CCS: CORBA-based Conferencing Service

Dirk Trossen, Karl-Heinz Scharer
Technical University of Aachen
Dept. of Computer Science IV
Ahornstr. 55
D-52074 Aachen, Germany
trossen@i4.informatik.rwth-aachen.de

Abstract: An efficient conferencing service facilitates the implementation and the run-time control of conference applications. Key features of a conferencing service are conference management, multicast communication support, application state synchronization and user data marshalling. The paper defines a conferencing service to fulfil the requirements of conferencing applications. An object model of the service is defined and implemented using the CORBA distributed platform. The conference control protocol, which is used for our conferencing service, is introduced together with architecture of our implementation.

1 Introduction
Conference applications enable the communication and cooperation of widely distributed end-systems. The key function in these applications is the multi-point communication among several users. In heterogeneous environments this involves marshalling and demarshalling of user data. These features are crucial for applications such as cooperative working on documents and multi-user games. However, additional functions are needed in most conference applications. Due to concurrent activities, the need arises for coordination and synchronization of distributed program entities and users. Furthermore, functionality to inquire for information about the state and structure of existing conferences may be needed, e.g. to join a conference or to invite other users. All these features of conference applications are mostly independent of specific scenarios. Therefore it is useful to provide them by a generic system layer. Usage of the generic functionality accelerates and simplifies the implementation of conference applications.

In our paper we aim at developing a CORBA-based conferencing service by introducing the main features of the conference control protocol as the basis of our conferencing service. An object model and architecture for a CORBA-based service is proposed. Multipoint communication is one of the key features of a conferencing service. CORBA [5] provides a service to support multipoint in heterogeneous environments. We shortly introduce one implementation for its usage in a conference environment.

The paper is organized as follows. In section 2, we present some related work. In section 3, the conference control protocol is described, before introducing the realization in a CORBA environment in section 4. In section 5 conclusions are presented.

2 Related Work
Most of the requirements of a conferencing service are common to some applications. To standardize protocols for data communication the ITU designed the T.120 protocol stack [4]. This protocol family includes multipoint communication and generic conference support over different networks using profiles. A tree topology of providers is used with a dedicated top provider. Due to the lack of delta refreshes of database contents and the missing support of multicast in the control and data stream, the T.120 protocol stack

1. The work presented in the paper results from a joint project between Philips Research and Technical University of Aachen. The project was funded by Philips
is not well scalable as shown in [2]. Approaches [3] exist to combine the T.120 and CORBA [5] platforms to fulfill the requirements of a conferencing application. The IETF (Internet Engineering Task Force) defines conference and multicasting services that can be compared with the features of the T.120. They, for example, define a simple conference control protocol [1], a simple conference invitation protocol [7] and multicast transport protocols such as RTP (real-time protocol) to transport real-time streams over IP multicast. The services and features of the conference control protocol of this approach are comparable with the T.120, but unfortunately this protocol is not described in more details.

Another approach for a distributed conferencing service was developed in the HORUS project [8]. There, a description is given how to realize a framework for distributed conference applications. User data marshalling is realized by using CORBA. Multipoint conferences are supported but there is no standardized communication platform.

3 Conference Control Protocol

A number of features are inherent to different conference applications. These features can be offered in a generic way by a conferencing service infrastructure. In [3] a detailed definition of application requirements and features of a conferencing service can be found. Since the wide variety of features can not be adequately treated in a single paper, we restrict ourselves in the following to only four essential features, namely conference management, multicast support, state synchronization and user data marshalling. Aspects like security, streaming, application functionality are not stated in this paper. In this section we propose a conference control protocol to fulfill these requirements of a conferencing service together with a high scalability in large scenarios.

The core of our conferencing service is the control protocol, which is used to provide the functionalities of our service. The aspect of user-data marshalling in heterogenous environments is provided by a layer on top of our service, which is not described in this paper. Figure 1 shows the architecture of the proposed protocol stack. The conference control protocol uses a reliable multicast transport protocol. It maps multipoint connections of the conference to multicast connections of the underlying multicast transport protocol, but user data may be sent directly using the multicast protocol. The protocol is provided by conference servers, which are organized in a hierarchical tree topology. Conference clients are connected to one of these servers. Each control request is sent upward in this topology, some indications like database refreshes are sent back via a multicast indication cloud, which is established during setting up the conference. This cloud is a dedicated multicast channel, each provider is connected to when joining the conference. The database of conference members and applications is replicated at each provider in the tree. Thus, read requests are minimized. If a database entry changed, the database is only partially refreshed. This is called delta refresh, which is sent using the multicast indication cloud. Therefore scalability is improved.