Collaboration Management in DiCE

H. M. VIN*, M.-S. CHEN AND T. BARZILAI
IBM T. J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, USA

Effectiveness of managing collaborations directly impacts the productivity of a group, and can be significantly improved through computer supported tools made available by the advent of high speed networks and multimedia computer systems. In this paper, we propose a framework for characterizing the requirements of person-to-person (e.g., telephony), person-to-service (e.g., collaboration with software agents), and service-to-service collaborations (e.g., business workflow applications). The framework defines a hierarchy of three abstractions, at the lowest level of which are *conferences* which characterize the access rights of participants involved in a collaborative endeavor. The higher two levels are *activities*, which represent a collection of semantically related conferences, and *collaborations*, which represent temporally ordered sequences of activities. We describe the mechanisms for establishing and controlling the progress of a wide variety of multimedia collaborations. The framework for modelling collaborations and the mechanisms for their instantiation described in this paper form the basis of *DiCE*—a *Di*stributed *C*ollaborative *E*nvironment being developed at the IBM T. J. Watson Research Center.

Received October 1992

1. INTRODUCTION

Recent advances in network technology (e.g., FDDI, SMDS, PARIS, ATM, etc.) coupled with the development of high performance workstations with digital audio and video capabilities are expected to transform computer systems from mere data processing units into highly effective means for carrying out multimedia collaborations. However, enabling a rich set of computer assisted collaborations requires the integration of multimedia collaboration management with distributed computing, applications, and communications.

In recent years, several conferencing systems have been proposed and prototyped to improve the effectiveness of person-to-person collaborations. Lantz [13], and Sarin and Greif [17] have proposed conferencing architectures for text and graphics. Forsdick *et al.* at BBN [9], Ahuja *et al.* at AT & T Bell Laboratories [2] and Aguilar *et al.* at SRI [1] propose architectures for person-to-person multimedia conferencing. Casner *et al.* at ISI [6] have developed media transport protocols for teleconferencing among geographically distributed sites in wide area packet networks, but do not address issues of flexible capabilities for user participation. Angebranndt *et al.* provide a client-server architecture analogous to X-Windows for integrating audio and telephony into a graphics workstation [4]. The PX system [12] developed at the Computing Research Laboratory of BNR and the Etherphone system developed at Xerox PARC [19, 21, 22] have explored workstation-controlled telephony and capabilities for editing and distributing recorded voice messages. However, the emphasis in much of the above work is on person-to-person teleconferencing and not on general paradigms for carrying out a wide range of structured collaborations among individuals. Harrison *et al.* [10] have developed a model for coordinating concurrent development. However, their model is restricted to concurrent update of text files. Holt [11] has presented a general framework for expressing coordination requirements in an organized work area, but does not address the requirements of digital multimedia on high speed networks. Danielsen *et al.* [8], and Prinz and Speth [15] have described group communication models for computer-based communications environments. A taxonomy of conferencing environments described by Rangan and Vin in [16] is a step in the direction of understanding structured collaborations. However, mechanisms necessary for developing a comprehensive framework for specifying various types of collaborations have not been fully explored.

In this paper we propose a framework for capturing the requirements of a wide range of structured collaborations, ranging from simple meetings and conferences to classrooms and examinations, and from corporate negotiations, work flow tasks, and team design endeavours to courtroom hearings, each with its own unique set of requirements for media communication. The framework defines a hierarchy of three abstractions, at the lowest level of which are *conferences* which characterize the access rights of participants involved in a collaborative endeavour. The higher two levels are *activities*, which represent a collection of semantically related conferences, and *collaborations*, which represent temporally ordered sequences of activities. We describe mechanisms for instantiating and managing the progress of structured collaborations captured by our framework. The framework for modelling collaborations and the mechanisms

* Current address of Harrick Vin: Multimedia Laboratory, Department of Computer Science and Engineering, Mail Code C-0114, University of California at San Diego, La Jolla, CA 92093. E-mail: vin@cs.ucsd.edu. This work was carried out when the author was visiting IBM T. J. Watson Research Center during the summer of 1991.

for their instantiation described in this paper form the basis of *DiCE*—a *Di*stributed *C*ollaborative *E*nvironment—being developed at the IBM T. J. Watson Research Center[1].

The rest of the paper is organized as follows: The framework for modelling structured collaborations is described in Section 2. Mechanisms for establishing and controlling the progress of collaborations are outlined in Section 3. The DiCE prototype is described in Section 4, and finally, Section 5 summarizes the paper.

2. A FRAMEWORK FOR MODELLING COLLABORATIONS

Informally, we refer to interactions among multiple agents (representing either humans or software services) as a *conference* (denoted by C), a set of conferences that exist simultaneously as an *activity* (denoted by $\hat{C}$), and a temporally ordered sequence of activities as a *collaboration* (denoted by $\tilde{C}$). We refer to the agents interacting in a conference as *participants* (denoted by P). Participants of a conference interact through a set of *applications* (denoted by A)—involving audio, video, text, etc. If $\hat{C}_1, \hat{C}_2, ..., \hat{C}_n$ represent the activities constituting a collaboration $\tilde{C}$, and $\forall i \in [1, n]$, $\{C_{i1}, C_{i2}, ..., C_{im}\}$ represent the conferences constituting activity $\hat{C}_i$, then a complete description of collaboration $\tilde{C}$ requires the knowledge of the following:

1. The interactions that characterize each conference C_{ij} (Section 2.1),
2. The conditions for the existence of conferences $C_{i1}, C_{i2}, ..., C_{im}$ that constitute $\hat{C}_i$ (and hence, the existence of $\hat{C}_i$) (Section 2.2), and
3. The conditions for activation of each activity $\hat{C}_i$, $i \in [1, n]$ (Section 2.3).

2.1. Characterizing interactions within a conference

Participants of a conference interact through a set of applications (involving audio, video, images, text, etc.). The semantics of a conference may impose some constraints on the access rights of participants with respect to applications. Generally, the access rights depend on:

1. the type of conference,
2. the type of application, and
3. the type of participant.

For instance, a course at a university is a collaboration which contains many different types of conferences, such as lectures, examinations, laboratory sessions, brainstorm sessions, etc.; two types of participants, namely, professor and students; and many different types of applications that involve audio and video, electronic blackboard, simulations, etc. During a lecture, all the participants (namely, the professor and the students) may have the right to transmit and receive audio or video. However, only the professor may be permitted to write on the blackboard. On the contrary, in a brainstorm session, students may also be permitted to write on the blackboard. Hence, complete characterization of the interactions within a conference requires the specification of the access rights of its participants with respect to its constituent applications.

Formally, let $P = \{P_1, P_2, ..., P_n\}$ be a set of participants, and $A = \{A_1, A_2, ..., A_m\}$ be a set of applications. Let receive ($\leftarrow$) and transmit ($\rightarrow$) represent the basic modes of participation in an application. Let D_{ij}, $i \in [1, n]$, $j \in [1, m]$ represent the access rights of the participant P_i with respect to application A_j. Hence, $D_{ij} \in \{\emptyset, \{\rightarrow\}, \{\leftarrow\}, \{\rightarrow, \leftarrow\}\}$. Thus, a participant P_i can participate in application A_j iff $D_{ij} \neq \emptyset$. We denote this by $P_i \bowtie A_j$, and refer to as *participation relation.* Specifically,

$$P_i \bowtie A_j \equiv D_{ij} \neq \emptyset \quad \text{and} \quad D_{ij} \in \{\emptyset, \{\rightarrow\}, \{\leftarrow\}, \{\rightarrow, \leftarrow\}\}$$

Given these semantics for the notions of participants, applications, and the access rights, we define a conference as follows:

Definition 1. A set of participants and applications constitute a conference if and only if each participant participates in at least one application, and for each application, there are at least two participants, among whom at least one has the right to transmit and at least one has the right to receive[2]. Furthermore, a conference aggregates applications associated with a semantic context, and can also be termed as an aggregator of applications.

Formally, if

$$D_i = \bigcup_{j=1}^{m} D_{ij}$$

represent the access rights of P_i with respect to all the applications, then a set of participants P and a set of applications A can be said to constitute a conference C iff

$$\forall i \in [1, n]: D_i \neq \emptyset$$

and

$$\forall j \in [1, m], \quad \exists i, i' \in [1, n]: i \neq i', P_i \bowtie A_j, P_{i'} \bowtie A_j$$

and

$$\{\leftarrow, \rightarrow\} \subseteq D_{ij} \cup D_{i'j}$$

The access rights of participants with respect to applications constituting a conference can be graphically represented using directed arcs from the participants to the applications. Figure 1 describes a lecture involving an Audio Conversation and an Electronic Blackboard.

[1] This prototype collaboration management system will be deployed in AURORA, which is one of the five CNRI national gigabit testbeds.

[2] Conditions, such as at least two participants associated with each application, are required to guarantee meaningful interaction among a group of participants. Any change in access rights during the progress of a conference must guarantee that these conditions hold for the entire duration of the conference.

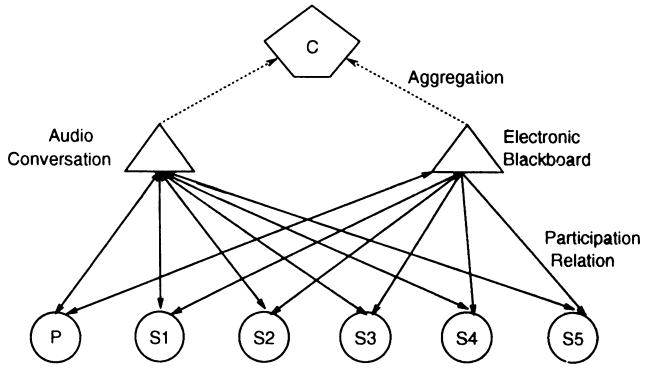

FIGURE 1. Representing the interactions within a lecture.

It illustrates that in a lecture, only the professor has the permission to write on the blackboard (as depicted by a bidirectional arc from P to Electronic Blackboard), and the students can only view its contents (as depicted by unidirectional arcs from Electronic Blackboard to S1, ..., S5). However, all the participants (namely, the professor and the students) can receive and transmit audio.

Note that the participation relation encapsulates the *permitted* access rights of both the sender and the receiver with respect to the media exchanges characterizing an application. However, during the progress of a conference, participants may specify any *desired* mode of participation within their permitted access rights. Furthermore, a conference may even permit participants to alter their permitted access rights and desired mode of participation during the progress of the conference. Depending on whether a conference permits the access rights and mode of participation once it is initiated, it can be classified as either *static* or *dynamic*. In reality, most conferences are partly dynamic and partly static, with varying degrees of dynamicity. For instance: in an *examination*, participants cannot be changed after initiation; in a *judicial court hearing*, witnesses cannot change their desired mode of participation after initiation; in a *research group meeting*, almost any of the attributes can be changed at any time.

Additionally, participants in our framework can represent either human users, software services (such as, a conference recording service), or other conferences. By permitting a participant to be a conference, the framework provides a recursive definition of a conference, leading to *nested* conferences [16]. It should be observed that, even though nested conferences can always be flattened into simple conferences containing purely individual participants, the former approach provides a mechanism for separating intra-group and inter-group communication, and serves as a better (more natural and efficient) abstraction to model inter-group and inter-organization conferences. To illustrate, consider an inter-organizational meeting to discuss policies for technical cooperation between two organizations. The nature of the collaborative activity requires that members of each organization have closed-door discussions amongst themselves, and then communicate the consolidated views of their organization to the other organization. Such a meeting is amenable to decomposition into three conferences: C_1, the meeting amongst the members of one organization; C_2, the meeting amongst the members of the other organization; and C, the meeting between the groups (see Figure 2).

2.2. Expressing existence of a conference

Each conference is associated with a duration of existence, which can be expressed either in terms of time or as an expression relating the existence attributes of various entities constituting the conference. In the simplest case, a conference terminates when there are less than two active participants (e.g., a telephone conversation). A more involved semantics may, for example, require that a conference should terminate or change when a key participant leaves the collaboration (e.g., a classroom lecture may transform into an informal meeting once its professor leaves).

When multiple applications are involved in a conference, certain applications may be considered more *critical* than others. For instance, during a lecture involving audio and video, interaction in the audio domain may be considered mandatory for receiving or transmitting video, thereby making audio transmission critical for the existence of the lecture.

The existential constraints among the participants and applications can be expressed through *existential relations*. If $P = \{P_1, P_2, ..., P_n\}$ denotes the set of participants and $A = \{A_1, A_2, ..., A_m\}$ denotes the set of applications, then $Z = P \cup A$ denotes all the entities constituting a conference. Let $\Rightarrow$ denote the existential relation. If $\mathscr{E}_{z_i}$ and $\mathscr{E}_{z_j}$ denote the boolean variables representing the existence of entities Z_i and Z_j, respectively, then the relation between Z_i and Z_j, denoted by

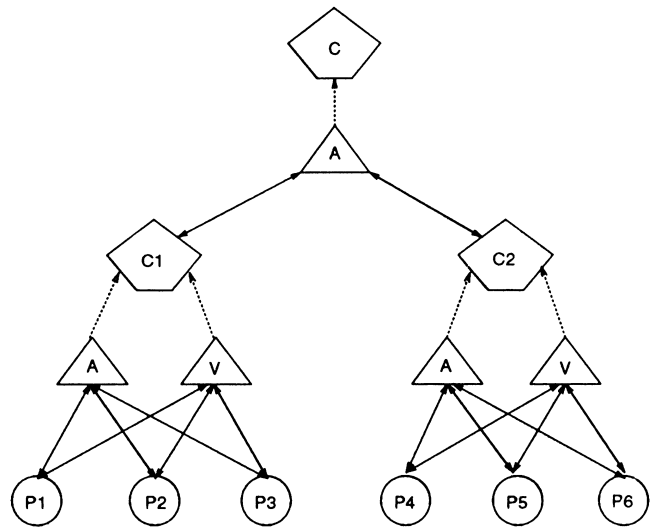

FIGURE 2. Inter-group meeting: group participation in a conference.

$\mathscr{E}_{z_i} \Rightarrow \mathscr{E}_{z_j}$, has the semantics that the existence of Z_i depends on the existence of Z_j. The interpretation of the semantics of the existential relation depends on the entities related by it. $\mathscr{E}_{A_i} \Rightarrow \mathscr{E}_{P_j}$ signifies that the existence of application A_i requires the participation of P_j. Similarly, $\mathscr{E}_C \Rightarrow \mathscr{E}_{A_i}$ has the semantics that the conference C exists only when application A_i exists. The interpretation of $\mathscr{E}_{A_i} \Rightarrow \mathscr{E}_{A_j}$ is that participation in application A_j is essential for participating in A_i. Graphically, the existential relationship $\mathscr{E}_{z_i} \Rightarrow \mathscr{E}_{z_j}$ can be represented as a dashed arrow from Z_i to Z_j.

In general, the existence of a conference may also depend on the existence of other conferences. For instance, consider a conference G_g between the members of a technical group and their manager to discuss a management report. In order to obtain the precise details of the management report, the manager may initiate a conference C_r with the meeting recording service to retrieve the proceedings of a previously conducted management meeting. Since the C_r is relevant only when C_g is in progress, we can say that the existence of C_r depends on the existence of C_g (see Figure 3). We refer to such a set of existentially related conferences as an *activity*.

Definition 2. An activity is an aggregation of existentially related conferences.

An activity terminates when all of its constituent conferences terminate. On the other hand, initiating an activity initiates all of its constituent conferences.

2.3. Collaborations

Whereas an activity provides a convenient abstraction for capturing collective interaction in a common time span, a complete collaborative endeavor may, in general, involve several periods of collective activity, possibly with dependencies between the outcomes of some periods and causations of others. All such activities, whose initiations are temporally related, constitute a *collaboration*. In general, the ordering of various activities constituting a collaboration can be expressed using any of the 13 possible temporal relationships (namely, *before*, *meets*, *during*, *overlaps*, *starts*, *ends*, and *equal* plus the inverse relations—except *equal*) [3, 14]. An activity is initiated as soon as the temporal constraints are satisfied. Using the notion of temporal relations, a collaboration can be formally defined as follows:

Definition 3. A collaboration is a temporally ordered sequence of activities.

Figure 4 illustrates the use of temporal relations in the specification of a collaboration (namely, a course at a university) consisting of three types of activities (namely, lectures, laboratory sessions, and examination). The temporal ordering shown in Figure 4 represents a course that consists of a sequence of lectures and laboratory sessions (starting with a lecture, as depicted by a bold pentagon in Figure 4), followed by an examination.

Recursive temporal relations among the activities constituting a collaboration makes the collaboration persistent (as opposed to transient collaborations such as telephone conversations). Mutual recursion yields periods of active interactions separated by periods of inactivity, and can be used to realize periodic meetings (such as, weekly scheduled meetings, lectures for a course, etc.). For example, a temporal relation $\hat{C} \overset{\tau}{\rightsquigarrow} \hat{C}$ indicates that activity $\hat{C}$ is to be re-activated τ units of time after the termination of its previous activation. For such collaborations, the needed system resources (such as network bandwidth) can be reserved in advance, and their periodic invocations can be made automatic.

2.4. Discussion

A conference within our framework represents synchronous interactions among a set of participants. An activity is defined as a set of existentially related conferences, and can model all types of synchronous interactions among groups. A collaboration is defined as a set of temporally related activities, and hence is capable of modelling asynchronous interactions. Thus, the abstractions of conference, activity, and collaboration together can capture all types of interactions in person-to-person, person-to-service, and service-to-service environments.

The three types of relations, namely, the participation, existential, and temporal, are orthogonal, and hence can be specified independently of each other. Specifically, the process of specifying collaborations can be divided into three stages:

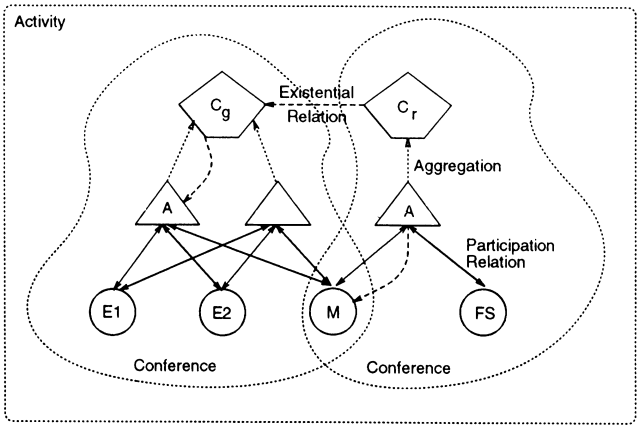

FIGURE 3. An activity as an aggregation of two conferences.

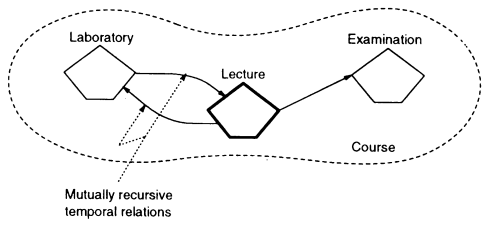

FIGURE 4. A course as a temporal ordering of lectures, laboratory sessions, and examination.

1. Defining a collaboration as a composition of temporally related set of activities;
2. Defining each of the activities as a set of existentially related set of conferences; and
3. Defining each of the conferences as an aggregation of participants and applications tied together by participation and existential relations. Figure 5 represents such a *hierarchical specification* of a course.

Such specifications of collaboration semantics, represented either graphically (as shown in this Section) or using a formal, object-oriented specification language [20], can be interpreted to synthesize low-level, network environment specific constraints, which can be retrieved while instantiating collaborations. Since conferences are the building blocks of activities and collaborations, instantiating a wide range of collaborations requires mechanisms for establishing and controlling the progress of their constituent conferences, and are elaborated in the next Section.

3. CONFERENCE MANAGEMENT

The framework for modelling collaboration presented in the previous Section combines efficiency and power via a hierarchy of three abstractions. Since, within our framework, conferences are the building blocks for modelling collaborations, the usefulness of the framework critically depends on the flexibility of the underlying conference management, which is guided by the following two principles:

- *Autonomous participation:* Every participant should be permitted to independently determine his/her mode of participation in conferences. Furthermore, even after the initiation of the conference, its participants may be permitted to alter their mode of participation independently, at their own will.
- *Negotiated conference management:* Establishing a conference is a negotiation process involving participants and their environments (such as, hardware and network environments). Negotiations may also be carried out during the progress of a conference to permit dynamic changes in the list of participants and their access rights.

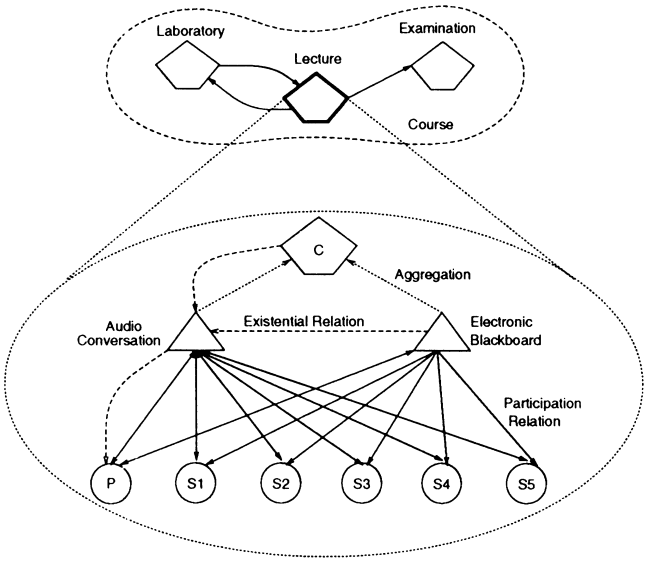

FIGURE 5. Hierarchical modelling of collaborations.

In this Section, we discuss the implications of the above two principles on the techniques for establishing and managing the progress of conferences.

3.1. Establishing a conference

The process of establishing a conference consists of three phases:

1. Invitation and arbitration,
2. Negotiation, and
3. Setup.

This three phase procedure not only provides a flexible platform for creating conferences, but also reduces the overhead of channel setup for real-time media communication (as we shall outline later).

3.1.1. Invitation and arbitration

The goal of this phase is to determine the set of invitees that are willing to participate in the conference. The main tasks of this phase includes: sending invitations to all invitees, arbitration (to elect one initiator among multiple agents who try to initiate a conference simultaneously), collecting initial responses confirming willingness to participate, and the establishment of a multicast connectivity[3] for the exchange of control information.

If conference management is based on a client-server model, requests for initiating a conference are sent by initiators to a conference server (a software entity responsible for managing establishment of conferences). Consequently, the server can uniquely identify the conference initiator (by ignoring contending requests for initiating the same conference), thereby simplifying the invitation and arbitration phase of conference establishment. However, the client-server model is fraught with inherent problems of reliability and scalability.

In a peer-to-peer communication environment, on the other hand, there are two approaches in dealing with multiple initiators:

1. setup-and-merge, and
2. arbitrate-and-setup.

In the setup-and-merge approach, participants ignore contending initiators and join different initiators to form a set of non-overlapping subgroups. These subgroups then try to merge themselves back into one complete group. However, this approach is quite often undesirable

[3] Multicast communication, in networks not providing support for multicast connectivity, can be emulated by a combination of several point-to-point and broadcast communication channels.

because it may yield an unacceptably long time to initiate a conference, may incur high network overhead, and may be overly complicated for non-specialist users.

On the contrary, the arbitrate-and-setup approach detects and resolves contentions from multiple initiators, thereby ensuring a unique initiator with each conference. Since media transmission channels are established only after all the participants of the conference are determined, the arbitrate-and-setup approach yields much lower network overhead. However, such an approach necessitates the development of techniques for uniquely identifying the conference initiator [7]. Assuming that:

1. the network transmission latency for a packet is bounded by D, and:
2. messages are delivered reliably, but not necessary in order,

we propose a timeout-based mechanism for detecting and resolving contentions.

During the arbitration phase, an agent can be either an initiator or an invitee. After sending invitations to all the invitees, an initiator collects replies for a period of $2D$. If the initiator receives no invitation during this interval, then there is no competing initiator. Otherwise, each initiator independently decides whether or not to give up the initiator role based on simple deterministic rules (such as the length of their user identifiers). Similarly, each invitee also waits for a period of $2D$ for receiving possible invitations from more initiators. If the later invitation is sent by a prioritized initiator, it reacts by returning a positive reply and changing its sign-on to the new initiator. Otherwise a late invitation is ignored. After the $2D$ period, the arbitration is considered complete. The winning initiator, however, waits for a time out of D before continuing to the next phase of actually setting up network connectivity.

3.1.2. Negotiation

This phase is designed to permit participants (and their hardware and network environments) of a conference to reach consensus in determining the operational parameters of the conference. The functionality of the negotiation phase can be subdivided as follows:

- *Managing hardware and network heterogeneity:* Negotiations among the participants of a conference provide mechanisms for managing heterogeneous hardware and network environments. For instance, consider an audio conference between two participants, whose audio digitizers encode audio stream differently (such as, mu-law and ADPCM). If a service for converting one audio encoding to another is available, then detection of heterogeneity during the negotiation phase can trigger an invitation to the service to join the conference. This would then enable the participants of the conference to communicate. Negotiations can also be used to resolve feature interaction problems (i.e., when a feature provided by the conferencing system interferes with the operation of another feature, e.g. call-forwarding-busy-line and call-waiting) [15].
- *Flexibility of participation in conferences:* The negotiation phase permits participants of a conference to reach a consensus on their mode of participation with respect to the conference. For instance, if the number of video streams that can be received at any workstation is limited due to processing requirements, then each participant may determine the set of video streams that he/she would like to receive by the process of negotiation. Negotiations may also be carried out between the participants of a conference even during its progress. For instance, participants of a conference may negotiate to admit a new user, or to alter the access rights of participants already active in the conference.

The negotiation phase yields a set of operational parameters, that define the set of multicast pipes required to be established among the participants of a conference, which is the input to the setup phase.

3.1.3. Setup

The main task of the setup phase is to establish long-term connectivity for media communication among the participants of each of the applications. There are two different approaches in establishing multicast pipes for media communication:

1. create exactly one pipe with the capacity (bandwidth) equal to the combined requirement of all the applications, or
2. establish a set of pipes, one for each application.

Although the former approach is simpler, and incurs a smaller network overhead for setting up the pipe, the latter approach is more desirable for the following reasons:

- Media information may have to be routed to different devices (and hence, to different processes) within a workstation, which may result in a significant overhead for demultiplexing the data stream received from a single multicast pipe.
- Different media have different transmission characteristics (such as the tolerance to packet loss or error in transmission). Hence, the process of establishing multicast pipes can exploit the transmission characteristics, so as to efficiently utilize the bandwidth.
- The participants in different applications may be different. Hence, if a single pipe, with combined capacity is created, it leads to a lot of wasted bandwidth.

Once multicast pipes for media transmissions are established, the conference is said to be established. It is important to reiterate that the negotiation and setup phases may take place at any time during the progress of a conference so as to reflect the changes in participants'

preferences or status. Successful completion of each round of negotiation and setup procedures during a conference results in a reconfiguration of the conference.

3.2. Controlling the progress of a conference

3.2.1. Coordination

Interactions in a conference may need to be coordinated due to the intrinsic nature of applications or resource constraints. For instance, the nature of applications such as shared electronic blackboards or concurrent editors may necessitate controlled access. Since no single set of rules can satisfy the coordination requirements of several different applications, we propose a simple generic baton sharing protocol, which can be tailored by the conference management to enforce the coordination requirements of different applications [7].

In this protocol, batons represent access rights with respect to applications. The number of batons is a parameter that is set to represent the maximum number of participants that can simultaneously exercise the access right. A participant needs one baton to exercise the access. It is possible that a participant holds more than one baton. In such a case, only one baton is *active* or in use and the others are considered *idle*. When a participant intends to exercise the access right but does not have any baton, it sends a request for a baton to all the participants of the conference. On receiving a request for baton, a participant either gives away all his idle batons or returns a negative reply. The negative reply indicates one of the two possible situations: the participant has one active baton (which he does not wish to release), or has no baton at all.

This protocol is sufficient for normal situations. However, additional considerations are necessary for exceptional situations, such as loss of batons due to unintended separation of participants. In steady state, the sum of the number of idle batons and the number of negative replies indicating active baton condition should be the same as the maximum number of participants that have the access right. This property can be used by a participant, who is waiting for a baton, to detect abnormal conditions. Upon detection, participants exchange status update messages describing the number of active and idle batons in their possession. If a shortage of batons is detected, then the participant with the smallest identifier generates batons to make up. On the contrary, if there is an excess of batons, participants, again according to their identifiers, first give up idle batons and, if necessary, then give up active batons.

3.2.2. Dynamicity in mode of participation

During the progress of a conference, participants may change their desired mode of participation or even terminate their participation at any time, thereby initiating changes in transmission or reception of media information. In response to such changes, conference management must verify the requested change against the permitted access rights of the participant. If verified, media transmission pipes are altered to reflect the new configuration. Furthermore, a *conference controller* may also alter the permitted access rights of participants, invite new participants, or change the conference controller. The extent of dynamicity of a conference governs the types of changes in conference attributes (such as participants, applications, access rights, mode of participation, etc.) permitted during its progress.

4. THE DiCE PROTOTYPE

A *Di*stributed *C*ollaborative *E*nvironment (DiCE), based on the framework for modelling collaborations presented in this paper, is currently being prototyped by the High Bandwidth Applications Group at the IBM T. J. Watson Research Center. In order to demonstrate its power, DiCE has been instantiated in the Multimedia Multiparty Teleconference (MMT) system [7]. MMT is a workstation based, hub-free (peer-to-peer) environment for managing multimedia conferences. The design of MMT exploits the functionality of the emerging high speed networks, and achieves functional integration of multimedia computing with high speed networking. MMT permits participants of a multimedia conference to exchange high quality motion video and audio, and supports a shared workspace platform over which existing applications can be used in a collaborative mode without any modifications. In this Section, we describe the hardware environment and the software architecture of our prototype.

4.1. Hardware environment

The multimedia workstation of our first prototype is based on IBM PS/2 running AIX. Each PS/2 is equipped with special purpose hardware for full duplex compression and packetization of full motion video, and for the composition of multiple simultaneous video and audio streams. Specifically, each PS/2 in our prototype environment is equipped with three adapters: multimedia front-end, multimedia processing, and communication back-end. The multimedia front-end is the IBM M-Motion Video Adapter/A. This adapter has the functions of:

1. capturing and digitizing analog video and audio,
2. delivery to another adapter for processing or networking, or direct playback locally, and
3. chroma keying for multiplexing video and graphics on the same workstation display.

The multimedia processing adapter, which is the heart of our prototype, can compress and decompress digital video at 30 frames/second, compose multiple compressed video streams from different remote sources using an innovative technique [18] and display them simultaneously at 30 frames/second each, and compose multiple audio streams from different remote sources and provide

the unconstrained audio effect of a face-to-face meeting. Consequently, each participant in a multimedia collaboration can independently decide the video and audio streams that he or she wants to receive. Furthermore, the distributed mixing of audio and video yielded by the multimedia processing adapter is more reliable than the typical hub-based composition of media streams.

The communication backends are either a research gigabit LAN prototype (e.g., MetaRing or ORBIT), or a FDDI adapter running at 100 Mbps. These adapters can exchange information with multimedia processing adapters directly without involving system CPU and memory. This is very important in achieving an efficient network and workstation interface, which becomes critical in dealing with multimedia computing. Figure 6 represents the inter-connectivity of the three adapters.

The hardware prototype, with the Multimedia Multiparty Teleconferencing system, has been demonstrated on the PARIS platform, and is expected to be deployed for various field trials, including the AURORA national gigabit testbed and various joint studies between IBM and its industrial partners.

4.2. Software architecture

In order to support a wide spectrum of multimedia collaborations that can be captured by the framework presented in Section 2, the software architecture of DiCE is designed to consist of three components: *collaboration management*, *media transmission control*, and *interfaces*.

4.2.1. Collaboration management

The functionality of collaboration management can be partitioned into three components: *Multimedia Server* (MMS), *Collaboration Management Unit* (CMU), and *Conference Control Unit* (CCU).

- *Multimedia server (MMS):* There is exactly one active instance of MMS in a workstation. The primary function of the MMS is to provide naming and addressing facilities for agents, so as to provide features such as invitation by name (rather than address), automatic invitation forwarding, and visiting [19]. When a user registers with an MMS on a workstation, the MMS creates a Collaboration Management Unit (CMU) responsible for managing all the collaborations in which the user may participate. Since multiple users may be simultaneously registered from the same workstation, several instances of CMUs may coexist within a workstation.
- *Collaboration management unit (CMU):* A CMU represents a user, and is the heart of the software architecture of DiCE. It encapsulates the behavioural semantics of collaborations (such as, the temporal relations among activities, and the conferences constituting each activity) in which a user may participate. In fact, it performs the functions of a 'schedule keeper', and initiates activities (and hence, their constituent conferences) in accordance with the temporal specifications of collaborations. For each conference initiated, the CMU creates a Conference Control Unit (CCU) responsible for controlling the progress of the conference. Furthermore, the CMU maintains existential relations among conferences constituting activities.
- *Conference control unit (CCU):* CCUs represent conferences and encapsulate their operational attributes and constraints (such as, access rights with respect to applications, critical applications and participants, etc.). The primary function of the CCU is to control the establishment and progress of a conference. This includes sending preliminary invitation to a list of invitees and accumulating acceptances, carrying out negotiations to decide the mode of participation for each of the participants, setting up the network connectivity for control as well as media communication, and modifying network connectivity based on changes in mode of participation of participants. Note that the functionality of CCU is application-independent, providing a flexible, extensible architecture. Each participant is represented in a conference by a CCU. Since a user may be participating in multiple conferences simultaneously, multiple active instances of CCU may coexist in a workstation.

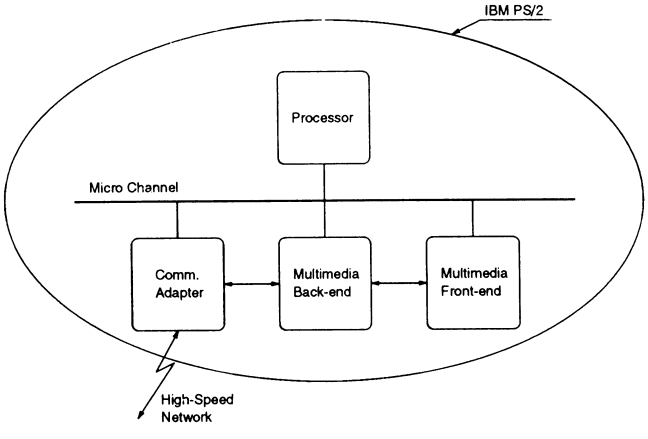

FIGURE 6. Hardware environment.

To illustrate the mapping of collaboration semantics to operational modules (namely, CMU and CCUs), consider the specification of a course shown in Figure 5. The semantics of a course can be captured using a CMU and three CCUs (one each for Laboratory, Lecture, and Examination, denoted by CCU_{Lab}, CCU_{Lect}, and CCU_{Exam}, respectively). The CMU will activate each activity in accordance with the temporal relations. Activating a Lecture, for instance, will result in the creation of the CCU_{Lect} for enforcing the operational attributes and constraints characterizing a lecture (see Figure 5). When the Lecture terminates (as defined by its existential relations), CCU_{Lect} also terminates, and the CMU is notified. The CMU, in turn, may activate either CCU_{Lab}

or CCU_{Exam}. The CMU maintains the semantics specification of the Course until its termination.

4.2.2. Media transmission control

Media Transmission Units (MTU) are used for managing applications constituting a conference. An MTU is created by CCU to support an application that becomes part of the conference. The functionality of an MTU is application and network dependent. For instance, the MTU corresponding to a shared workspace application, such as the electronic blackboard, should mediate information exchange as well as coordinate accesses from the participants, i.e., floor control; whereas the MTU corresponding to a motion video application initiates the video transmission protocol.

4.2.3. User interface

The user interface components in the software structure include:

- *Collaboration Interface (CI):* The Collaboration Interface provides the linkage between the user and the underlying collaboration management system. It provides facilities to initiate different types of collaborations (such as, a course, a panel session at a convention, etc.). In order to initiate a collaboration, the initiator first selects the type of a collaboration, which triggers the invocation of a window-based interface for specifying collaboration-specific parameters. For instance, if a user decides to initiate a course, then the interface prompts the user to specify the names of the professor, assistants, and students as well as the duration of course.

  Once a collaboration is initiated, CI provides mechanisms for controlling its progress. Specifically, CI provides two types of modules: a conference control module, which provides a window-based interface for placing, joining, and terminating conferences; and an application linkage, which provides mechanisms for users to add and delete applications to the conference.

  To create a conference, the initiator first specifies the set of invitees (if different from the one described by the specification of the parent collaboration), which results in invitations being sent to all of the invitees. On receiving a request to participate in a conference, CI can perform automatic filtering of invitations (based on the criteria specified by the user) using the conference-specific parameters (such as the reason for creating a conference, urgency, etc.) specified in the invitation, and then prompts the user about the invitation.

  Once the invitee accepts the invitation, the application linkage permits the user to specify the applications with respect to which he/she would like to participate in the conference. The preferences can also be altered dynamically (i.e., during the progress of the conference). Since the applications constituting a conference are logically independent entities, CI initiates a different interface for each application.
- *Video Interface (VI):* The functionality of the Video Interace can be subdivided into two parts:

  1. Video Window, and
  2. Video Control Panel.

  A Video Window is used for displaying motion video at 30 frames/sec, and multiple such Video Windows can be active simultaneously. Each Video Window maintains information regarding all the participants of the conference, and provides a menu-based interface to change the source of the video stream being displayed in it.

  The Video Control Panel provides an interface to create, destroy, move, and resize Video Windows; as well as mechanisms for controlling display-specific features such as brightness, contrast, colour, etc. Furthermore, the Video Control Panel is responsible for communicating information describing the configuration of Video Windows (such as, height, width, overlap, etc.) to the video processing adapters, which can also be used by the adapters to perform real-time digital mixing of multiple video streams.
- *Audio Interface (AI):* Unconstrained audio transmission is the default in the DiCE multimedia conferencing environment. Hence, each participant receives audio streams from all other participants, which are then digitally mixed before playback. The Audio Interface provides a window-based interface to selectively enable the mixing of various audio streams. In addition, it provides an interface for controlling certain audio-specific parameters such as volume of the speakers. Such parameters are also communicated to the multimedia adapter to facilitate the process of audio mixing.
- *Shared Workspace Interface (SWI):* The functionality of the Shared Workspace Interface can be subdivided into two parts:

  1. Shared Workspace Control Panel, and
  2. Application Window.

  The Shared Workspace Control Panel provides the interface to invoke various applications within a conference. Once initiated, an Application Window provides application-specific interface to the user.

Figure 7 represents the inter-relationship between the components of the software architecture of DiCE prototype.

5. CONCLUDING REMARKS

We have presented a framework for modelling collaborations in person-to-person, person-to-service, and service-to-service domains. We have provided formal semantics for collaborations, and have derived a set of properties (namely, the participation, existential, and temporal relations) essential for characterizing the beha-

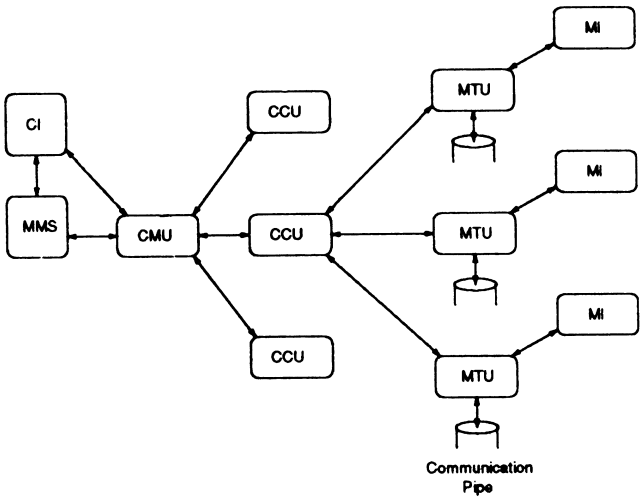

FIGURE 7. Software architecture of the DiCE prototype.

vioural semantics of collaborations. We have described mechanisms for instantiating different types of collaborations, and the design of a software architecture that controls the progress of collaborations. The framework for modelling collaborations and the software architecture described in this paper form the basis of *DiCE*—a desk-top *D*istributed *C*ollaborative *E*nvironment being prototyped at the IBM T. J. Watson Research Center, and will be installed on the PARIS fast packet switched networks in the AURORA testbed.

REFERENCES

[1] L. Aguilar, J. J. Garcia-Luna-Aceves, D. Moran, E. J. Craighill and R. Brungardt, Architecture for a multimedia tele-conferencing system, *Proceedings of the SIGCOMM '86 Symposium on Communications Architectures and Protocols, Stowe, VT*, pp. 126–136 (1986).

[2] S. R. Ahuja, J. Ensor and D. Horn, The rapport multimedia conference system, In *Proceedings of COIS' 88 Conference on Office Information Systems, Palo Alto, CA*, pp. 1–8 (1988).

[3] J. F. Allen, Maintaining knowledge about temporal intervals, *Communications of the ACM*, **26**(11), pp. 832–843 (1983).

[4] S. Angebranndt, R. L. Hyde, D. H. Luong, N. Siravara and C. Schmandt, Integrating audio and telephony in a distributed workstation environment, In *Proceedings of Summer 1991 USENIX Conference, Nashville, TN*, pp. 419–436 (1991).

[5] T. F. Bowen, F. S. Dworak, C. H. Chow, N. D. Griffeth, G. E. Herman and Y. J. Lin, The feature interaction problem in telecommunication systems, *Bellcore Technical Report* (1989).

[6] S. Casner, K. Seo, W. Edmond and C. Topolcic, N-way conferencing with packet video, *Proceedings of the Third International Workshop on Packet Video, Morristown, NJ* (1990).

[7] M.-S. Chen, Z.-Y. Shae, D. Kandlur, T. Barzilai and H. M. Vin, A multimedia desktop collaboration system, *To appear in the proceedings of IEEE GLOBECOM '92*, Orlando, Florida (1992).

[8] T. Danielsen, U. Pankoke-Babatz, W. Prinz, A. Patel, P. Pays, K. Smalland and R. Speth, The amigo project: Advanced group communication model for computer-based communications environment, In *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW '86)*, Austin, Texas, ACM Press, pp. 115–142 (1986).

[9] H. C. Forsdick, Explorations in real-time multi-media conferencing, *Proceedings of the 2nd International Symposium on Computer Message Systems, IFIP*, pp. 331–347 (1985).

[10] W. Harrison, H. Ossher and P. Sweeney, Coordinating concurrent development, In *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW '90)*, Los Angeles, California, ACM Press (1990).

[11] A. W. Holt, Diplans: A new language for the study and implementation of coordination, *ACM Transactions on Office Information Systems*, **6**(2), pp. 109–125 (1988).

[12] R. Kamel, K. Emami and R. Echert, PX: Supporting voice in workstations, *IEEE Computer*, **23**(8), pp. 73–80 (1990).

[13] K. A. Lantz, An experiment in integrated multimedia conferencing, In *Proceedings of CSCW '86*, pp. 267–275 (1986).

[14] T. D. C. Little and A. Ghafoor, Synchronization and storage models for multimedia objects, *IEEE Journal on Selected Areas in Communications*, **8**(3), pp. 413–427 (1990).

[15] W. Prinz and R. Speth, Group communication and related aspects in office automation, In *Proceedings of the IFIP TC 6/WG 6.5 Conference on Message Handling Systems*, Munich (1987).

[16] P. Venkat Rangan and H. M. Vin, Multimedia conferencing as a universal paradigm for collaboration, In *Multimedia Systems, Applications, and Interaction*, Chapter 14, Lars Kjelldahl (editor), Springer-Verlag, Germany (1991).

[17] S. Sarin and I. Greif, Computer-based real-time conferences, *IEEE Computer*, **18**(10), pp. 33–45 (1985).

[18] Z.-Y. Shae and M.-S. Chen, Mixing and playback of JPEG compressed video, *IBM Research Report, RC 16068*, Also to be presented at GLOBECOM '92 (1990).

[19] D. C. Swinehart, Telephone management in the etherphone system, In *Proceedings of the IEEE/IEICE GLOBECOM '89, Tokyo*, pp. 1176–1180 (1989).

[20] H. M. Vin, M.-S. Chen and T. Barzilai, A framework for modeling collaborations, *In Proceedings of the 1992 IFIP International Conference on Upper Layer Protocols, Architectures and Applications (ULPAA '92)*, Vancouver, Canada, Ed. G. Neufeld and B. Plattner, Elsevier Science Publishers (1992).

[21] H. M. Vin, P. T. Zellweger, D. C. Swinehart and P. Venkat Rangan, Multimedia conferencing in the etherphone environment, *IEEE Computer—Special Issue on Multimedia Information Systems*, **24**(11), pp. 69–79 (1991).

[22] P. T. Zellweger, D. B. Terry and D. C. Swinehart, An overview of the etherphone system and its applications, *Proceedings of the 2nd IEEE Conference on Computer Workstations*, pp. 160–168 (1988).