Managing Telemeetings

Constantin Arapis
Dimitris Thanos

Abstract

Group meetings are one of the most frequent activities in an office environment. Tele-meetings - meetings where participants are geographically dispersed and use a computer and networking infrastructure for communicating - are becoming increasingly pervasive. Telemeeting participants may use various media types for communicating including audio, video, images, graphics, animation and text. We present a telemeeting environment for modelling and managing the complete lifetime of telemeetings. The environment implements common teleconferencing services as well as services for storing and querying meeting sessions in a telemeeting database.

1 Introduction

Group meetings are one of the most frequent activities in an office environment. Meetings may be improvised or either regularly or occasionally scheduled in advance. Examples of meetings include conferences, university courses, research group meetings, medical consultation meetings and board of directors meetings.

One may classify meetings into different meeting types according to the various activities occurring during the meeting session. For example, during a board of directors meeting the following activities would occur in sequence: some participants may present proposals on a given subject, a debate on these proposals then follows and finally a decision will be taken on which proposal to accept by voting. Voting will be absent from a medical consultation meeting: a doctor responsible for a particular patient may schedule a meeting with a number of his colleagues in order to better evaluate the patient’s case. The doctor informs her/his colleagues of the patient’s symptoms and the results of medical tests. An examination of the patient then takes place. After the examination a debate follows where each doctor contributes with her/his expertise for choosing the most appropriate treatment. In both, the board of directors meeting and the medical consultation meeting, a decision is taken but in a different way. In other types of meetings there is no decision to be taken. In a course a tutor presents her/his course material and students raise their questions either spontaneously or at precise points in time.

A telemeeting is a meeting where participants are geographically dispersed and use a computer and networking infrastructure for communicating. Telemeetings are becoming increasingly pervasive due to recent technological advances in computers and networking [3]. Currently most desktop computers are adequately equipped for being used as telemeeting stations. They have sufficient processing power for real-time compression and decompression of audio and video signals and they are equipped with multimedia devices such as cameras, microphones and speakers. Telecommunication companies are improving and extending their
networking services for satisfying the needs of multimedia applications. For example, ISDN networks are relatively cheap and readily available for transmitting good quality audio and for most types of teleconferencing applications acceptable quality video images.

In a telemeeting session several services may be necessary. For example, in addition to an audio communication channel which is fundamental, several other forms of communication channels may be useful and even necessary including video, email and text chat. Each of the above communication mechanisms satisfies different communication needs. Another service often required in a telemeeting session is the ability to broadcast a slide-like presentation to all participants. Finally, groupware functionality, such as shared editor and shared whiteboard, is yet another type of useful service. We classify the services that we have mentioned in this paragraph as session services.

In addition to session services, it is often essential to take into account the various activities occurring during the whole life period of a meeting. More precisely, an agenda describes the temporal structure of the meeting session and the topics to be discussed for meetings which are scheduled in advance. Some time after the meeting, the minutes of the meeting are given out to participants. A software environment for telemeetings should include services allowing users to specify an agenda for telemeetings, verify/validate the constraints specified in the agenda, record the telemeeting session in a telemeetings database (TDB) and allow users to query the TDB. We classify the services that we have mentioned in this paragraph as meeting services.

Most meeting software environments limit meeting services to the storage of meeting sessions into files. The environment gives the user the ability to play back the session recording and step forward and backward much like a video tape on video tape player. Note however that browsing in a meeting session recording is more complex that browsing a video tape. The complexity depends on the session services available during the meeting session. We believe that it should at least be possible to browse on recordings of session services like shared editors, shared whiteboards and presentations.

Since spring 1997 we have undertaken the development of a telemeeting environment named TeME (TeleMeeting Environment). The starting point for this development effort was our participation in a tele-education project [1] in which we aimed to provide two services. The first service is the broadcast of a lecture in real-time over the Internet\(^1\). The service is interactive in the sense that a feedback channel allows students to communicate with lecturers. The second service is the development and maintenance of a database containing both the course material presented in a lecture and the lecturer’s speech [8]. Since a telelecture is a special case of a telemeeting session we targeted the development of a telemeeting environment flexible enough for use in many types of meetings including telelectures. To test the suitability of TeME for various types of telemeetings, we started using it in our weekly research group meetings.

\(^1\) For each telelecture there are two lecturers. One is located in a classroom in CUI (Centre Universitaire Informatique) in Geneva. The other lecturer is located at our project partner’s location in GMD (German National Research Centre for Information Technology) in Bonn. Students are either present in the CUI’s classroom or follow the lecture over the Internet.
The originality of TeME lies on the implementation of meeting services. Currently, TeME stores meeting sessions partially in files and partially in a relational DBMS. This enables users to readily enrich a meeting session recording with semantics relevant to a particular meeting type. It also allows users to query the TDB on collections of meeting sessions and not only on the contents of one meeting session. For example it is possible to issue queries like: retrieve (and play) those audio parts of research group meetings held during the years 1996 and 1997 in which issues on multimedia databases were discussed.

In this paper we present TeME’s meeting services and their implementation. More precisely this paper is structured in the following way. In the second section we present a typical telemeeting scenario using TeME. In the third section we briefly describe the session services implemented in TeME. In the fourth section we describe TeME’s meeting services. In the fifth section we comment on TeME’s implementation effort. In the last section we present our concluding remarks.

2 Telemeeting Scenario

We identify three user profiles for telemeetings: the participant, the presenter, and the meeting administrator (MA). Participants are users who participate in the telemeeting. Presenters are participants who are granted the privilege to broadcast a presentation to all other participants. Participants can only follow the presentation and possibly address questions and comments to presenters. Note that during the same meeting session several presentations may be scheduled and performed by different presenters. The MA is responsible for conducting a telemeeting in its entirety and tuning TeME to service participants and presenters.

We identify three consecutive periods that cover a telemeeting: the pre-meeting period, the meeting session period and the post-meeting period. Figure 1, Figure 2 and Figure 3 illustrate TeME’s module configuration and use during the three periods of a telemeeting. Rectangles are used for illustrating hosts, ovals for representing software processes, and unidirectional or bi-directional arrows for representing data transfers.

During the pre-meeting period the MA uses TeME for defining the meeting’s agenda, probably taking into account requests from other participants. The telemeeting’s agenda is stored in the TDB.

During the meeting session TeME’s reflector module with the synergy of client session modules supply users a set of session services. The telemeeting’s agenda stored in the TDB during the pre-meeting period is used by the reflector for verifying the temporal structure of the telemeeting session and the various constraints specified by the MA. TeME’s reflector module records the meeting activity in the TDB. Finally, during the meeting session, the MA may enrich the contents of the TDB with various pieces of information. For example, in a board of directors telemeeting, the MA may insert the decision of acceptance or refusal of a particular proposal in the TDB.
The post-meeting period corresponds to the period during which participants query the contents of the TDB. Note that the MA may continuously upgrade the TDB with data that were not available during the telemeeting session and with data that she/he considers to be relevant to the meeting session.

**Figure 1** Pre-meeting period

**Figure 2** Meeting session period
3 Session Services

TeME’s session services are used exclusively during the meeting session. They are implemented according to a client/reflect architecture. The client part of TeME provides an interface to the user. Each client is connected to the reflector. The reflector acts as an intermediary between all clients. The main role of the reflector is to repeatedly accept a message from a client. The reflector then either forwards the same message to all clients or does some processing before broadcasting a different message to all clients. For example, a user may broadcast an email to all other users. The message sent to the reflector contains the email and is forwarded as such from the reflector to all participants. However, when an audio packet is sent to the reflector, the reflector first “mixes” the packet with other audio packets which have been concurrently received and then broadcasts the mixed packet to all clients.

The message received from the reflector might not contain all necessary information for the receiving client for further processing. In that case the message sent to the client is a sort of reference for where to find the target data. For example, using the telepresentation service for broadcasting a particular Web page to all clients, a message containing a URL is first sent to the reflector. The reflector then broadcasts the message to all clients. Finally, each client loads and then displays the data corresponding to the URL from the server indicated in the URL. Figure 4 illustrates the implementation of TeME’s session services.
The session services implemented in TeME are the following:

**Telepresentation Service:** TeME implements a Web navigation service: each participant has the possibility of visiting any Web site during a session. TeME uses SUN’s HotJava component [9] to load and display HTML pages. The Web navigation capability of TeME is further exploited for providing a telepresentation service. One of the participants can become a presenter. The HTML pages the presenter is displaying on her/his screen are displayed on the screens of all the participants. More precisely, whenever the presenter loads a new page the HotJava component provides TeME with the URL of the page. This URL is then sent to the reflector within a TCP packet. The packet is then forwarded to all clients. Each client requests the HotJava component to load and display the page corresponding to the URL. Participants playing the role of presenter can change dynamically and are identified with a presenter password set by the MA. To join a session there is no need to perform any previous downloading through the net. In addition, the number of participants connected to the reflector may change dynamically. Figure 5 shows TeME’s user interface of the telepresentation service.

**Audio Connection:** Whenever the reflector starts executing it immediately spawns an audio reflector/mixer process. The audio reflector/mixer is responsible for establishing the audio communication between participants. The audio reflector mixes all audio streams coming from clients and broadcasts a set of mixed streams back to them. TeME’s clients record and play audio sampled in one channel, at 8KHz rate with a sample size of 16 bits. The audio connection of a client may be full-duplex or half-duplex depending on the audio card of the workstation. The connection allows a user to send either compressed or uncompressed audio. Compressed audio requires 13Kbits/sec bandwidth\(^2\) while uncompressed audio requires 128Kbits/sec but users enjoy better audio quality.

**Image Broadcast:** A user may instruct its TeME client to spawn one or more image servers. An image server gets images from some image source, for example a camera, and upon request sends the current image to a client using the HTTP protocol. The refresh rate at which image servers refresh their images is variable. Each image server is advertised to all clients via the reflector. Thus each client builds the list of all image servers and their associated refresh rates. From the list of image servers, a participant may choose to display one or more images. For each image the client sends a request directly to the image server to get image data. Sending the refresh rate of image servers to all clients prevents clients from requesting images at rates higher than the refresh rate of the servers. Figure 5, contains three images broadcast during a telelecture, two images are captured at CUI and one at GMD. Figure 6 illustrates the implementation of the image broadcast service: ovals represent processes, rectangles hosts, thick arrows represent image data transfers, thin arrows represent message data transfers. Currently there is not sufficient bandwidth for transmitting acceptable quality of video stream (25 frames/sec) over the Internet. However, displaying images refreshed at regular intervals,

\(^2\) TeME uses the GSM algorithm for compressing the audio streams.
Figure 5  TeME’s user interface of telepresentation and image broadcast services
for example a few images per minute, allows users to have a visual clue of what is happening at other sites of the session. In a LAN the situation may be different. There are LAN providing 100 Mbit/sec bandwidth and soon this figure will rise to 1Gbit/sec. In such an environment TeME users could send and receive video streams at 25 frames/sec.

**Electronic Mail:** A participant can write a message and send it to all participants of the session. This service is useful when there is a presentation and the presenter does not want to be interrupted during her/his presentation. The participants can progressively email their questions during the presentation. The presenter will answer the questions after having finished his presentation. It should be noted that the service of electronic mail is a simplified form of the existing mail programs: a message is broadcast to all participants, and there is no way to send a message to an individual participant. The reason for which we have implemented this service instead of using existing electronic mail is that we can identify which are the messages related to the session and store them in the TDB. Such messages may be useful for further processing, for example, for generating frequently asked question lists and for evaluation measures on the clarity of a presentation.

**Text Chat:** The chat service allows users to broadcast lines of text to the session’s participants. The difference to electronic mail is that each line of text is immediately transmitted to all other participants. It is thus a more interactive service than the electronic mail and is intended to simulate a "written discussion" between several users. Chat is a very useful service for tuning various parameters of a session, for example for setting the microphone and speaker levels, or for reminding a participant to introduce a
name allowing him to be identified by that name during the session instead of by his
machine IP number.

4 Meeting Services

An important requirement TeME should satisfy is extendibility to new meeting types. TeME
achieves extendibility as follows: the agenda definition and manipulation module provides a
graphical interface allowing the MA to specify prototypes of meeting agendas. Figure 7
shows the interface of the agenda definition and manipulation module. An agenda prototype
describes the generic structure of a set of meeting sessions. For a particular meeting session
an agenda is instantiated out of the agenda prototype. For example, the MA may define a
researchGroupMeeting prototype agenda describing the common structure of this type
of meeting. A particular research group meeting session then would be instantiated from the
researchGroupMeeting prototype.

Figure 7 Agenda definition and manipulation module interface

The TDB’s database schema used for storing agenda prototypes, agenda instances and
meeting sessions consists of a few relational tables. We name these tables meeting system
tables (MST). Figure 8 shows the relational schemes of MSTs. More precisely, the
meetings table stores information relative to meetings. Each meeting is identified with a
unique meeting_id attribute value. The intervals table contains the temporal intervals
into which a meeting session is decomposed. The notion of temporal interval is described in
detail in the remainder of this section. The events table stores events occurring during a
meeting session. The streams table holds for each meeting the URL where stream data are
located, for example, the meeting’s session audio stream recording.

meetings(meeting_id, meeting_name, prototype_id,
        start_date, start_time,
        stop_date, stop_time,
        duration)

intervals(interval_id, meeting_id, parent_interval_id, prototype_id,
          start_date, start_time,
          stop_date, stop_time,
          duration,
          interval_type, table_name,
          executed, ordered, child_number, mandatory)

events(event_id, meeting_id,
       event_date, event__time,
       event_type, description)

streams(stream_id, meeting_id, stream_type, stream_reference)

Figure 8 Meeting System Tables

Data stored in MST are related to data stored in tables defined by the MA. Tables
defined by the MA are named meeting data tables (MDT). MDTs are used for storing data
that are relevant to a particular meeting type. Relationships between MST and MDT are
based on data values stored in attributes meeting_id, interval_id, event_id and
stream_id. It is exactly this feature that makes TeME extendible to new types of meetings
and allows us to enrich meeting sessions with semantics relevant to a particular meeting type.

4.1 Defining Agendas

An agenda prototype (or a meeting type) describes the temporal structure of telemetings in
terms of temporal intervals. A temporal interval is associated with a starting date and time, an
ending date and time, a duration and a MDT name. Note that it is not necessary to specify the
duration, date and time values at the moment of agenda definition. Specifying date, time
and/or duration values for temporal intervals become constraints that are verified during the
meeting session. For example, specifying for a particular temporal interval $I$ a start time
value 15:00 and a duration of 15 minutes means that during the meeting session the interval $I$
should start at 15:00 and stop at 15:15. Failing to verify the constraint during the meeting
session will lead the system to warn the MA and prompt her/him to undertake an action for
satisfying the constraint. For example the action could be to modify the start and/or duration
values of $I$.

We classify temporal intervals into three types: basic temporal interval, sequence
temporal interval and alternative temporal interval. The interval_type attribute in the
intervals table contains this information. A basic temporal interval cannot be further

---

3 The definition of attributes meeting_id, interval_id, event_id and stream_id in MDTs should be the same with the definitions of these attributes in MSTs.
decomposed. A sequence temporal interval represents a sequence of temporal intervals. An alternative temporal interval represents a choice of temporal intervals.

Temporal intervals in an agenda are structured in a tree-like structure. Thus an agenda prototype is identified with a tree $T$ of temporal intervals. Basic temporal intervals can only be leaf nodes of $T$. Sequence and alternative temporal intervals cannot be leaf nodes. Figure 9 shows the temporal structure of the research group meeting type. Rectangles represent nodes of the tree and are labelled using a hierarchical notation prefixed with the letter $N$. Dark grey rectangles represent sequence nodes, light grey rectangles represent alternative nodes and white rectangles represent basic nodes. Arrows connect parent and children nodes. Parents are placed at the tails of arrows and children are placed at the heads of arrows. No starting or ending date and time values have been specified. These values will be filled during the meeting session.

For a particular meeting session of type $T$ a tree $T'$ is instantiated out of $T$: a replica $T'$ of $T$ is created. In tables meetings and intervals rows modelling prototypes are assigned the value 0 in attribute prototype_id. Rows modelling an instance $T'$ of $T$ are assigned the positive value of attribute meeting_id of $T$ in table meetings to their prototype_id attribute. Nodes of $T'$ may be in one of three states: created, started and stopped. Initially all nodes of $T'$ are in state created. During the meeting session some nodes of $T'$ may reach the started state depending on the MA’s actions and choices. The fact that a node enters the started state models the fact that the temporal interval corresponding to the node is now taking place. The boolean attribute executed in table intervals is initialised to the truth value false and is assigned the truth value true when the interval reaches the state started. A node changes state from state started to state stopped when the corresponding temporal interval ends.

Nodes are classified as mandatory and non-mandatory. The boolean attribute mandatory in the intervals table contains this information. In Figure 9 mandatory nodes contain the label MANDATORY. A mandatory node will eventually start. A non-mandatory node may never start meaning that the corresponding temporal interval may never take place. Note that when a node is “started” and has children then at least one of its children should also start.

A sequence node is modelling the fact that its children intervals should start in sequence. A sequence node may be ordered or unordered. The boolean attribute ordered in the intervals table contains this information. In Figure 9 ordered nodes contain the label ORDERED. For an ordered sequence node its children should start in the order in which they have been defined. For an unordered sequence, the order in which children nodes start is determined during the meeting session. A sequence node should satisfy the following constraint: the total duration of its children nodes should be the same as the parent node duration. In addition, the time periods of children should be contiguous. In Figure 9 node N.1.1 is an example of a mandatory unordered sequence node for which no MDT has been specified. One child of node N.1.1 is associated with MDT PhDprogressReport. The
second child of node N.1.1 is associated with MDT courseReport. Even though all children are non mandatory at least one of them must be instantiated since their parent is mandatory. Figure 10 illustrates the schemes of the two MDTs PhDprogressReport and courseReport. PhDprogressReport associates the name of a PhD student with its
progress level. courseReport associates a course with an evaluation report produced the date. In both tables the attribute interval_id allows the MA to schedule meetings’ temporal intervals during which progress reports and course evaluations are discussed.

PhDprogressReport (interval_id, name, progress_level)
courseReport(interval_id, course_name, date, evaluation)

**Figure 10** Meeting Data Tables.

An alternative node is used for modelling the fact that one of the children nodes will be started during the meeting session. However before the meeting session it is not known which one will be chosen. An alternative node respects the following constraint: the duration, time and date values of all its children are the same as the duration, date and time values of the alternative node. In Figure 9 node N.1.2 is an example of an alternative node. It has three children associated with MDTs projectProposal, paperRehearsal, researchIdeas. Note that an alternative node is always exclusive: only one of its children will be started. A non exclusive alternative node (one or several of its children are started in sequence) is modelled as an unordered sequence node with non mandatory children nodes.

### 4.2 Recording Sessions

Recording the meeting session is a task performed by the reflector since the reflector is aware of all events that occur during a meeting session. Recorded data include the mixing of all participant’s audio streams, system events and user events.

Audio streams are not directly stored in the TDB: they are stored in files. The streams MST maintains the associations between audio files and meeting sessions. More precisely, the stream_reference attribute contains the name of the file containing the data and the stream_type attribute the type of the stream, for example stream of images or stream of audio samples. Note however that we do not currently store images broadcast during meeting sessions due to our limited storage capacity.

System events are managed by TeME and are the following: broadcast of a URL from presenter to participants during presentations, the broadcast of an email message, the broadcast of chat lines, the joining and leaving of users from the session, the change of presenter. System events are automatically stored in the events MST when they occur. The event_type attribute allows us to distinguish between the various types of events. The description attribute maintains information further describing the event. For example when the event type is the broadcast of a URL the URL reference is stored in the description attribute.

User events are those events that the MA decides are worth storing in the TDB. An example of a user event is the request from a participant to include in the meeting’s minutes a document not available in electronic form. The MA may enter this event in the TDB and later
associate with a scanned version of the document. For storing user defined events the MA should define a MDT for that purpose.

4.3 Temporal Interval Management

Another important task of TeME during the meeting session is the management of the agenda instantiation. Let us assume that the meeting session has started and that we are currently at temporal interval $I$, that is, $I$ has started but has not yet stopped. Depending on whether or not the duration of $T$ is known, the system determines the end of $I$ in the following way. If $I$’s duration is known then the system monitors a clock and warns the MA whenever the clock value exceeds the date and time values of $I$. Then the MA may decide either to extend the duration of $I$ or to confirm the end of $I$. If $I$’s duration is not known, the system simply waits until the MA signals the end of $I$.

Once the end of a temporal interval $I$ is specified the system tries to determine which temporal interval should be started next. This is achieved in two phases. First, the system tries to identify a set $A$ of temporal intervals (ancestors of $I$) whose end date and time can be deduced from end of $I$. In fact the end date and time values of the elements of $A$ will be identical to that of $I$. Thus end date and time values of $I$ are propagated recursively from children to parents. Note that $A$ may be empty. In a second phase a temporal interval $N$ is identified which is the closest ancestor of $I$ of type sequence and for which at least one of its children has not yet been started. One of $N$’s children will be chosen to be started, say $S$. The start of $S$ may imply the start of a set $D$ (may be empty) of temporal intervals which are descendants of $S$. All elements in $D$ will be assigned the same starting date and time values which are assigned to $S$. Thus the start dates and time values are propagated from a parent to children down to the leaf nodes.

More precisely, the following algorithm is used for identifying the sets $A$, $D$ and node $N$. First the function find_ancestors is called recursively with a formal parameter $P$. find_ancestors is initially called with $I$. When find_ancestors stops executing, $N$ is assigned the temporal interval to be started next. Then the function find_descendants is called recursively executed with a formal parameter $P$. find_descendants is initially called with $N$. Figure 11 contains the pseudo code description of the two functions.

If find_ancestors is called with $P$ being a sequence temporal interval either ordered or unordered and all temporal intervals that are children of $P$ are ended, then $P$ is inserted into $A$ and a new call of find_ancestors is executed with the parent of $P$ as parameter. If there is a child of $P$ not yet started then assign $P$ to $N$ and return. If $P$ is an alternative temporal interval then insert $P$ into $A$ and a new call to find_ancestors is executed with parameter the parent of $P$. 
find_ancestors(P) {
    if (type(P) = sequence) {
        INTERVAL_SET ← get_children(P);
        if (all_member_are_stopped?( INTERVAL_SET)) {
            insert P into A;
            find_ancestors(parent_of(P))
            return;
        } else {
            N ← get_child_not_yet_started(P);
            return;
        }
    } else { /* type of P is alternative */
        insert P into A;
        find_ancestors(parent_of(P));
        return;
    }
}

find_descendants(P) {
    if (type(P) = ordered_sequence) I ←get_next_children(P);
    else if (type(P) = unordered_sequence) I ←ask_MA_chose_children(P);
    else if (type(P) == alternative) I ←ask_MA_choose_children(P);
    else { /* basic node */
        insert P into D;
        return;
    }
    insert I into D;
    find_descendants(P)
    return;
}

**Figure 11** Algorithms used for determining the next interval to be started

If find_descendants is called with P being an ordered sequence temporal interval then let I’ be P’s child that should be started first. I’ is inserted into D and find_descendants is called with parameter I’. If find_descendants is called with P being an unordered sequence temporal interval the MA should indicate which child of P not yet started, say I’, should be inserted into D. find_descendants is then called with parameter I’. If find_descendants is called with P being an alternative temporal interval the MA should indicate which child of P, say I’, should be inserted into I. find_descendants is then called with parameter I’. Finally if find_descendants is called with P being a basic temporal interval then P is inserted into D.

4.4 Meeting post-production

The description we have given for the use of TeME’s services is that of the “waterfall” model: a meeting agenda is instantiated first, the meeting session is recorded and finally TDB queries are executed.

However recording meetings may be performed differently. TeME’s services can be used for recording events and the audio stream of an “unclassified” meeting session. After the meeting session, the MA may instantiate an existing agenda prototype or define a new meeting agenda and associate the new agenda instance with the unclassified meeting session.
The meeting agenda’s temporal interval tree will provide the temporal decomposition of the meeting session. Data in MDTs would complement the non-temporal semantics of the meeting session.

The above two schemas of modelling meetings are at opposite extremes of one another. An intermediate schema is also possible. An agenda prototype is instantiated and the telemeeting session is recorded. The MA then may update the agenda which means that the temporal decomposition of the meeting session will be altered. Finally, additional data may be inserted into MDT related with the meeting session.

4.5 Querying the TDB

Querying the TDB is relatively straightforward. Users familiar with database technology can directly query the TDB. For users not familiar with databases the MA should provide stored queries for accessing the TDB. Querying the TDB is simply performed using the query engine of the DBMS implementing the TDB. Since we require TDB to be implemented on top of relational DBMS the query facility most commonly provided would be an (extended version) of SQL.

The only assistance TeME provides the MA for querying the TDB is an audio server and a client program for streaming the audio data from the audio server. The audio server streams the part of the audio file its client has requested. Thus whenever the result a TDB query contains temporal intervals the start and end time and date values can be used for requesting the sound corresponding to the temporal interval from the audio server.

5 Implementation

TeME is an inherently distributed environment: TeME’s services are provided to users connected through Internet and LANs. It is therefore natural to expect TeME to be highly portable in order to run on a variety of hardware/software platforms. Requiring TeME to run on a single platform would be an important limitation for its usage. We failed to completely satisfy this requirement: whenever TeME is ported into a new environment some programming effort should be invested. However we minimised this effort as much as possible. The majority of TeME is written in Java 1.1 and is highly portable. Only parts that access audio devices and modules which should execute efficiently in real time, like compression/decompression of audio data, are written in C/C++ and are less portable. The portability benefit of using Java is counterbalanced by performance penalty. Running TeME’s session client module on a 166MHz Pentium PC is unacceptably slow. A 200Mhz or higher Pentium PC is required for running TeME’s sessions client. TeME currently runs on SUNs under Solaris, on SGIs under IRIX, and on PCs under Windows 95 and Windows NT.

---

4 Until now, the Java part of TeME has been ported to a variety of environments without requiring any adaptation effort.
Another operational requirement for TeME is that the relational DBMS used for storing the TDB should provide a JDBC interface [10]. This actually holds for most popular relational DBMSs. Note also that the data type of MSTs’ attributes are either integer or string. Thus the TDB could be supported by any DBMS satisfying the JDBC requirement.

In general the development of TeME has been influenced by the following guideline: reduce the programming effort to a minimum. Thus our strategy consisted in integrating existing software components and tailoring them to satisfy our specific needs. A development effort was undertaken only when components either did not exist or could not be properly integrated. The implementation of the telepresentation service is an example of this strategy. Most of the functionality of this service is carried out by the HotJava component. The drawback of the above strategy is that limitations and/or bugs of the software components, which cannot be easily worked around, become TeME’s limitations. For example, we cannot broadcast the presenter’s navigation changes whenever she/he follows tagged links in the same HTML page and the reason is that the HotJava component does not signal such URLs to the requesting software component.

TeME uses the TCP/IP protocol suite for process communication. More precisely, for broadcasting URLs, text lines for the chat service and emails TeME modules establish TCP connections. For the real-time audio communication during meeting sessions there are two network connections between a client and the audio reflector/mixer: a TCP connection and a UDP connection. The TCP connection is used for signalling. The UDP connection is used for sending audio data. The choice of different network connections is justified by performance and reliability considerations. For the transmission of signalling data reliability is more important than ensuring small transmission delays. Therefore TCP connections are appropriate for the transmission of signalling data. A UDP connection is faster than a TCP one but packets may arrive out of order or be lost. Depending on the amount of audio packets lost or out of order, a UDP connection may or may not provide an acceptable quality audio transmission. For audio data we have privileged small transmission delays over reliability and therefore we have not implemented any algorithm for recovering lost or out of order audio packets. A lost packet causes a silence period of 250 ms (a packet contains audio data corresponding to 250 ms of audio stream) on the output audio stream. An out of order packet is simply discarded whenever arrived at the destination.

The audio communication between a user and the TDB server used when querying the TDB is implemented again using a TCP connection for signalling and a UDP connection for the audio data transmission. Since the audio transmission is non-interactive and one way, from the TDB server to the user, introducing a small initial delay (up to three seconds) to the audio transmission is acceptable by the user and allowed us to implement a simple schema for recovering lost packets thus improving the quality of the audio communication. The client

---

5 We use an Oracle DBMS in our current implementation.
6 The version of the HotJava component we have used is 1.1.2
7 We expect TCP packets with sizes smaller than 10 bytes to arrive at destination in less than 500 ms. Receiving signaling packets with greater delays than 500 ms causes delays between the user’s actions and system docility to become noticeable to the user.
keeps a FIFO buffer of packets. Whenever a packet is lost the client sends a request to the TDB server to retransmit the lost packet.

6 Conclusions

In this paper we have presented TeME an environment for modelling and conducting telemeetings. TeME integrates two types of services: session services and meeting services. Session services allow a set of network connected users to attend a telemeeting. Meeting services allow the meeting administrator to model telemeetings store and query telemeeting sessions in a telemeeting database.

There are several software packages, either freeware or commercially available, implementing session services including CUSeeMe [4] NetMeeting [11] Intel’s Internet Video Phone [12]. Session services in these software packages are implemented in a similar or different way to TeME. A survey of these systems, their services and implementation can be found in [7]. However none of them provides the means to model the meeting activity and to store it in a database.

Research efforts [5] [6] have experimented with implementations of meeting services in addition to session services. Their approach is summarised as follows: record the meeting activity in files using either a customised or a some standard format\(^8\) and enable users to add annotations to the recorded stream. Then the recorded activity can be “replayed” much like a tape in a video tape recorder. Annotations serve two purposes: they add semantic content and can be used as indices for the recorded meeting.

In TeME we have adopted a different approach since we let users define meeting agenda prototypes and associate agendas with meeting recordings. An agenda provides a structure to a meeting. It is exactly this feature that allows us to provide users the power of dbms engines for querying meetings. In addition, our approach is flexible enough for two reasons. First, users may define their own agendas. Second, the association between meeting and agenda can be performed either before or after the meeting session. The flexibility and extendibility of meeting types offered in our environment is counterbalanced by the programming effort the meeting administrator should provide for users not familiar with dbms query capabilities.

Our future development plans target the following directions. Improve the querying capabilities of TeME. More precisely, provide the meeting administrator with a high level scripting language and/or tool allowing her/him to extract, synchronise and replay various data streams, for example, stream of audio data, stream of various types of events and enable the specification of actions to be executed depending on the type of event. Extend TeME’s session services with new capabilities, for example integrate voice based recognition of speakers and provide support for new media types like video and animation.

---

\(^8\) For example Quicktime.
References


