP identify a particular user, his or her function, and authentication within the session, comprising both the assigned task (note taker, speaker etc.) and the current status with regard to floor usage (floor controller, floor holder, or participant). A priority value can be attached to allow for preemptive task completion or preferred floor attribution for specific holders. AId identifies the application in use, from which a particular RT, i.e., instance of some resource such as text, voice, video, emanates. FId and F# characterize the specific floor in use and its number of concurrent instances. Their values depend on the number of floors that a resource concurrently allows, as in the case of a whiteboard with multiple cursors and telepointers. Finally, FS reflects the actual state of the floor-control protocol, i.e., whether the respective floor is free, granted, requested, or in migration. Such states must be tracked for local and remote actions in order to capture and control collaborative events to and from the local site. For time-critical media the control packets need to be transmitted on behalf of a real-time transport protocol [4, 68].

Until now most sessions are on a small scale. Dissemination of bulky data streams, as in the case of video, can be a problem with point-to-point connections and unicast transmission, since each packet must be sent multiply to all sites. Rather than transmitting multiple copies of the same packets to different receivers, multicasting packets can reduce the session traffic. Compared to data streams, the size of floor control information is rather negligible. However, reliable delivery (no lost packets and “correct” ordering) must be ensured.

Concurrent floors can be used for the same resource if the collaborating node sets are disjoint, and floor coupling can account for synchronization constraints between resources. Figure 2 depicts a “session graph” with a simple topology of five collaborating nodes. The resource type is indicated next to each layer and solid arrows indicate live messages. Since user pairs (U_2, U_1) and (U_3, U_5) do not overlap in combined use of video and speech, two concurrent floors can regulate these parallel interactions. Coupling constraints are

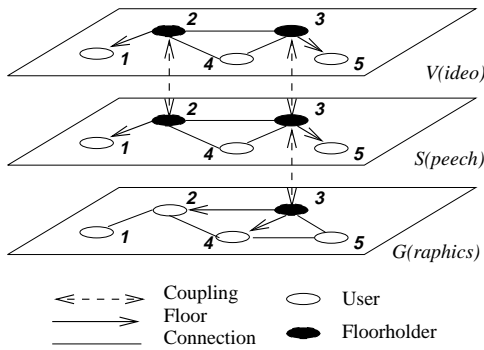


Fig. 2. Intramedia floor concurrency and intermedia coupling

expressed as pairs (floor holder, resource) c . For the scenario in Fig. 2, we have $(U_2, (V, S))_c$ and $(U_3, (V, S, G))_c$. The latter constraint represents, for example, synchronized usage of telepointers with speech, as it may be the case for a lecture. Floor coupling applies to temporal synchronization in the merging of separate media streams as well as resources that are inherently synchronized through their presentation or recording process. An example for provision of video-audio coupling is given with joint usage of the Mbone tools `vat` and `vic` [48] where video streams are presented together with audio.

Due to the variety of resources, sharing modalities, and online activities, the number of actual floors wandering in the collaborative environment can vary greatly. An upper bound is given by the number of all resource components shared in disjoint partitions of all users, i.e., the cardinality of the union of all resource sets. Users can control as many floors in parallel as there are sharable resources compatible in usage, unless activities are preempted by a chair or floor arbiter.

In the ideal case, sessions are stable – machines never crash and connections never fail. In a more realistic scenario, however, sessions must be resilient to failure. For small sessions, a network partition caused by a failing link might end (and possibly postpone) the affected session. For large sessions, loss of members must allow for continuation of the remainder session within the partitions and eventual healing by resynchronization of all intermediate work efforts and data. Exception handling includes preventive tasks such as checking for liveness of session members' sites, logging of control information and activities, repairing tasks such as re-election for social and system roles, and re-establishing coherency among sites.

Security in collaborative systems needs to be ensured, since new security holes emerge due to the many sharing modalities. To prevent *floor forgery* and unauthorized modification or copying of shared information, cryptographic measures must be employed with regard to the storage of floor state tables, their replication, and the transmission of any control packets altering the session state. While only those sites actively participating in a session need to ensure global coherency, security must be enforced for all live or idle sessions and each floor request must be authenticated, particularly in the case of rejoining sites.

Table 4. Sample floor control primitives and functional semantics

Call	Semantics
floor add	Add template for f to active floors
floor claim	Request idle or used f and enter queue
floor create	Bind new f uniquely to r and s
floor expand	Enhance scope of f to multiple subjects
floor freeze	Freeze usage of f for withdrawing r
floor grant	Grant f for r to s
floor info	Give binding attributes and usage state
floor kill	Eradicate all floors for specific r or s
floor migrate	Make f move to other s without release
floor relinquish	Give up f after activity completion
floor remove	Remove inactive f
floor reset	Reset floor attributes for whole session
floor revoke	Force release of f from s
floor shrink	Reduce scope of f to less subjects
floor track	Track history of f during session

6.3 An API for floor control

In order to enable an integrated floor attribution service, each application needs a well-defined interface to the floor daemon. The suggested application programmer's interface (API) in Table 4 is a first step towards standardized procedures for collaborative applications. The calls relay control information between subjects at different sites. f depicts a specific floor; r , a specific resource, and s , a specific subject (user, agent). The postcondition of each call is stored in the floor state table, which is replicated to all sites with active session members.

6.4 User-related design

Floor control is an application-level concept made tangible by the user interface. Sessions have a self-organizing quality and users impose expectations on the system in the triangle between standalone work, group work, and the performance of their machine and the underlying network. Essential user-related issues are the presentation of floor information and control options, triggering of floor passing, consistency maintenance across all sessions, and the timely stages of interaction with regard to floor bookkeeping. The alignment of related information during shared work, participant autonomy, distinction between shared and private spaces, separation of conference-control and application-related commands, agreement upon conference roles for participants, and a continuous presentation of the conference status are related issues, which we do not discuss in more detail.

For each resource, the QOS priorities are different. Depending upon the application type and tasks, users' expectations may focus on high throughput (e.g., a high video frame rate) or low delay (e.g., for audio), and accept trade-offs such as a certain degree of lossiness or jitter. Next to quantifiable QOS figures, other important qualitative criteria fostering acceptance of collaboration control are fairness, intuitive correctness, practicability, and nonintrusiveness. Also important is the responsiveness, measured by the time to locally process requests and reflect them in the user interface (response time), and by the time to propagate them to other sites (notification time) [18].

Floor control protocols must observe user expectations, as well as system constraints, by taking into account resource allocation and scheduling measures in order to fulfill performance guarantees linked to the usage of a shared resource. Ideally, each user should have the impression that the shared “session space” is seamless, similar to a file system tied together from distributed sites via a network-file system (such as NFS). Furthermore, users must be able to understand the reasoning behind floor assignment, to influence, and switch on or off the control at each level of granularity, ranging from resource components to their applications or, with group consensus, the entire session. If users feel that floors are controlled reasonably by the system, fallback to manual floor management and “social power plays” can be avoided.

We now briefly discuss important user-interface issues. Information on private, sharable, and shared resources needs to reflect the current control status with regard to arbiters, owners, and holders. Only public resources are floor controlled. Preset floor lists, similar to access control lists as protection and capability structures, can be provided as default. Individual user settings can then override such default floor attribution. Control granularity of specific resources should be adjustable, i.e., certain properties or functions could be exempt from being floor controlled depending upon users or the session purpose. Floors can be coupled with window states; iconization of a window could, for example, disable all floors for resources accessible through that window. Customization of control parameters should be uniform for all resource types, as in pull-down menus that let the users select floors for all desired resources originating at the local site or from remote sites.

Cognitive cues such as transparency or color in the visual representation of a shared resource are helpful to indicate its current floor state. An opaque or green rendition of a widget could depict a locally held floor, a lightly shaded or yellow rendition could indicate that a floor is requested by a remote site, and a transparent or red object icon could signify that the floor is held by a remote site. Auditory cues can also be used as support and pointing at a locked resource could hence trigger an auditory signal to depict the sharing status. A common solution to mark multiple floors for the same visual resource, such as telepointers, is to label them with (colored) name tags identifying their holders, although this approach is only practical for small sessions. For a “stateless” floor, which is in transition between two users (already assigned but not yet taken), the interface must mark or block the associated resources to avoid inconsistencies or multiple assignment of the same floor.

Once a floor is granted, there are various options by which floor release and consecutive assignment can be triggered or enforced:

1. *Chaired* control leaves decisions about whomever is allowed which activity to the current chair or arbiter; however, a human facilitator can only overlook a limited number of users, resources, and activities.
2. *Signaled* control is based on cues given by input devices. Examples are mouse-triggered, voice-activated, or gesture-based floor requests or releases.

3. *Timed* control allows us to set time limits on the holding duration, either in real time or other metrics, such as the number of requests received from remote sites. Users must then be signaled to “finish up” or “get ready” with regard to resource access.

4. *Event-based* control makes floor movements dependent upon the presence or absence of system events, such as certain work conditions or arrival of specific packets.

Methods 1 and 2 exemplify explicit floor control, triggered directly by users, whereas methods 3 and 4 represent implicit floor control by an automatic mechanism. Explicit floor control is particularly suitable for real-time interactions, whereas implicit control works well for asynchronous task completion involving resource competition. A queuing scheme that buffers multiple simultaneous requests may cause delays in service completion, but generally does not allow handling of time-critical media. Absence of any limitation on the floor-holding time leaves mediation of competition for floors to social protocols. Such a scheme may result in unfairness for larger work groups. Floor passing may not depend solely on the activity of a shared resource, since a pause in the activity does not necessarily imply that a user is ready to release the floor.

Once floors have been granted and shared data has been modified, the problem of the collaborative undo arises [57]. Of course, the transfer of live streams (video, audio) cannot be reversed. Also, certain collaborative actions may involve the transfer of bulky data that cannot be logged. Otherwise, *undo floors* can be granted in reversible order of the activity sequence that led to the current resource state. “Undos” may then be carried out either with user assistance or automatically. Furthermore, a session history allows for collaborative redo or macro definitions of joint work steps. This also helps in analyzing or teaching work procedures, either to improve the workflow or to understand the results gained. Floor caches can support “redos” and improve responsiveness. The concepts of the collaborative undo and redo are also helpful if data on different sites become inconsistent and recovery or resynchronization is not practical. Total coherency of all distributed data, as well as that of interfaces and displays on demand need not necessarily be preserved at any time. It can rather adhere to a model of viewer independence and lazy consistency [51], which supports incremental updates on specific resources only in case of their usage.

Collaborative user interfaces need to reflect user roles, session organization, and activities. In that sense, the local “look and feel” needs coherent integration with the remote sites’ desktop and must observe the reciprocity rule [22] – local-to-remote activities imply a remote-to-local notification or feedback. Furthermore, very large sessions involve many resources and collaborative activities that can cause cognitive overload for users. Traditional windows, icons, menus, pointer (WIMP) interfaces reduce on-screen information by opening, closing, or iconifying documents. This may clutter the work space with too many icons, overlapping windows, and multimedia activities. To let the user gain a better overview of the collaborative space, a multiscale interface [24, 62] allows for zooming and shift of attention on live floor-controlled activities within a seamless session space. Fluid navigation and switching between visually clustered

session activities may help to cope with the amount of information. Also, a panoramic view on remote collaborative spaces and their peripheral activities may improve group awareness.

6.5 Interaction patterns and stages

Interactions go through various timely stages that affect floor generation, bookkeeping, and passing.

Initiation. Floors are created with session start-up or joining of a running session. For persistent sessions, adherent floors are always live. Temporary sessions induce the creation of floors for resources that are declared public. Floors are also reserved for sharable resources not made public. In the case that a resource consists of several components, floor instances are created for each sharable component. Examples are a microphone with mute and volume buttons and a window with a brush pen and a telepointer as sub-components. New shared resources and floors can also be introduced on the fly by any user. Control information is to be stored in control state tables. Depending upon the location of shared resources (at the local site, at remote sites, or at another distinct location, as in the case of an instrument, some field location), sender floors filter out which sites are able to transmit their data to the local site (What I See Is What I Want) and receiver floors regulate which sites are eligible to receive locally generated data (What You See Is What I Share). In addition to multicast groups, such floors help reduce traffic. Users can specify their sender and receiver floors and hence “customize” the multicast groups at start-up or run time.

Conduction. During a session, the floor state table keeps track of all local and remote movements that affect the work process and data consistency. Requests for a floor and its state information are multicast to all sites that use the associated resource. Depending upon the floor policy, floor requests can be buffered and serviced in a different order, or perhaps only the first request is serviced and follow-ups are discarded, until the same floor becomes available again. The latter solution refers to face-to-face meetings, in which a tentative speaker in a lively discussion must claim a floor repeatedly in order to obtain it.

From daily conversations and activities, we observe that next to a main “interaction theme”, the provision of feedback can signal the floor holding sender that the receiver is alive and confirms or objects to the progress of the interaction. Similarly, and in order to allow for short-term interjections in collaborative activities, a primary floor controls the main task, whereas very short-term feedback is provided by a secondary floor. This way, bidirectional cooperation is established, and accounts for interruptions by “out-of-turn speakers”, which may delay completion and cause overlap [44] of a floor holder’s action. The duration and condition under which such back channels are established can vary with session characteristics. This dualistic alternation between floors is not only applicable to real-time media, but also to any

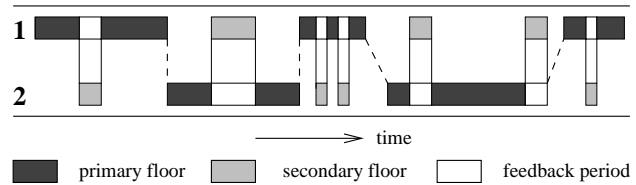


Fig. 3. Sample timeline for alternating primary and secondary floors

resource with a possibility of short-term feedback. Common sense suggests that the need for back channels rises with the degree of interactive work and frequency of turn-taking. Figure 3 displays the conversational process in a two-point connection and the “flip-flop pattern” of action and affirmation via a back channel floor.

The sharp hand-off times between primary and secondary floors, marked by dashed lines, are in reality minute switching intervals due to round trip delay. To ensure consistency, a floor is assumed to pertain to a participant until its migration to another user is confirmed.

Termination and exceptions. Two distinct cases of exception that lead to floor loss are intentional and involuntary withdrawals. The former is caused either by expelling a user from a session or the regular leave of a user, or termination of the entire session. Involuntary loss is due to site or link failures. Floor state information needs to be kept during a session in case a member leaves temporarily or a damaged session recuperates. If a session is chaired and the site of an arbiter crashes, another member must be elected as the arbiter. Similarly, if a floor arbiter leaves a session, we must ensure that all outstanding activities supervised by this arbiter are concluded. If a floor owner crashes, the data originating from that site disappears from the session, and the remaining session body must be notified. If a floor holder’s site crashes, the floor must be reassigned to a successor. In any case, the orphaned floors need to migrate, and the loss or change in control roles must be recorded in the control state table. In case of network partitions, sessions can continue with the consent of their “live” members, and a coherency protocol can be started to realign the control information uniformly into a healed session. It must also be possible to allow for remote usage or authorized copying of shared data even though the owner of such data is no longer present in a session. Floor proxies can be granted to users or sites as place holders for withdrawing session participants or vanished sites to allow the session to continue.

7 Related work

Floor control for multimedia architectures was first addressed a decade ago [2]. Its roots can be traced back to psycholinguistic studies on turn-taking in conversations [65]. Work in conversation analysis [45] suggests that turn-taking relevance points (TRPs) indicate possible hand-over times for floors. Similar issues have been researched in the context of computer-mediated communication (CMC) [49, 52]. Turn-taking in online meetings seems to be unaffected by

video, although the subjective quality of interaction improves if eye gaze and synchronized speech are supplied with sufficient quality [36].

The interdisciplinary nature of the problem of modeling and optimizing cooperation for dynamic group work pertains studies on floor control also to workflow management [19, 29] and coordination science [47]. Concurrency control and real-time delivery issues with regard to groupware user-interfaces have recently been investigated as well [31]. Current work on session control protocols focuses on session scalability [33, 67] or protocol standardization [16, 32], with floor control as a subservice to session control. Interdependencies among resources and temporal and causal synchronization constraints have been discussed in the context of the MCP protocol framework [78], which employs a token-based floor control mechanism. Formal models for multimedia collaboration proposing hierarchical session organization [59] and group membership in asynchronous and synchronous scenarios [58] have also been subjects of research.

To clarify the question of how floor control fits into various task groups and collaboration types, a simple taxonomy for collaborative environments is useful. Unlike the focus of other more general taxonomies [18, 61, 74], our focus is on floor control provision. We distinguish between two main categories of systems: those embedding or overlaying computers with real-world meeting spaces and those shifting the entire meeting space within networked computers. For both categories, race conditions for shared resources can be resolved with floor control:

1. Face-to-face meetings with computer support. This category entails all systems that facilitate collaboration with teleconferencing and telecommunication tools in “same space/same time or different time” meetings. The goal is to increase communication and productivity in face-to-face scenarios. The paradigm of enriching “real” meeting spaces with telecommunication equipment is exemplified with the MEDIA SPACES [6] paradigm, the DIGITALDESK (which overlays analog and digital work spaces with cameras) [77], the shared drawing tools CLEARBOARD-1/2 [38] (which are working with glassboards as digitizer-screens allowing for local work with awareness of remote gestures and processes), the TEAMWORKSTATION [37] (which also merges real desktop activities with computer-represented data via a camera interface and translucent overlay), and Xerox’s LIVEBOARD (a shared large LCD whiteboard combining pen-based input by a chair and wireless transmissions from workstations equipped with groupware).

2. “Virtual” meetings through computers. Systems of this category support “different space/same time or different time” meetings. The goal is to overcome spatial separation. A simple rendition has already been given in UNIX with commands such as `write`, `talk`, and `confer`. Inviting and floor control are performed through `msg` or grabbing the cursor by pressing specific keys in order to write a message. `wall` broadcasts preemptively. Stefik suggested roving locks for the Xerox COLAB system [73] to serve alternating floor requests, where auditory signals and “manual” negotiation between parties indicate floor passing. The idea of automated floor control was proposed for the scheduling

and calendaring systems RTCAL and MPCAL [30, 66]. Lantz’ V-CONF system [41] first realized a floor control unit to interface with a conference agent mediating I/O between shared applications. MMCONF by Crowley et al. [9] features a centralized architecture in which a token-based scheme regulates usage of telepointers between replicated applications, but the floor control protocol can lead to deadlocks or unfairness. A notion of fault tolerance is introduced by recreating floor tokens from a log in case the current floor holding site vanishes. A first attempt to supply a variety of floor passing policies for telepointers was made with the JVTOS system [10].

The current suite of MBONE tools [40] provides a set of generic multimedia collaboration tools for a multicast testbed overlaid on the Internet. The VAT tool enables speaker-activated floor passing in audio-conferencing and sender-floor control via “mute” buttons. The VIC tool supports a similar notion for video-audio transmissions. A collaboration-unaware architecture has been realized with CECED [8], which uses a distributed activity-sensing control algorithm COMET to capture and control shared events nonintrusively. Examples of collaboration-transparent systems are SHASTRA [3] for engineering and telemedicine, and CSPRAY, a “spray-rendering” visualization tool used for geoscientific collaboration [56]. Toolkits like GROUPKIT [63] have also been developed to speed up rapid prototyping of group aware applications. A comprehensive list of CSCW systems [46], which are mostly group-aware, shows that despite many different architectures for supporting collaboration with generic tools or dedicated task types, only few implement a richer notion of floor control.

8 Conclusion and future work

The *raison d’etre* for floor control lies in mitigating race conditions in multiparty collaboration with multimedia resources, especially for large sessions with diverse domains of shared information, multimodality of inputs, and rich functionality of applications. This applies to synchronous collaboration, i.e., instantaneous control needs, as well as to asynchronous scenarios. Examples are agent-based task execution or disconnected work groups in wireless systems. Floor control can be used as “soft” or stringent, autonomous or chair-aiding control to augment meeting organization, to increase “collaborative throughput” and to resolve conflict in diverse group activities. It also allows for application-level congestion control by balancing the activity load according to current system conditions.

The basic framework for floor control described here in conceptual terms needs to be extended in various directions. Too little is known about group dynamics in telecooperation to present a precise account of how to implement floor control optimally for any given application. Also, it is unclear, whether a generic protocol is valid for all types of applications across organizational and cultural boundaries in intranets or the Internet. At this time, it is not clear how user behavior in direct interactions with other users and resources affects system behavior and the QOS, since users expect service fairness despite the fact that they may follow their own goals and display “greed”. Game-theoretic analysis

has been used to model network behavior and protocols in order to better understand such user-system dynamics [71]. Other engineering problems address integration of floor control protocols with scalable reliable multicast [23], and with object-oriented environments [27]. For example, CORBA [53] allows for type safety, encapsulation, reuse, portability, and extensibility, yet currently provides no concurrency control methodology for shared usage of objects.

Tool design for CSCW has to observe quantitative and qualitative constraints both from an engineering and human factors perspective. Nonverbal communication across computers cannot be conveyed and perceived with the same subtlety as in face-to-face meetings due to limited image resolution, transmission delays, speaker identification, and engagement problems. Such problems are caused by detachment, lack of awareness, and the substitution of telepresence for physical presence. Gestures account for 35% of all interactions that convey ideas, signal turn-taking, or refer to objects [34]. Similar to voice activation for speech, movement-activated floor attribution may qualify for collaborative virtual reality. New metaphors and “light” control methods are needed to transfer behavioral elements from face-to-face meetings into the electronic domain in order to foster synergy among individuals and work groups.

Interaction patterns vary with each resource and turn-taking can possibly be predicted from previous behavior. For example, a self-improving floor control model for speech could predict the next speaker with syntactic segmentation and disambiguation to capture rhetorical pauses, interrogatives, and hence the illocutionary force, which is the intent of a speaker’s statement [50]. Other learning algorithms may apply to different resource types to improve the turn-taking flow, fostering “naturalness” in collaborative work. Another salient research problem is the integration of “smart” floor control protocols with goal-based collaboration roles, such as communicators, synchronizers, archivers, etc., in workflow models [20] and agent-based collaboration [15, 64].

Dynamic sharing of online work is a new paradigm whose consequences for communication and data processing will become more apparent for the years to come. Applications will increasingly offer collaborative services, as can be seen by the current trend to make web browsers more interactive. Floor control is a promising methodology to improve cooperative behavior between users and agents to utilize distributed system components and improve the work process.

Acknowledgements. This work was supported in part by the Office of Naval Research (ONR) under contract N-00014-92-J-1807.

References

1. Agrawal D, Bruno JL, El Abbadi A, Krishnaswamy V (1994) Managing concurrent activities in collaborative environments. Technical Report TRCS94-05, University of California, Santa Barbara, Calif
2. Aguilar L, Garcia-Luna-Aceves JJ, Moran D, Craighill EJ, Brungardt R (1986) Architecture for a multimedia teleconferencing system. Proceedings of SIGCOMM, Stowe, Vt., ACM Press, New York, NY, pp 126–136
3. Anupam V, Bajaj C, Schikore D, Schikore M (1994) Distributed and collaborative visualization. *Computer* 27:37–43
4. Banerjee A, Knightly EW, Templin FL, Zhang H (1994) Experiments with the Tenet real-time protocol suite on the Sequoia 2000 wide area network. Proceedings of ACM Multimedia, San Francisco, Calif., ACM Press, New York, NY, pp 183–191
5. Barghouti NS, Kaiser GE (1991) Concurrency control in advanced database applications. *ACM Comput Surveys*, 23:269–317
6. Bly SA, Harrison SR, and Irwin S (1993) Media spaces: bringing people together in a video, audio and computing environment. *Commun ACM (Special Issue on Multimedia in the Workplace)* 36:28–47
7. Cellary W, Gelenbe E, Morzy T (1988) Concurrency control in distributed database systems. Series “Studies in computer science and artificial intelligence 3”, North-Holland, Amsterdam
8. Craighill E, Lang R, Fong M, Skinner K (1993) CECED: A system for informal multimedia collaboration. Proceedings of ACM Multimedia, Anaheim, Calif., New York, NY, ACM Press, pp 437–445
9. Crowley T, Milazzo P, Baker E, Forsdick H, Tomlinson R (1990) MM-Conf: an infrastructure for building shared multimedia applications. Proceedings of CSCW’90, Los Angeles, CA, ACM Press, pp 637–650
10. Dermier G, Gutekunst T, Plattner B, Ostrowski E, etal (1993) Constructing a distributed multimedia joint viewing and tele-operation service for heterogeneous workstation environments. Proceedings of the 4th Workshop on Future Trends of Distributed Computing Systems, IEEE, Lisbon, Portugal, Los Alamitos, CA, Sept., pp 8–15
11. Dommel HP, Garcia-Luna-Aceves JJ (1995) Floor control for activity coordination in networked multimedia applications. Proceedings of the 2nd Asian-Pacific Conference on Communications, Osaka, IEEE Comm Soc Press, pp405–409
12. Dommel HP, Garcia-Luna-Aceves JJ(1995) Design issues for floor control protocols. Proceedings of Multimedia and Networking, IS&T SPIE 2417:305–316
13. Dourish P, Bellotti V (1992) Awareness and coordination in shared work spaces. Proceedings of CSCW’92, Toronto, Ont., Can., ACM Press, New York, NY, pp 107–114
14. Dourish P, Bly S (1992) Portholes: supporting awareness in a distributed work group. Proceedings of CHI’92, ACM, Monterey, Calif., ACM Press, New York, NY, pp 541–547
15. Edmonds EA, Candy L, Jones R, Soufi B (1994) Support for collaborative design: agents and emergence. *Commun ACM* 37:41–47
16. Edwards WK (1994) Session management for collaborative applications. In: Proc. CSCW ’94, Chapel Hill, NC, USA. ACM Press, New York, NY, pp 323–330
17. Ellis CA, Gibbs SJ (1989) Concurrency control in groupware systems. Proceedings of the ACM SIGMOD International Conference on Management of Data, Portland, OR, ACM Press, New York, pp 399–407
18. Ellis CA, Gibbs SJ, Rein GL (1991) Groupware – some issues and experiences. *Commun ACM* 34:38-58
19. Ellis C (1994) A workflow architecture to support dynamic change. *SIGOIS Bulletin* 15:23
20. Ellis C, Wainer J (1994) Goal-based models of collaboration. *Collaborative Comput* 1:61–86
21. Elmagarmid AK (ed) (1992) Database transaction models for advanced applications. Morgan-Kaufmann, San Mateo, Calif
22. Fish R, Kraut RE, Root RW, Rice RE (1993) Video as a technology for informal communication. *Commun ACM* 36:48–61
23. Floyd S, Jacobson V, McCanne S, Liu CG, Zhang L (1995) A reliable multicast framework for light-weight sessions and application level framing. Proceedings of Sigcomm, Cambridge, Mass., ACM Press, New York, NY, pp 342–56
24. Furnas GW, Bederson BB (1995) Space-scale diagrams: understanding multiscale interfaces. Proceedings of CHI’95, ACM SIGCHI, Denver, Colo., www.cm.org/sigchi/Chi95/Electronic/documnts/papers/gwf_bdy.htm
25. Garcia-Luna-Aceves JJ, Craighill EJ, Lang R (1989) Floor management and control for multimedia computer conferencing. Proceedings of the 2nd IEEE Comsoc International Communications Workshop, Ottawa, Canada
26. Garcia-Molina H, Barbara D (1985) How to assign votes in a distributed system. *J ACM* 32:841–860
27. Gay V, Leydekkers P, Huis in’t Veld R (1995) Specification of multiparty audio and video interaction based on the reference model of open distributed processing. *Comput Networks ISDN Syst* 27:1247–1262
28. Greenhalgh C, Benford S (1995) MASSIVE: A collaborative virtual environment for teleconferencing. *ACM Trans Comput – Human In-*

- teraction 2:239–261
29. Gulla JA, Lindland OI (1994) Modeling cooperative work for workflow management. Proceedings of the Advanced Information Systems Engineering 6th International Conference, CAISE '94, Utrecht, The Netherlands, Springer, Berlin Heidelberg, New York, pp 53–65
 30. Greif I, Sarin S (1988) Data sharing in group work. Computer supported cooperative work: a book of readings, Morgan-Kaufman, San Mateo, CA, pp 477–508
 31. Greenberg S, Marwood D (1994) Real time groupware as a distributed system: concurrency control and its effect on the interface. Research Report 94/534/03, Department of Computer Science, University of Calgary, Alberta, Canada
 32. Handley M, Crowcroft J, Bormann C (1996) The Internet Multimedia Conferencing Architecture. Internet Engineering Task Force. Internet Draft. <ftp://ftp.isi.edu/internet-drafts/draft-ietf-mmusic-confarch-oo.ps>
 33. Handley M, Wakeman I, Crowcroft J (1995) The conference control channel protocol (CCCP): a scalable base for building conference control applications. Proceedings of Sigcomm, Cambridge, Mass., Comp. Comm. Review 25(4), ACM Press, New York, NY, pp 275–287
 34. Hayne S, Pendergast M, Greenberg S (1993/94) Implementing gesturing with cursors in group support systems. J Management Inform Syst 10:43–61
 35. Hughes PH (1993) Going off the rails: understanding conflict in practice. In: Easterbrook S (ed) CSCW: cooperation or conflict? Springer, Berlin Heidelberg New York, pp 161–169
 36. Isaacs EA, Tang JC (1994) What video can and can't do for collaboration: a case study. Multimedia Syst 2:63–73
 37. Ishii H, Miyake N (1991) Toward an open shared work space: computer and video fusion approach of TeamWorkStation. Commun ACM (Special Issue on Collaborative Computing) 12:36–50
 38. Ishii H, Kobayashi M, Grudin J (1992) Integration of interpersonal space and shared work space: clearboard design and experiments. Proceedings of CSCW'92, Toronto, Can., ACM Press, New York, NY, pp 33–42
 39. Kamel N (1993) An integrated approach to shared synchronous groupware work spaces. Proceedings of 4th Workshop on Future Trends of Distributed Computing Systems, IEEE, Los Alamitos, Calif., IEEE Comput. Soc. Press, pp 157–163
 40. Kumar V (1994) The Mbone information web homepage. URL <http://www.best.com/~prince/techinfo/mbone.html>
 41. Lantz KA (1988) An experiment in integrated multimedia conferencing. In: Greif I (ed) Computer supported cooperative work: a book of readings, Morgan-Kaufman, pp 533–552
 42. Larrue J, Trognon A (1993) Organization of turn-taking and mechanisms for turn-taking repairs in a chaired meeting. J Pragmatics 19:177–196
 43. Lauwers JC, Lantz KL (1990) Collaboration awareness in support of collaboration transparency: requirements for the next generation of shared window systems. Proceedings of CHI'90, Seattle, WA, ACM Press, New York, NY, pp 663–671
 44. Lerner GH (1989) Notes on overlap management in conversations: the case of delayed completion. Western J Speech Commun 53:167–77
 45. Levinson SC (1983) Pragmatics. Cambridge University Press, Textbooks in Linguistics, Cambridge, UK
 46. Malm PS (1994) The unofficial yellow pages of CSCW – groupware, prototypes and projects. Technical Report, University of Tromsø, Norway, URL <http://www11.informatik.tu-muenchen.de/cscw/yp/>
 47. Malone TW, Crowston K (1994) The interdisciplinary study of coordination. ACM Comput Surveys 26:87–119
 48. McCanne S, Jacobson V (1995) vic: A flexible framework for packet video. Proceedings of ACM Multimedia, San Francisco, Calif., ACM Press, New York, NY, pp 511–522
 49. McKinlay A, Procter R, Masting O, and Woodburn R and others (1994) Studies of turn-taking in computer-mediated communications. Interacting Comput 6:151–171
 50. Nagata M, Morimoto T (1994) An information-theoretic model of discourse for next utterance prediction. Trans Inform Processing Soc Japan 35:1050–1061
 51. Narayanaswamy K, Goldman N (1992) “Lazy” consistency: a basis for cooperative software development. Proceedings of CSCW'92, Toronto, Can., ACM Press, New York, NY, pp 257–264
 52. Novick DG, Walpole J (1990) Enhancing the efficiency of multiparty interaction through computer mediation. Interacting Comput 2:227–246
 53. Object Management Group (1995) OMG concepts and CORBA platforms. GMD Fokus, Germany, URL <http://www.fokus.gmd.de/minos/sig/omg.html>
 54. O'Connell B, Whittaker S, Wilbur S (1993) Conversation over video conferences: an evaluation of the spoken aspects of video-mediated communication. Human-Comput Interaction 8:389–428
 55. O'Connell DC, Kowal S, Kaltenbacher E (1990) Turn-taking: a critical analysis of the research tradition. J Psycholinguistic Res 19:345–373
 56. Pang A, Wittenbrink C, Goodman T (1995) CSpray: a collaborative scientific visualization application. Proceedings of Multimedia and Networking, San Jose, Calif., IS&T SPIE 2417:317–326
 57. Prakash A, Knister MJ (1994) A framework for undoing actions in collaborative systems. ACM Trans Comput – Human Interaction 1:295–330
 58. Rajagopalan B (1993) Consensus and control in wide-area group communication. AT&T Bell Laboratories, Holmdel, NJ 07733-3030,
 59. Rajan S, Rangan PV, Vin HM (1995) A formal basis for structured multimedia collaboration. Proceedings of the 2nd IEEE Multimedia Comput and Systems Conference, Washington, D.C., IEEE Comput. Soc. Press, Los Alamitos, CA, pp 194–201
 60. Rangan PV, Vin HM (1991) Multimedia collaboration as a universal paradigm for collaboration. Multimedia – principles, systems and applications, Springer, Berlin Heidelberg New York, pp 3–15
 61. Reinhard W, Schweitzer J, Völkens G (1994) CSCW tools: concepts and architectures. Computer 27:28–36
 62. Rennison E (1994) Galaxy of News. An approach to visualizing and understanding expansive news landscapes. In: UIST '94 Seventh Annual Symposium on User Interface Software and Technology. Marina del Rey, CA, ACM Press, New York, NY, pp 3–12
 63. Roseman M, Greenberg S (1995) Building real-time groupware with GroupKit, a groupware toolkit. ACM TOCHI Trans on Comp Human Internet 3(1), pp 66–106, March
 64. Rosenschein JS, Zlotkin G (1994) Rules of encounter – designing conventions for automated negotiation among computers. MIT Press, Cambridge, Mass
 65. Sacks H, Schlegloff EA, Jefferson G (1974) A simplest systematics for the organization of turn-taking for conversations. Language 50:696–735
 66. Sarin S, Greif I (1988) Computer based real-time conferencing systems. In: Greif I (ed) Computer supported cooperative work: a book of readings. Morgan-Kaufman, San Mateo, pp 397–420
 67. Schooler EM (1993) The impact of scaling on a multimedia connection architecture. Multimedia Syst 1:2–9
 68. Schulzrinne H, Casner S, Frederick R, Jacobson V (1995) RTP: a transport protocol for real-time applications. Internet Engineering Task Force – Internet Draft draft-ietf-avt-rtp-*.txt
 69. Sellen AJ (1995) Remote conversations: the effects of mediating talk with technology. Human-Comput Interaction 10:401–444
 70. Shen H (1994) Access control for collaborative environments. PhD Thesis, Purdue University, Lafayette, Ind
 71. Shenker SJ (1995) Making greed work: a game-theoretic analysis of switch service disciplines. IEEE Trans Networking 3:819–831
 72. Srimani PK, Das SR (1992) Distributed mutual exclusion algorithms. IEEE Computer Society Press, Los Alamitos, Calif
 73. Stefik MA, Foster G, Bobrow D, Kahn K, Lanning S, Suchmann L (1987) Beyond the chalkboard: computer support for collaboration and problem solving in meetings. Commun ACM 30:32–47
 74. Szyperki C, Ventrè G (1993) A characterization of multi-party interactive multimedia applications. Technical Report TR-93-006, International Computer Science Institute (ICSI), Berkeley
 75. Waldspurger CA, Wehl WE (1994) Lottery scheduling: flexible proportional-share resource management. Proceedings of the 1st USENIX Symposium on Operating Systems, Design and Implementation, USENIX Association, Monterey, Calif., pp 1–11
 76. Walker MB (1982) Smooth transitions in conversational turn-taking: implications for theory. J Psychol 110:31–37
 77. Wellner P (1993) Interactive with paper on the DigitalDesk. Commun

- ACM (Special Issue on Computer Augmented Environments) 36:86-97
78. Yavatkar R, Lakshman K (1994) Communication support for distributed collaborative applications. *Multimedia Syst* 2:74-88
79. Yavin D (1995) Replication's fast track. *Byte* 20:88A-90



HANS-PETER DOMMEL is a full-time PhD student at the Baskin Center for Computer Sciences and Engineering of the University of California at Santa Cruz (UCSC). In 1990, he received his German "Diplom Univ." in Computer Science and Theoretical Linguistics from the Technical University and the University of Munich. He came to the United States in 1992 on a Fulbright Scholarship, receiving his M.S. in Computer Engineering from UCSC in 1994. His current research interests are multimedia systems, CSCW, ubiquitous computing, game theory, distributed algorithms, and data compression.



J.J. GARCIA-LUNA-ACEVES was born in Mexico City, Mexico on October 20, 1955. He received his BS degree in Electrical Engineering from the Universidad Iberoamericana, Mexico City, Mexico, in 1977, and his MS and PhD degrees in Electrical Engineering from the University of Hawaii, Honolulu, HI, in 1980 and 1983, respectively. He is an Associate Professor of Computer Engineering at the University of California, Santa Cruz (UCSC). His current research interest is the analysis and design of algorithms and protocols for computer networks. Dr. Garcia-Luna is an editor of the *ACM Multimedia Systems Journal*.

He has been Chair of the ACM special interest group on multimedia, General Chair of the first ACM conference on multimedia, *ACM MULTIMEDIA '93*, Program Chair of the *IEEE MULTIMEDIA '92 Workshop*, General Chair of the *ACM SIGCOMM '88 Symposium*, and Program Chair of the *ACM SIGCOMM '87 Workshop* and the *ACM SIGCOMM '86 Symposium*. He has also been program committee member for numerous *IFIP 6.5*, *ACM*, and *IEEE* conferences on computer communication. He received the *SRI International Exceptional-Achievement Award* in 1985 for his work on multimedia communications, and again in 1989 for his work on adaptive routing algorithms.