Scalable Group Communication in Tightly Coupled Environments

Approved Dissertation Thesis of
the Faculty of Mathematics, Computer Science, and Natural Science
of the University of Technology North-Rhine Westfalia Aachen
to obtain the Doctorate in Computer Science

Submitted by
Dirk Trossen
M.Sc.

Advisor: Professor Dr. Otto Spaniol
Professor Dr. Boudewijn Haverkort

Date of Oral Defense: July, 11th 2000

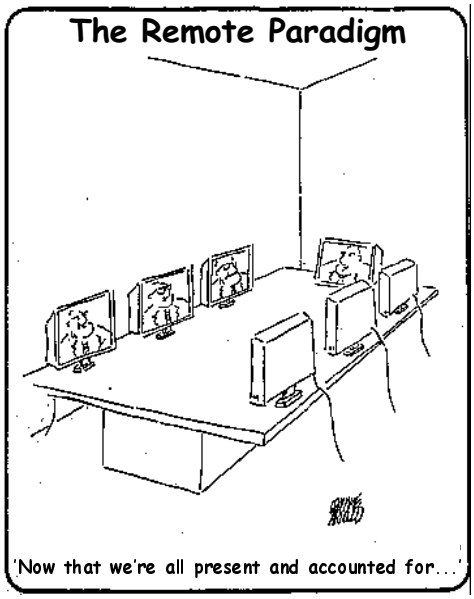
Acknowledgements

I wrote this dissertation during my time as a researcher and Ph.D candidate with the Chair of Computer Science 4 at the University of Technology Aachen, Germany.

I would like to thank my advisors, Professor Otto Spaniol and Professor Boudewijn R. Haverkort. I am thankful to Professor Spaniol for offering me the opportunity and freedom to choose my fields of interests and to carry out my ideas in this thesis. I am also grateful to Professor Haverkort who acts as the co-advisor of this thesis. His comments helped me to improve and finalize my thesis during the last months.

I would also like to thank my colleagues for giving me the opportunity to work in this specific scientific environment for more than four years. It was a pleasant time and I learned a lot. I'd like to thank Christian Cseh for his fruitful comments to my work, but also for his contributions to our professional and non-professional talks at 'Downunder'. Furthermore, I want to thank all people who were patient enough to read my thesis for correction, namely Christian Cseh, Marko Schuba, Karl-Heinz Scharer, and Andreas Ropers. Moreover, I would also like to express my gratitude to the student workers and diploma students. Pascal Papathemelis, Arpad Katona, Karl-Heinz Scharer, Wolf-Christian Eickhoff, Peter Kliem, and Erik Molenaar contributed to my work and this thesis and I enjoyed immensely to work with them. A special thank to Nils Seifert who made the protocol implementation of MTP-SO available to me.

A very personal thank you goes to my family and friends. Thank you to Martin for his patience and understanding during our billiard evening discussions. Thank you also to Andreas and Arlett for their moral support as friends and best neighbors. Thank you to my mother and Claus for their encouragement and love. And finally thank you to my brother Axel and his family for their understanding. His son and my godson Kevin showed me impressively how irrelevant a Ph.D thesis can be when looking into the eyes of a newly born child.



Source: Unknown

Table of Contents

CHAPTER 1

Introduction

Pushed by the development of the global Internet, the need for systems to enable cooperative work among distributed users has become more and more popular in the past fifteen years. Well-known tools like e-mail, file transfer, or especially the World Wide Web have been used for *asynchronous* exchange of information, leading to an exploding usage of the Internet in the past five years. But also more and more *synchronous* conferencing applications have been considered in recent research. Some of them led to first products for collaborative group communication between distributed sites. Additionally, the development of these tools is pushed by the increasing costs and distribution of institutions and companies, since conferencing applications promise to be an efficient utility for communication and cooperation between widely spreaded sites. The notion of *Computer Supported Collaborative Work* (CSCW) has been used in the research community to describe a wide spectrum of scenarios of collaborative communication among distributed users. In [Lyy90], CSCW is characterized as "*...computers in cooperative, coordinated, and collaborative work groups under various task, time and space*".

According to the definition presented in [Lub95], s*ynchronous, distributed* cooperation takes place at the same time between distributed sites. As a consequence, a minimal synchronization between the participating users is necessary, both on control and data transfer level. Typical scenarios are, among others, audiovisual meetings, e.g. in a virtual conference room, a panel discussion or lecture via the Internet, or lessons of a school. Applications used in these scenarios are videoconferencing tools for audiovisual exchange, shared whiteboards for distributed editing of documents, or shared applications to enable a common view of available data. Despite the variety of the scenarios, most of the required functionality in these scenarios is the same. Hence, it is desirable to design conferencing systems suited for different situations independent from the size of the scenarios in terms of participating users, i.e. to enable the *provision* of scalable group communication.

For a proper design of conferencing systems, a proper *understanding* is crucial of what happens in the considered scenarios. On the one hand, *a posteriori* approaches, like user or field tests, are not well suited for the development of new systems, because they suffer from the long development cycle to run the tests. On the other hand, *a priori* approaches, like simulation of the system, enable to shorten the development time of conferencing systems. For that, appropriate descriptions of the systems are crucial. Due to the complex interactivity in these scenarios caused by the mapping of more or less complex social rules on distributed environments, sim-

ple techniques are not suited to model these systems. As a consequence, appropriate *modeling methodologies* are required to support the development of conferencing applications and to evaluate the system without implementing and installing it.

This thesis proposes a scalable approach for synchronous, collaborative work and a description methodology to model and evaluate groupware systems as an *a priori* approach. In chapter 1.1, an overview of the investigated research fields in the area of group communication is given, and the goals of this thesis are presented. Chapter 1.2 outlines the structure of the thesis to clarify how to reach the defined goals.

1.1 Fields of Research and Goals of the Thesis

As mentioned above, synchronous, collaborative scenarios form a new group of interactive conferencing applications, which have become more and more popular in the past fifteen years. Most of the proposed systems in the past have only been designed for small scaled scenarios in terms of participating users. Events like the IETF (*Internet Engineering Task Force*) meeting in March 1992, which was partially distributed via the Internet, extended the field of synchronous cooperation to a larger scale, i.e. several hundreds of users. This event opened a new chapter of research, namely the *development of services for scalable group communication in the Internet*.

The key functionality which is required in most of these scenarios is *management* of the conference environment, *multipoint communication* among several users, involving marshaling and demarshaling of user data, and functionality for *coordination* and *synchronization* of distributed applications and users due to concurrent activities. All these features are mostly independent from specific scenarios. Thus, it is useful to provide them by a generic layer to accelerate and simplify the implementation of conferencing applications. Due to the globalization of the Internet and the distribution of the participants, *scalability* in terms of group size and geographical group distribution is a big issue in the development of a generic conferencing service. Furthermore, the provided services as well as the used protocol mechanisms have to be designed in a *robust* manner to ensure the acceptance by the users.

Concerning the control mechanisms of conferencing systems and the degree of coupling the participating entities, the paradigms of *loosely* and *tightly coupled* environments are distinguished. The first only provide poor synchronization functionality and are mainly used for streaming scenarios with small interactivity between the participating members. As a consequence, systems following this paradigm are not well suited for scenarios with a high interactivity following specific rules similar to a real world scenario. Tightly coupled environments provide a richer functionality for synchronization of distributed applications by interconnecting the different sites using point-to-point connections for control purposes, mainly as tree or star topologies. Scenarios with a high synchronization and interactivity need, i.e. scenarios with a complex *social protocol*, are realized following this paradigm. Both paradigms should be evaluated with respect to their applicability in different scenarios.

In addition to the design of a conferencing service, the problem of modeling groupware scenarios is another big research topic in the group communication community concerning mainly two problems. The first is related to the provision of means for *structurally modeling* groupware scenarios by describing objects and the participating entities accessing these objects.

The second considers the performance evaluation of the modeled system. For that, realistic workload is crucial to obtain realistic performance evaluation results. As a consequence, a *dynamic description* of the system and user behavior should be provided. Simple models, often

assuming simplified traffic characteristics, do not properly reflect the high interactive characteristics of typical groupware scenarios, e.g. a panel discussion. The traffic in these scenarios is highly correlated which has to be taken into account by the used model.

Common for both modeling issues is the necessity to provide a model being suited even for large scaled configurations. Using this model should enable the evaluation of specific services and mechanisms of the system without implementing and installing the conferencing system on hundreds of machines and running expensive user tests.

In this thesis, two research issues are addressed, namely the *design* as well as the *evaluation* of a conferencing service. On the one hand, requirements of conferencing applications are defined before proposing the services as well as the basic mechanisms for a *Scalable Conferencing Control Service* (SCCS) facilitating the implementation of conferencing applications.

On the other hand, a modeling methodology is presented which provides a framework enabling both analytical and simulative evaluation of conferencing systems. This framework is used to evaluate two basic mechanisms of the proposed service to demonstrate the feasibility to obtain performance evaluation results even for realistic groupware scenarios of larger scale.

In the following, both research topics are explained in detail as the main goals of the thesis.

Goal 1: Designing and Prototyping a Scalable Conferencing Control Service

Lots of conferencing systems, products, and standards have been proposed during the past fifteen years. The scope of these platforms and systems reaches from simple client-server-based solutions to fairly complex tree-based provider environments. Lots of research has been performed for the provision of a generic conferencing system to facilitate the development of conferencing applications following both paradigms concerning the realized topology, namely loosely and tightly coupled environments.

The first step towards a design and prototype of a generic conferencing service is the definition of requirements which are necessary to be fulfilled for the implementation of conferencing applications. In addition to the service requirements, scalability and robustness of the offered services and mechanisms are crucial from a technical point of view.

While most of the existing systems fulfil the service requirements, which is shown in a review of related work, the *scalability* issue is only poorly covered. As a consequence, existing platforms are restricted either in their applicability with respect to the conference size or in their service functionality to provide a higher scalability. Especially loosely coupled solutions restrict their synchronization and conference management functionality to improve their scalability in terms of supported conference members. Available tightly coupled systems are restricted to several users only mainly due to inefficient underlying protocol mechanisms.

As a consequence, this thesis proposes the *Scalable Conferencing Control Service* (SCCS) as a tightly coupled solution to fulfil the service requirements by the provision of sophisticated conference management functions, multipoint transfer capabilities, and application synchronization functionality. Beside the design of the services of SCCS, the presented work focuses on a scalable provision of the service, i.e. to use mechanisms which are suited for larger conferences in terms of participating members. For that, efficient multicast functionality of the underlying transport system is extensively used in the service. Moreover, an own resource management scheme is proposed which allows to improve the responsiveness of the system. For a further improvement of the system, a dynamic reconfiguration algorithm is presented which dynamically rebuilds the conference environment during runtime. The services and protocol mechanisms are properly defined giving a detailed description of the service and protocol data units and the sequence of messages to provide a specific functionality.

In addition to the service and protocol definition, prototyping the presented conferencing service is another big task to demonstrate the feasibility of the proposed service as a generic solution for the development of conferencing applications. For that, the object and process model of a service implementation is presented as well as core activity diagrams describing the main functionality of the service. Moreover, a shared whiteboard application is presented to demonstrate the main functionality of the service.

As a starting point for the second goal of the thesis, general performance evaluation goals are defined in terms of performance measures to be obtained and evaluation parameters to be considered during the evaluation. It is the task of the second part of the thesis to provide methods fulfilling these presented performance evaluation goals.

Goal 2: Modeling Groupware for Evaluation of the Conferencing Service

Modeling groupware scenarios is also an issue which is investigated by a large number of research projects. The proposed approaches reach from simple stochastic models over workflow management schemes to behavioral methods.

Similar to the first part of the thesis, group communication modeling requirements are defined covering aspects like communication and user behavior description, but also the ability to define a sophisticated access control model to distributed objects in the system. Furthermore, it is a crucial requirement for groupware scenario models to enable an analytical as well as a simulative performance evaluation of the system.

The defined requirements are used to assess the related work in the area of group communication models. It is shown that existing frameworks are not well suited for conferencing scenarios. Especially the aspect of describing user data and control traffic is only poorly supported by existing modeling techniques. Furthermore, none of the presented methods provides a sophisticated access control model to describe the dynamic change of access rights to distributed objects in a conferencing scenario. As a consequence, the related work is hardly suited to evaluate communication and protocol mechanisms of conferencing environments for realistic groupware scenarios.

To overcome these weaknesses of the related work, this thesis proposes a modeling methodology for groupware systems. The presented model is based on *roles* within a scenario interacting according to defined *rules* which are based on a *social protocol*. The performed actions regarding a defined social protocol are assumed to fully describe the communication which is carried out between the conference members, and the dynamic change of access rights. For the description of these actions, standard methods like stochastic Petri nets or specification languages are used. With this framework, analytical as well as simulative evaluation of the underlying conferencing system is enabled with an additional structural model of the groupware scenario.

In the presentation of the modeling technique, the object model is defined to statically describe the modeled system. Furthermore, the process model is outlined for the dynamic description of the participating entities in the conference in terms of *role instances.* Finally, a syntax definition of the used description language is presented. A major issue in the proposal of the modeling framework is the independence of the framework from the realization of the behavior description of the different roles. Two examples for this realization are presented enabling the analytical evaluation of the system by using stochastic Petri nets and providing a simulative evaluation by using a specification language which is mapped to an event-based simulation. For the latter, a simulation framework is presented which includes an object and process model resulting in an extendable simulation tool for groupware scenarios.

The main purpose of proposing a modeling framework in the context of SCCS is the evaluation

of basic protocol mechanisms of the service. From a quantitative point of view, the evaluation outlines the achieved improvement when applying the proposed mechanisms even in large scaled scenarios. Hence, the evaluation should prove the applicability of SCCS for a larger scale in terms of the conference size.

From a qualitative point of view, the evaluation demonstrates the feasibility and computational effort of different modeling techniques in complex scenarios with respect to the size of the conference and the described system behavior. For that, analytical as well as simulative methods are applied to outline the applicability of the different methods, especially of the proposed modeling framework.

1.2 Overview

In the following, the structure of the thesis is presented which is used to reach the described goals of chapter 1.1. In figure 1-1, the chapters of the thesis are presented as an overview to

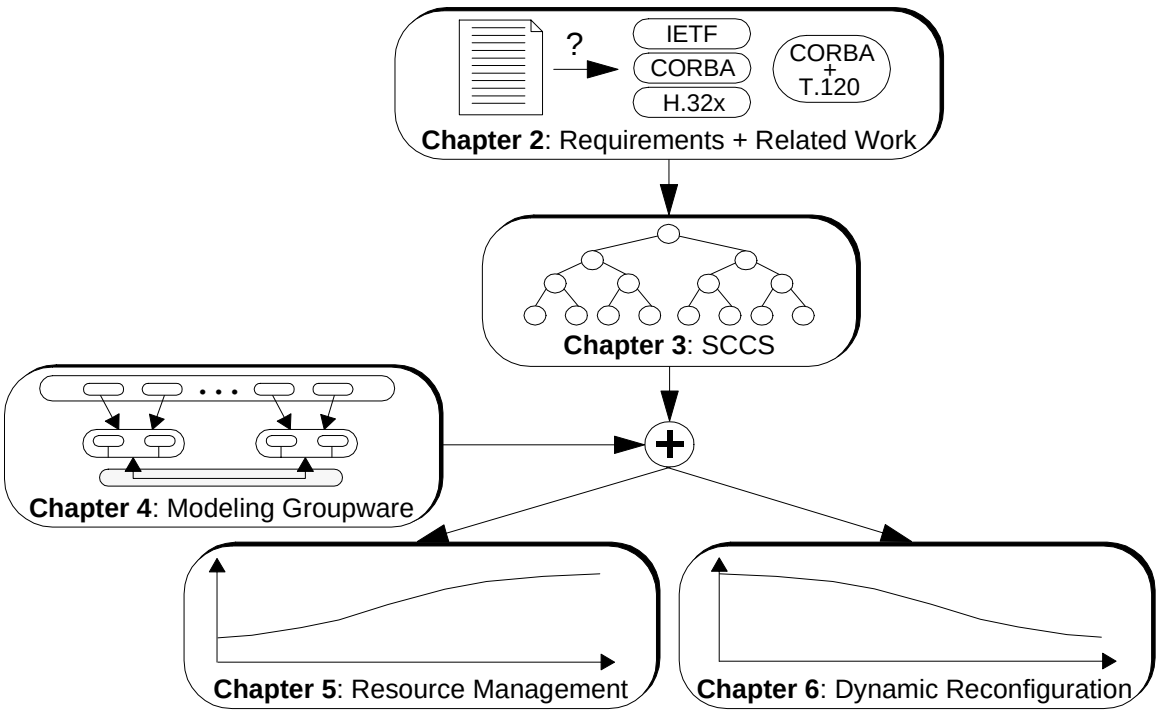


Figure 1-1: *Structure of the Thesis*

illustrate the relations between the different chapters of the thesis.

The purpose of chapter 2 is twofold. On the one hand, it defines service and technical requirements of groupware applications for the provision of a generic conferencing service. Furthermore, technical definitions and paradigms for conferencing systems are introduced to clarify the considered environments in the following. On the other hand, chapter 2 gives an overview of related work in the area of group communication and conferencing systems. Specifically, the H.32x standards of the *International Telecommunication Union* (ITU) are presented as well as the Internet Multimedia Architecture preferred by the *Internet Engineering Task Force* (IETF). Additionally, the *Common Object Request Broker Architecture* (CORBA) is introduced as a more general distributed platform, which is, as a forth platform, combined with the ITU-T.120

standard as proposed by the author.

All presented group communication environments are described and assessed with respect to the defined requirements of conferencing applications for a better comparison of the platforms.

Chapter 3 presents the *Scalable Conferencing Control Service* (SCCS) as a proposal for a generic service to facilitate and accelerate the development of conferencing applications. For that, the considered environment, the provided services, and the used protocol mechanisms are presented with respect to the defined requirements of chapter 2. Beside the definition of the services and protocol mechanisms, the chapter deals with the presentation of a prototype implementation of SCCS. For this realization, an object oriented approach is chosen. Hence, the used object as well as a corresponding process model are outlined. Additionally, activity diagrams of the main control functionality of the implementation are depicted. Similar to the related work, SCCS is presented and assessed with respect to the defined service and technical requirements of chapter 2.

As a preparation for the evaluation of SCCS, this chapter also defines the main performance evaluation goals, outlines the performance measures which are of interest during the evaluation, and considers the parameters of the system and their impact on the evaluation goals.

Modeling groupware scenarios is the topic of chapter 4. Requirements are defined to be fulfilled by modeling approaches. A review of related work in this area is presented, which leads to the conclusion that especially communication and access control aspects are only poorly supported by current approaches. Hence, an own modeling framework is proposed fulfilling the defined requirements and enabling the analytical as well as the simulative evaluation of SCCS. Two realizations for the behavior description in this framework are depicted, namely stochastic Petri nets and protocol specification languages. For the latter, a framework is introduced enabling the event-based simulation of large scaled scenarios with a realistic system workload.

The modeling framework of chapter 4 is used to perform the performance evaluation of main SCCS mechanisms, namely the resource management and the dynamic reconfiguration, following the presented evaluation goals of chapter 3.

Chapter 5 performs the first evaluation which compares the proposed resource management scheme in SCCS with a standardized, centralized scheme of the ITU. The purpose of the evaluation is twofold. First, it is meant to demonstrate the applicability of different modeling approaches. For that, a simple stochastic model as well as the stochastic Petri net and event-based simulation realization of the proposed modeling framework are used. Second, the evaluation determines the performance gain which can be achieved when applying the proposed scheme to show the improvement of the responsiveness of the conferencing system even in large scaled conferencing systems.

The evaluation of the dynamic reconfiguration, which is proposed in SCCS to reconfigure the conference environment during runtime for optimization, is performed in chapter 6. Considering the first goal of the evaluation, it is demonstrated that the proposed algorithm is able to detect certain *activity patterns* in the conference and groups the corresponding members nearby in the conference environment. Furthermore, the improvement of the responsiveness is determined when applying the proposed technique. At last, the impact of the reconfiguration parameters on the specific performance measures is investigated leading to setting guidelines for these parameters. Due to the complexity of the dynamic reconfiguration, only the simulative method based on the modeling framework of chapter 4 is used for this evaluation.

Chapter 7 concludes the thesis and summarizes the main results of the presented work.

CHAPTER 2

Group Communication Systems

A lot of research has been performed during the past fifteen years concerning the topic of *Computer Supported Cooperative Work* (CSCW). Applications and systems ([BeKo98][BrRe98] [CBV92][GuPl93][HKS93]) were developed to enable synchronous collaborative work among distributed users in a wide spectrum of scenarios, e.g. with different group size or different specific application functionality.

As a common result from this research, several key features of conferencing applications can be outlined as scenario-independent functionality, namely *conference management facilities*, support of *multipoint transfer* of user data, *distributed applications support*, and provision of commonly used functionality as *standardized application protocols*. Furthermore, *scalability* in terms of group size and geographical distribution of entities and *robustness* of services and mechanisms are big issues in the design of such conferencing services.

As a consequence, some of the research work is focussed on the provision of a generic conferencing service (e.g. proposed in [AF91][ADH93][AnBa93][Ar92][BiRe94][Cr90][RBM96]) to accelerate and simplify the development of group communication applications. Even standardization organizations in the Telecommunication (ITU) and Internet (IETF) domain defined or are currently defining a common basis for such applications leading to several standards for conferencing environments.

In this chapter, typical group communication scenarios are introduced leading to a definition of requirements for the provision of a scalable conferencing service in chapter 2.1. These requirements are used for the assessment of existing systems as well as for the proposed conferencing service in chapter 3. As examples for group communication systems, three approaches are presented. The first one is based on the ITU standards for tightly coupled conferencing environments, while the second one covers the research effort in the Internet domain and is more focussed on loosely coupled scenarios. As a third environment, the CORBA framework is introduced as a generic platform for distributed applications. For all approaches, the service and system model is outlined with a brief introduction of the different system components. Furthermore, their main protocol mechanisms are presented.

Additionally, it is evaluated to what extent the conferencing systems fulfil the service and technical requirements. It is shown that all systems lack of certain functionality, e.g. missing scalability or poor conference management facilities.

2.1 Group Communication Applications

Many conferencing tools and applications were developed to cover either generic or specific scenarios to enable collaborative cooperation among distributed users. In this section, three typical synchronous application scenarios are depicted as examples for conferencing among distributed users. These scenarios are used to define requirements, on service as well as on technical level, for a generic provision of conferencing facilities.

2.1.1 Application Scenarios

During the past fifteen years, several group communication scenarios have been realized using different tools and systems. The scenarios cover examples from simple audio/video streaming situations, e.g. the first audio transmitted IETF meeting in 1991 [CaDe92], over more sophisticated Internet lectures ([Eff95][MASH99]) to scenarios for tele-education [BGL96] and teleworking.

In this section, three application scenarios, namely tele-conferencing, tele-education, and teleworking, are presented for a non-formal description of the functionality which is needed from a conferencing system. This description is used to define requirements for the provision of a generic service to facilitate and accelerate the application development for different scenarios.

Tele-Conferencing

The scenarios, covered by the first environments for group communication, are more or less pure *audiovisual* scenarios to support different kinds of distributed meetings. The difference between the facets of meetings can be found in the conference size and the degree of the interaction among the users. The first criteria might range from a few participants only up to several hundreds or thousands, while the latter range from high interactive conferences to simple streaming scenarios with almost no interactivity among the participants.

An example for a tele-conferencing scenario with high interaction is a *project meeting* which takes place mostly in small groups. For that, facilities like special conferencing rooms with special audio/video equipment [TrRao99] are often used. The potential group of users is normally well-known, i.e. often the conference is announced beforehand according to a given timetable. A conference database might reflect the participating members together with additional information, e.g. site address or phone number. Functionality like authentication and authorization is needed to provide secure and restricted access conferences. The users are exchanging video images and an audio stream, either mixed from the streams of all participants or selected according to a given scheme. Furthermore, a participant is often defined to conduct the meeting based on certain *social rules* for the interaction among the participants, e.g. to indicate a question before sending the own audiovisual data. Thus, the communication among the conference members takes place according to rules which have to be implemented by the underlying system.

Especially the progress in multipoint transfer in the Internet raised another kind of tele-conferencing scenario, called *Large Meeting*, in a larger scale in terms of participants. Audiovisual data is streamed from a central point to a number of participants which might even range up to several hundreds or thousands of conference members. Normally, the content of the audiovisual data is dedicated to a certain topic and the interactivity is rather small. Examples for this scenario are Internet lectures ([Eff95][MASH99]) or IETF meetings [CaDe92]. The interactivity in these examples is mostly concentrated on the location of the central streaming point. Nevertheless, there might be situations, especially in the lecture example, where a higher

interactivity back to the presenter is desirable, e.g. to allow questions to the presenter.

The need for extensive interaction increases with the provision of more complex scenarios in terms of social rules to be implemented. For instance, for the provision of *Panel Discussions* or *Expert Rounds* dedicated to a certain topic, higher interactivity among the participants of the conference must be supported, e.g. questions from the auditorium.

Similar to normal face-to-face meetings, an agenda and conference announcement might be defined and distributed beforehand. Users might also be invited to join the conference.

Tele-Education

Tele-Education scenarios might be seen as a Lecture scenario with extended functionality. In addition to the plain audiovisual information which is sent out to the participants non-real-time data might be used to emphasize the pedagogical goal of the lecture. For instance, accompanying slides might be provided or an application might be shared among the users in terms of control and display content. For the first case, the content of the slides has to be distributed and displayed on the different sites. Annotations and saving operations have to be supported. Thus, common formats or at least format conversion functionality have to be provided by the system.

In the latter case, the state of an application has to be shared among different sites. Each local change is displayed on each participant's terminal. For interactive learning, the system should support to give the control of the shared application to certain users.

Depending on the scope of the participating students and the economical interests of the providing institution, functionality for restricted access to the conferences, i.e. authorization, and accounting of the received information has to be supported.

The scale of tele-education scenarios varies depending on the goal of the session. It might range from several participants only for small distributed group lectures ([BGL96][Eff95]) to large education scenarios with several tens or hundreds of students similar to university lectures.

Tele-Working

An extended version of the small group project meeting is the tele-working scenario. In addition to the exchange of audio/video streams, project data like pictures, slides, or diagrams are exchanged among the participants. Furthermore, similar to the tele-education scenario, a shared application might be used to commonly work together. Due to the sensitivity of the exchanged data, authorization and authentication functionality is crucial to provide secure exchange. Normally, the scale of the scenario is rather small similar to the size of project teams.

As a summary of the above sketched scenarios, it can be said that conferencing systems have to provide facilities to support multipoint transfer of streaming and non-real-time data. Furthermore, management facilities like invitation and announcement as well as a conference database have to be provided. Crucial for the mapping of well-known real-world interactivity to the distributed case is the provision of functions that implement social rules. But also functionality for authorization, authentication, accounting, and billing is important to protect *intellectual property rights* (IPR) in conferencing scenarios. Furthermore, it is important to provide the conferencing functionality independent from the size of the conference in terms of participants.

It can be seen from this non-formal functionality description that certain functions are common in a wide spectrum of scenarios. As a consequence, this non-formal description above can be used to define requirements, both on service and technical level, for the provision of a generic conferencing service.

2.1.2 Requirements

In the last section, several conferencing scenarios were presented for synchronous collaborative work in distributed systems to describe needed functionality in these scenarios. This section is used to define more formal requirements, both on service and technical level, for a group communication system to enable collaborative work among distributed users. These requirements are used to assess existing systems as well as the service proposed by the author in chapter 3.

2.1.2.1 Service Requirements

As presented in chapter 2.1.1, certain service functionality is requested by conferencing applications even in different scenarios. *Conference Management* facilities are needed as well as *Multipoint Transfer* of streaming and non-real-time data. Furthermore, a basic *Distributed Applications Support* has to be provided together with a pool of *Standardized Applications* which are commonly used in several conferencing scenarios. These four topics are outlined in more detail in the following presenting generic functionality to be provided to conferencing systems to facilitate the development of these applications.

Conference Management

The management of conferences covers four main topics, namely

- *management of the conference environment* : Functionality to *announce* and to *initiate* a conference is needed. For that, advertisement features like a *lookup service* for conferences or a dedicated announcement channel are required. Furthermore, a conference setup functionality is crucial including features to invite other users to a (running) conference or to merge conferences by appending or inviting entire conferences. Additionally, splitting capabilities can be used to form new sub-conferences out of existing ones. In general, sophisticated *reconfiguration* features have to be provided by the management services. The used mechanisms for these features highly depend on the used environment in terms of chosen topology (see chapter 2.1.2.2), i.e. functionality for topologies like simple stars is much easier to provide than sophisticated tree topology management functionality. Furthermore, the conference management should enable *conducted* conferences in the sense that a dedicated user is defined having special management rights, like expelling other users from a running conference or denying the access to a conference.
- provision of a *common conference database* : A common conference database is crucial for the provision of membership information containing commonly used information (like naming and location information) as well as extended, scenario-specific data.
- handle *security aspects* : For the provision of secure and restricted conferences, authorization and authentication have to be provided as well as mechanisms to support secure transfer of user data.
- manage *network resources* : The conferencing architecture should provide facilities to control the network resources, e.g. by providing a defined and guaranteed *Quality of Service* (QoS [Part93]).

Multipoint Transfer

Services for multipoint transfer of user data cover the aspect of secure, location-transparent exchange of user data among several conference members independent from specific networks and endsystems. The services should provide communication patterns which range from one-

to-one to many-to-many communication (see figure 2-1).

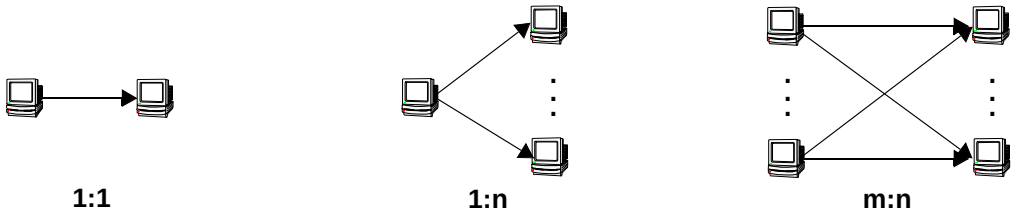


Figure 2-1: *Communication Patterns for Multipoint Transfer*

The management of security aspects is covered by the conference management services. However, the provision of the secure transfer has to be implemented by the multipoint transfer service. For the location transparency of the transfer, a transport layer abstraction has to be provided to address a certain user group in the conference. This transport layer abstraction should also allow to use different kinds of transport facilities depending on the transmitted data type, e.g. streaming protocols for real-time audiovisual data. As a consequence, multipoint transfer on the conference layer should not duplicate transport layer functionality.

The multipoint transfer services should also enable the transfer of user data among heterogeneous endsystems. For that, *marshaling capabilities* [Tan96] should be supported by the service to provide a common transfer syntax of the exchanged user data.

The support of different kinds of networks is shifted to the transport layer when using the facilities of this layer. Thus, the internetworking aspect highly depends on the used transport layer.

Distributed Applications Support

This topic covers features to facilitate the operation of distributed applications. This means in particular the provision of mechanisms to synchronize the state of applications. It enables a coordinated concurrent access to shared resources and data like hard-discs or printers but also to objects or streams. The access to resources can be abstracted by the notion of a *floor* which was introduced by turn-taking psychological studies ([KPMW94][Walk82]) and is defined as "the right to speak in a ...body, e.g. a parliament" [Coll88].

Pure audiovisual conferencing scenario often need just a *loose* floor control, e.g. selecting the current speaker based on the audio volume, as proposed in [Scho96]. For more complex access control patterns, Dommel et al. proposed a floor control protocol in ([DoGLA95][DoGLA98]) with basic operations like *allocate*, *ask for*, and *pass* a floor. These floors might be defined exclusively or non-exclusively. Additionally, with a *member query* operation, it was shown that these operations are sufficient to abstract access control and conventional turn-taking in conferencing scenarios. Thus, this functionality should be provided by a generic conferencing service for the access control in distributed applications.

Standardized Applications

Conferencing applications use the features offered by a generic conference service. Certain application functionality is commonly used in different conferencing scenarios. For the provision of interoperability among applications within a given scenario, a standardization of the application functionality is crucial. This has to be provided by a generic conferencing service to accelerate the development of scenario-specific applications. Relevant widely-used application functionality in the conference context are, among others, *shared whiteboard*, *audio* and *video conferencing*, *multipoint file transfer*, *chatting*, and *shared application functionality*.

As a summary, the following table shows the service requirements to be covered by a generic conferencing service.

Aspect	Features
Conference Management	- management of conferencing environment - conducted conferences - conference database management - security management - network resources management
Multipoint Transfer	- location-transparent addressing scheme - transport layer abstraction - provision of media-dependent transport layers - support of heterogenous networks and endsystems
Distributed Applications Support	- access and floor control
Standardized Applications	- audiovisual functionality - shared workspace - shared application

Table 2-1: *Conferencing Service Requirements*

2.1.2.2 Technical Requirements

Beside the services themselves, a generic conferencing system should fulfil certain technical requirements which are the foundation of protocols in general, namely *robustness* and *scalability*. Furthermore, several realizations of the conferencing environment are outlined with regard to the robust and scalable provision of the services.

Robustness

A crucial issue in conferencing environments is the *robustness* of the system. Features of a conferencing service as well as the implementation of these have to be provided in a robust manner to ensure the acceptance by the users.

On *service level*, the management functionality of the conferencing environment should be offered in the sense that the existing conference status should not be affected when applying any changes on the conference environment during the runtime, e.g. when joining or deleting users, or merging existing conferences. Otherwise resolving mechanisms should be provided. It is worth mentioning that it is not a solution to improve robustness of the services by removing certain *critical* functionality from the service repertoire. A minimal management functionality is crucial to provide natural operations within conferencing scenarios, e.g. join/leave mechanisms without affecting the rest of the conference.

Robustness on protocol level is even more difficult to provide. Communication among distributed users involves a number of different endsystems, configurations, and networks. Lots of components are present in the conference, either directly, e.g. the endsystems, or indirectly, e.g. network components like routers. As a consequence, the protocol mechanisms which are used in these multipoint end-to-end scenarios have to be designed very carefully to provide robustness on the protocol level. Functionality to recover from errors on lower layers, e.g. caused by network components or transport layer failures, is crucial to guarantee a robust conferencing environment. Fault tolerance for certain parts of the conferencing system might be used to improve the robustness of the entire system.

Furthermore, the protocol mechanisms have to be defined and specified correctly using state-of-the-art technology. There are several methods to show the robustness on protocol level in terms of *correctness* of the provided functionality using *testing and validation methods*. These methods can be used to validate a protocol specification either defined as a finite-state automata description (using SDL [Z100a] or LOTOS [ISO8807]) or by using *Message Sequence Charts* (MSC [Z120]). For the first kind of specification, *test cases* can be defined using notations like TTCN [X292] which define the input workload to the system to test. The results are compared to the requirements of the system denoting the expected behavior.

Methods proposed in [AHP96] or [ObKe99] use MSCs for testing a protocol specified by the sequences of messages. Furthermore, formal verification methods like proposed in ([CrRe99] [CrRo99]) might be used to check *deadlock* as well as *lifelock freedom* of the protocol mechanisms in a conferencing environment.

Scalability

The second crucial issue in the provision of a generic conferencing service is *scalability* in terms of conference group size and geographical distribution. The conference management should be able to handle small as well as large conferences in terms of members ranging from few participants to several hundreds or thousands. Thus, the mechanisms for database administration and environment management have to be designed for a larger scale to provide a generic mechanism.

Another sensitive area in terms of scalability is the multipoint transfer of user data. The transport mechanisms for media streaming and reliable data transfer have to be designed to enable efficient and timely delivery of the data to the participating terminals even in large scaled scenarios. Efficiency also includes optimal usage of network resources, e.g. usage of underlying multicast mechanisms. Furthermore, the timing constraints of certain media types, e.g. audiovisual data ([Part93][StNa95]), have to be taken into account for the timely delivery of the data even in large scaled scenarios.

The access control to distributed objects is another critical scalability issue. In general, scenario-dependent access control functionality is implemented using *floor control*. Related work, presented in ([DoGLA95][Er99][GrSt85][IKW98][KPMW84][Lyy90]), pinpoints the importance of *time-constrained* floor control in scenarios with a need for tight control among users and objects independent from the size of the conference. The presented results emphasize the effect of time-constrained floor control on the acceptance by the participating users. Thus, the floor control mechanisms have to be designed for smaller and larger scaled scenarios due the direct impact on the responsiveness of the conferencing system.

It can be summarized that several mechanisms have to be considered when designing a conferencing service suited for small as well as for large scaled scenarios. These mechanisms can be supported by lower layers, e.g. network or transport layer, but have also to be considered on the conferencing layer itself.

Chosen Topology Environment

From a service point of view, the chosen topology for the realization of the services is not an issue which has to be taken into account. But from a scalability point of view, the chosen topology is crucial for the provision of certain services in a scalable manner. Depending on the given scenario, the selection of a certain topology might be useful for a higher scalability, especially if some services, like management functionality, are not needed in the scenario.

In the following, a comparison of two paradigms for conferencing environments are presented,

namely *tightly coupled* and *loosely coupled environments*, outlining the connectivity among conference members. The first request dedicated point-to-point connections among conferencing entities, while the latter do not communicate via point-to-point connection but by using multicast-based exchange of control and user data. It is briefly demonstrated what the advantages and disadvantages of both paradigms are. For the tightly coupled environments, two different topology realizations are outlined additionally, namely *tree-based* and *central server* topologies.

- Tightly coupled vs. Loosely coupled Environments

Conference environment topologies are usually built according to two different paradigms with respect to the coupling of the participating users [HCBO99]. *Tightly coupled environments* build the topology by interconnecting the participating endsystems with point-to-point connections for at least transferring control traffic via this topology. Normally tree or central server topologies are used. For the transfer of user data, multicast capabilities might be used or the control topology might be used as well.

The advantage of using tightly coupled environments is the simple provision of a membership control in terms of a conference memberlist and management functions like admitting or expelling members. Additionally, the restricted access to the conference is easy to implement using central authorization. Furthermore, the implementation of *floor control* mechanisms is simplified when using tightly coupled environments [DoGLA95] due to the stringent routing between the point-to-point connections within this environment.

The main weakness of a tight coupling is the bad scalability in terms of conference size. This is because the number of point-to-point connections increases. As a consequence, the realization of the management for large tightly coupled environments is very difficult, especially for systems with several tens or hundreds of users. Moreover, error recovery mechanisms have to be provided to ensure tolerance in the case of connection errors like node failures and to improve the reliability of the conferencing system. The integration of these mechanisms complicates the design and development of a generic conferencing system with a high reliability of the provided services. Another disadvantage of this environment approach is, that for the support of multicast transport layers, a parallel communication infrastructure has to be established by the conferencing system. Using the same topology, which means to map multipoint to point-to-point traffic, results in an inefficient usage of network resources and therefore in an even worse scalability of the approach.

Loosely coupled environments do not use point-to-point connections, but communicate mostly using multicast transfer. For that, the participants listen to a dedicated multicast group as shown in figure 2-2. The entire control traffic is sent to this multicast group in addition to the user data.

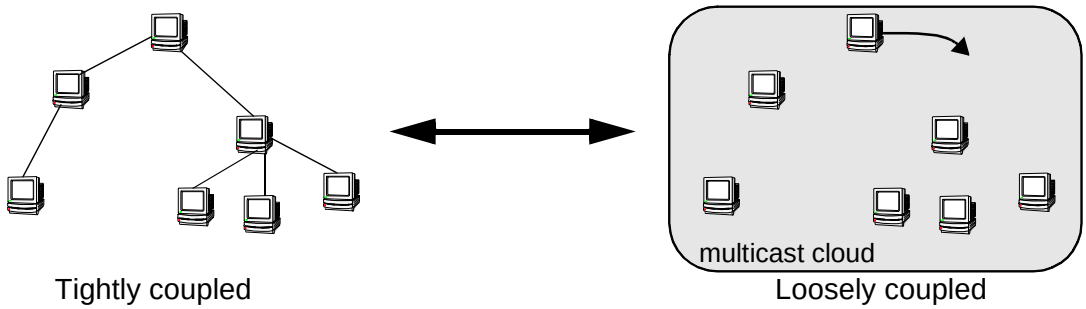


Figure 2-2: *Tightly coupled vs. Loosely coupled Environments*

The main advantage of using these environments is the higher *scalability* in terms of conference size. Due to the usage of underlying multicast techniques, conferences of several thousand users are possible. These systems offer a higher tolerance in the case of temporary failures on network level. Furthermore, complex environment management functionality is not needed, e.g. for managing a tree topology. This functionality is provided by the lower network layer as multicast services like dynamic join/leave of members. Additionally, the same transport medium, i.e. the multicast group, can be used both for control and user data traffic, which simplifies the realization of the system.

But there are several weaknesses when using loosely coupled environments for conferencing. Membership control, e.g. member authorization or conductorship functionality, is hard to implement. Not even a list of current conference members is normally supported in these systems to avoid multicasting the entire list to the multicast group. Furthermore, floor control mechanisms are much harder to design and to implement due to the missing stringent routing of floor control requests. Approaches like proposed in [ADH93] or [HaCr97] use *quorum-based techniques* or *temporary inconsistency* to realize access control to objects in these environments. Or a *central floor manager* can be used to administrate conference floors. But the response time which can be obtained using these techniques is very high in large conferences. Thus, the applicability of floor control services in these environments is restricted to either small or special scenarios (e.g. only a few floor holders).

It can be summarized that using tightly coupled environments enables the provision of membership and floor control, but introduces difficulties in the realization of the mechanisms. As a consequence, providing scalable systems based on a tight coupling is much more difficult to realize. Using loosely coupled environments improves the scalability of the system and therefore the usability of the system in larger scale. But some of the services are either not or only poorly supported within these environments, which restricts the applicability of these environments. As shown in [HCBO99], tightly coupled environments are normally used for scenarios with a tight control of members and resources, while the loosely coupled approach is mostly used for streaming scenarios with no need for membership and floor control.

Due to the defined service requirements in chapter 2.1.2.1 which include conference management and floor control functionality, the focus in the following sections will be on conferencing systems based on tightly coupled environments. However, conferencing approaches which are currently discussed in the Internet are presented in chapter 2.3, which are using loosely coupled environments as well. It will be seen in the proposed conferencing service in chapter 3 that both paradigms might be combined by the notion of loosely coupled *non-SCCS listeners* (see chapter 3.1.3). This will lead to a higher scalable system which also fulfils the service requirements. With that combination of both paradigms, it is possible to support scenarios, where a tight control among a few participants is crucial, but the rest of the conference is coupled loosely. This might be the case for instance in a panel discussion with tightly coupled members participating actively in the conference, and a large auditorium listening passively to the discussion.

- Tree-based vs. Central Server

As mentioned above, point-to-point topologies are used to tightly couple participants within a conference to provide membership and floor control functionality. In many approaches for conferencing systems ([AF91][AnBa93][BeKo98][BrRe98][MaRo97][OAB99]), a simple *central server* approach is used for the control traffic topology. This topology model simplifies the implementation of the conference management functionality, because complex mechanisms for tree management are not needed.

Another approach for tightly coupled environments is building a *tree topology* with a dedicated

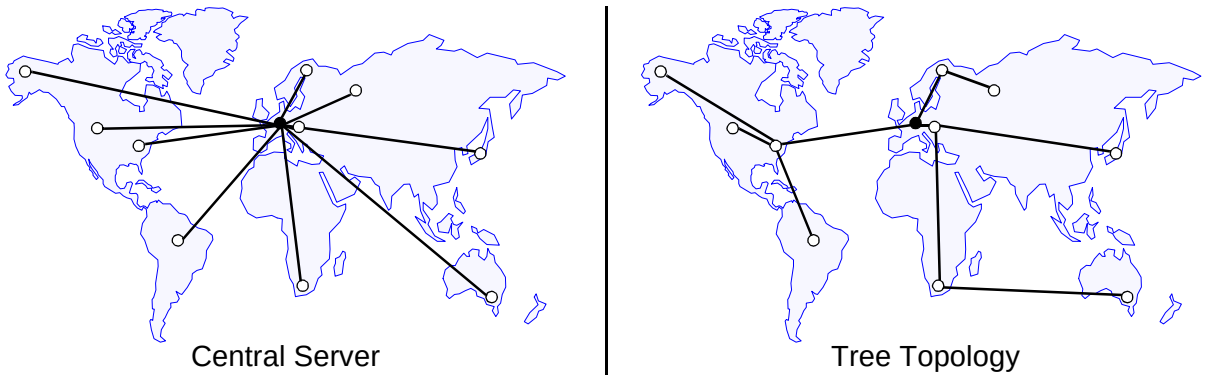


Figure 2-3: *Central Server vs. Tree Topology*

top most node. Standards like the T.120 (see chapter 2.2.3 and [T120]) or the proposed conferencing service in chapter 3 use this approach to build tightly coupled topologies. Due to the more complex topologies which may be built with trees, more sophisticated management functionality has to be implemented. This complicates the design and development of services with the support of these topologies.

But concerning the scalability of the chosen topology in terms of conference size and geographical distribution of the participating endsystems, conferencing services based on tree topologies scale much better.

In terms of *conference size*, the central server model emphasizes the bottleneck of the system if the size of the conference is increased. The number of connections which have to be administrated by this server grows linearly with the number of participating endsystems. Normally, the number of threads in the central server implementation grows as well. Thus, the central server would reach a saturation point of supported endsystems, depending of course on the performance of the central entity. In addition to the bottleneck problem, the central server is also the *single-point of failure* in the system.

When using tree topologies, the central bottleneck can be avoided or at least reduced by developing appropriate mechanisms for the routing of data within the topology. In fact, the proposed conferencing service in chapter 3 uses a newly developed routing scheme to reduce the load in the topmost node. Furthermore, error recovery from failures in the topmost node is easier to provide for a reduction of the single-point of failure problem.

Furthermore, in terms of *geographical distribution* of participating endsystems, a central server approach might lead to problems, especially in terms of *delay*. In figure 2-3, a conferencing scenario is presented with a large, world-wide, geographical distribution of the endsystems. In this example, the central server is located in Europe as a more or less optimal solution for the placement of this server in terms of mean delay for the control data. The right picture outlines a possible tree topology for this scenario.

But consider the case, that most of the communication would take place among "local" users, e.g. in South and North America. Despite this local communication, all requests have to be sent to the central point in Europe, instead of being served in the local area. This effect is emphasized when considering more users in the scenario. When using tree topologies, routing schemes might be used to handle requests locally. If the tree is built according to the geographical distribution (e.g. shown in figure 2-3 right), this might lead to smaller delays for control requests in the conference. Hence, this results in a higher responsiveness of the system.

In this thesis, conferencing systems for tightly coupled environments are presented using tree

topologies instead of following the central server approach. But it is obvious that these systems also support central servers as a degenerative form of a tree.

2.1.3 Summary

This section presented a brief survey of conferencing application scenarios in the area of tele-conferencing, tele-education, and tele-working. The last two scenarios can be seen as more sophisticated extensions of plain tele-conferencing scenarios by adding functionality like distributed editing of documents or controlling shared applications. Many conferencing systems have already been realized to provide more or less specific functionality for specific scenarios.

As a common result from the overview of conferencing scenarios, several key features of conferencing applications were outlined as scenario-independent. This led to the definition of service requirements for a generic conferencing service provision covering several key aspects, namely *conference management*, *multipoint transfer* of user data, *distributed applications support*, and provision of *standardized commonly used functionality*. Beside these service requirements, some technical points were outlined which are of interest when designing and developing conferencing services. *Robustness* and *scalability* of provided services and mechanisms were pinpointed as crucial issues for the development of a conferencing service. Furthermore, endsystem coupling and topology aspects were discussed in this section under the constraint to fulfil the required services in a scalable manner.

The defined requirements are used as a basis for the presentation as well as the assessment of existing conferencing systems which are presented in the next sections. Furthermore, the approach proposed by the author in chapter 3 is designed and assessed with respect to the defined service and technical requirements.

2.2 ITU H.32x

The *International Telecommunication Union* (ITU) initiated the standardization for audio/ video and data equipment enabling the cooperative communication among several users in the late 1980s. This standardization effort led to the ITU-T recommendations of the H.32x series. Each of these recommendations is an umbrella standard consisting of several other protocols for a certain functionality. The first standard suite H.320 [H320] was finished in 1990 for the usage in narrowband ISDN systems. In 1995, this standard was adapted to broadband ISDN systems leading to the definition of the H.321 series [H321]. The standard suites H.322 [H322] and H.323 [H323] were defined for *Local Area Networks* (LANs) which provide quality of service (QoS) guarantees or not respectively. Due to its IP foundations, the standardization efforts for the H.323 standard increased in the past two years to emphasize its importance for the usage in the Internet.

In the following chapter, the H.323 is introduced in more detail, starting with the service model of an H.323 system. After that, the architecture of the system is presented. Some of the architectural views are common to the other H.32x systems, some are special for this kind of environment. Additionally, the data conferencing part of an H.32x system, standardized in the ITU-T T.120 recommendation series [T120], is presented in more detail providing data conferencing and multipoint data transfer functionality. The section will conclude with a short assessment of the H.323 system regarding the requirements defined in chapter 2.1.2.

2.2.1 Service Model

As mentioned in the introduction, H.323 is defined as an umbrella standard specifying the components to be used within an H.323-based environment. For the different components, existing standards are used providing the functionality, i.e. the services of an H.323 system. Due to the standardization history of the H.32x protocol series, the provided services in an H.32x system are divided in services for audio/video and data conferencing. As a result of this separation, some functionality is implemented twice, especially for conference management and multipoint transfer of data, while other functionality is only defined for a certain part, either for data or audio/video conferencing. In the following, the services of an H.323 system are introduced based on the requirements defined in chapter 2.1.2.

2.2.1.1 Conference Management

The H.323 provides conference management functionality for audio/video conferences using the *call signaling* functionality of H.245 [H245]. This standard provides *call set-up* and *call transfer* of real-time connections. Multipoint real-time conferences are also supported, but this requires additional equipment in the H.323 environment (see chapter 2.2.2). Users may join or leave existing conferences. The provided functionality is very similar to modern telephony equipment, which results from the origin of the standardization as a basis for video telephony.

For the data part of a conference, the conference management of the T.120 recommendation [T120] is used (see section 2.2.3.6). This standard suite provides the establishment of data *conferences* (e.g. for shared whiteboard or shared application functionality), either as a conducted or unconducted conference. Each T.120-based end system can be a member of one or more conferences defining the highest level of a communication relation. Each conference consists of one or more *sessions* in which several *application protocol entities* (APEs) are grouped.

Existing conferences may be merged leading to a release of existing resource identifiers in one of the merged conferences. More sophisticated reconfiguration functionality is not implemented. Queries for existing conferences on an endsystem are also supported. A domain-wide lookup service for conferences is not provided. The T.120 also includes a conference database of all members, applications, and resources in the conference.

The set-up for a data conference is incorporated in the call set-up procedure of an H.323 conference using the functionality of the H.245 signaling. Data conferences within an H.323 system can be set-up independently from audio/video conferences. For the provision of secure conferences, the H.235 standard [H235] is used. The H.323 also supports address translation between aliases and IP addresses and bandwidth management within a LAN for controlling the network bandwidth e.g. by restricting the number of established conferences.

2.2.1.2 Multipoint Transfer

For multipoint conferencing for audio/video, the H.323 provides *centralized* conferences using the functionality of a central entity, called *Multipoint Communication Unit* (MCU, see chapter 2.2.2.4). This device performs the audio mixing, video switching, and re-distribution of the resulting combined audio/video stream. *Decentralized* conferencing, using multicast technology of the underlying network, is supported for the transmission of the audio/video data. The control processing of the conference, e.g. selection of the video stream, is still located in the centralized MCU using the control channel of the H.245 protocol [H245]. But processing the incoming audio/video stream has to be implemented by the terminals themselves. H.323 also

supports *mixed* conferences consisting of a centralized and decentralized part.

For the data part of an H.323 conference, the T.120 recommendation provides a *channel* concept as an abstraction for multipoint transfer. These channels offer a flexible concept for addressing different groups of receivers within a conference. Three different kinds of channels are supported, namely *user identification channels*, *public channels*, and *private channels*.

Users are identified by a user identifier which corresponds to a unique channel number. This channel identifier serves as an address for point-to-point communication within the domain, e.g. for notification of detaching other users.

Multipoint channels are multi-member channels. These channels can be used to send data to a subset of users within a conference. Send and receive access to individual multipoint channels may be controlled by a *private channel* mechanism. A user may convene a private channel, becoming the *private channel manager* of this channel. The private channel manager may invite other users to join the channel or force a user to leave the channel. The admitted users may join a dedicated channel. *Public channels* may be joined and created by any member of the conference. By joining a combination of channels, a user can select to receive messages sent to these channels and ignore message sent to the other channels. An application does not need joining to a channel to send on it, but must be member of it to receive data. Channel membership is maintained in a distributed fashion.

It is not standardized how the channel concept for data conferencing is mapped to the underlying network. Usage of multicast technology is not mandatory. Different *network profiles* are defined to provide a wide range of possible networks to be used, ranging from ISDN, B-ISDN (ATM) to IP networks.

Two different modes of multipoint data transfer are provided by the T.120 using the above described channels. When using the *simple send data* service, data from different senders may arrive in a different order at each receiving site, because the data is sent via the shortest route within the underlying communication system. *Uniform send* data transfer is necessary when data is being sent simultaneously from several senders, but must be received in the same sequence by all receivers.

Furthermore, a simple priority model for data transfer is supported. Control data is transmitted with highest priority. Three additional priorities are possible.

2.2.1.3 Distributed Applications Support

For the audio/video part of an H.323 system, distributed conferencing is supported by synchronizing several audio/video streams if an MCU (see chapter 2.2.2.4) is available. For this, the packetization functionality of the H.225.0 [H225] is used. Two video conferencing modes are standardized, which are *speaker-selection* (the active speaker is selected) or *conducted mode* (a conference conductor selects the active video stream). Both modes are implemented in the MCU entity providing appropriate mixing, coding, and synchronization functionality.

For the data conferencing part, the notion of *tokens* is used to realize synchronization between distributed applications for the *floor control* [DoGLA95] (see chapter 2.1.2.1). Tokens can be allocated *exclusively* (*grabbed tokens*) or *non-exclusively* (*inhibited tokens*). It can be asked for the status of a token. Furthermore, the token can be given to another user by the (exclusive) owner of the token, e.g. after asking for it (*token please*). It is not possible to ask for a list of current token holders.

It can be seen that simple floor control operations like *floor-asking* and *floor-passing* are pro-

vided for the data part. If the floor control functionality should be used for the audio/video part of the H.323, a parallel data conference has to be established.

2.2.1.4 Standardized Applications

The main focus of the H.323 standardization is the definition of applications which are commonly used in conferencing scenarios. For that, standard application protocols are defined for the usage within an H.323-based environment.

For the audiovisual part, the standard defines codecs, control, and delivery standards in point-to-point or multipoint scenarios (see chapter 2.2.2 for further details). Furthermore, a *capability exchange procedure* is integrated in the conference set-up to negotiate a common set of functionality to ensure interoperability among different H.323 systems.

For data conferencing, the T.120 defines several standard applications which are *still image transfer* (shared whiteboard), *multipoint binary file transfer*, *shared applications*, and *text chat*. All these standard applications use the service functionality of the data conferencing part of an H.323 system. As a consequence, these application protocols can be easily adapted to systems providing the same service model of the underlying data conferencing system.

2.2.2 Architecture

The H.323 defines the technical requirements and the components for audiovisual and data conferencing equipment in LANs which do not provide a guaranteed quality of service to provide the services described in the previous section. The standard is an *umbrella* recommendation referring to other ITU-T standards for the different components of an H.323 entity. Due to

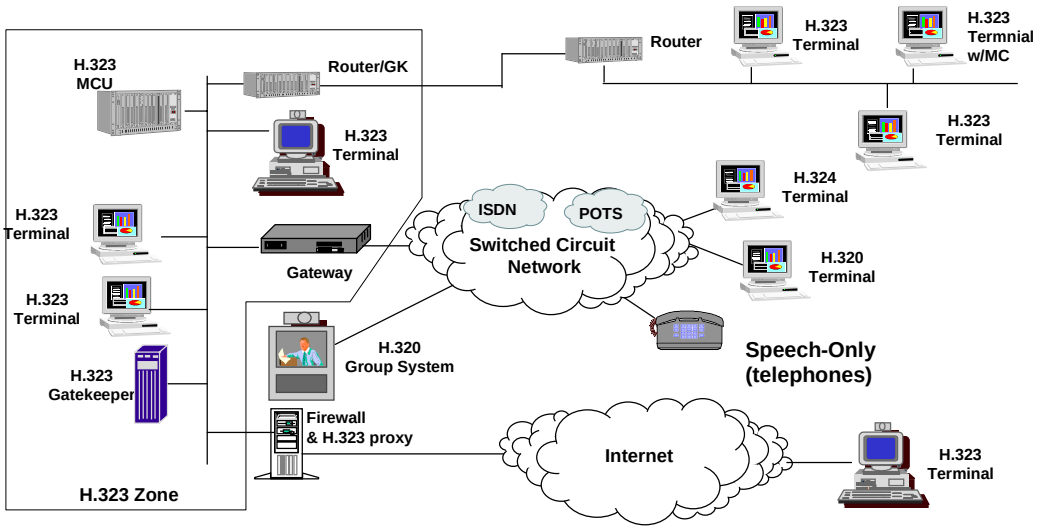


Figure 2-4: *H.323 Environment*

the history of the H.32x recommendations, an H.323 system consists at least of terminals, connected to the network, providing a minimal functionality. Adding other components enables the provision of different services, e.g. the provision of multipoint video conferences or the connectivity to other H.32x systems. Figure 2-4 shows the different entities of an H.323 environment, which are explained in the following.

2.2.2.1 Terminal

The terminals within an H.323 environment are the endpoints of the communication providing a real-time two-way communication as the minimal functionality of an H.323 system. The H.323 recommendation defines the technical requirements in terms of supported components within an H.323 terminal. The components are divided in *audio*, *video*, *data*, and *control*.

Media	Required	mandatory recommendation	optional recommendation
Audio	Yes	G.711	G.722, G.723, G.728, G.729
Video	No	H.261	H.263
Data	No	T.120	-
Control	Yes	H.245, H.225.0	RAS

Table 2-2: *Supported Media Types of an H.323 Terminal*

Table 2-2 shows the different media types and the corresponding ITU recommendations which define the appropriate functionality. It can be seen that for the mandatory audio transmission lots of different codecs are supported. Furthermore, the optional video transmission part allows several video codecs supporting different formats. For the optional data part (like shared white-board or shared applications), the T.120 standard is used (see section 2.2.3).

During setting up a connection using an H.323 terminal, the capabilities of the connecting terminals are exchanged and the minimal supported *capability set* is chosen for the connection. For that, the second mandatory component of the terminal is used, defined in the H.245 standard [H245], responsible for control functionality. As a second control component, the H.225.0 [H225] is used for media stream packetization and support of the *Real-time Transport Protocol* (RTP [RFC1889]). Figure 2-5 shows the different components in the H.323 terminal.

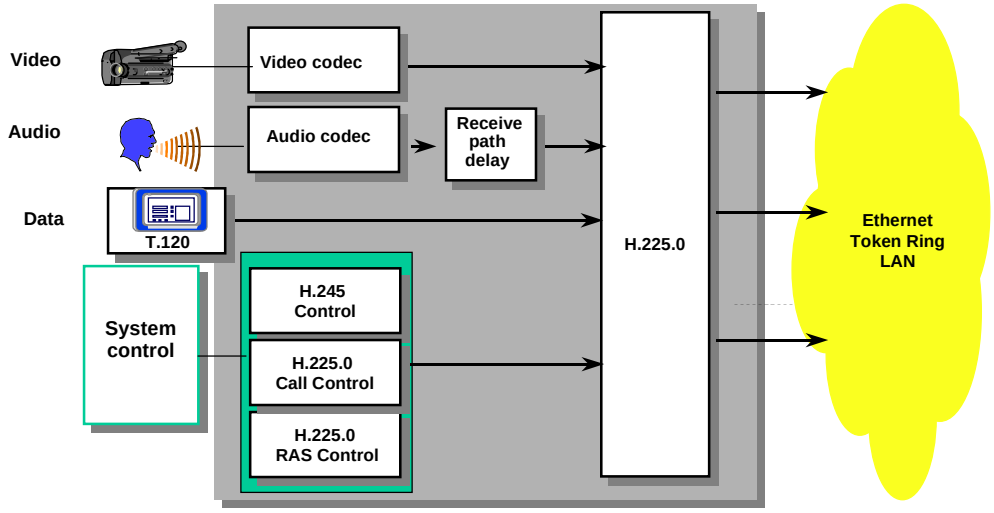


Figure 2-5: *Components of an H.323 Terminal*

2.2.2.2 Gateway

The purpose of an H.323 gateway is to reflect the capabilities of an H.323 endpoint to other

H.32x systems and vice versa. This is realized by translation functionality between transmission formats (e.g. H.225.0 to H.221 [H221]) and communication procedures (e.g. H.245 to H.242 [H242]). In addition, the gateway also performs translation procedures between different audio and video codecs. Gateways are located at the edge of the LAN environment to enable connectivity to Wide Area Networks (WANs) or ISDN-based H.32x systems. If there is no connectivity to these systems, the gateway functionality is not needed in an H.323 system, because H.323 endpoints may communicate directly with other H.323 systems. However, it is possible to transparently connect two LAN-based H.323 systems, bypassing some routers or a low bandwidth link. This gateway functionality is not provided by H.323 gateways.

Moreover, gateways provide multipoint control functionality, e.g. by acting as an MCU at the H.323 side or as a T.120-based data provider in the system. Gateways are normally registered with the H.323 gatekeeper to indicate the available functionality. Many capabilities of an H.323 gateway are left to the designer and vendor of the gateway, e.g. the number of supported H.323 terminals at the gateway is not standardized.

It can be seen that the gateway technology allows an H.323 system to extend the connectivity to other H.32x-based systems to provide a larger basis for collaborative working.

2.2.2.3 Gatekeeper

An H.323 gatekeeper plays a managing role within an H.323 system. The collection of H.323 equipment managed by a single H.323 gatekeeper is defined as the *H.323 zone* (see figure 2-4).

Gatekeepers provide two important housekeeping functions which help to preserve the integrity of the network. The first one is the *translation* of addresses between LAN aliases for terminals or gateways and IP addresses using a simple registration mechanism.

The second function is the *management of bandwidth* within the network. For instance, this can be realized by limiting the maximum number of conferences in the network or the streaming bandwidth for each conference based on managing the bandwidth. It is outside the scope of the standard how to determine the available bandwidth in the system.

A gatekeeper is optional for an H.323 system. However, if a gatekeeper is present, the services *address translation, admission control, bandwidth control,* and *zone management* are mandatory. Furthermore, a gatekeeper may provide optional services, namely *call control signaling*, *call authorization*, *bandwidth management,* and *call management*. For further information of these services, see [H323].

2.2.2.4 Multipoint Control Unit (MCU)

The MCU within an H.323 environment consists of two functional components. The first one is the *Multipoint Controller* (MC) being responsible for handling H.245 negotiations between terminals during set-up of the conference to define a common capability set of the participating systems. The MC does not deal with media streams. This is done by one or more *Media Processors* (MPs) which are responsible for audio mixing and video switching based on the chosen policy (speaker selection or conducted mode). The functionalities of the MC and the MPs can be realized in dedicated hardware or coexist in other elements like an H.323 terminal.

2.2.3 Data Conferencing - T.120

The data communication part within an H.323 system is provided by a parallel data conferencing system defined in the T.120 recommendation [T120]. Similar to the H.323, this is an umbrella standard consisting of several other standards providing the appropriate functionality.

In chapter 2.2.1, the service model of an H.323 system was presented containing also the services for data conferencing. In addition to the service model, this chapter outlines the system model of a T.120-based system enabling multipoint data conferencing. For that, the different components of a T.120 system are presented. Two of the components, which provide the conference management, multipoint, and distributed applications support functionality, are depicted in more detail.

2.2.3.1 System Model of T.120

The T.120 system model is comprised of the *communication infrastructure* and the *application protocols* using this infrastructure. In the T.121 recommendation [T121], an *application template* is defined for the design of T.120-compliant applications, which is shown in figure 2-6. The grey shaded part depicts the communication infrastructure which is composed of three components: *Generic Conference Control* (GCC), *Multipoint Communication Service* (MCS) and *Network Profiles*. It provides multipoint connections with reliable data delivery within a *conference* context. It can accommodate multiple independent applications using the same multipoint environment within a single conference. Multipoint connections are mapped to the underlying communication infrastructure which can be any combination of circuit switched telecommunication networks and packet-based LANs.

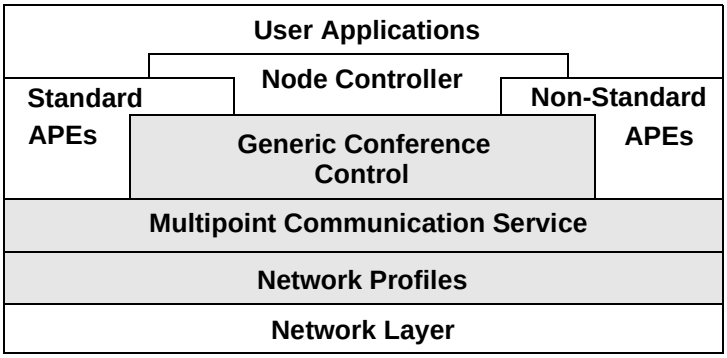


Figure 2-6: *ITU Protocol Suite T.120*

In the following, a short overview of the different components of the system model is given.

2.2.3.2 User Applications

User applications are not within the scope of the T.120 standard series. They are assumed to be *multipoint aware* and designed to use the services of the *Generic Conference Control* (GCC) and the *Multipoint Communication Service* (MCS) as the main parts of the communication infrastructure. Furthermore, the applications may use any combination of standardized and non-standardized protocols to communicate with peer user applications.

2.2.3.3 Application Protocols

Application protocol entities (APEs) comprise a set of services for commonly used application functionality and the provision of associated protocol data units for peer-to-peer(s) communication. These may be proprietary (*non-standard*) or *standard* application protocols. The T.120 series includes a set of application protocols designed to support facilities for data conferencing. These protocols define a minimum requirement in order to ensure interworking between different implementations like *multipoint binary file transfer* (T.127 [T127]), *audiographic protocol for still image viewing and annotation* (T.126 [T126]), or *application sharing* (T.128 [T128]). Furthermore, a simple *text chat* (T.134 [T134]) functionality is defined. An application protocol is divided into two functional components: the *Application Resource Manager* (ARM) providing functionality common to all protocols and the *Application Service Element* (ASE) providing the application specific functionality. The *generic application template* (T.121 [T121]) proposes an implementation framework for T.120-based applications using provided application protocols.

2.2.3.4 Node Controller

In a T.120-based system, the *Node Controller* provides the T.120 management function at a terminal or *Multipoint Control Unit* (MCU). It is the role of the Node Controller to issue the primitives to the GCC which starts and controls the communication session. The node controller is not within the scope of the T.120 recommendations, but the interfaces to communicate to the GCC and other standardized elements are defined by the corresponding service primitives in the appropriate standards.

2.2.3.5 Multipoint Communication Service (T.122/T.125)

The main part of the T.120 communication infrastructure is defined in the T.122 standard [T122] as a generic service designed to support highly interactive multimedia conferencing applications. It supports full-duplex multipoint connections with a wide variety of addressing patterns with a provision of several networks types. A *channel concept* is introduced as an addressing scheme for multipoint data transfer (see chapter 2.2.1). Additionally, services for synchronization of distributed applications are provided to realize floor and access control in group communication scenarios. For that, a *token* concept is provided (see chapter 2.2.1). Furthermore, two different modes of data transfer (*uniform* and *simple* mode) are realized by multipoint routing within MCS, and priorities for user data are supported. The MCS assumes a transport protocol which is conform to an OSI TP4 [X224] to enable network independence. Different network profiles are defined in the T.123 standard [T123] (see chapter 2.2.3.7) providing TP4 functionality over different networks.

The mechanisms used in the MCS to realize the services as mentioned above are defined in the T.125 recommendation [T125]. This recommendation defines the MCS protocol mechanisms by introducing an informal description of the mechanisms combined with a formal definition of the exchanged protocol data units together with an SDL [Z100a] specification of the protocol. In the following, the system model and main mechanisms are described informally.

System Model of T.122

The MCS provides its services, namely multipoint communication and token management, within an closed administration area called *MCS domain*. Applications attach to a domain for

using the services of the MCS.

For the realization of the services, *MCS providers* are located within this domain offering the services of the MCS to applications which are attached to one of these providers. Domains are identified locally via a unique *domain selector*. The MCS providers in a single domain are interconnected hierarchically as a tree topology via *MCS connections*. Each MCS connection is realized via reliable transport connections. The uppermost provider of the tree topology is called *MCS Top Provider*. It is possible that a dedicated MCS provider is located in several domains. However, inter-domain communication is not supported due to the closed character of an MCS domain. Figure 2-7 shows an example for the domain concept of the MCS and its realization as a tree topology.

Domain Management

MCS provides simple functionality to manage the MCS domains. *Domain parameters* are defined for each MCS domain. These parameters are set during creation of the domain and are exchanged among the providers when creating a new MCS connection, i.e. when adding a new provider to the domain. For instance, the domain parameters include values for the maximum height of the domain tree, maximum number of tokens, channels, and users. Furthermore, when connecting two domains between MCS providers via MCS connections, MCS performs a *merging* operation of these domains. Within this operation, token or channel numbering conflicts are simply resolved by releasing conflicting resources in the lower domain. More sophis-

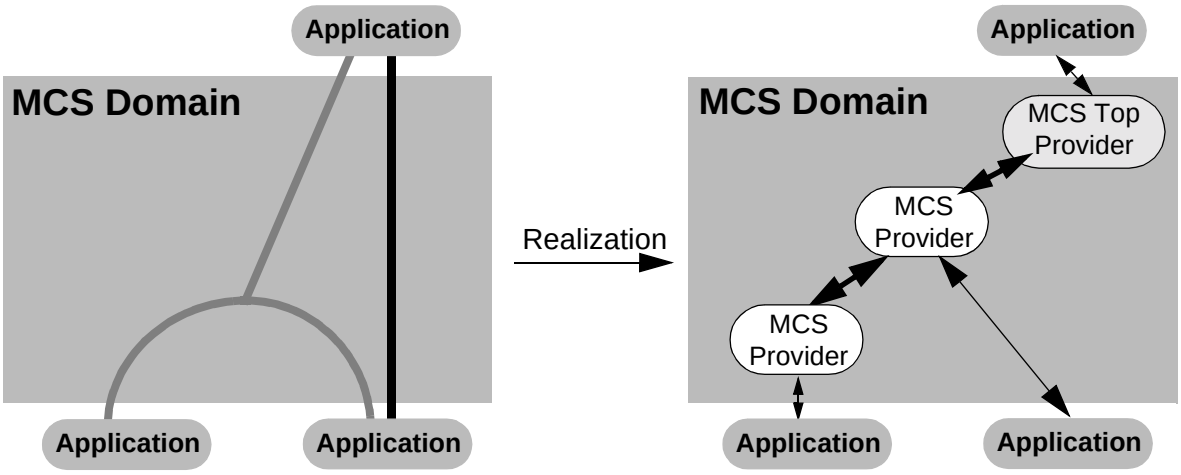


Figure 2-7: *Domain Concept of the MCS*

ticated domain management features like splitting or re-ordering of providers are not provided.

Due to the mapping of GCC conferences to a dedicated MCS domain (see chapter 2.2.3.6), this results in very simple conference management functionality with respect to topology administration.

Resource Management

Application resources, for instance hard discs or files, are not within the scope of MCS. Resources in the context of MCS are *tokens* and *channels*. Tokens are mostly used for application state synchronization or floor control [DoGLA95], while the latter are used for addressing of multipoint data. Channels are allocated in the topmost provider if the requested channel is not yet in use.

Otherwise, the channel request is routed to the first provider with routing information for this channel and confirmed by this provider. Administration requests, e.g. to admit other users, are routed based on the user identifier given in the request.

The token resources of MCS are centrally administrated in the MCS top provider. As a consequence, each allocation request for a token is routed upward until it reaches the top provider to ensure consistency of the identifiers. But also further requests, e.g. inquiring the token holder or testing the status of the token, are routed to the topmost provider. This results in a high response time for token requests and therefore also for the floor control within the T.120. As a consequence, the scalability of the floor control functionality is restricted with respect to the conference size.

Multipoint Data Routing

After establishing a domain and attaching users to the domain, channels can be joined and data might be transferred in a multipoint fashion using the *simple send data* or *uniform send data* services for multipoint delivery of data to multiple sites.

Simple send data from different senders may arrive in different order at each site, because they are transferred via the shortest route up and down the MCS tree of the domain. For routing user data, the channel information in the MCS providers is used.

Uniform send data transfer is necessary when data is being sent simultaneously from several senders, but must be received in the same sequence by all receivers. All requests are sent to the MCS top provider. From there, the packets are dispatched in the same order to all receivers, including the sender, if it is a member of the destination channel.

It can be seen that both data transfer services use the MCS tree for routing the data. Hence, the multipoint transfer is mapped to point-to-point data transfer within the tree on the conferencing layer instead of using transport layer functionality. In annex A of the T.125 standard [T125], a *multicast adaptation layer* is proposed, located between the MCS layer and the corresponding network profile (see figure 2-6), for the usage of multicast on the transport layer. For that, the multipoint transfer is mapped to corresponding multicast groups. But in the protocol specification, there is no separation for channels with or without mapping to multicast groups in the MCS provider. As a consequence, the multipoint data is handled in the old way (mapping to point-to-point, i.e. duplicating packets appropriately), and the adaptation layer has to handle multiple data copies for the same multicast group. This is very inefficient, because the internal load of the provider remains the same. At the time of writing this thesis, there is no implementation of this adaptation layer available.

MCS combines flow control on the individual connections into a global domain-wide flow control to adapt the sending rate to the receiving rate. The throughput of a domain may be limited by its slowest receiver. This throughput is negotiated during domain establishment, but the mechanisms are neither standardized nor realized.

2.2.3.6 Generic Conference Control (T.124)

Within a teleconference, there are communication relations between applications of remote systems. The administration of these relations is done by generic functions of the *Generic Conferencing Control* (GCC), defined in the T.124 standard [T124]. It provides functions for management of conferences and definition of the communication relations.

Each T.120-based endsystem (either terminal or MCU) can be a member of one or more *conferences* defining the highest level of inter-application communication relation. Each conference

consists of one or more *sessions* in which several *application protocol entities* (APEs) are grouped together. Within GCC, each conference is associated with a unique *MCS domain* (see

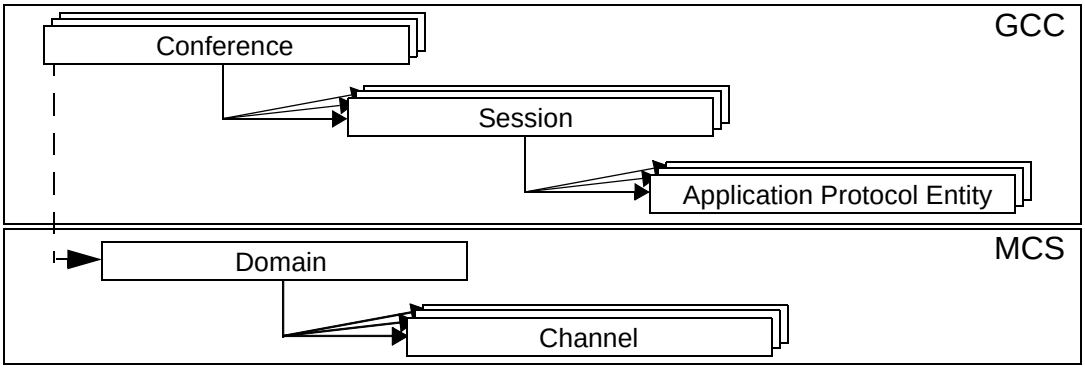


Figure 2-8: *Communication Relations within T.120*

chapter 2.2.3.5). The communication relations are realized by mappings to corresponding MCS channels. In figure 2-8, the mappings between GCC and MCS concepts are illustrated.

GCC provides several functions for the conference management, e.g. asking for existing conferences, creating new conferences, or admitting other endsystems. Mechanisms exist to signal changes of the conference structure, e.g. when deleting an APE. GCC supports the negotiation of *capabilities* of the APEs of a conference, password protected access of a conference, and a generic administration of resources. Furthermore, GCC implements the concept of *conducted* conferences where a defined conference member controls the admittance and other functionality of the conference. The conductorship of a conference might be given to another member by the current conductor. For this functionality, the underlying MCS token functionality is used.

The GCC does not provide conference reconfiguration mechanisms like re-ordering of endsystems in the logical structure, because the used MCS does not provide this functionality. Furthermore, GCC does not implement a global directory of conferences. Hence, the conference provider must be known when looking for a specific conference.

As a second service, GCC implements a conference database as a hierarchical database within *each* endsystem of the conference. With this design decision, the duration of read requests for database entries is minimized, while the duration of write requests is maximized, because consistency of the database content has to be ensured. The GCC database consists of a the following parts:

1. *conference roster*, containing information of conferences and their members
2. *local application roster* and *conference applications roster*, holding information about created APEs on an endsystem, the conferences where an APE participates, and the capabilities of an APE
3. *application registry*, where the resources used by a conference are stored, i.e. tokens and channels of MCS

The conference roster and conference applications roster exist in two versions. On the one hand, there is the *full version* which contains complete information of endsystems, APEs, and sessions of a conference. On the other hand, there is a *subhierarchy version* containing the information of the endsystems, APEs, and sessions in the endsystem's subtree.

The database is distributed among all endsystems, when an entry within the database is changed. In the simple version of the GCC, the entire database is refreshed if only one entry was

changed (*full refresh*). This results in a high load for conference data exchange. As an extension of the proposed full refresh model, a more sophisticated database exchange model is proposed by defining *delta refreshes* (sending only the changed entries of the database). Furthermore, different types of nodes are defined which differ in the reception of conference data to reduce the network load.

It is common to both conference database exchange mechanisms that for the transfer of the content the MCS data transfer functionality is used. Thus, the efficiency of the database exchange depends on the implementation of this functionality (see chapter 2.2.3.5). If the multipoint to point-to-point mapping is implemented in the used MCS stack, the entire database is exchanged using the tree routing of the MCS. This results in an even less efficient exchange of the content in terms of response time and generated load on the network (see chapter 2.2.4).

2.2.3.7 Network Specific Data Protocol Stacks (T.123)

This recommendation which defines protocol stacks for terminals and MCUs specifies network-dependent aspects of the T.120 protocol suite as profiles for each network. Each profile defines a set of protocols that provides transport functionality of the OSI reference model. Network specific profiles are defined for ISDN, CSDN (*circuit-switched digital networks*), PSDN (*public-switched digital networks*), and PSTN (*public-switched telephone network*). The scope of the recommendation also includes B-ISDN and LAN. The handling of audio and video signals in a multimedia conference is not part of this recommendation, because the T.123 only defines protocol stack profiles for the T.120 data conferencing part.

2.2.4 Experiments and Results

Recently, several H.323-based systems have been shipped out by different vendors. Some of them implement the audiovisual part only, while other products offer the entire functionality of an H.323 terminal. At the time of writing this thesis, the availability of other H.323 components like gatekeepers, gateways, or MCUs increases slowly. But unfortunately the price of these components rapidly reaches several thousands of dollars. As a consequence, no experimental results were gathered for the audiovisual part of the H.323. Instead, the mechanisms of the data conferencing part were evaluated, standardized in the T.120 protocol suite.

As mentioned in chapter 2.2.3.5, the MCS standard was defined by a formal protocol specification using the *Specification and Description Language* (SDL [Z100a]). The author of this thesis evaluated this specification and the used mechanisms by translating the protocol specification to C code with an integration of marshaling functions and a mapping to TCP/IP as the underlying transport layer protocol (see [HeTr97]). The multicast extension was not evaluated, because during the time of the investigation, the proposal for this extension was ongoing.

Several scenarios and experiments were performed with the translated protocol stack of the MCS, published in [HeTr97]. Domain creation and merging functionality was investigated together with token requests and data transfer performance. The main results of this investigation can be summarized as follows:

- Domain creation and extension within MCS is a very heavy weight mechanism with low functionality. Due to an error in the specification, the performed merging operation results in a complete release of resources (tokens, channels and users) within the lower of the two subconference. Thus, the functionality is not usable in a proper way.

- Token experiments were used to gather results for the service time to forward resource requests within an MCS provider (see [HeTr97]). The service time depends on the location of the node within the topology (topmost node, leaf node, or intermediate node), the network load, and the processing power of the MCS provider. The measured mean service times, shown in table 2-3, will be used as a basis for the performance evaluation of the proposed protocol in chapter 3. The values were measured on a Sun Sparc 5 with low network load [HeTr97].

Location of Node	Average Service Time
Top Node	2.0 ms
Intermediate Node	1.8 ms
Leaf Node	1.4 ms

Table 2-3: *Service Time to forward Token Requests in the MCS Implementation*

- The data transfer showed the expected results concerning the resulting bandwidth requirements when using the multipoint to point-to-point mapping. The measurements were performed in a multicast-capable Ethernet-LAN. With an increasing number of participants, the required bandwidth for the transfer increased linearly due to the mapping to single point-to-point connections [HeTr97]. This impressively shows the inefficiency of the multipoint service of the standard.

In addition to the experiments with the MCS implementation, the GCC was evaluated on a functional basis. Beside some inconsistencies in the standard concerning non-defined resource identifiers and non-existing functionality requested from the underlying MCS stack, the database exchange mechanism was outlined as the main weakness of the GCC in terms of scalability of the proposed mechanisms [HeTr97].

As presented in chapter 2.2.3.6, the decision was made to hold information about conferences in an internal database which is kept as a copy in each participating GCC system. Hence, the overhead of changing the database is maximal, since every copy must be kept consistent on every endsystem. In addition to this design decision, an inefficient replication protocol was chosen in spite of more sophisticated mechanisms with better performance. The inefficiency of the data administration of T.124 results from the exchange of the complete conference database when adding a new endsystem, although the information is stored in all providers except the new one. Together with the inefficient multipoint transfer of the MCS (by using point-to-point connections), this results in a high load for the administration of the conference database, which leads to poor performance in higher scaled conference scenarios (a few tens of participants).

2.2.5 Assessment of H.323

The H.323 umbrella standard was defined to enable audiovisual conferencing among several endsystems including data conferencing functionality. This section assesses the services and mechanisms with respect to the requirements given in chapter 2.1.2.

Conference Management

The conference management functionality of an H.323 system is divided into audiovisual and data conferencing. For the first one, point-to-point and multipoint conferencing is supported. Due to the tightly coupled character of the chosen system model and the telecommunication

origins of the standard, a call-control procedure was chosen which is very similar to telephony equipment. Several call-control features like call-setup, call-forwarding, and call-extension are implemented. Due to the centralized design of the call control and the complex control exchange procedures, high scalable audiovisual conferences in terms of participants are not supported by this system. But extensions for loosely coupled environments with a more light-weight conference setup are proposed in the H.332 [H332] using IETF protocols like SIP (see chapter 2.3.2). For the provision of secure conferences, an encryption standard is used.

When using a Gatekeeper instance, a network resource management in terms of bandwidth is implemented by H.323, e.g. by restricting the number of established conferences. But this does not lead to the provision of guaranteed QoS.

For the data part of the H.323, a tree-based conference management approach is chosen. Simple management functionality like conference setup and local conference inquiry is standardized. Functionality to reconfigure the conference tree topology is not defined, not even a dynamic deletion of providers from the conference tree.

A conference database is implemented by the GCC standard of the data conferencing part. A very inefficient replication mechanism for this database restricts the applicability of the approach. Improvements of this scheme were proposed but have not yet been realized.

Both conference management facilities, for audiovisual and data conferencing, are incorporated in a system-wide conference setup procedure. It can be seen that this setup procedure was first separated for audiovisual and data part of the system and then again merged in a central conference setup. This leads to a heavy-weight conference management part of the H.323 system. For that, some proposals for a more light-weight management of an H.323 session were defined.

Aspects like conference announcement, billing, and a directory service are currently under discussion, but are not yet integrated in the standard.

It can be summarized that the complex call setup procedure for the audiovisual part and the inefficient database replication protocol of the T.124 standard are not well suited for conferences with a large number of endsystems. Furthermore, the separation of the conference management for audiovisual and data conferences leads to different concepts for both parts. This results in a heavy weight overall management structure with only basic functionality.

Multipoint Transfer

Similar to the conference management, the multipoint aspect is separated for audiovisual and data conferencing as well. Thus, different concepts are used for both traffic types.

For the audiovisual conferencing, the notion of *Multipoint Communication Units* (MCUs) enables multipoint transfer of audiovisual data. But due to the high technical requirements, e.g. mixing, transcoding, and switching, the costs for this equipment are very high (several thousands of dollar), so normally these components are not used. Furthermore, the centralized approach enables audiovisual conferencing only in small scenarios, i.e. few participants. With the decentralized version, multicast functionality of the underlying network layer is used, but the control of the multipoint stream is still realized by a central multipoint control entity.

For the data conferencing part, the *Multipoint Communication Service* (MCS) provides any pattern for multipoint data transfer. Furthermore, private as well as public addressing is supported. But unfortunately, the service is realized by mapping multipoint on point-to-point connections built as a tree topology. This definition leads to several problems with regard to performance and applicability of the standard, because multicast or broadcast capabilities of networks are not used. Hence, transport layer functionality is duplicated on the conference layer instead of

using existing functionality on the transport layer. As a consequence, media- or application-dependent transport protocols are not supported by the channel concept of the standard.

A new design of the MCS implementation is required to support any kind of real multicast. Approaches like annex A of the T.125 try to replace the OSI TP4 requirement by a T.120 specific multicast transport protocol for networks which provide multicast transport. But this approach is realized only by an extension of the existing standard instead of introducing the notion of multicast-capable channels in MCS.

Marshaling facilities for the transfer of typed user data like objects are not supported by the T.120 protocol, so this functionality has to be implemented by the application.

Both multipoint transfer capabilities were defined to work with different types of networks. The audiovisual part is IP-based in the H.323, and components were defined to interoperate with other H.32x-based systems (like H.320 for N-ISDN systems). For the data conferencing part, different protocol stack profiles were defined to operate with different networks.

As a consequence, it can be summarized that the multipoint transfer capabilities of the H.323 are not well suited for large-scaled conferences and offer a poor flexibility in terms of supported transport protocols. Approaches were proposed to extend the applicability, but have not yet been realized.

Distributed Applications Support

The H.323 supports floor control facilities for audiovisual and data conferencing. But floor control for audiovisual conferencing is restricted to simple *conducted* and *speaker-selected* conferences. More complex floor control based on given scenarios according to *social rules* have to be implemented using the facilities of the data conferencing part, the T.120. For that, a parallel data conference has to be established to use this functionality.

The floor control in the T.120 is realized by providing a *token* concept. Almost all of the proposed functionality for floor control [DoGLA95] is integrated in this proposal, only the capability to inquire a list of the current floor holders is not provided. Unfortunately, this function is required to implement a shared application, so this missing feature is emulated in the corresponding standard (T.128 [T128]).

Standardized Applications

An important issue of the H.323 was the standardization of commonly used applications. The audiovisual part of the standard defines all procedures to interoperate among different H.323 systems of different vendors. Furthermore, different data conferencing applications like multipoint file transfer, shared whiteboards, and shared applications were standardized as well as simple chatting functionality. As a consequence, the H.323 provides a wide range of standard application protocols to build group communication applications. These standardized protocols are built on top of the service model of the underlying conferencing communication infrastructure, i.e. the GCC and MCS for data conferencing. Thus, these application protocols might use other conferencing protocols if they provide an equivalent service model.

Robustness

Robustness is only poorly supported on service level. Fault tolerance and error recovery are provided by a re-setup of a connection or a conference which leads to a complete conference re-setup instead of recovering from the error during a running conference. Furthermore, only poor conference management functionality is provided, so operations like deleting providers from running conferences are not provided.

For the robustness on protocol level, the used protocols are defined using *Message Sequence Charts* (MSC [Z120]), but only the MCS protocol (T.125) is formally specified using SDL. Neither for the MCS nor for the other protocols of the H.323, test cases are defined by the ITU to ensure the conformance of products to the standards. The *International Telecommunication and Multimedia Consortium* (IMTC), an industry foundation, organizes quarterly events to test the interoperability among products of IMTC members. These tests improve the interoperability of products, but do not ensure conformance of the products to the supported standards.

More sophisticated formal methods like proposed in [CrRe99][CrRo99] were not used during the standardization process of the H.32x.

Scalability

The scalability aspect was mentioned as one of the most important in the requirement definition of chapter 2.1.2. Unfortunately, the standardization effort of the H.323 was not focussed on the provision of a high scalable system. In the audiovisual transmission part, a strictly centralized or at least centrally controlled approach was used for the transmission of multipoint data. Hence, it does not scale well despite of the assumption of an IP-based LAN environment. Furthermore, the poor mechanisms which were designed for the data conferencing part, namely conference database management and multipoint transfer, restrict the applicability of the system to a small number of users. Moreover, the centralized floor control realization leads to a poor responsiveness of T.120-based systems. Approaches to support a higher number of participants are currently under development, but have not yet been realized so far. Additionally, in annex C of the T.120 recommendation, a light weight version of the T.120 protocol suite is proposed with some functionality restrictions, but the used protocol mechanisms are the same.

2.2.6 Summary

This section presented the H.32x system family which was standardized by the ITU to enable audiovisual conferencing scenarios. As an example for one of these umbrella standards, the H.323 standard was introduced due to its IP foundations and its increasing relevance to IP telephony systems.

The different components of an H.323-based system were presented in this section. These components are used to extend the system functionality by adding modules to the system. This modular concept is one of the advantages of the system design. Capability exchange procedures among components in the H.323 system ensure the negotiation of a minimum level of interoperability between the participating devices.

But this modularity is also the main weakness of the system. Most of the different components were designed in parallel or without taking concepts of other components into account. As a result, some functionality was implemented twice, which led to a heavy weight system. Furthermore, the centralized design concept outlines many bottlenecks in an H.323-based system. Also the newer T.120 part of the system, designed for the provision of data conferencing, suffers from some design drawbacks for the service provision. Inefficient multipoint transfer of user data as well as a poor conference database replication mechanism restrict the number of participants in a conference to a few members only. Furthermore, the used floor control mechanisms do not scale to a large number of members due to the centralized management approach.

In general, it can be said that an H.323-based system provides main parts of the requirements presented in chapter 2.1.2. However, some of the service and mechanism design decisions

restrict the applicability of the standard even in moderate sized scenarios (few tens of members). As a result, the design of an H.323-based system is very static and cannot fulfil complex audiovisual scenarios with integrated data conferencing. Proposals were defined to support loosely coupled environments or more light weight versions of the protocols, but products and implementations with these new protocol stacks are not widely available up to now.

2.3 Group Communication in the Internet

Group communication in the Internet has become a growing field of research during the past ten years. Pushed by technology improvements in the area of IP multicast and provided bandwidth of the used networks, conferencing applications and toolkits were proposed and developed (e.g. [AF91][ADH93][BrRe98][Herr92][Scho93][SchoCa92]), enabling the audiovisual cooperation among researchers including data applications like shared whiteboards.

Most of these applications have been designed following the *Application Level Framing* (ALF) approach proposed by Floyd et al. [FJC97]. This design framework assumes that all protocol functionality above simple transport layers are application-specific. As a consequence, the functionality should be provided by the appropriate application itself instead of designing a generic protocol for conferencing, multimedia delivery, or distributed control of applications. Service units should be framed on application level containing all semantics needed for the purpose of the packet being delivered by simple transport layer facilities like reliable multicast or media streaming protocols. Some of the first multimedia applications of the early 1990s were designed in this way, although Floyd et al. published this design principle first in 1997.

Following this design paradigm, the *Internet Engineering Task Force* (IETF) started with the standardization of conferencing protocols for several purposes in 1996. For that, an *Internet Multimedia Conferencing Architecture* was proposed as a framework for the design of conferencing applications for the Internet. Several of the proposed protocols of this framework are still in the standardization process or have been standardized recently. Most of these components are extracted from the monolithic tools which were designed following the ALF design approach as a reverse engineering effort.

In this section, an overview of group communication in the Internet is given, starting with the presentation of the underlying multicast technology, the *Multicast Backbone* (MBone), which is being used in the Internet. After that, the Internet Multimedia Conferencing Architecture and its components are described in detail. A third section gives a brief overview of available tools for conferencing in the Internet before concluding with an assessment of the proposed architecture with respect to the requirements of chapter 2.1.2.

2.3.1 Multicast in the Internet - MBone

Multicast on IP level has been integrated in the development of the Internet by introducing an own class of IP addresses (class D) which is reserved for *IP multicast* traffic. The semantic behind the definition of this class is to address several receivers which have joined the *multicast group* being represented by the appropriate IP multicast address. With this semantic, more sophisticated multipoint addressing schemes (one-to-many, many-to-many, one-to-all) are provided on IP level.

The transfer of IP multicast packets on network level (e.g. Ethernet, Token Ring) is defined by several mappings of the IP multicast address to corresponding MAC (*Medium Access Control*)

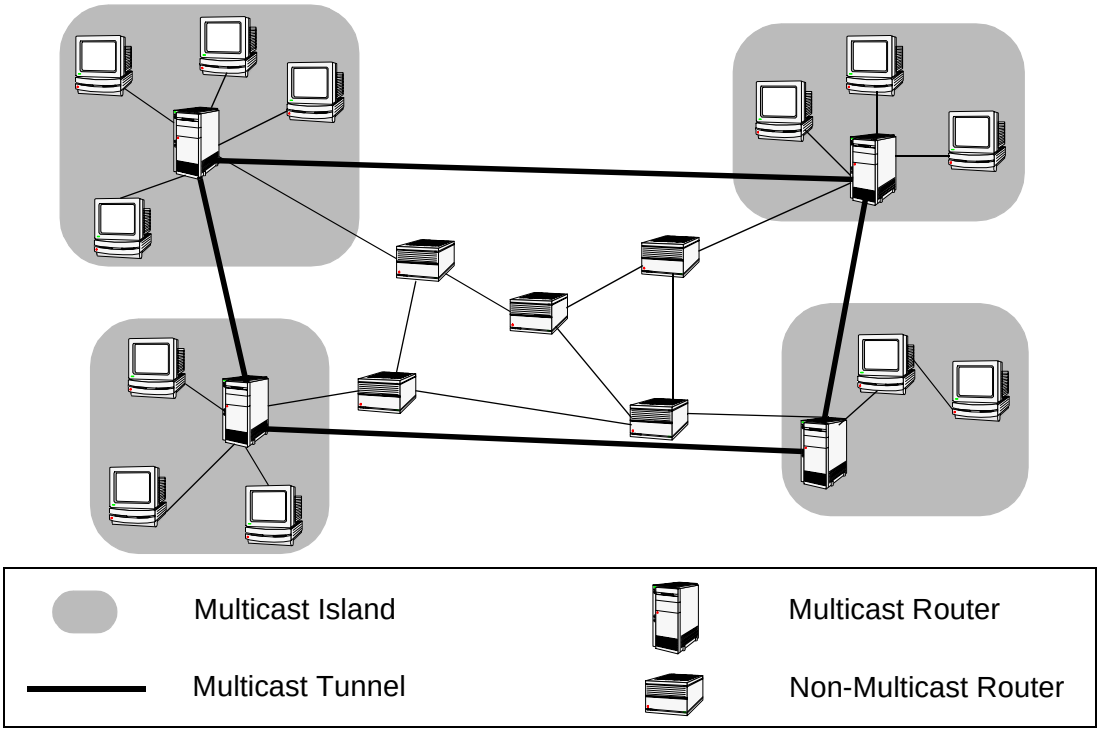


Figure 2-9: *Multicast Backbone - Topology*

addresses of the used network. For the connection of different multicast-capable networks via the Internet backbone topology, the *Multicast Backbone* (MBone [Erik94]) was designed. First introduced during an IETF audiocast meeting in 1991 [CaDe92], a virtual overlay network topology was implemented to connect *multicast islands* via *multicast tunnels*. For that, an IP multicast packet is encapsulated by special *multicast routers* [RFC1112] at the edge of a multicast-capable network and sent via the *normal* Internet backbone topology. If the encapsulated packet is received by another multicast router, the original multicast packet is restored and the data is sent to the local multicast island if there is at least one locally registered receiver for the corresponding multicast group. Thus, the more efficient multicast technology is used on local level. Figure 2-9 presents the design of the MBone topology with its different components.

To avoid the worldwide distribution of multicast packets for a specific group, the notion of the *scope* of a packet is introduced. For each multicast packet, the scope defines the area of the distribution of the packet. This is realized by using the TTL (*Time-to-live*) field of the IP header when transferring the packet. This field is used within multicast routers by decrementing this value when leaving a defined *administrative area*. If the counter reaches zero, the packet is discarded. With this simple mechanism, the scope of the packet in terms of used routers is limited.

For the routing of the IP multicast packets, a spanning tree is built for the multicast route from the sender to different receivers using multicast routing protocols like *Distance Vector Multicast Routing Protocol* (DVMRP [RFC1075]) or *Protocol Independent Multicast - Sparse Mode* (PIM-SM [RFC2362]). Local multicast reception is announced by receivers to their local multicast router using the *Internet Group Management Protocol* (IGMP [RFC2236]) providing a *join/leave* mechanism of local users to dedicated multicast groups. An evaluation of multicast routing protocols can be found in [Her97].

The multicast functionality of the Internet, provided by the MBone topology, is extended to the transport layer by several protocols. Approaches for multimedia streaming like RTP

[RFC1889] or RTSP [RFC2326] are built on top of UDP using the IP multicast facility. Multicast transport protocols like RMTP [PLSB97], MTP/SO [BOGTS94], or SRMT [Schu99] provide reliable data delivery functionality to the upper layers.

This basic multicast functionality in the Internet is used in several conferencing architectures, including the presented H.323 standard (see chapter 2.2) and the following Internet approach. But also the proposed conferencing approach in chapter 3 uses the functionality of IP multicast and multicast transport protocols.

2.3.2 Internet Multimedia Conferencing Architecture

Handley et al. proposed in [HCBO99] an architecture for Internet multimedia real-time conferencing enabling a general and scalable framework to implement group communication applications even in very large scaled scenarios. This proposal reflects the work which has been done (and is currently ongoing) in the IETF working group MMUSIC (*Multiparty Multimedia Session Control*) as a standardization effort for conference management and setup.

In this section, the main components of the architecture are presented covering conference setup and control in addition to real-time and reliable data delivery using multicast. The presentation starts with the services provided by the architecture before outlining the system model which is used in the architecture together with the proposed protocol stack and its components.

2.3.2.1 Service Model

The proposed architecture was designed to support a wide range of conferencing scenarios together with a large degree of freedom when designing the applications for these scenarios. Similar to the H.323 approach, the architecture is proposed as an umbrella framework using different standardized components to provide specific functionality. This section outlines the service model of this framework without specifying the different components. The latter are introduced in the following sections in more detail.

Conference Management

The architecture provides management functionality for *loosely* as well as *tightly coupled* conferences. The provided management services are divided into services for *conference setup and discovery* and facilities for *conference course control*. For the first one, services for conference announcement are defined as well as for active invitation of certain users to a conference. For the description of the conference and the used media, a meta data description is defined. For the location of conferences, a lookup service is provided. Existing web-based techniques are used as well as directory services for conference announcement.

The management services for conference course control provide functionality for resource allocation by using existing resource reservation protocols for the Internet. For *tightly coupled conferences*, H.323-based services are used for conference management. Thus, all services of these systems are available in the architecture when establishing tightly coupled conferences.

For *loosely coupled scenarios* (or *light weight sessions*), the provision of a conference database depends on the application and is not explicitly provided. This follows the ALF design paradigm by integrating this functionality in the application semantics. A logical *conference-session-application* mapping similar to the conference management in the T.120 standard (see chapter 2.2.3) is not provided as a generic service for light weight sessions.

The Internet architecture also provides authentication to join a *secure* conference. Appropriate

standards like PGP (*Pretty Good Privacy* [RFC1991]) are used for this functionality. The corresponding keys are distributed for instance by e-mail in the conference announcement. Furthermore, the session announcement itself might be encrypted as well as the transferred multipoint data.

For the management of network resources and the provision of a *Quality of Service* for a dedicated stream within the conference, a reservation scheme is defined as a service approach in the Internet.

Multipoint Transfer

The multipoint transfer of streaming and non-real-time data is realized by using IP multicast-based techniques and protocols. Streaming as well as reliable multicast transport protocols are directly used with a common notion of *multicast groups* to provide multipoint addressing patterns. Furthermore, existing unicast technologies like TCP or HTTP [RFC2616] are supported. Management services for multicast groups (like private channels or security aspects) are left to the application according to the ALF paradigm. Thus, these services are not provided in a generic way to simplify the design of group communication applications.

Distributed Applications Support

For the support of distributed applications, the services are divided for tightly and loosely coupled conferences. For the first, the ITU-based H.32x approaches are used integrating these functionalities in the Internet architecture. For light weight sessions, the floor control is implemented as an application-specific functionality following the ALF paradigm, i.e. a generic service is not provided.

Standardized Applications

Similar to the other services, standardized ITU-based application protocols are available for tightly coupled scenarios. But in light weight sessions, standard application protocols are not defined according to the ALF design paradigm. De-facto standards provided by certain tools are presented in chapter 2.3.3, known as the *MBone Tools*.

It can be summarized that the service model of the Internet architecture provides services for conference setup and discovery. For other services, existing approaches in the area of conference control and multicast delivery are used in an application-specific manner. Thus, generic services for floor control or commonly used standard application protocols are not defined.

2.3.2.2 System Model

It can be seen from the service model that the architecture does not define a stringent model for the provision of conferencing functionality. Most of the functionality is left to the application following the application level approach proposed by Floyd et al. [FJC97]. Thus, the resulting system model of the proposed architecture is rather simple by proposing new standards for conference setup and discovery and using existing technology for media streaming and resource allocation. In the following, the proposed system model and its components are presented by first introducing the proposed protocol stack and the division in different components. In the following sections, the components and the used protocols are described in more detail.

Figure 2-10 shows the proposed protocol stack of the Internet conferencing architecture. The grey shaded part is defined as application-specific functionality, while the rest is provided as a generic service to the applications. Following this proposal, a user application is divided into a

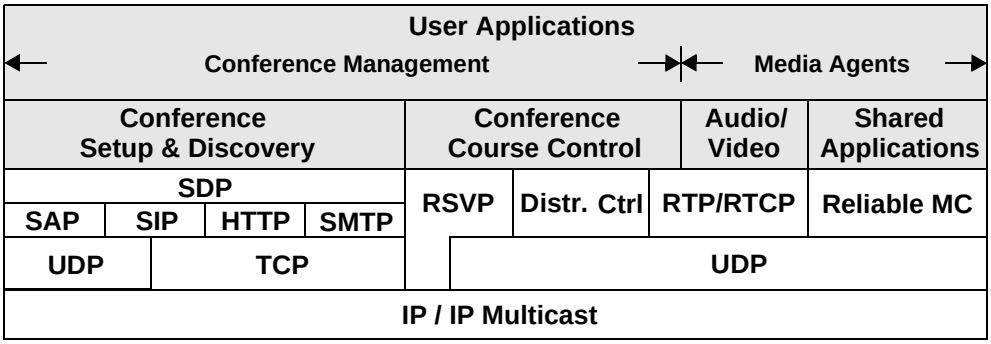


Figure 2-10: *Protocol Stack of the Internet Multimedia Architecture*

component dealing with *conference management* and into agents dealing with media handling (*media agents*). The latter use existing streaming technology (like the *Realtime Transfer Protocol* [RFC1889]) or reliable multicast transport protocols (like RMTP [PSLB97] or MTP/SO [BOGTS94]) to implement audiovisual or shared application functionality. The conference management functionality is divided into setup and discovery of conferences as well as control of running conferences. The latter deals with allocation of network resources (e.g. bandwidth) and floor control aspects [DoGLA95] for distributed applications support.

On the transport layer, TCP [RFC793] and UDP [RFC768] functionality is used based on IP [RFC791] and IP multicast on the network layer. In the following, the four functional blocks, *conference setup and discovery*, *conference course control*, *media streaming*, and *shared applications* are described in more detail by introducing the used protocols for the corresponding functionality.

2.3.2.3 Conference Setup & Discovery

For the setup of conferences, two mechanisms are provided, namely *announcement* of sessions and *initiation* of a session. For both, protocols are proposed to provide the corresponding functionality. Furthermore, web- and SMTP-based mechanisms may be used to announce conferences. For the description of conferences, a meta data language is proposed for the definition of conference specific information. In the following, the different protocols are described which provide the setup and discovery functionality.

Session Desciption Protocol - SDP

For the description of conferences, a text-based description language is proposed in [RFC2327]. Despite the chosen name, SDP does not define how to exchange the conference and media description. It just defines the exchange format for the conference description as a metadata language.

A typical conference description contains information about the location of the origin host, where the conference is hosted, a description of the media being transferred (used protocol, codecs, quality of service parameters), or resource requirements for the reservation of appropriate resources in the network.

Session Announcement Protocol - SAP

For the advertisement of conferences, the *Session Announcement Protocol* (SAP [HPW99]) is proposed. With this protocol, the conference description (defined using SDP) is distributed

among the potential session members, that are specified in the SDP definition of the conference, or by using multicast transfer capabilities to send the announcement message using a dedicated multicast group. It is also intended to announce conferences for tightly coupled conferences. In this case, the invitation mechanism has to be specified in the session description, e.g. H.323-based. But it is expected by the authors of the proposal, that only large scaled light weight sessions are announced with SAP to restrict the number of announced sessions by excluding smaller, more private, tightly coupled conferences.

Session Initiation Protocol - SIP

For the initiation of a conference, the *Session Initiation Protocol* (SIP [HSSR98]) is proposed for the functionality to initiate a conference regardless of the used conference model (tightly or loosely coupled). This functionality is independent from whether the conference is advertised (using SAP) or not. Furthermore, the invitation of users during a running conference is provided.

As users may be mobile, SIP provides a *lookup service* to enable location independent invitation of users by introducing *redirection mechanisms* instead of routing invitation calls to a specific terminal. Additionally, alternative responses are supported like leaving a message in the case of temporary unreachability of the invited user.

SIP does not standardize how to initiate a conference setup. Thus, it is independent from the conference control and setup mechanism itself. Basically, it is just an invitation message delivery service supporting location transparency of the message delivery. The application semantic, defined in the SDP descriptions, should allow to setup the conference using the specified application given in the SDP message. With this system, advance setup features are provided which could be used e.g. for advance reservation of the required resources.

HTTP/SMTP

The exchanged data units in SAP and SIP are based on the *Hypertext Transfer Protocol* (HTTP [RFC2616]) transferred as text messages using TCP as a reliable transfer protocol. The text-based SDP descriptions may also be advertised using HTTP for web-based exchange of conference setup information or SMTP for mail exchange of the conference description.

Security

The Internet architecture uses standard encryption mechanisms like PGP [RFC1991] to encrypt session announcements as proposed in SAP. Furthermore, authentication and key distribution is used to restrict the admission to certain conferences to a defined group of members. For the secure transmission of multipoint data, *on-the-fly encryption* is used, but research in this area is still in progress due to the stringent timing requirements which make the encryption process very difficult.

2.3.2.4 Conference Course Control

Beside features to announce and initiate conferences and to invite users for conferences, even to running ones, the proposed Internet architecture provides features for *conference course control* covering aspects for resource reservation and distributed control of the applications.

RSVP

Due to the critical timing constraints for audio and video transmission (see chapter 2.1.2),

audiovisual conferencing demands a high quality of the provided services to ensure a high acceptance of the participating users. Network techniques like the *Asynchronous Transfer Mode* (ATM [Part93]) provide guarantees for certain *Quality of Service* (QoS) parameters like *bandwidth*, *delay*, or *delay variation* (jitter). But these features are not provided when using IP as the standard protocol in the Internet. As a framework to provide QoS independent from the used network infrastructure, the *integrated services* [RFC1633] approach is proposed. With this framework, resource reservation and the provision of a certain level of performance is enabled. It is not within the scope of this approach how the resources are provided by the underlying system. Only the definition is given how to reserve provided resources. Additionally, a router architecture is proposed to implement the resource reservation.

For the reservation of resources from a dedicated sender to a group of receivers, the *Resource Reservation Protocol* (RSVP [ZDE93]) is proposed. Resources can be reserved within routers in a multipath environment. It is a receiver-initiated protocol, i.e. the receiver starts the resource allocation procedure. RSVP uses a *soft-state* approach, i.e. the resource allocations are refreshed based on periodic reservation messages generating a permanent load for the RSVP flow. As mentioned above, it is neither within the scope of the protocol how the resources are provided to the system nor how the admission control of a new connection is realized.

For the integrated services framework, two different classes of traffic are defined in [RFC2211] and [RFC2212], namely *Guaranteed Load* and *Controlled Load Service*. The first provides guaranteed bandwidth and delay to the application, while the latter only supports measurement-based QoS, i.e. the QoS is load-dependent.

The integrated services approach, and especially RSVP, is very difficult to implement. First implementations of RSVP are now available, e.g. for the Microsoft Windows system. Furthermore, RSVP lacks of scalability due to the complex algorithms which are hard to be implemented, especially in the backbone domain of the Internet.

For simplicity of the realization of a QoS-based network infrastructure, a new approach is currently discussed in the Internet workings groups, called *Differentiated Services* ([Kilk99] [RFC2475]). Instead of using the classification mechanisms for flows in the integrated services approach, each transferred packet is explicitly marked using the *Type of Service* (TOS) field of an IP header. Due to this *flow aggregation* technique, the per-hop behavior within an administrative domain of an *Internet Service Provider* (ISP) is much better compared to the integrated services scheme. This improves the applicability of this approach, especially in the backbone domain and when traversing different ISPs. For more details concerning the architecture of this type of service, see [RFC2475].

Conference Control

The services for conference control are divided into services for light weight and tightly coupled conferences. For the latter, approaches used in the ITU world, i.e. the H.323 (see chapter 2.2), might be used for explicit membership and floor control facilities. For more scalable tightly coupled conferences, Borman et al. proposed the *Simple Conference Control Protocol* (SCCP [BOR96]) providing the same service as proposed for an H.323 system but with higher scalability. Unfortunately, the mechanisms for this provision of high-scalable conferences are not proposed in the approach. So it is not clear how the authors want to provide this scalability.

For loosely coupled conferences, no explicit conference control is proposed. Hence, this type of conferences lacks of explicit membership control and sophisticated floor control facilities.

2.3.2.5 Media Streaming

Streaming media data like audio and video requires only little in terms of transport protocol functionality. Retransmissions are normally not feasible, when the data is sent over non-trivial distances (e.g. in MAN or WAN) due to the stringent timing constraints for real-time data transfer [Part93]. This simplifies the implementation of a streaming protocol, because retransmission buffers are not necessary.

For the transfer of multimedia data, it simplifies the reception of data if different logical streams are carried in different logically separated units. For this separation, the notion of *flows* is defined which may be realized by using different ports on transport level for the flows or by sending the flows to separate multicast groups. Furthermore, different QoS parameters might be associated to the different flows, e.g. a smaller packet loss for audio than for video. The flow separation also allows to configure the reception of multimedia data more flexible, e.g. by selecting only the audio stream of a conference when connected to a slower link or to subscribe only to the basic video layer when a smaller amount of video data can be received.

Especially the latter is realized in the framework of *receiver-driven layered multicast* proposed in [MJV96], where video streams are separated in different layers which are sent to different multicast groups. Subscribing to all layers enables the reception of the best quality video, while disabling more and more *extended layers* lowers the quality of the received video. In this framework, the receivers control the reception by testing the available local bandwidth applying a *learning mechanism* of local multicast group subscription and deletion. Thus, the control algorithm is receiver-driven. On the one hand, the subscription and deletion of multicast groups causes an additional overhead due to the needed IGMP traffic for these operations, but on the other hand, the available (or needed) bandwidth might be adapted optimally to the user needs. Following the application level approach (ALF framework) in the Internet architecture, the realization of these separation mechanisms are within the scope of the specific scenario and therefore of the specific application.

For the transport of the resulting media flows, the *Realtime Transport Protocol* (RTP [RFC1889]) is used which provides a corresponding feedback facility defined in the *Realtime Transport Control Protocol* (RTCP [RFC1889]). The protocol data consists of a standard packet header which includes, among other fields, a traffic specific timestamp, payload information, and sequence numbering. The timestamp information is normally used for *media synchronization* which is not within the scope of RTP. The payload information is a standardized field defining the contained media type (e.g. H.263 [H263]). These numbers are assigned by the *Internet Assigned Numbers Authority* (IANA) for each media type. The sequence numbering information of the RTP header might be used to detect losses or out-of-sequence packets to implement *adaptive applications* [CSZ92]. Each RTP source is identified by a *source identifier* contained in each packet. RTP allows mixing functionality in gateways, combining several flows to combined flows. As a streaming framework (including central server elements and video on demand functionality like forwarding in a stream), the *Realtime Streaming Protocol* (RTSP [RFC2326]) might be used. Implementations of this framework are still in progress.

For control purposes (e.g. bandwidth adaptation or bandwidth management) and media synchronization, RTCP is used. This protocol provides the relationship between the real-time clock at a sender and the RTP timestamp of the flow packets to enable inter-flow synchronization, e.g. between video and the corresponding audio flow. Furthermore, textual information about the sender is transferred using RTCP. But the protocol is also used for feedback information of the reception quality and partial membership information. The membership information does not provide an exact list of members, so it does not replace a conference database. The receiver

information is sent to the entire multicast group with a constant RTCP rate for each conference. Hence, if the conference grows, the RTCP rate of each member decreases.

Approaches like the *receiver-driven layered multicast* [MJV96] use RTCP for the learning mechanism and to indicate the reception quality of the different layers. It can be seen that the approaches for media streaming follow the ALF approach to implement application specific semantics.

2.3.2.6 Shared Applications

The Internet architecture does not provide a generic application to simplify commonly used functionality like shared whiteboards. Following the *Application Level Framing* approach, many data applications have been designed during the past ten years, e.g. wb [JaCa98b] for shared whiteboard functionality or NTE (*Network Text Editor* [HaCr97]) for distributed text editing. An overview of commonly used MBone tools is presented in chapter 2.3.3.

It is common to all these shared applications that packet loss is not tolerable. As a consequence, some sort of multicast reliability is mandatory on transport layer level. The requirements for the reliability may differ depending on the application. Some of the applications, e.g. a shared whiteboard, add per-participant information to a shared state. This state might be temporally inconsistent as proposed in [HaCr97] for distributed editing. Other applications like the central administration of a speaker list require a proper sequence of the information to provide fairness.

The Reliable Multicast research group of the *Internet Research Task Force* (IRTF) is currently discussing several approaches to provide reliability for these different applications. But some important points to be provided by reliable multicast transport protocols, e.g. congestion control, are still in progress. This can impressively be seen from the high number of proposed protocols ([BOGTS94][Schu98][PSLB97][WMK97]) in the past. As a consequence, it is not expected that a standard protocol will be defined in the near future.

2.3.3 MBone Tools

At development of conferencing applications in the Internet, the approach to integrate functionality directly into the corresponding application instead of providing generic application protocols was taken into account very early. This was realized in the earliest tools for the MBone, years before Floyd et al. proposed their ALF framework as a design pattern for this kind of applications. In the following section, some of the available tools for group communication in the Internet are presented. Only the main functionalities are described in a short overview.

Voice Audio Tool - VAT

One of the first conferencing tools for the Internet was presented in the early 1990s for audio transmission (*vat* [JaCa98a]). The tool provides several codecs and transmits the data using RTP over IP or IP multicast. Speech control is supported selecting the current speaker depending on the volume (*volume-controlled floor control*).

Video Conferencing - VIC

For live video transmission, the *vic* tool [MCa98] was developed supporting different software codecs (on some platforms hardware codecs are supported as well). Due to the usage of RTP over IP multicast, multipoint conferences are provided. Different views of the current video streams are supported (splitted, speaker-centered). Another video tool which is used in the

MBone is *nv* (network video [Fred92]). It was developed for the X11 Unix environment.

Session Directory - SD

For the provision of a conference list, the session directory tool (sd) was developed as a first approach to advertise conferences. Currently, the tool is adapted to the new mechanisms proposed in the Internet architecture using the announcement and initiation mechanisms of SAP [HPW99] and SIP [HSSR98] for the provision of the functionality.

Network Text Editor - NTE

Handley developed a network-based text editor (NTE [HaCr97]) for distributed editing of plain text. This tool deals with the concept of *temporary inconsistencies* to provide reliable exchange of the text which is marked as inconsistent while the update process is in progress. This approach is motivated by the high packet losses in the MBone which were investigated in [Hand97].

Shared Whiteboard - WB

Together with the audio transmission, a shared whiteboard presented in [HaCa98b] was one of the first applications which was available for the Internet. It provides creation of standard drawing objects and integration of pictures (e.g. in postscript format) for presentation on different sites. The reliability mechanism of the *wb* implementation was later extracted in the definition of the *Scalable Reliable Multicast* (SRM) framework [FJC97].

Questionboard - QB

For the provision of a rather simple floor control in conferencing scenarios, the question board (*qb* [MaRo97]) was proposed. It establishes a central instance for the management of incoming questions and indicates them by a *question list*. This approach does not provide a generic floor control framework because floor control in this context is meant as asking questions to the central speaker. Distributed floor control, e.g. for synchronization of applications, is not supported.

Most of the tools described above are available for several platforms, either UNIX or Windows-based. In [HWC97], a *control channel model* is proposed to interconnect the different components among distributed systems.

2.3.4 Assessment of the IETF Approach

This section outlines to what extent the architectural approach of the IETF working group fulfils the requirements for conferencing systems defined in chapter 2.1.2.

Conference Management

The conference management functionality of the Internet architecture is divided into services for *conference setup* and *conference control*. For the first, an announcement and initiation protocol is proposed, based on multicast advertisement messages in a standard text-based format. This information contains a description e.g. of the transferred media, required resources, or the used initiation mechanism. Announcements might also be advertised using web-based techniques (e.g. a central WWW page for announcements). Features like invitation redirections to support mobile users are provided as well as a directory mechanism for conference lookup. For the provision of secure announcements, encryption of advertisement messages is supported using standard encryption mechanisms like PGP.

For the control of conferences, mechanisms for tightly coupled conferences are used by integrating H.323-based systems in the architecture. An own approach for tightly controlled conferences was proposed, but the mechanisms to support higher scalability of this approach are not mentioned in the protocol.

According to the ALF approach, the control for loosely coupled conferences is left to the applications, i.e. the functionality is assumed as application and scenario specific and therefore, a generic service is not provided.

For the control of conference resources in terms of network quality of service, RSVP and techniques like Differentiated Services are proposed to provide QoS for conferencing applications by appropriate resources allocation mechanisms.

Due to the modular approach of the architecture, different conference control frameworks can be integrated into this architecture by providing the appropriate functionality. It will be seen that the proposed conferencing service SCCS, presented in chapter 3, can be embedded into this architecture as well.

Multipoint Transfer

Within the Internet architecture, several multipoint transfer capabilities are provided. All of them are based on the notion of *multicast groups* as an addressing scheme which is used for the data. With this notion, all communication patterns are supported using a common multipoint addressing scheme.

For the transfer of user data on transport layer level, RTP is proposed as a media streaming approach using UDP via IP multicast or unicast. The protocol design is very simple in terms of additional header information and protocol functionality. RTP is also used in other conferencing frameworks, e.g. the decentralized version of H.323 (see chapter 2.2.1). The media streaming is supplemented by a control protocol to distribute approximate group size information and to provide feedback facilities.

For the transfer of reliable multicast data, no generic protocol is proposed. Application functionality is implemented application specific according to the ALF approach. But research and development efforts are in progress to specify standard protocols for different kinds of reliability requirements. Furthermore, marshaling aspects for the transfer of typed data are not provided, because this functionality is covered by the application semantics which is not included in the architecture.

On-the-fly encryption is used to support secure multipoint communication by using encryption mechanisms like DES or PGP. But the stringent timing constraints, especially for real-time data, restrict the applicability of these mechanisms.

Distributed Applications Support

For tightly coupled environments, control mechanisms as provided by the H.323 standard are used. As a consequence, services of these systems are available in the Internet domain as well.

For loosely coupled conferences, no generic service for floor and access control is provided. It is assumed that this application-specific functionality is implemented in the application directly, following the ALF approach. Applications like the *question board* (qb [MaRo97]) demonstrate how these application specific semantics have to be implemented.

Standardized Applications

There are no standardized applications defined in the Internet architecture. Commonly used

functionality like audiovisual conferencing and shared workspaces is not defined, so interoperability among conferencing applications is not assured. But commonly used applications, known as the *MBone tools* (see chapter 2.3.3), can be seen as de-facto standards. Some of the functionality of these tools has been extracted in the past to be proposed as a standard component (e.g. SRM [FJC97] which was extracted from the *wb* implementation).

Robustness

The proposed architecture standardizes only a few components, namely announcement, invitation, and media streaming. All these mechanisms are multicast-based, either for the transmission of announcement messages or the transmission of the streaming data. As a consequence, the robustness of these services against local errors is very high. For the control of tightly coupled conferences, the same restrictions as for the H.323 systems hold (see chapter 2.2.5) due to the usage of this system. Other functionality of the architecture is left to the applications, so the robustness depends on the implementation of the application functionality.

Protocol specification and validation techniques are not widely used in the Internet domain for the proof of the correctness of used mechanisms. The standardization process of the IETF requires only a running reference implementation of proposed standards.

Scalability

The main scalability aspect of the Internet architecture is the consequent usage of multicast for most of the functionalities. For that, the MBone was designed as an overlay network in the Internet, using multicast capable networks wherever possible and connecting these *multicast islands* by unicast tunnels. With this design, a high scalability is reached for the transfer of multipoint data. Additionally, the definition of the *scope* of the packet restricts the distribution of multicast packets to a defined administrative area.

The advertisement functionality of the announcement protocol SAP is based on multicasting a message to a defined multicast group. Also, the feedback mechanisms of the media streaming, provided by RTCP, are based on multicast. For both mechanisms, there is a certain *break-even point* where the usage of multicast becomes more effective than using unicast messages. For the SAP functionality, especially announcements with larger scale in terms of participants and geographical distribution of the participants scale better when using multicast. In contrast, announcements for small conferences (which might also be several tens) should be better announced using centralized mechanisms, e.g. web-based announcement on a certain server. This decision has to be done by the application or the user (if using the same application for both scenarios). There is no service switching between both options transparently for the user.

For the RTCP mechanism, the rate of feedback data is restricted to a certain amount of bandwidth to improve the scalability of the feedback. But this might limit the responsiveness of the implemented algorithm if the group size increases.

Other scalability issues, e.g. for floor control or receiver feedback, have to be implemented by the applications, following the ALF approach. As a consequence, the scalability cannot be assessed concerning these features before appropriate standard services or applications are available.

2.3.5 Summary

In this section, group communication in the Internet was presented. In a brief introduction, an overview was given how multicast transfer in the Internet is realized by introducing the MBone

(Multicast Backbone) topology and some basics of this technique.

In the main part of the section, the *Internet Multimedia Conferencing Architecture* was introduced as the framework of the session control working group of the IETF. This architecture is mainly based on the *Application Level Framing* (ALF) approach. As a consequence, main parts of conferencing functionalities, especially for distributed applications support, have to be implemented by the applications. Only mechanisms for conference setup and resource reservation are proposed as generic services. For tightly coupled scenarios, systems like H.323 are integrated in the architecture for the provision of the appropriate services. Furthermore, the RTP/RTCP combination is proposed for media streaming and feedback implementation. The feedback mechanism itself has to be implemented by the application, i.e. no generic templates are provided.

Regarding the requirements of chapter 2.1.2, it can be seen that the proposed Internet architecture covers only a few services as generic functionality. The entire distributed applications support is not provided by a generic service. Thus, only basic services are available. Higher functionality is open to the implementation of the application. This is seen as a consequent extension of the ALF architectural paradigm to a multimedia conferencing architecture.

The main advantage of this framework is the open aspect of the architecture. It does not restrict the functionality of applications to the services which are provided within this framework. Thus, it is possible to extend the functionality by adding application specific semantics to the system. But this *open concept* is also the main disadvantage of this approach. It complicates the interoperability among conferencing applications, because the application semantics have to be the same for interoperability between these applications.

As a consequence, the proposed Internet architecture can only be seen as a first step for the architecture of conferencing software of the future by defining a modular framework for these applications. Dedicated functionality for specific scenarios has to be investigated and integrated into the framework in the future as proposed standard application protocols to ensure the interoperability of future group communication applications.

2.4 CORBA

The *Object Management Group* (OMG) started with a definition for an architecture of a generic distributed platform in the early 1990s. The platform was not specifically designed as a solution for conferencing scenarios, but as an architecture to simplify the development of distributed applications in general. The resulting *Common Object Request Broker Architecture* (CORBA) [CORBA93] is the OMG's answer to the need for interoperability among different vendors of software products transparent in terms of used hardware platforms and locations. The main design goals of CORBA are

- reuse of software components,
- portability and interoperability of software, and
- forming main parts for commercial software

by allowing applications to communicate with one another no matter where they are located or who has designed them. The architecture specifies interfaces, not the implementations within the system. For each proposed interface, at least one implementation of an OMG member must exist. As already mentioned, CORBA was not defined as a special conferencing architecture providing services for that purpose. As a consequence, no special services are standardized to

enable conferencing applications. However, CORBA enables the design and implementation of this kind of applications.

In this section, the service model of the current CORBA 2.0 standard is outlined with respect to the conferencing requirements introduced in chapter 2.1.2. It is not meant to present a general overview of the CORBA framework. After that, the architecture is presented outlining specific functionality to enable the development of conferencing services before depicting some extensions of the newly proposed CORBA 3.0.

As a framework to integrate standard-based conferencing functionality in a CORBA environment, the CORBA via T.120 approach is presented outlining the architecture and some results of this approach. At last, it is evaluated to what extent the CORBA functionality fulfils the conferencing requirements defined in chapter 2.1.2.

2.4.1 Service Model

Due to the general approach of CORBA, the introduction of the service model in this section deals more with the means to provide the services than with the provision of the services itself. Hence, the overview is presented with respect to the requirements of chapter 2.1.2 instead of presenting a general overview of CORBA services. For an overview of the latter, see [CORBA93].

2.4.1.1 Conference Management

The distributed object access in CORBA follows the client-server paradigm. Hence, the communication in CORBA takes place as an interaction between an object interface and its implementation which resides on a certain server (see chapter 2.4.2.3). In [TrSch98a], a CORBA framework was presented to realize a tightly coupled environment by using *crossed client-server implementations*. This approach allows to establish a communication infrastructure similar to the tree-based approach in the T.120 standard (see chapter 2.2.3). But of course, the services to manage this topology have to defined by the application, i.e. they are not provided by CORBA.

For provision of fault-tolerance in a CORBA environment, the *Persistency Service* [COSS97] might be used together with approaches to *migrate* objects for fault recovery or redundancy ([Go91][Nutt96]).

Facilities to control network or endsystem resources are not provided by CORBA.

2.4.1.2 Multipoint Transfer

Object communication in CORBA is strictly point-to-point following the client-server paradigm. However, CORBA also supports multipoint transfer of data by providing the *Event Service* [COSS97] for the exchange of messages between consumers and suppliers of events. Furthermore, CORBA supports *typing of data*, i.e. marshaling the parameters of an object invocation to ensure interoperability among different software and hardware platforms.

Additionally, location transparency of objects is implemented in CORBA by providing the *Trading Service* [COSS97] as a lookup service.

But there are no generic management facilities like administrating private and public channels similar to the T.120 approach (see chapter 2.2.3), e.g. to manage a member list for certain transfer capabilities. This has to be implemented by the application. Furthermore, it is not intended

to use arbitrary protocols within a generalized channel concept to support different media-dependent transport protocols.

2.4.1.3 Distributed Applications Support

Due to the distributed object approach in CORBA, the support for distributed applications can be implemented on application level. Generic concepts like *tokens* or floors are not defined in CORBA, because these concepts are not really necessary. The access control to objects can be realized by the object implementations directly without using explicitly allocated tokens.

2.4.1.4 Standardized Applications

Standard application protocols for conferencing are not defined in CORBA. The *Common Facilities* (see chapter 2.4.2.1) of CORBA are meant as commonly used functionality provided by certain packages within CORBA. But up to now, commonly used conferencing functionality has not been in the scope of the CORBA standardization.

It can be seen from the description above that CORBA provides certain functionality to design and implement conferencing systems. However, it does not offer generic services to simplify these implementations.

2.4.2 Architecture

The following section gives a brief overview of the *Common Object Request Broker Architecture* of the OMG. The description starts with the system model of the architecture which is used to provide interactions among distributed objects. Furthermore, communication and interfacing aspects are outlined which are of interest when designing conferencing applications. For a more detailed introduction to CORBA, its services, and applications, see [LiP98].

2.4.2.1 System Model

The system model is based on the client-server programming paradigm. Applications which reside within clients are comprised of *application objects.* These objects and the corresponding *service implementations*, provided by servers, communicate via a software bus, called *Object Request Broker* (ORB), as an abstraction for the distributed communication among objects.

The OMG defines an *Object Management Architecture* (OMA [CORBA93]) which is depicted in figure 2-11. Application objects may use *CORBA services* (like *Naming Service, Event Service, Concurrency Control* as defined in [COSS97]) and *Common Facilities* (for user interfaces, information management, system management). The communication between created objects and other services takes place via the ORB, abstracting from the communication infrastructure in a CORBA-based distributed system.

CORBA provides an *Object Model* which enables a standardized presentation of object concepts and terminology. An *object* is an identifiable, encapsulated entity that provides one or more services. These services can be requested by a client by communicating with the object implementation on a server. Only the object interfaces are defined, but neither the implementations nor the sequence of interface calls are specified. The standard, initially introduced as CORBA 1.1 in 1991, defines the *Application Programming Interfaces* (API) that enable client/

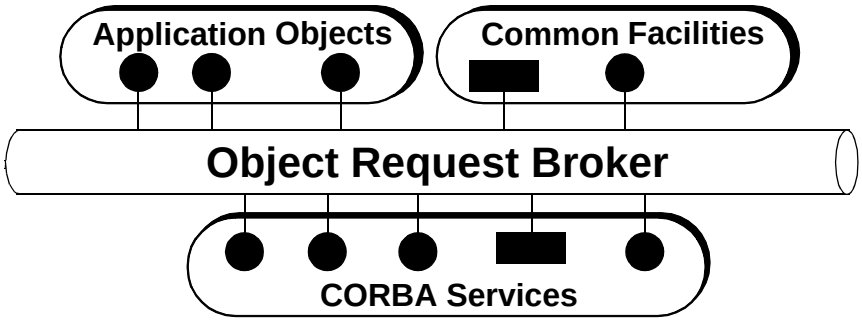


Figure 2-11: *Object Management Architecture (OMA)*

server object interaction within a specific implementation of an *Object Request Broker* (ORB). CORBA 2.0 [CORBA93], adopted in December of 1994, defines true interoperability by specifying how ORBs from different vendors can interoperate. For that, the *General Interoperability Protocol* (GIOP) is proposed translating requests from one ORB to another.

2.4.2.2 Interfacing in CORBA

Because the OMG only specifies interfaces and not the implementations, the interface definition is a central work within the OMG. For the specification of object interfaces, an implementation-independent language is proposed, the *Interface Description Language* (IDL [CORBA93]). It defines the interfaces for the clients and the object implementations. The language is a subset of ANSI C++, but was designed to be programming language and network-independent. No algorithms are specified. Hence, no variables and sequences exist.

The IDL separates interfaces from its implementations, as shown in figure 2-12. The IDL compiler generates *stub* and *skeleton* functions which are used in the client and server for the communication (see chapter 2.4.2.3). These functions may be generated for different languages like C++, Smalltalk, or Ada. The stub function parameters are extracted and marshaled before being transferred. The skeleton routine of the object implementation demarshals the parameters and invokes the server object implementation. Thus, marshaling of user data may be done via interface routines of CORBA. For the encoding rules within CORBA, the CDR (*Common Data Representation* [COSS97]) notation is used.

2.4.2.3 Communication Mechanisms in CORBA

This section introduces object-to-object communication mechanisms used with the ORB concept of CORBA. Furthermore, multipoint communication mechanisms are introduced. For the provision of location transparency between client and server within CORBA, the *trading mechanism* is depicted.

Object-to-Object Communication

As shown in figure 2-11, the ORB is used for the communication between client objects and the corresponding object implementation on a server. The ORB communication realizes a higher level *Remote Procedure Call* (RPC [RFC1057]) mechanism, including method invocation in a certain object and marshaling of the exchanged parameters.

Figure 2-12 shows the mechanism of requesting a service of an object implementation by a client. Using the ORB, a client can transparently invoke a method on a server object located on

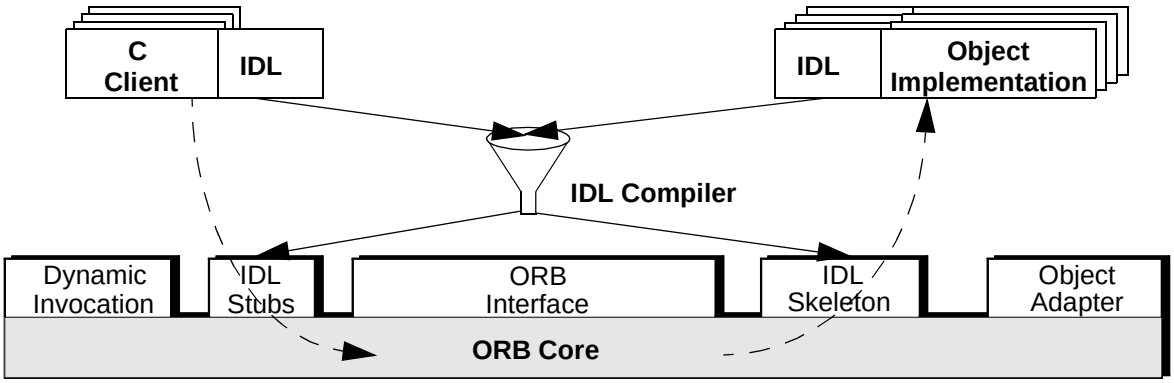


Figure 2-12: *ORB Communication in CORBA*

the same machine or across a network, using the *IDL stub* routines which are generated by the IDL compiler based on the IDL definition (see chapter 2.4.2.2). The *ORB core* intercepts the call and is responsible for finding an object implementation on a server, transferring the parameters, invoking the corresponding method, and returning the results. For this, the appropriate object implementation is called via the generated *IDL skeleton* routine on the server side. The object is addressed at the client side with an *object reference* which contains information about the location and implementation of the server object. A local *Naming Service* [COSS97] at the server can be used to lookup a specific object reference if the reference is known to this naming service.

In the CORBA 2.0 specification, the *Internet Interoperability Protocol* (IIOP) is defined for interoperability among different ORB vendors over the Internet. A more general protocol, the *General Interoperability Protocol* (GIOP), is also defined for the support of other network protocols. This allows the invocation of object implementations among different ORB implementations.

As a consequence of this communication concept, the client does not have to be aware of where the object is located, its programming language, the used operating system on the server side, or any other system aspects that are not part of the object interface defined in IDL.

Multipoint Communication

As mentioned above, the communication between a client object and its implementation is strictly point-to-point via the ORB. Multipoint ORBs are neither implemented nor standardized. Thus, multipoint exchange of user data is not possible. There are approaches to use other communication infrastructures like the HORUS system [BiRe94] for the provision of *object groups* [Maff95] in CORBA. But these approaches are restricted to a specific UNIX environment.

For the provision of an *event messaging* concept for multipoint communication in CORBA, the *Event Service* [COSS97] was defined which provides a noticeboard-like communication between objects following the *consumer-supplier* communication model.

In the simplest form, messages are exchanged using a FIFO-ordered model (push/pull messages) between *suppliers* and *consumers* of events in a point-to-point manner. Requests may also be sent asynchronously. Notifications to the consumer of a message are realized either by blokking (*pull*-operation) or non-blocking operations (*try_pull*-operation). Events may be *typed* (using IDL definition for user data) or *generic* (exchange user data as bitstreams). Thus, marshaling of events is supported.

The consumer-supplier concept is extended to a multipoint communication between several

Figure 2-13: *Communication via Event Channels*

consumers and suppliers by creating *Event Channels* as CORBA objects which act as *supplier* and *consumer* of events (see figure 2-13). Events are exchanged via these event channels in a reliable fashion. After creation of an event channel, applications may be added to this channel as a consumer and/or supplier of events (messages) to communicate with other applications via this channel. In the CORBA standard, applications are added by the *administrator* of the channel (the creator of the event channel object). This is similar to the private channel model of the T.120 (see chapter 2.2.3.5). It is not specified how the *underlying* communication is realized, because CORBA does not standardize the implementation of a service.

The CORBA implementation ORBIX from Iona, Inc. [Iona99] supports a public channel model only. The event channels are identified by CORBA string types and joined by the consumers and suppliers. Private channels are not supported. For the implementation of the multipoint communication, IP multicast is used to provide multicast on network level. Building the multicast address groups is not standardized in the *Event Service* specification [COSS97]. In [Scha98], an evaluation of the Orbix Event Service V1.1 implementation was performed. It was shown that the applicability of this implementation is restricted to the transfer of small packets only. Transmission of larger packets (e.g. streaming data) leads to unstable behavior of the protocol stack and poor performance in terms of transmitted data (less than 20% of comparable TCP throughput). At the time of writing this thesis, a revised version of the Orbix system and the Event Service are available which promise a more stable implementation.

Trading

The communication among objects using the ORB is realized between a client requesting a service of an object implementation and a server providing the implementation of the object. The mechanism to connect to a specific service implementation in CORBA is known as *binding*. Initially, the binding process was defined as static in the first CORBA specification. This means that the client has to know the server address and the specific service implementation. A local *Naming Service* [COSS97] can be used to lookup a certain object implementation, but this mechanism is restricted to the server where the Naming Service is implemented.

In a distributed system, it is desirable to have a dynamic selection of required services. Thus, a provision of a dynamic binding is crucial. The mechanism that enables a client as the service consumer to choose appropriate service providers at run time without knowledge of the object location is called *Trading Service* [COSS97]. An ODP *trader* is an object providing *service location transparency* to the client object.

Figure 2-14 shows the interactions between a trader and its users. A trader receives *service offers* from *exporters* of service implementations (step 1). These offers are stored in a central or distributed database. The trader accepts *service requests* from *importers* of a service (step 2). This request is an expression of requirements. The trader searches its service database to find the matching service offer. The most appropriate service offer which satisfies the service request is returned to the importer (step 3). This may be a list of matched services if more than one exists. Finally, the client invokes the object at the server (step 4) which, in the end, returns

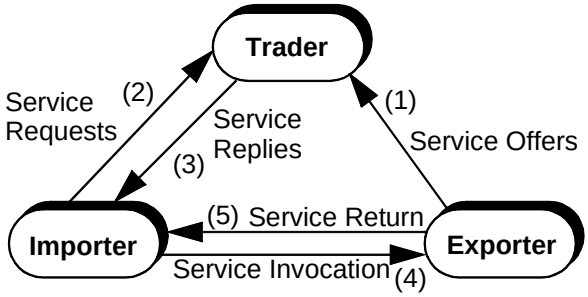


Figure 2-14: *Interactions of a Trader and its Users*

after service completion (step 5).

There are approaches to trade in a more user friendly way by specifying a given *quality of service* [LPT99]. The service requirements are expressed in terms of *constraints* which may be parametrized functions. Thus, service trading may lead to an optimum of service provision.

2.4.3 CORBA 3.0 Extensions

During the time of writing this thesis, the OMG is standardizing CORBA extensions incorporating new functionalities for better *Internet integration*, *messaging functionality*, and *real-time extensions*. This section gives a brief overview of this work as a summary of the current standardization effort.

Internet Integration

For a better integration of CORBA systems in Internet-based corporate networks, *firewall extensions* are going to be integrated in future CORBA systems. The firewall specification provides defined ports, being defined at the *Internet Assigned Numbers Authority* (IANA), for the IIOP and GIOP (see chapter 2.4.2.3) to enable communication among different ORBs through firewalls.

A second extension addresses the object reference problem in CORBA 2.0. As mentioned in chapter 2.4.2.3, the easiest way to get a remote object reference in CORBA is to use the server's naming service for a local object instance lookup. A problem occurs if the object reference for the server's naming service is not known beforehand. The *Interoperable Name Service* overcomes this problem by defining an URL-like object reference, `iioploc`, that may be defined within a program to reach defined services at a server, including the Naming Service. A second URL format, `iiopname`, invokes the remote Naming Service, looks for the object, appended at the URL, and retrieves the object reference of the remote object.

Asynchronous Messaging

CORBA 3.0 defines a *messaging specification* for the support of a number of asynchronous and time-independent invocation modes to realize both static and dynamic invocations. The results of an asynchronous invocation may be retrieved by either polling or callback which is defined by the client in the original invocation. *Invocation policies* allow to control the ordering (by time, priority, or deadline), priorities, deadlines, time-to-live of requests, and define a start and end time for time-sensitive invocations. Furthermore, the control of a routing policy and network routing hop count is provided.

CORBA for Specific Systems

For the provision of CORBA for specific systems, the OMG proposes extensions for systems with restricted resources in terms of memory and CPU power or systems with real-time capabilities. Furthermore, fault-tolerance management is proposed.

Minimum CORBA systems are primarily intended for embedded systems which are static, once they are designed and realized in hardware. Their interactions with the outside network are predictable, so they have no need for the dynamic aspects of CORBA (dynamic invocation, Naming Service). As a consequence, this functionality is not included in the Minimum CORBA system to lower the resource requirements for these systems.

Real-time CORBA systems provide control of resources in terms of threads, protocols, and connections using priority models to achieve predictable behavior for both hard and statistical real-time environments. Scheduling aspects are also proposed. This proposal incorporates recent research results in the area of real-time extensions for CORBA ([GoS96][SGHP97]).

Fault-tolerance for CORBA is being addressed by a proposal based on entity redundancy and fault management control.

CORBA Components

As structural elements for the software design of CORBA systems, the OMG proposes *CORBA components* to facilitate the development of large scaled distributed software and grouping element to functional units.

CORBA components consist of *containers* as the main part of a component. A container packages *transactions, security,* and *persistence* and provides interface and event resolution of the container objects. Containers keep track of event types emitted and consumed by components and provide event channels to carry events. The containers also keep track of interfaces provided and required by the components they contain, and connect one to another at dedicated interfaces. Thus, a container provides its functionality to the application programmer at a higher level of abstraction than the CORBA services do. As a consequence, the application programmer does not need to take care about (secure) transactions and different interfaces of the objects within a container.

Furthermore, the specification defines a multi-platform software distribution format, including an installer and XML-based configuration tool to enable a CORBA component marketplace. Additionally, a separate scripting specification will map the CORBA environment and the component assembly to a number of established scripting languages.

2.4.4 CORBA via T.120

As already mentioned, the CORBA environment does not support conferencing applications by providing generic functionality for the development of such applications. An approach to integrate this functionality, even standard-based, in a CORBA environment is proposed by the author in [HTT97]. In this approach, the T.120 functionality is provided as a CORBA conferencing service. As a consequence, a client may access this service with well defined CORBA interfaces by using the T.120 standard architecture for tightly coupled environments.

In [HTT97], a comparison of both environments (T.120 and CORBA) is presented leading to the conclusion that a combination of both benefits from the advantages of both systems. In the following sections, the architecture of the resulting conferencing environment is presented before summarizing some experimental results of the platform.

2.4.4.1 Architecture

The architecture which was used to combine CORBA and T.120 is shown in figure 2-15. Each

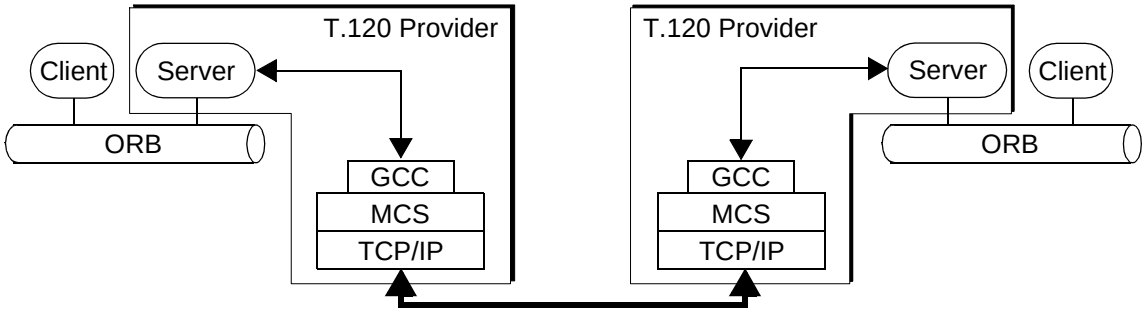


Figure 2-15: *CORBA via T.120 - Architecture*

T.120 provider is implemented as a CORBA server, *bridging* requests of the CORBA client, acting as a T.120 application, to requests for the T.120 environment and vice versa. Hence, *bridging* in this sense means to translate requests from one system to the other.

Because the T.120 protocol implements also notifications for the clients, e.g. when a database entry has changed, the server also implements functionality to invoke registered notifications of the client. For that, the notion of *crossed client-server* implementations is used in [HTT97] passing the CORBA object reference of the client notification object to the server, which is then used to invoke the notification by the server.

With this architecture, the T.120 conferencing functionality is provided as a conferencing service in CORBA. Features like trading and naming services in CORBA might be used to extend the T.120 by CORBA functionality.

2.4.4.2 Experiments and Results

The proposed architecture presented above is implemented using the Iona Orbix [Iona99] runtime environment of CORBA. For the T.120 functionality, a free available tool is used. Some of the used CORBA mechanisms are proprietary due to the missing standardization of implementation details in CORBA, e.g. the multithreading of requests. However, most parts of the server are portable to other CORBA systems.

It can be seen from the proposed architecture that the CORBA invocation of the services adds additional overhead to the system compared to a *pure* T.120 system. Especially, when server and client are located on the same host, i.e. when using the system only as a local bridge between two different systems, a lot of unnecessary overhead is produced:

- The communication between server and client takes place via inter-process communication. This is much more costly compared to inter-thread communication which would be used in a pure T.120 environment.
- Marshaling of requests is not necessary when invoking requests at the same host, because the encoding of data types is the same.
- Server and client are normally two separated programs. This causes an additional *scheduling* overhead when invoking a request at the client being sent to the local distribution daemon and then forwarded to the (local) server.

The implemented system was evaluated by the author with regard to this overhead in [Tro99a] by adding *monitoring points* in the appropriate functions of the system. It was shown that the scheduling overhead, caused by the separated programs, is the worst compared to the other costs which are more or less constant. Using the *collocated mode* programming model [Tro99a] of the CORBA implementation Orbix from Iona, the scheduling overhead can be eliminated by linking the server implementation to the client. This results in one application program which enables CORBA's look-and-feel of a conferencing system based on a standardized conferencing environment. However, a CORBA-like remote invocation of the T.120 services is also feasible by using the server implementation only.

2.4.5 Assessment of the CORBA Functionality

The CORBA environment was designed as a generic platform to enable inter-object communication in distributed systems. It was not meant as a conferencing platform. Hence, it does not provide any conferencing specific functionality as a generic service to enable the development of conferencing applications. As a consequence, most of the requirements defined in chapter 2.1.2 are not covered by a plain CORBA system. However, CORBA provides means to implement conferencing services, e.g. shown in ([OAB99][TrSch98a]).

Conference Management

Tree-based topologies may be built from a federation of CORBA servers. Object migration and *object groups* [Maff95] might be used to add fault-tolerance to a CORBA system. But this conference management functionality, among others like provision of a database or reconfiguration of a server tree, is not yet integrated in CORBA as a generic conference management service. Furthermore, security and network resource management are not provided by the architecture.

Multipoint Transfer

CORBA provides event and messaging services which also support multipoint communication patterns. CORBA extensions are proposed to realize real-time transfer of user data within a CORBA system [SGHP97] even with typed, i.e. marshaled, user data. But a homogeneous concept to realize multipoint transfer, e.g. using a generic channel concept, is not provided.

Distributed Systems Support

CORBA does not provide a generic floor control protocol. But the distributed implementation of an application inherently includes the access control to objects, e.g. by using *semaphores* in the local implementation. However, these mechanisms can only be used for a centralized access control, not for a distributed *floor control* as in an H.323 environment.

Standardized Applications

The *Common Facilities* of CORBA were designed to provide commonly used functionality in CORBA systems, but they do not cover conferencing application functionality. Thus, commonly used functionality like shared workspaces is not provided as generic services within CORBA.

Robustness

CORBA provides several means to implement robust applications and therefore robust conferencing services. For instance, the *Persistency Service* [COSS97] might be used to save system

state information to recover from errors. Furthermore, *object migration* techniques might be used for redundancy of objects. In [Maff95], *object groups* are proposed to implement fault tolerance in CORBA systems.

Scalability

Due to the missing conferencing service functionality, it is not possible to assess the scalability of CORBA-based solutions. However, multicast is supported by CORBA which enables scalable transfer of user data. But multicast-based object invocations, implemented by a multipoint ORB, are not yet defined in CORBA which restricts the applicability of the entire environment.

2.4.6 Summary

This section presented an overview of the *Common Object Request Broker Architecture* (CORBA) proposed by the OMG as an architectural framework for the development of distributed applications. The approach was not designed as a conferencing platform as such, i.e. conferencing specific functionality is not provided in the basic architecture. Thus, the requirements of chapter 2.1.2 are not covered by the system. But services for this purpose might be provided in the system, for instance integrated as CORBA *Common Facilities*.

From the presentation of the CORBA framework and its CORBA 3.0 extensions, emphasized by related work ([OAB99][TrSch98a]), it can be stated that CORBA might be used as a platform for conferencing services. But still missing functionalities like QoS support, real-time streaming, and a homogeneous multipoint addressing restrict the development of conferencing services. Additionally, available implementations still suffer from details like inefficient multipoint transfer [TrSch98a], missing multithreading functionality [Scha98], and missing real-time capability [SGHP97], which restrict the development of conferencing services in CORBA.

As an approach to provide conferencing services in CORBA, the integration of a T.120-based system and CORBA was presented which implemented a CORBA server as a *bridge* between CORBA and T.120. The architecture of the system was presented together with a summary of experimental evaluation results.

2.5 Summary

This chapter dealt with the presentation of group communication scenarios, their required functionality, and existing examples of conferencing systems.

It started with a brief overview of conferencing scenarios like tele-conferencing, tele-education, and tele-working. The description of these scenarios presented key conferencing functionality which is mostly independent from the specific scenario, namely *conference management*, *multipoint transfer of data*, *distributed application support*, and *provision of standardized applications*. These key features were defined as service requirements for the provision of a generic conferencing functionality covering a wide spectrum of application scenarios. Furthermore, *robustness* as well as *scalability* of provided services and mechanisms were outlined as crucial technical requirements for a conferencing service.

Additionally, two paradigms were compared concerning the coupling of participating endsystems within a conference, namely *tight* or *loose* coupling. It was concluded that for the provision of a wide spectrum of service requirements, tightly coupled environments are better suited for scenarios with tight control of members and resources. But also a combination of both para-

digms is an interesting option for certain scenarios where tight control is only needed for a few conference members.

In the second part of the chapter, existing conferencing systems were presented and assessed with respect to the defined requirements. The first system is standardized by the *International Telecommunication Union* (ITU) and comprises several standards for certain functionality in tightly coupled environments. The umbrella standard H.323 combines all these different components to a conferencing system for audiovisual and data conferencing. Systems based on this standard provide most of the requirements defined for conferencing functionality. But the systems suffer from some design decisions which lead to poor performance for large scaled scenarios in terms of conference size and geographical distribution of the participants. Furthermore, the modularity of the standard causes that some functionality is duplicated in the system. This leads to a heavy weight system if providing a wide spectrum of scenarios. For the support of large scaled scenarios, several extensions of the standard components are currently under discussion in the different working groups of the ITU.

The second approach reflects the ongoing work in the Internet domain concerning real-time conferencing. For a better understanding, how multipoint transfer is realized in the Internet, the *Multicast Backbone* (MBone) was briefly introduced as the realization of a multicast overlay topology. This infrastructure is commonly used for the realization of multipoint communication patterns in the Internet. The main part of the IETF work presentation was covered by the discussion of the proposed *Internet Multimedia Conferencing Architecture*. This proposal is meant as a framework for realizing conferencing applications for the Internet. It defines components for conferencing functionality support. But only a few of these components are currently in the standardization process, namely for conference announcement and initiation, and realization of multipoint transfer of audiovisual data. For the control of conferences in terms of membership and floor control the paradigm of loosely coupled conferences is used. This leads to the provision of conferences with no control of application resources. For scenarios with tight application resource control, the H.323 standard family is proposed. Additionally, the IETF is currently developing mechanisms to provide network Quality of Service (QoS) using protocols like RSVP or approaches like *Differentiated Services*. In contrast to the ITU approach, the proposed architecture does not standardize commonly used functionality as application protocols. However, well known tools (MBone tools) can be seen as de-facto standards for certain functionality like shared whiteboard or videoconferencing. Due to some missing functionality which is left to the applications the proposed Internet architecture can only be seen as a first step for the architecture of conferencing software of the future. Dedicated functionality for specific scenarios has to be investigated and integrated in the framework in the future as proposed standard protocols to ensure the interoperability of future group communication applications.

As a third platform, the CORBA framework of the OMG was presented for the development of distributed applications in general. Despite the provision of common facilities for distributed scenarios, special conferencing services are not standardized by the OMG. Multipoint transfer of user data is poorly supported by a simple messaging service which was not designed to stream real-time data. Extensions for Internet support, real-time and embedded systems, and components are currently in the standardization process. Other conferencing related functionality is neither standardized nor in discussion for standardization.

An approach to combine the T.120 standard of the ITU and CORBA was proposed by the author to develop a standard-based conferencing system with a CORBA look-and-feel from the user’s point of view. The architecture to realize the system was presented, and experimental results were depicted.

As a summary, the following table outlines the different requirements of a conferencing system

Aspect	Feature	H.323	IETF	CORBA
Conference Management	conferencing environment management	(✓)	✗	✗
	network resources management	(✓)	✓	✗
	conducted conferences	✓	✗	✗
	conference database management	✓	✗	✗
	security management	✓	✓	✗
Multipoint Transfer	location transparent addressing	✓	✓	✓
	transport layer abstraction	(✓)	✓	✓
	provision of media-dependent transport	✗	✓	✗
	support of heterogeneous systems	✗	✗	✓
Distributed Applications Support	access and floor control	✓	(✓)	(✓)
Standardized Applications	audiovisual functionality	✓	(✓)	✗
	shared workspace	✓	(✓)	✗
	shared application	✓	✗	✗
Robustness	Error Recovery, Fault Tolerance	✗	(✓)	(✓)
	Specification	(✓)	✗	(✓)
	Tests	✓	✓	✗
Scalability		✗	(✓)	✗

Table 2-4: *Comparison of Presented Conferencing Systems*

and shows a comparison of the presented systems with respect to these requirements. It can be seen that the IETF approach (which includes the ITU approach for tightly coupled environments) covers a wide spectrum of the requirements. But the usage of the ITU standards for tightly coupled environments leads to a poor scalability for scenarios with tight control among users and resources. Furthermore, both paradigms of tight and loose coupling of endsystems are hardly supported in mixed scenarios, i.e. tight control among a few participants with a large loosely coupled auditorium.

As a result of the presentation of existing conferencing systems, the next chapter will present a conferencing service proposed by the author. This system is designed for tightly coupled environments with a service provision even in large scaled scenarios. The proposed service will be designed and assessed with respect to the requirements for conferencing systems defined in chapter 2.1.2.

CHAPTER 3

Scalable Conferencing Control Service

In the previous chapter, existing group communication systems were presented in detail, some of them standard-based, some not. Regarding the requirements of a conferencing system layer, defined in chapter 2.1.2, it can be seen that the existing systems fulfil these demands differently. But even the most sophisticated T.120 standard approach, offering a rather generic functionality which covers a wide range of the requirements, lacks in some points like management issues and marshaling of typed data. Furthermore, the mechanisms used in the existing systems often do not offer a high scalability of the system.

As a consequence, this chapter presents the *Scalable Conferencing Control Service* (SCCS) as an approach to cover a wide range of the requirements of chapter 2.1.2 together with a high scalability by using more sophisticated protocol mechanisms.

The development of the service follows the specification criterion proposed by G. Holzmann in [Holz91], which are

- specification of the *service,*
- assumption of the *environment,*
- definition of the *vocabulary,*
- definition of the *encoding* format, and
- specification of the *procedure rules.*

The presentation of the service starts with the *service model* of SCCS in chapter 3.1 presenting the environment in which the service is provided and outlining the different services offered from a user's point of view. Chapter 3.2 presents the *protocol mechanisms or procedure rules* which are used to provide the services of SCCS in an efficient and scalable manner. For both parts, the *vocabulary* and *encoding* is defined by introducing the service and protocol data units exchanged among the different protocol entities. These data units are defined using an abstract notation (ASN.1 [X680]). But due to the complexity of the protocol, the entire definition of the service and data units and the message sequences are not presented in the thesis, but are specified in a detailed *service* and *protocol specification* of SCCS given in [SCCS98] and [SCCS99].

In chapter 3.3, the *implementation design* is outlined which was chosen to realize the protocol both on object and process model level with an additional demo application for the demonstration of the SCCS functionality.

SCCS is assessed in chapter 3.4 regarding the requirements given in chapter 2.1.2. The scalability aspect of the requirements is evaluated for two SCCS protocol mechanisms in the chapters 5 and 6 after introducing the modeling approach in chapter 4. As a starting point, the SCCS presentation finishes in chapter 3.5 with outlining the general evaluation goal and the performance measures which are of interest, among others, when evaluating the service mechanisms. Furthermore, the parameters which are assumed for the different system components and their impact on the results are presented.

3.1 Service Model

This section presents the service model of SCCS from a user's point of view. A first version of the SCCS service model was proposed in [TrSch98b], which is extended by several features in the current version.

The classification of the services is similar to the one used for the requirements in chapter 2.1.2. After presenting the SCCS service environment, the services for managing a conference are outlined, including database, user management, conference administration, and voting policy issues. The second class of services deals with multipoint transfer issues, while the third one supports access and floor control in distributed applications. Finally, marshaling of typed user data is presented. Figure 3-1 shows an overview of the services provided by SCCS.

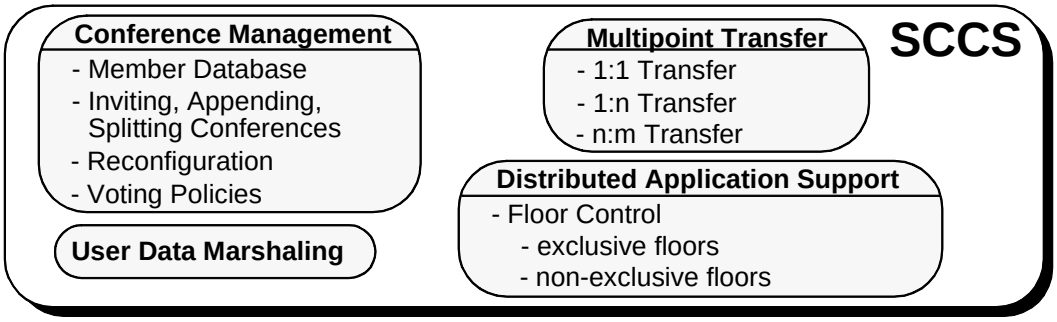


Figure 3-1: *Services of SCCS*

3.1.1 SCCS Service Environment

SCCS provides the creation of *conferences* between several users. Within each conference, the services of SCCS are provided by conference servers (*SCCS providers*) to which conference clients (*SCCS users*) are locally connected. The SCCS providers establish an *SCCS control topology* which is associated to the corresponding conference.

Services of SCCS are invoked by appropriate *service data units* (SDUs). These SDUs are sent from the SCCS user to the corresponding SCCS provider and back. The services of SCCS are provided by certain protocol mechanisms to fulfil the needed service functionality leading to an exchange of *protocol data units* (PDUs) among the SCCS providers. The corresponding protocol mechanism chapters are referred to when presenting the services.

A first version of SCCS was proposed in [TrSch98a] with a realization using CORBA as the underlying communication infrastructure. The service model is specified and described in more detail in a *service specification* for SCCS [SCCS98].

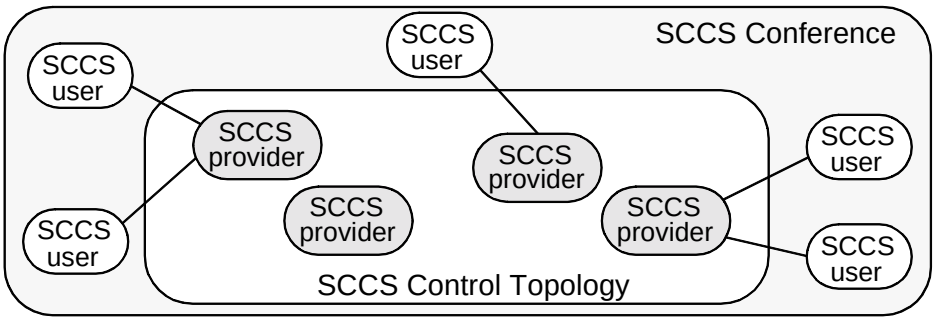


Figure 3-2: *SCCS Service Environment*

3.1.2 Conference Management

SCCS provides main features for conference management and control. This means that it offers means to create conferences locally at an SCCS provider. Other conferences may also be invited to join the current one (see chapter 3.2.9), or a conference might be actively appended to other conferences (see chapter 3.2.8). Conferences may also be splitted into two sub-conferences (see chapter 3.2.11). SCCS deals with the aspects of merging conferences (see chapter 3.2.10), e.g. due to inviting another one. Resource conflicts which may occur when merging conferences with similar resource identifiers are resolved by remapping conflicting resources. Furthermore, SCCS provides the reconfiguration of parts of the entire conference by *deleting* or *moving* SCCS providers during a running conference without destroying or blocking (parts of) the conference during the operation (called *static reconfiguration*, see chapter 3.2.15). This enables to delete providers during runtime, e.g. when the corresponding computer is shut down. Additionally, the reconfiguration of the conference may be used to optimize the environment (called *dynamic reconfiguration*, see chapter 3.2.16) resulting in faster response time, e.g. for the floor control of SCCS (see chapter 3.1.4).

Conference management functions like inviting, appending, or splitting use *voting policies* to decide whether the operation is valid or not. During creation of the conference, the voting policy of the conference is set. Two modes are supported, *conducted* and *democratic*. In the first mode, a defined *conference conductor* is asked whether a certain operation should be performed or not. SCCS provides passing the conductorship to other members during the conference. In the conducted conference mode, the conductor is also able to expel a specific member from the conference. When using the democratic voting policy, all members of the conference are asked about the validity of an operation which is performed only if consensus among all providers is given.

As a major task of conference management, SCCS provides a database of conference members, which is readable for every conference member. This database contains information, among other, like participant name and site information. Chapter 3.2.4 describes the conference database content in detail.

SCCS does not cover aspects like conference lookup and announcement, network resource management, group security, or accounting. For that functionality, existing approaches might be used, e.g. from the Internet domain (SIP [HSSR98], SAP [HPW99], or RSVP [ZDE93]).

3.1.3 Multipoint Transfer

SCCS provides simple point-to-point as well as multipoint communication in order to support

all communication patterns from 1:1, 1:n to m:n. For point-to-point communication, a conference-wide user identifier is used. For multipoint communication, a flexible *channel concept* is proposed similar to the T.120 standard (see chapter 2.2.3) supporting *public* and *private* channels being identified by a conference-wide address.

Private channels are convened by a dedicated member of the conference who may invite other members to join the channel. Members can also ask for invitation. The convener may expel members or release the channel.

Public channels can be joined by all members in the conference and are created upon request by any member. The channel is released if all members have left it.

SCCS does not provide data priorities or different transfer modes as proposed in the T.120 standard (see chapter 2.2.3). This is because SCCS does not realize the multipoint routing of user data by own mechanisms. Instead, it uses transfer facilities of underlying transport protocols. Thus, features like message sequencing or prioritizing data depend on the used protocol.

Note that the SCCS service *multipoint transfer* does not define how the transfer of data is realized on lower layers. It will be outlined that SCCS supports different protocols on the transport layer (see chapter 3.2.12). The appropriate protocol is chosen during the creation of the channel. The send/receive operations within SCCS are only interfaced to the specific transport layer. As a consequence, real-time transfer can be realized as well as the transfer of reliable user data by choosing the appropriate multicast transport protocol, e.g. RTP [RFC1889] or RTSP [RFC2326] for real-time streams. Thus, SCCS provides homogeneous multipoint transfer facilities in contrast to the H.32x systems, where the transfer of real-time and non-real-time data is provided by separate units (see chapter 2.2.1).

Non-SCCS Listener

As mentioned above, SCCS does not realize the user data transfer by own mechanisms to avoid duplicated functionality of the transport layer. As a consequence, it is possible to bypass the SCCS layer for the reception of user data, which dodges the channel concept introduced above. This causes some problems in the provision of real privacy for private channels. Thus, additional mechanisms like authentication have to be used. But these mechanisms are not provided by SCCS. Hence, the channel concept can only be seen as a basis service for the implementation of public and private transfer scenarios which have to be supplemented by other mechanisms.

But the possibility to bypass SCCS for the multipoint transfer of user data is also a chance to increase the scalability of the conferencing system. With the notion of a *non-SCCS listener* in a conference, it is possible to transfer data in SCCS which is received by alternative endsystems. For that, an SCCS conference plus an SCCS channel is created for the users in the conference using SCCS services like floor control, membership control, and so on. The multicast group is announced beforehand, using mechanisms from the Internet domain, e.g. SAP and SIP (see chapter 2.3.2). As a consequence, the multicast data on the SCCS channel can be received by far more users, the *non-SCCS listeners* who join the multicast group directly on transport level.

Thus, the entire conference consists of users who have joined the SCCS conference for using SCCS services like floor control, and *non-SCCS listeners* who have joined the multicast group directly on transport level to receive the multicast data only. Hence, SCCS provides a combination of tightly coupled environments for tight controlled conferences together with loosely coupled reception of multicast data.

3.1.4 Distributed Applications Support

For the support of distributed applications in terms of *application state synchronization* and *floor control* [DoGLA95], SCCS implements the floor control protocol proposed in [DoGLA98]. For that, a *token concept* is provided similar to the ITU-T.120 standard (see chapter 2.2.3.5). Tokens are used as an abstraction of states or access rights from (real) resources in a group communication environment, e.g. the right to send an audio stream in a videoconference. The mapping of resources or states to tokens is beyond the scope of SCCS.

Tokens in SCCS may be *grabbed* exclusively by one member or *inhibited* non-exclusively by several. The status of a token may be requested, indicating whether it is allocated exclusively or non-exclusively or whether it is free. Furthermore, the ownership may be given to another member if the token is grabbed exclusively. It is also possible to ask for an exclusive token and to inquire for the owner(s) of a token. The latter functionality is not supported in the ITU standard. However, it is an important feature when implementing e.g. shared applications [T128].

It can be seen that the proposed token concept provides *floor-asking* and *floor-passing* operations for application synchronization together with status and owner inquiry functionality. The service definition does not define how tokens are managed by the system. In chapter 3.2.13, a resource management scheme is introduced to provide floor control functionality in SCCS.

3.1.5 User Data Marshaling

Another important issue demanded in the requirements of chapter 2.1.2.1 is the transfer of *typed user data* among heterogeneous systems. Standard representations like the *Abstract Syntax Notation One* (ASN.1 [X680]-[X691]) or the *Common Data Representation* (CDR [COSS97]) are used to describe objects or data structures in a system-independent format. The CORBA environment [CORBA93] uses the *Interface Description Language* (IDL [COSS97]) to define objects and their signatures on an abstract level. For that, it uses the CDR notation for the translation of the objects into an octet stream.

Common to these approaches is that the data description, either in ASN.1, CDR, or IDL, has to be translated using appropriate tools which generate encoding/decoding functions based on the abstract notation. For instance, the ASN.1 tool SNACC [SNACC94] parses an ASN.1 description of data structures and generates C data structures and corresponding encoding/decoding routines. These functions are used at sender and receiver side to generate an octet stream from the data structure or vice versa. For this translation, the standardized *Basic Encoding Rules* (BER [X691]) are used to ensure that the resulting octet stream is the same even in a heterogeneous system.

Figure 3-3 illustrates the different steps of the problem. In the first step, the user data is defined using abstract notations like ASN.1 or CDR, while the second step generates encoding/decoding routines from this definition. The sender transfers typed user data by using the encoding functions to translate the typed data stream into an octet stream which is sent using the transfer capabilities of SCCS. At the receiver side, the octet stream is translated back to a typed data stream using the generated decoding functions.

For this purpose, SCCS proposes a *user data marshaling layer* (see chapter 3.2.1), containing the generated encoding/decoding functions. It is not in the scope of SCCS what kind of representation (ASN.1, CDR) and what kind of tools could be used.

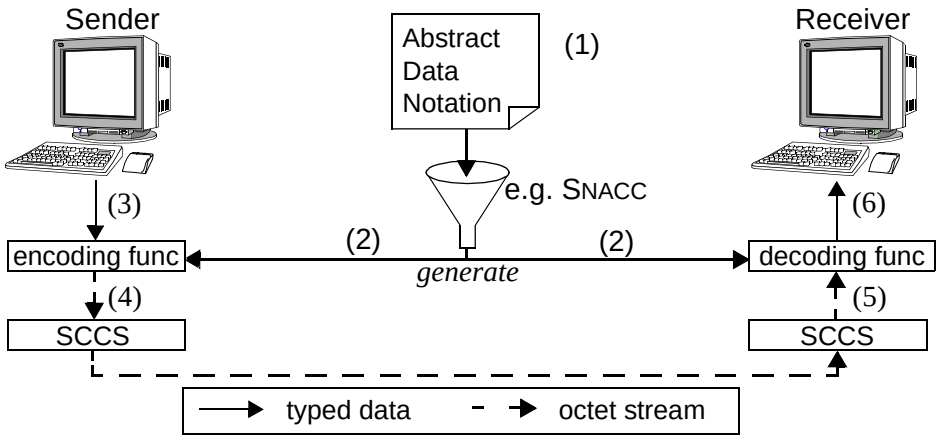


Figure 3-3: *User Data Marshaling in SCCS*

3.1.6 Service Data Units in SCCS

The services of SCCS are used by sending corresponding *Service Data Units* (SDUs) to the *Service Access Point* (SAP) of the appropriate SCCS provider. For each functionality being defined, there may exist at most 4 different types of SDUs, a *request*, an *indication*, a *response*, and a *confirm*. The `request` SDU invokes the corresponding service at the SCCS provider and is usually confirmed by a `confirm` SDU. If an SCCS service notifies an entity about a certain action, an `indication` SDU is sent to the SCCS user. Sometimes, the SCCS user has to respond to an indication, which is done by the corresponding `response` SDU.

The *Message Sequence Chart* (MSC [Z120]) in figure 3-4 shows the different types and the *SCCS Service Access Points* (SCCSSAPs) at which the PDUs are invoked. Note that an `indi-`

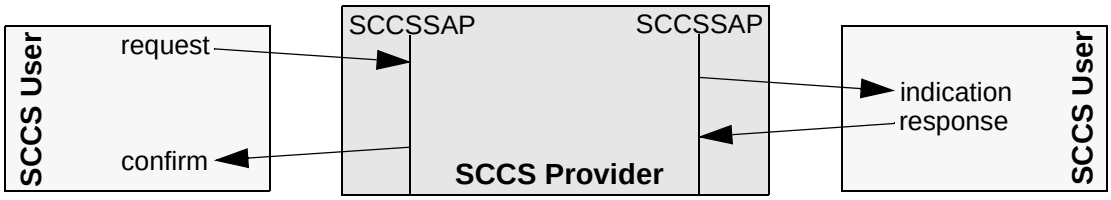


Figure 3-4: *Message Sequences in SCCS*

`cation` SDU does not necessarily need to be answered by a `response` SDU. It depends on the functionality of the corresponding SCCS service. Furthermore, the SCCSSAPs are divided in *control SAPs* and *user SAPs* [SCCS98] depending on the requested functionality.

For each SCCS functionality, the SCCSDUs, the corresponding parameter lists, and the MSCs are defined in the *SCCS service specification* [SCCS98]. In table 3-1, an overview of the different SCCSSDUs is presented separated into several *functional units*, namely:

- **conference management**: creating, joining, appending, inviting conferences
- **user management**: conductorship and expelling users from the conference
- **channel management**: for any kind of administration of private and public channels
- **token management**: for any kind of token administration
- **data transfer**: for point-to-point and multipoint transfer of user data

- **reconfiguration management**: for resolving conflicting resources during reconfiguration
- **database management**: for member database functionality

For more detailed information about SCCSSDUs, their provided functionality and processing, see [SCCS98]. If an SCCS service is invoked by an SCCS user, certain protocol mechanisms are used to provide the requested service. These protocol mechanisms are described in the next section.

Functional Unit	SCCSSDUs
Conference Management	SCCS_Conference_Create-request/confirm SCCS_Conference_Join-request/indication/response/confirm SCCS_Conference_Invite-request/indication/response/confirm SCCS_Conference_Append-request/indication/response/confirm SCCS_Conference_Split-request/indication/response/confirm SCCS_Conference_Leave-request/indication/response/confirm SCCS_Ask_Join_Permission-indication/response SCCS_Ask_Invite_Permission-indication/response SCCS_Ask_Append_Permission-indication/response SCCS_Ask_Split_Permission-indication/response
User Management	SCCS_Conductorship_Give-request/indication/response/confirm SCCS_Member_Expel-request/confirm
Channel Management	SCCS_Channel_Join-request/confirm SCCS_Channel_Leave-request/indication/confirm SCCS_Channel_Convene-request/confirm SCCS_Channel_Admit-request/indication/confirm SCCS_Channel_Expel-request/indication/confirm SCCS_Channel_Disband-request/indication/confirm SCCS_Channel_Members-request/confirm SCCS_Channel_Ask_Admit-request/indication/confirm
Token Management	SCCS_Token_Grab-request/confirm SCCS_Token_Inhibit-request/confirm SCCS_Token_Release-request/confirm SCCS_Token_Test-request/confirm SCCS_Token_Please-request/indication/confirm SCCS_Token_Give-request/indication/response/confirm SCCS_Token_Members-request/confirm SCCS_Token_Reconf-request/confirm
Data Transfer	SCCS_Data_Send_MP-request/indication/confirm SCCS_Data_Send_P2P-request/indication/confirm
Reconfiguration Management	SCCS_Reconf_Remap_Members-indication SCCS_Reconf_Remap_Channels-indication SCCS_Reconf_Remap_Tokens-indication
Database Management	SCCS_DB_Entry_Update-indication SCCS_DB_Entry_Read-request/confirm SCCS_DB_Lock-indication SCCS_DB_Unlock-indication

Table 3-1: *Service Data Units in SCCS*

3.2 Protocol Mechanisms

When invoking an SCCS service, certain protocol mechanisms or *procedure rules* [Holz91] are used to provide the requested service to the SCCS user. Problems like proposing a protocol stack, defining the used topology, outlining the database structures, building topologies, and providing services like joining, appending, or splitting conferences as well as managing resources like token and channels have to be solved efficiently.

All these mechanisms and design issues lead to a definition of an SCCS control protocol providing the SCCS service functionality. It consists of a definition of *Protocol Data Units* which are exchanged among SCCS providers to provide the services, and of a functional description of the used mechanisms. In this section, the description of the used mechanisms and a brief overview of the PDU definition is presented. For further details about MSCs of the PDUs and their exact ASN.1 definition, see [SCCS99].

3.2.1 Protocol Stack

Figure 3-5 shows the proposed protocol stack using IP and IP multicast on the network layer. SCCS supports any multicast transport protocol for reliable multicast data-transfer as well as for reliable multicast control traffic, e.g. RMTP [LiPa95], MTP/SO ([BOGTS94][BOS97]), SRMT [Schu98], RMP [WMK97]. Furthermore, RTP [RFC1889] is supported for media-streaming and a reliable unicast transport protocol (like TCP [RFC793]) is used for the unicast control traffic. It is not shown in this figure that there might be additional point-to-point communication outside the scope of SCCS. The user data conversion layer is provided by using standard encoding/decoding routines (see chapter 3.1.5).

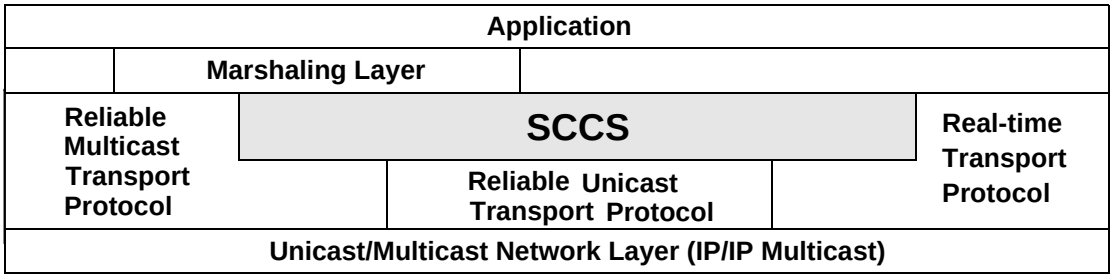


Figure 3-5: *SCCS Protocol Stack*

3.2.2 SCCS Protocol Environment

The SCCS service environment, presented in chapter 3.1.1, is implemented in the protocol environment of SCCS which is divided into the *control* and the *user data* part of SCCS.

For the former, a tree-based control topology is used. For that, *SCCS providers* establish *SCCS connections* building a non-cyclic tree with an uppermost provider in the tree, called *SCCS top provider*. The routing of most of the SCCSPDUs is done within this tree. The number of supported SCCS connections within each SCCS provider is restricted to an upper bound which depends on the implementation of the SCCS protocol in the provider. This number of provided connections is exchanged when an SCCS connection is established and is stored in the topology database of the upper provider of the connection (see chapter 3.2.4.3). When creating a conference and therefore building the tree topology, a multicast group is established associated to

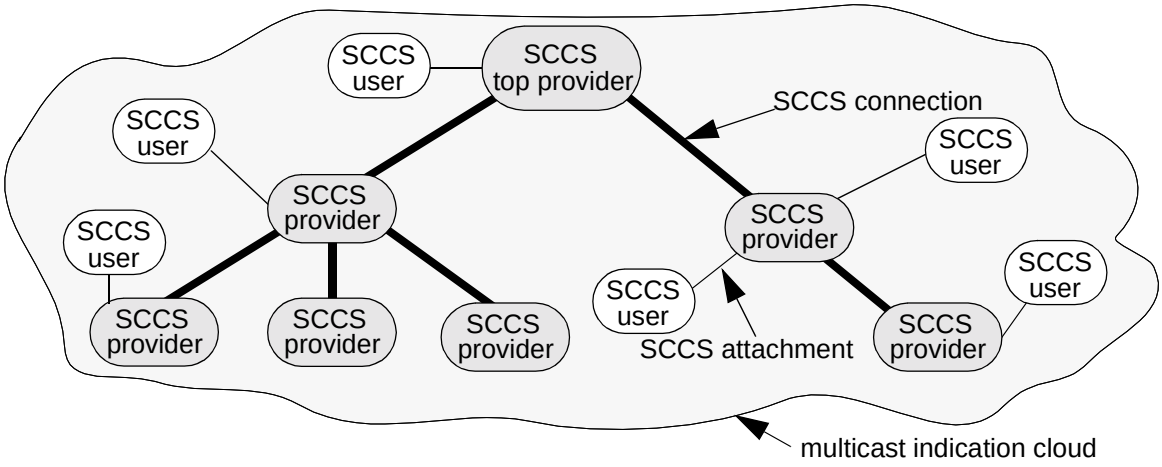


Figure 3-6: *SCCS Control Protocol Environment*

this conference. Each SCCS provider joins this multicast group when entering the conference. This multicast group, called *multicast indication cloud*, is used for some control indications and conference database refreshes (see [SCCS99]). Thus, multicast capabilities of the underlying transport layer is used for SCCS control traffic, which improves the scalability of SCCS compared to other approaches (see chapter 2.2.3). To each of the SCCS providers, an *SCCS user* may be connected establishing an *SCCS attachment* for transferring SCCSSDUs to the appropriate provider. Figure 3-6 shows an example for the SCCS control protocol environment.

For the *user data part* of SCCS, no specific transfer capabilities are established, because SCCS does not duplicate transport layer functionality. In contrary, the transfer of user data is realized by a simple mapping to a multicast group of an underlying multicast protocol (see chapter 3.2.12) if supported.

3.2.3 Services assumed by the Transport Layer

SCCS requests services for the establishment of unicast connections and for the control data transfer from an underlying unicast transport layer. Additionally, a multicast-capable transport layer (or an appropriate adaptation layer) is needed by SCCS providing services for administrating multicast groups and multicast user data transfer. This section defines the services to be provided from both multicast and unicast transport layer.

3.2.3.1 Unicast Transport Service

The unicast transport service shall offer the following features to the upper SCCS layer:

- *Establishment* of a transport connection (TC) towards another transport service provider to exchange TSDUs.
- *Transfer* of TSDUs, consisting of stream of octets, on a specific TC. This transfer is transparent in the sense that boundaries and content of TSDUs are left unchanged by the transport service.
- Unconditional *release* of TCs.

The model of the transport service is very similar to relevant parts of the ITU recommendation X.224 [X224]. Each TC is modeled by a separate pair of queues connecting two TSAPs. Each queue is limited in its capacity. Operations like connect, disconnect, or transfer of TSDUs result

in inserting or removing objects from the appropriate TC queue. These objects are removed in the same order they were added to the queue. The only exception is that an object may be removed if the following one is a disconnect object. A TC endpoint identification has to be provided locally to distinguish between TCs at a TSAP.

The unicast transport service provides functionality to compensate any error resulting from underlying networks, so this functionality does not need to be duplicated on the SCCS layer. If there is an unrecoverable error, this is signaled by a `TP_UC_Disconnect-indication` and the SCCS connection is released with a reason code `provider_initiated`.

A user initiated release of an SCCS connection results in a transfer of an appropriate SCCS-PDU, containing the reason (`user-requested`) and the graceful release of the SCCS connection.

In table 3-2, the primitives for the unicast transport service are shown.

Primitive	Parameters
TP_UC_Connect-request/indication/response/confirm	Caller address, Callee address
TP_UC_Disconnect-request	-
TP_UC_Disconnect-indication	reason code
TP_UC_Data-request/indication	user data

Table 3-2: *Unicast Transport Service Primitives*

3.2.3.2 Multicast Transport Layer

The multicast transport service shall offer the following features to the upper SCCS layer:

- *Creation* of a multicast group (MG) for the purpose of exchanging TSDUs among the members of the multicast group.
- *Joining* to an existing MG with the result of receiving TSDUs exchanged via this MG.
- *Transfer* of TSDUs, consisting of stream of octets, on a specific MG. This transfer is transparent in the sense that boundaries and content of TSDUs are left unchanged by the transport service.
- Unconditional *deletion* of an MG.

Each MG is modeled by an entity globally identified by a unique address. Multicast transport services may join this MG which is modeled by a separate pair of queues connecting the TSAP of the MG endpoint with the MG. Each queue is limited in its capacity. Transferring TSDUs to the MG results in inserting these TSDUs in all TSAPs queues which are connected to the MG. Leaving the MG by an MG endpoint means to destroy the appropriate queues of the TSAP. Deleting the MG results in destroying all queues and the MG entity itself. An MG endpoint identification has to be provided locally to distinguish between MGs at a TSAP.

The multicast transport service provides functionality to compensate any error resulting from underlying networks, so this functionality does not need to be duplicated on SCCS layer. If there is an unrecoverable error, this is signaled by a `TP_MC_Leave-indication` and the SCCS channel is released with a reason `provider_initiated`.

A user initiated release of an SCCS channel results in a transfer of an appropriate SCCSPDU, containing the reason (`user-requested`), and the graceful deletion of the MG on the local

endpoint. If the last endpoint has left the MG, it is deleted and the network resources are freed.

In table 3-3, the primitives for the multicast service are presented.

Primitive	Parameters
TP_MC_Join-request/confirm	Caller address, Group identifier
TP_MC_Leave-request/indication	Caller address, Group identifier, reason
TP_MC_Data-request/indication	Group identifier, user data

Table 3-3: *Multicast Transport Service Primitives*

3.2.4 Databases in SCCS

This section gives an overview of the SCCS information base within each SCCS provider divided in three different parts:

- *Conference Database*: contains information concerning each user in the conference
- *Resource Database*: contains information of resources allocated in the subtree below each SCCS provider
- *Topology Database*: contains information of the subtree topology below each SCCS provider

The next three subsections describe these three parts of the SCCS information base.

3.2.4.1 Conference Database

The conference database is the main information base of the conference containing details about each user in the conference. SCCS users may query this information.

The database is replicated on each site and updated using *delta refreshes* which are sent via the multicast indication cloud using the `DB_entry_update_indication`. Thus, multicast facilities of the underlying layers are used for the write requests. Each read request is confirmed by the local provider, so the data traffic of read operations is minimized as well. For access to the database, a *tag* is defined for each field of a user entry in the database. The entries are indexed by the user identifier. When requesting an entry, the user identifier 0 may be used to get the first entry in the database. Thus, the identifier 0 is not used as a real user identifier. The next entries may be enumerated with the help of the NEXT_ID tag entry, which builds a linked list of entries. Table 3-4 shows a user entry of the conference database together with the tags needed to access the fields of the entry.

Tag	Parameter	Content
USER_ID	UserID	SCCS user id of this user
NEXT_ID	UserID	SCCS user id of next user in the database
USER_NAME	User Name	name of the user
SITE_NAME	Site Name	name of the node

Table 3-4: *Conference Database in SCCS*

Tag	Parameter	Content
ADDRESS	Network Address	network-specific address of the node
USER_DATA	User Data	user-(or application-)specific data

Table 3-4: *Conference Database in SCCS*

3.2.4.2 Resource Database

The resource database is a local database stored in each provider and contains information about the resources (users, channels, and tokens) which are allocated in the subtree below the provider. For the users, only the identifier is stored. The user information (like address, name) is stored in the conference database. The resource database is mainly used for routing resource requests, e.g. for tokens, within the tree topology based on the rules of the proposed resource management scheme of SCCS (see chapter 3.2.13). This database has to be updated upon

- a new resource request
- a merge operation, i.e. by remapping identifiers
- a reconfiguration of the tree, i.e. by updating the resources due to a changed subtree

Table 3-5 shows a resource entry of the resource database, containing a pointer to the (resource) specific entry. The specific user entry is given in table 3-6, the token entry is shown in table 3-7, and the channel entry is presented in table 3-8. In contrast to the conference data-

Name	Parameter	Content
TYPE	USER, CHANNEL, TOKEN	type of resource
ENTRY	ChannelEntry/TokenEntry/UserEntry	pointer to entry describing specific resource

Table 3-5: *Resource Database Entry for a Resource*

Name	Parameter	Content
ID	UserID	identifier of the user
LINK	Integer	link number where user is located in the subtree below

Table 3-6: *Extended Resource Database Entry for a User Resource*

Name	Parameter	Content
ID	TokenID	identifier of the token
EXCLUSIVE	boolean	flag whether exclusive token (true) or not (false)
STATUS	TokenStatus	current status of token
LINK	Integer	link number(s) where token is used in the subtree below
TIMESTAMP	Integer	timestamp of last received timeout
TIMEOUT	Integer	timeout value for fast token give operation

Table 3-7: *Extended Resource Database Entry for a Token Resource*

Name	Parameter	Content
ID	ChannelID	identifier of the channel
PRIVATE	boolean	flag whether private channel (true) or not (false)
CONDUCTOR	UserID	SCCS user id of conductor if channel is private
GROUPNAME	Channel Name	string representation of underlying multicast group
PROTOCOL	Protocol ID	identifier of underlying protocol to be used
LINK	Integer	link number(s) where channel is used in the subtree below

Table 3-8: *Extended Resource Database Entry for a Channel Resource*

base, there is no entry to build a linked list. This is because the implementation of the resource database is not within the scope of SCCS.

3.2.4.3 Topology Database

The topology database is a local database stored in each provider and contains information about the subtree below that provider. This database is mainly used for reconfiguration of tree topologies (see chapter 3.2.15). For a network-independent representation of the providers, a *site number* is used which is assigned during joining a conference (see chapter 3.2.7). Each provider knows the site numbers of the subtree below itself and describes the topology of the subtree by a context-free grammar based on the EBNF (*extended Backus-Naur form*) notation [ISO14977]:

```
Node ::= sitenr
Tree ::= Node [ ( Tree { ,Tree } ) ]
```

Additionally, the SCCS top provider stores the number of supported links of each provider. This number is given when a provider joins the conference (see chapter 3.2.7).

Figure 3-7 shows an example of the topology databases within each provider of the conference. The information stored in each provider can be determined from the overall information in the SCCS top provider, which can be seen in figure 3-7 (the topology information in the leaf nodes is not shown). Table 3-9 shows the entries of the topology database. With the NEXT_SITES entry, a linked list is build according to the grammar above. Note that the information about the maximum link number in each site is only stored in the top provider (node 12 in figure 3-7).

Tag	Parameter	Content
SITE	SiteNr	number of current site
NEXT_SITES	SiteNr	list of next sites on the next level in the subtree below current site
LINK_NUMBER	Integer	number of provided links

Table 3-9: *Topology Database Entry for a Site*

3.2.5 Protocol Data Units in SCCS

After invoking a specific service of SCCS, a corresponding PDU may be sent, if necessary, from the *Transport Service Access Point* (TSAP) of the SCCS provider to the TSAP of the superior or successor SCCS provider in the tree topology. Similar to the SDUs of SCCS on ser-

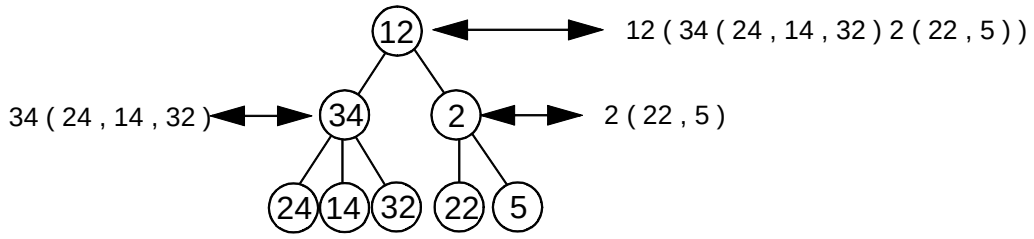


Figure 3-7: *Topology Database Entries*

vice level, there exist 4 different types of PDUs (see chapter 3.1.6) which are abbreviated with *rq* for request, *cf* for confirm, *in* for indication, and *rs* for response.

Table 3-10 gives an overview of the PDUs separated in functional units similar to chapter 3.1.6. For further details about the exact definition of the PDUs in ASN.1 notation and the MSCs of the exchanged PDUs, see [SCCS99]. It is worth mentioning that some of the PDUs are sent using the multicast indication cloud to improve scalability.

Functional Unit	SCCSPDUs
Conference Management	Connect_rq/cf ConferenceJoin_rq/cf ConferenceLeave_rq/cf ConferenceInvite_rq/cf ConferenceAppend_rq/cf ConferenceSplit_rq/cf AskInivitePermission_rq/cf AskAppendPermission_rq/cf AskSplitPermission_rq/cf AskJoinPermission_rq/cf
User Management	GiveConductorship_rq/cf/in/rs MemberExpel_rq/cf
Channel Management	ChannelJoin_rq/cf ChannelLeave_rq/cf ChannelConvene_rq/cf ChannelDisband_rq/cf ChannelAdmit_rq/cf ChannelExpel_rq/cf ChannelAskAdmit_rq/cf/in/rs
Token Management	TokenGrab_rq/cf TokenInhibit_rq/cf TokenGive_rq/cf/in/rs TokenPlease_rq/in TokenRelease_rq/cf TokenTest_rq/cf TokenMembers_rq/cf/in/rs
Data Transfer	SendData_rq/in

Table 3-10: *Protocol Data Units in SCCS*

Functional Unit	SCCSPDUs
Database Management	DB_entry_update_in DB_lock_in DB_unlock_in
Reconfiguration Management	MergeDB_rq/cf UpdateBC_rq/cf/in Reconf_Remap_Channels_in Reconf_Remap_Tokens_in Reconf_Remap_Members_in Reconf_Start_rq/cf Reconf_Update_rq/in/cf Reconf_Delete_rq/cf Reconf_Frequency_rq/cf Reconf_LinkTime_rq/cf

Table 3-10: *Protocol Data Units in SCCS*

3.2.6 Building Topologies

For some operations within SCCS, the topology must be extended by a new provider, e.g. when joining, inviting, or appending a conference. Therefore, a sequence of SCCSPDUs has to be exchanged among different providers, which is shown in this section. Note that rebuilding a topology due to a reconfiguration does not fall into the category of building topologies, because all providers are known, and the topology is well established (no cycles) when reconfiguring.

The way SCCS builds its topology is very simple. A new provider connects to the existing topology by establishing a connection on transport level. After that, a `Connect` primitive is exchanged among the connecting and connected provider. If there is already an upward connection in the lower provider, the topology is not built to avoid cycles. Figure 3-8 shows an

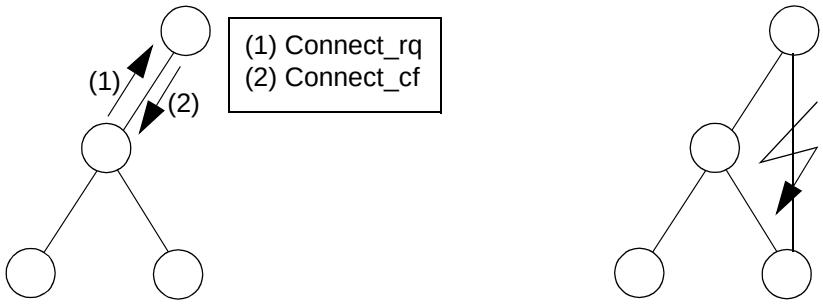


Figure 3-8: *Building SCCS Topologies*

example for an append operation of a three node subtree. The left part is a successful extension, while the right part shows a cycle in the topology with a suppression of the connection establishment.

3.2.7 Joining Conferences

Joining on service level is quite different from joining on protocol level. On service level, the join operation may be a simple join of a user to a conference which is already hosted on the local provider. But it may also be a shortcut for creating a local conference and then appending

to another conference hosted on a remote host. The second join operation can be divided in two steps:

- build the extended topology constisting of the joined conference and the new provider
- join the conference

The second step is similar for the service level shortcut and the ordinary join operation if the conference already exists on the local provider. For the first step, the topology building mechanism explained in chapter 3.2.6 is used. In the following, the join operation to an existing conference (step 2) is explained.

The `ConferenceJoin` request is routed upward to the SCCS top provider of the joined conference. If the provider of the joining user does not have an assigned site number, it sets its own number to 65535 and forwards the tree entry. During propagation of the request to the SCCS top provider, each provider along the path extends the *tree entry*. Hence, the SCCS top provider gets the entire information of the extended subtree used for its topology database. Each provider along the propagation path stores the new extended subtree below itself. If the provider of the joining user has already an assigned site number, it sets its own number to 65534 in the tree entry. This entry is used to indicate that the tree topology is unchanged and a tree entry does not need to be forwarded. Additionally, the number of provided links at the connecting provider is forwarded in the request, and it is stored in the topology database of the SCCS top provider.

Depending on the voting policy, the top provider asks for admission to join (for instance asking the conductor or all members). When this operation has been realized successfully, the `ConferenceJoin` request is confirmed successfully, and the assigned user identifier and site number is forwarded back to joined provider. If the temporal site number 65535 was used, it is replaced during propagation of the confirmation by the assigned site number.

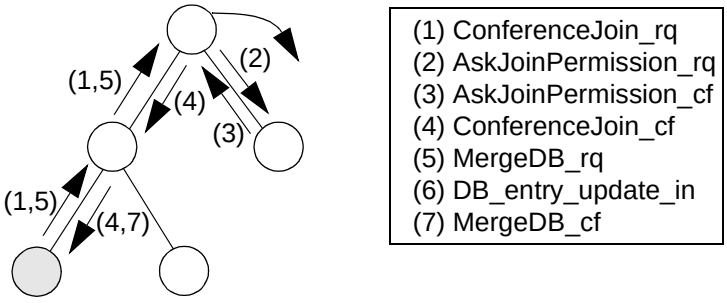


Figure 3-9: *Joining Conferences in SCCS*

The database of the joined provider which consists only of the local participant information is sent to the SCCS top provider (`Merge_DB_rq`). After updating the database, the top provider indicates the conference database update by sending a `DB_entry_update_in` to the multicast indication cloud. The superior provider of the joined one confirms the database merging with a `Merge_DB_cf`.

If the join operation was not successful, the request is confirmed with an error code. If a new connection was established during the operation, the upper provider in the topology destroys this new connection.

Figure 3-9 shows the second step of the join operation in the conducted conference case, which means that the `AskJoinPermission` request is sent only to the conductor of the conference.

3.2.8 Appending Conferences

Appending conferences in SCCS is initiated by the SCCS top provider of a lower conference A which might be appended to another upper conference B. Only a top provider is allowed to invoke that operation. The append operation can be divided into four steps:

1. ask for appending in conference A
2. build the new topology
3. ask for appending in conference B and confirm the operation
4. merge both conferences by remapping the identifiers

First, the SCCS top provider of conference A asks for permission to append to conference B according to the voting policy of conference A. If this is confirmed, the new topology is built as described in chapter 3.2.6 by establishing a new connection from the SCCS top provider of conference A to any provider of conference B. After that, the SCCS top provider of conference A requests for appending. The request is forwarded to the SCCS top provider of conference B. According to the voting policy of conference B, the top provider asks for appending in its conference. If this is confirmed positively, the append request is confirmed back. In the last step both conferences are merged (see chapter 3.2.10).

Figure 3-10 shows the steps 1 and 3 as described above. Both conferences are conducted for

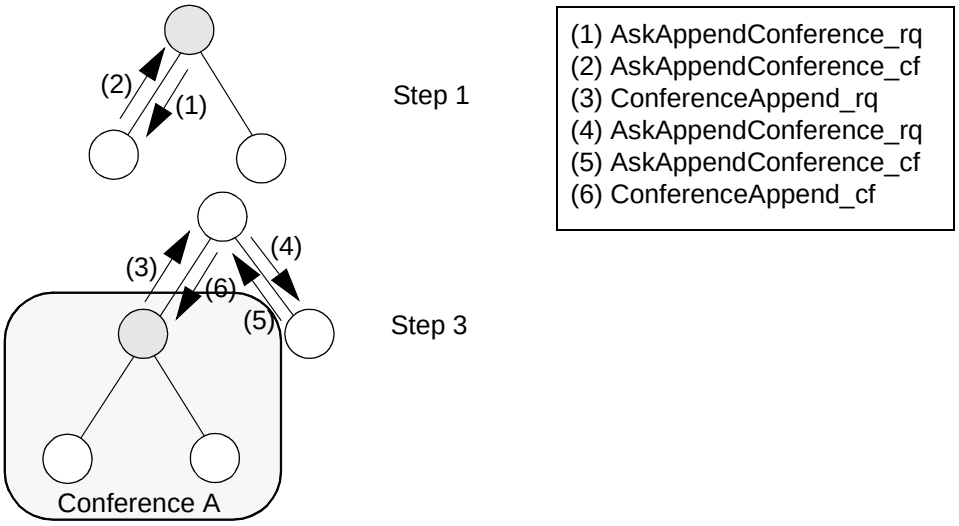


Figure 3-10: *Appending Conferences in SCCS*

simplicity of the picture. The steps 2 and 4 are performed according to chapter 3.2.6 and chapter 3.2.10.

3.2.9 Inviting Conferences

Inviting conferences in SCCS is initiated by any provider of an upper conference A which wants to invite a lower conference B. The peer provider of conference B must be the top provider of that conference. The invite operation can be divided in four steps:

1. ask for inviting in conference A
2. build the new topology

3. ask for inviting in conference B and confirm the operation
4. merge both conferences by remapping the identifiers

First, the provider of conference A, which wants to invite conference B, requests for invitation

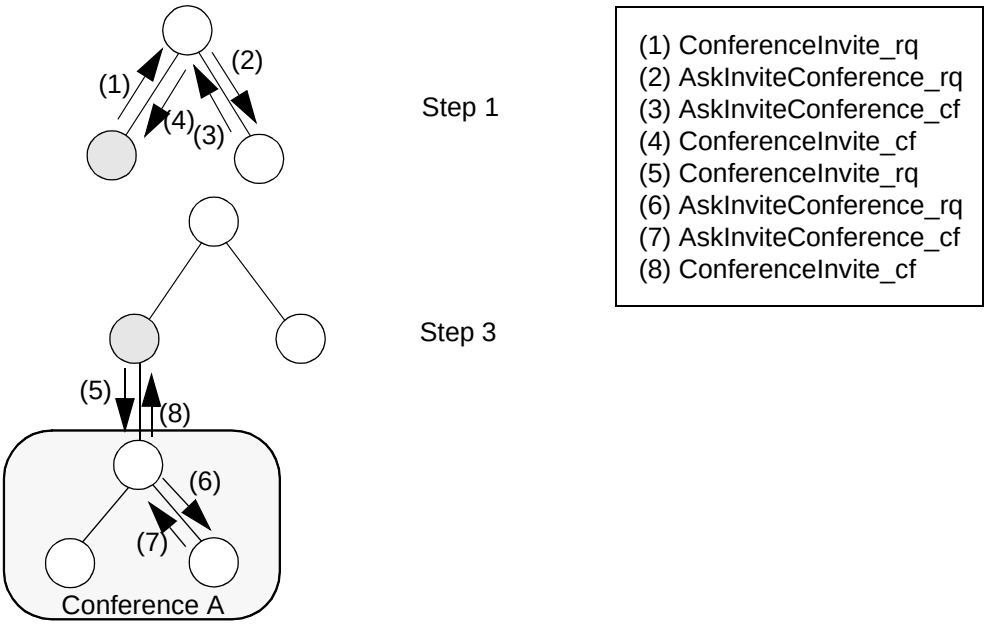


Figure 3-11: *Inviting Conferences in SCCS*

by sending a `ConferenceInvite` to the SCCS top provider of conference A. The top provider asks for permission to invite conference B according to the voting policy of conference A and confirms the `ConferenceInvite`. After a positive result, the new topology is built according to chapter 3.2.6 by establishing a new connection from the inviting provider of conference A to the top provider of conference B. After that, the SCCS top provider of conference B requests for inviting according to the voting policy of conference B. If this is confirmed positively, the invite request is confirmed back. In the last step, both conferences are merged (see chapter 3.2.10).

Figure 3-11 shows the steps 1 and 3 as described above. Both conferences are conducted for simplicity of the picture. The steps 2 and 4 are performed according to chapter 3.2.6 and chapter 3.2.10.

3.2.10 Merging Conferences

Merging of conferences is a feature of SCCS to append or invite conferences. The merging mechanism provides resource resolution mechanisms by remapping conflicting resources (user, channel, and token identifiers) during the merge operation. Merging is done during two main operations in SCCS

- when appending a lower conference to an upper or
- when inviting a lower conference from an upper.

First, the merging operation is explained regardless of the operation which invoked the merging. In the second step, the starting point of the merging operation is defined with respect to the calling operation.

In figure 3-12, the PDUs are presented which are exchanged during the operation. The steps for

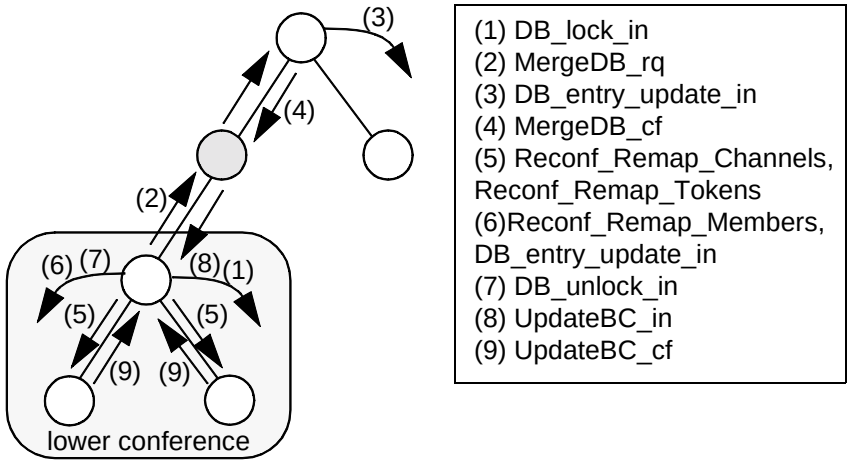


Figure 3-12: *Merging Conferences in SCCS*

performing the operation are as follows, using the numbering of figure 3-12:

1. The conference database of the lower one is locked using a `DB_lock_in`.
2. A `MergeDB_rq` is sent from the SCCS top provider of the lower conference to the SCCS top provider of the upper conference containing the entire conference database and all channel and token identifiers of the lower conference.
3. The new merged conference database is sent via the upper multicast indication cloud with a `DB_entry_update_in` in the upper conference.
4. The upper SCCS top provider determines the conflict resources of the merged conference and maps them to other identifiers. The remapped token and channel identifiers are sent back to the SCCS top provider of the lower conference. During the propagation downward, the provider to which the lower conference is connected extends the `MergeDB_cf` by the new database of the merged conference.
5. The top provider of the lower conference generates a `Reconf_Remap_Channel_in` and `Reconf_Remap_Token_in` for the appropriate subtrees containing the remapped resources.
6. The remapped user identifiers and the new merged database are sent via the multicast indication cloud of the lower conference using a `Reconf_Remap_Member_in` and `DB_entry_update_in`.
7. The merged conference database is unlocked.
8. The SCCS top provider of the lower conference invokes an `UpdateBC_in` via the multicast indication cloud to urge the providers of the lower conference to join the new multicast indication cloud of the merged conference.
9. Each provider of the lower conference confirms the join operation upward with an `UpdateBC_cf`, which is accumulated on the upward path to the SCCS top provider of the lower conference.

Note that after sending the `MergeDB_rq` upward in the second step, all resource requests (e.g. for tokens or channels operations) in the lower conference are sent upward to the SCCS top provider of the merged conference to ensure consistency of the merged databases.

The starting point of the merging operation is different in the *append* and *invite* case. When inviting conferences, the merging operation is invoked after sending the `ConferenceInvite_rq` back to the upper conference. When appending conferences, the merge operation is invoked after receiving the `ConferenceAppend_rq` from the upper conference.

3.2.11 Splitting Conferences

Splitting of conferences is a feature in SCCS to allow subtrees within the topology to be splitted off from the rest of the conference. For that, the top provider of this subtree has to initiate the split operation by sending a `ConferenceSplit_rq` to the SCCS top provider. Depending on the voting policy of the entire conference, the SCCS top provider asks for splitting the conference. In the next step, the split request is confirmed to the SCCS top provider of the splitted lower subconference. This provider disconnects from its upper provider. The SCCS top provider of the lower conference asks all members of the lower subconference to join a new multicast indication cloud. It should be noticed that, in contrast to merging a conference, an `UpdateBC_rq` is sent via the tree topology. This is because the multicast indication cloud cannot be used because ALL providers of the old conference are still listening. Hence, it is not clear what provider should join the new multicast indication cloud. All providers of the lower subtree confirm the update by sending back an `UpdateBC_cf`, which is accumulated during propagation back to the SCCS top provider of the lower conference.

In the last step, the SCCS top providers of both subconferences free all 'useless' resources by disbanding, i.e. releasing, private channels or freeing tokens. For instance, if the convener of a private channel is located in the lower subconference, all joined channel members in the upper subconference are notified by a `ChannelDisband_in` that the channel is not longer valid in this subconference.

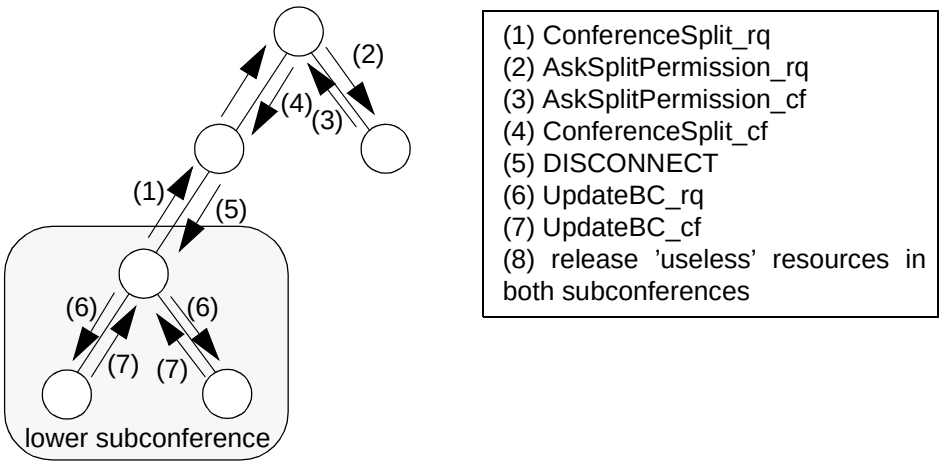


Figure 3-13: *Splitting Conferences in SCCS*

Figure 3-13 shows the sequence of the PDUs when splitting a conference. In this example, only the conductor is asked for splitting.

3.2.12 Data Transfer

The SCCS layer provides two modes of data transfer, *multipoint* or *point-to-point*. Point-to-point data to a single user is routed directly using the tree topology of SCCS. The data is transferred using the shortest route within the tree. This point-to-point transfer is mainly used for notifications within application protocols.

For the multipoint transfer, SCCS maintains a group member list for each multipoint channel providing private channels or queries for group members. The data is transferred using the underlying multicast transport protocol, which is determined during creation of the channel. The SCCS provider interfaces the incoming data to the different stub routines for the appropriate protocol. Different stub routines for several protocols may be supported by the local SCCS provider, which is not within the scope of the service definition. If at least one local user has joined a specific channel, the local SCCS provider joins the corresponding multicast group and acts as a receiver for multicast data within this group. The data is transferred from the local SCCS provider to the appropriate user by local attachments.

SCCS provides error-free reception of point-to-point data, as long as the source and destination remains attached to the conference. The reliability of the multipoint transfer depends on the used multicast transport protocol.

3.2.13 Resource Management Scheme

This section presents the resource management scheme of SCCS. *Resources* in the context of conferencing environments like SCCS are *tokens* (as an abstraction of states or access rights to real resources) and *channels* (for the multipoint transfer). While operations for channels like joining or creating channels are not that time-critical, operations for tokens are used to realize the floor control within SCCS. As stated in section 2.1.2, the response time of these operations is very critical for the acceptance by the user. As a consequence, a proper design of the used resource management scheme is crucial for the usability of the conferencing system especially in large scaled scenarios where large delays might occur.

Centralized resource management schemes, e.g. used in the T.120 standard [T120], are very inefficient in hierarchical topologies, because each resource request has to be sent upward within the tree topology to the top provider resulting in high response times for the request (see also chapter 2.2.4). On the other hand, due to the temporary inconsistency and the additional complexity, a *distributed* scheme, e.g. used in the X.500 directory [X500], is not suited either. Furthermore, the usage of tokens for inconsistency avoidance which are temporary inconsistent themselves is not useful. [OnSch93] and [TPH98] give an overview of resource management schemes in tightly coupled environments.

SCCS uses an improved resource management scheme developed by the author which results in higher performance of resource requests. In this scheme, after a successful allocation of a specific resource in the SCCS top provider, routing information is distributed along the path from the SCCS top provider to the corresponding SCCS provider where the resource was allocated. This routing information is used for subsequent requests concerning this resource by sending the requests upward in the tree only until a provider is reached being able to generate a confirmation or indication. The requests are replied directly, if possible, instead of forwarding the request up to the SCCS top provider. State changes (e.g. from allocated to free or vice versa) are only performed by the SCCS top provider, but not by any intermediate SCCS provider. Thus, the routing finally leads to the SCCS top provider, so the routing scheme terminates.

The scheme can easily be demonstrated using the example in figure 3-14. In the following, it is assumed that the requested resource is an SCCS token. Operations for channels are even simpler, because the channel operations are less complex, but the used routing is the same.

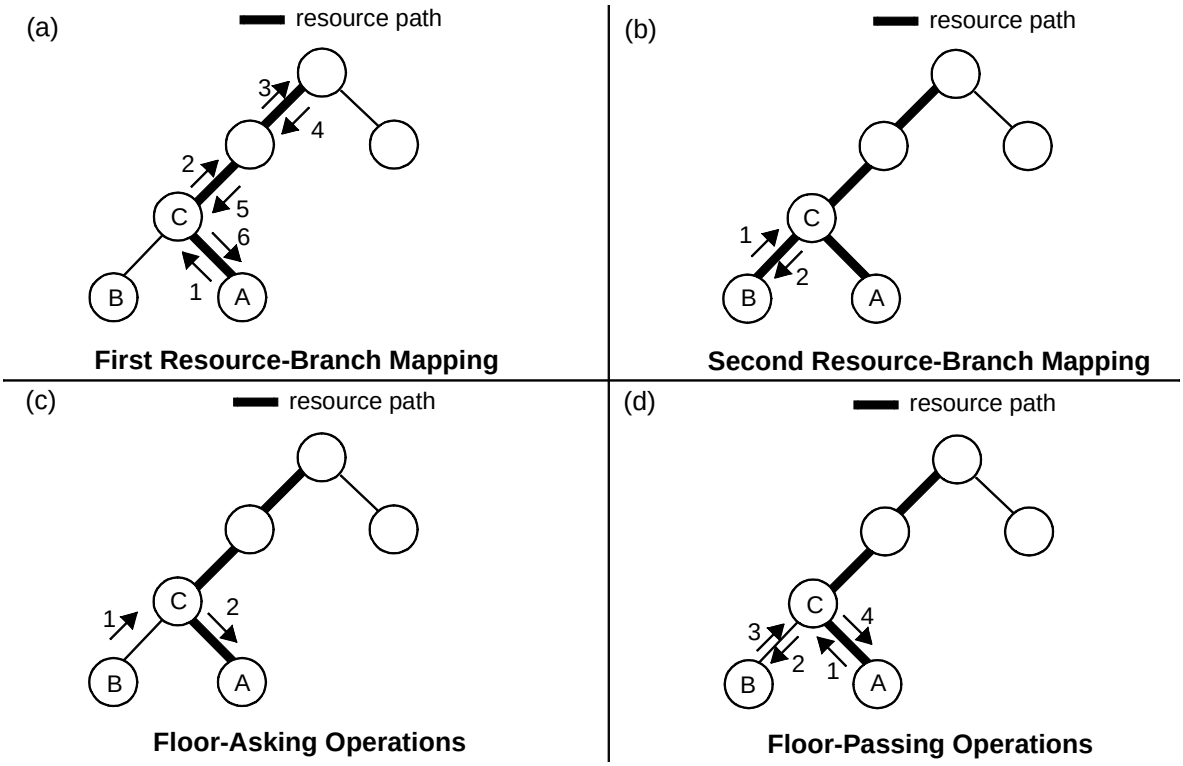


Figure 3-14: *Resource Management Scheme in SCCS*

It is assumed that a specific token is not yet allocated in the conference. When a specific SCCS user allocates the token, the request is sent upward in the tree. Due to the fact that the token is free, the request is sent to the SCCS top provider. The request is confirmed positively (the token is not yet allocated !) and sent back downwards in the tree. During the propagation of the confirmation, a *resource-branch mapping* is determined within each intermediate provider along the return path. With this mapping, the information is stored what resource is used in which subtree of the provider. It is not stored, *what* SCCS user has allocated the resource. Therefore, each provider holds an internal resource database for each resource (see chapter 3.2.4.2) which is used for further requests. When the confirmation has reached the requesting provider, a *resource path* has been established for this specific resource (see figure 3-14 (a)). Thus, the routing information for further requests is distributed among the tree implicitly during the confirmation propagation. That is why the scheme is called *Implicit Information Distribution* (IID), when proposed by the author in [TPH98].

It is also possible to establish resource paths to several providers. For instance, if a non-exclusive token is requested, the second resource path is established using the information of the first. This can be seen in figure 3-14 (b). Provider B allocates the (non-exclusive) token, after it was allocated by provider A before. The request is routed to provider C who is able to confirm the request immediately, because the information is stored in its resource database (see chapter 3.2.4.2) that this resource is non-exclusive and already allocated in the right subtree. Therefore, the request is not forwarded to the SCCS top provider anymore and the response time of this request is much faster.

Requests for *floor-asking* and *floor-passing* operations are also improved in terms of the response time using the proposed routing scheme. In most group communication scenarios, these floor-asking/-passing operations occur after initially allocating the floor. If provider B in figure 3-14 (c) asks for the specific (exclusive) token, the request is routed to provider C. This provider is able to route the request immediately to provider A instead of routing the request upward to the SCCS top provider as it is done in pure centralized schemes like the T.120.

Giving a token from one provider to another results in a (partial) refresh of the resource path, which can be seen in figure 3-14 (d). Provider A gives the token to provider B. The request is immediately routed to provider B by provider C using its routing information. When responding the token give request (this operation is a two-way operation), the resource path information in provider C is changed to the left subtree, i.e. the routing information in the resource database is changed.

It can be seen that the resource management scheme of SCCS consists of a central database in the SCCS top provider to ensure consistency of the resource identifiers together with a routing scheme which uses routing information distributed implicitly during the allocation of the resource. The usage of this scheme results in higher performance for resource requests and location updates in terms of response time which is very important for token operations. In the example of figure 3-14, it can be seen that the response time can be reduced especially if the resource requests occur locally in the tree topology. But also non-local resource requests are improved. The evaluation of the resource management scheme in chapter 5 compares the performance of the proposed scheme with the centralized scheme used in the T.120 standard.

As a further advantage, the complexity of the proposed scheme is not much higher than using the pure centralized scheme. The resource-branch mapping which is used as the routing information is stored during the propagation back to the requesting user. This information can easily be extracted from the confirmation message. When routing resource requests, an additional lookup in the resource database has to be done compared to simply forwarding the request. The complexity of this operation is of linear order. The proposed scheme was implemented in the demonstration implementation of SCCS (see chapter 3.3). It can be seen from that implementation that the resulting forwarding times are only slightly higher than the observed values of the T.120 scheme (see chapter 2.2.3).

Due to the resource-branch mapping in this scheme, there exists routing information which is distributed among the providers along the resource path (stored in the resource database). This information has to be updated during a reconfiguration of the topology as a major task of the static reconfiguration which is presented in chapter 3.2.15.

3.2.14 Fast Token Give

SCCS provides a *fast token give* operation, proposed by the author, which allows to shorten the response time of a complete token please/give operation. This operation sequence is very important for the floor control within group communication applications, because typically a token is requested from the current token holder and then given to the asking user.

Normally, an application would invoke a `TokenPlease_rq` which is signaled to the token holder by a `TokenPlease_in`. If the token holder wants to give the token away, the token holder invokes a `Token_Give_rq` being routed to the former application which responds the request. Therefore, this mechanism implements a *three way handshake* for a complete token pass operation.

To improve this operation, SCCS provides a fast token give operation as shown in figure 3-15.

For that, application B may specify a time-out in the `TokenPlease_rq`. This value is stored in the resource database (see chapter 3.2.4.2) along the propagation path of the request/indication of the token please operation. This time-out value has to take the distance between floor asking and floor holding entity into account. It is not within the scope of SCCS how to set this time-out. Normally, the value would be chosen large enough to be sufficient even for large scaled environments, e.g. several seconds.

If the token holder A answers (by sending a `TokenGive_rq`) within the given time period, the current token owner is directly changed to B without waiting for a response. So application B does not need to respond to the `TokenGive_in`. As a consequence, the asking application is not allowed to refuse the token. Due to the fast token give operation, application B becomes token owner if the previous token owner A is willing to pass the token. If there are further

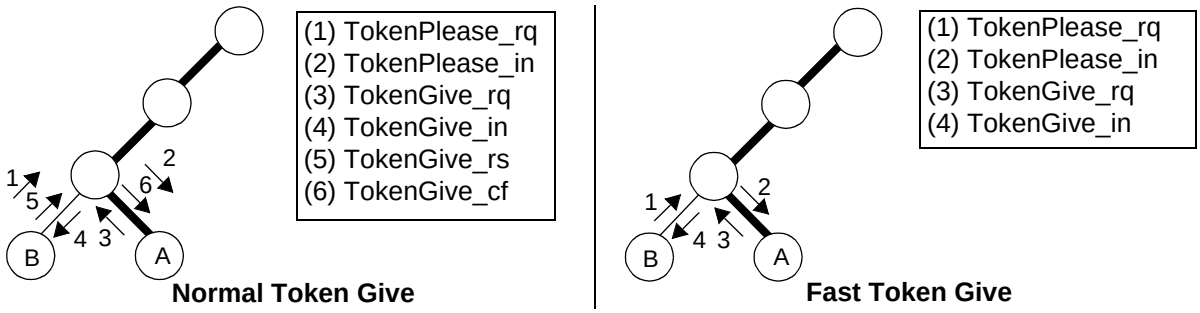


Figure 3-15: *Fast Token Give Operation*

requests for the token, while a fast token give operation is in progress, only the first request is served assuming a FIFO ordering of the token requests at the current token owner. Figure 3-15 shows the difference of the fast token give operation compared to the three way handshake scheme. It can be seen that the response/confirm messages are deleted. Hence, this mechanisms leads to a significant reduction of the entire response time in typical *floor ask/floor pass* scenarios.

3.2.15 Static Reconfiguration of Conferences

SCCS providers may totally be re-ordered by invoking a *reconfig* operation. With this operation, providers and complete subtrees may be deleted or removed from the topology. The new topology is determined by the top SCCS provider, building the *reconfig* messages for its subtrees and so on. After the determination of the new tree, a (static) reconfiguration mechanism has to build the new topology with the constraints

- to avoid inconsistencies and
- to avoid blocking (parts) of the conference during reconfiguration.

The reconfiguration scheme in SCCS fulfils these constraints and offers a complete non-blocking re-ordering of providers in the tree topology. A first version of the scheme is proposed in [Ei99]. The reconfig operation provides two sub-operations: *delete* and *move*. In both cases, the existing tree of providers has to be rebuilt according to the new topology. Deleting a provider from the tree means to build a new tree without this provider. Moving a subtree (consisting of one or several providers) means to delete the subtree from its original position and to integrate it at a new position in the tree. A delete operation can easily be realized by moving the subtree from below the provider to be deleted to the upper provider and deleting the single node.

In the following description of the reconfiguration, the algorithm is restricted to the *move* operation of one subtree within the topology to simplify the description.

The reconfiguration is divided in four steps to avoid blocking the conference:

1. Initiation of the reconfiguration
2. Establishing the new topology in parallel to the old one
3. Updating the databases
4. Switching to the new topology

In the following, all four steps are explained to clarify the reconfiguration mechanism.

3.2.15.1 Initiation of the Reconfiguration

The reconfiguration operation may be initiated by any provider in the tree. The operation is forwarded to the top provider who has the complete information of the tree. The decision, whether the operation is valid, depends on the *voting policy* chosen for the conference.

After this decision, the top provider determines the new topology based on the topology information stored in each provider (which is collected when building the topology) and sends `Reconf_Start_rq` messages to the appropriate providers in the tree. Only providers which have to start reconfiguration operations are involved in the reconfiguration operation, i.e. receive a `Reconf_Start_rq` message. When a provider along the message path receives such a message, it determines the new future subtree. If there are changes in its subtree, the provider generates a new `Reconf_Start_rq` message which is forwarded to the lower subtree. The transfer of `Reconf_Start_rq` messages is divided in two steps. First, the `Reconf_Start_rq` messages for providers to which a new subtree is moved are transferred. These messages are confirmed by the appropriate providers. After that, the messages for the providers which are moving are transferred. The confirmation is delayed until the connection establishment was successful (see next section). Otherwise, the message is confirmed with a negative result. In figure 3-16, an example is given. Provider B has to be moved to provider A.

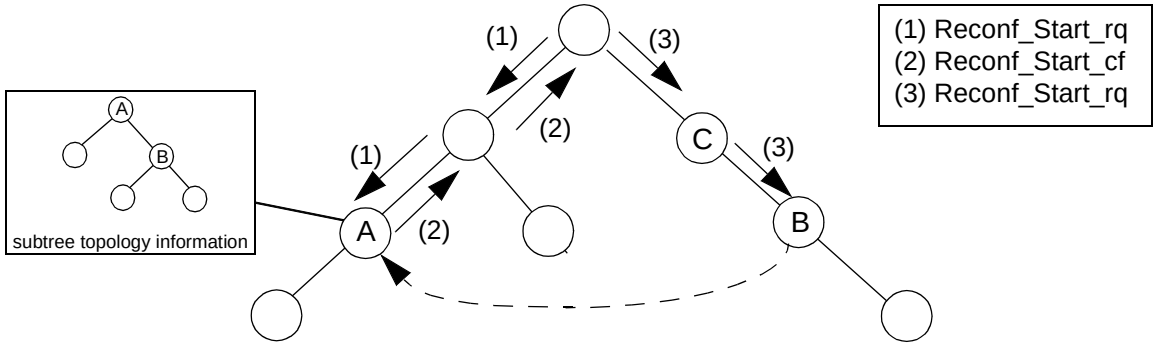


Figure 3-16: *Initiation of the Reconfiguration*

Thus, the top provider sends two `Reconf_Start_rq` messages. The first one is send to provider A consisting of the address of B and the indication to wait for a move. During the propagation of the `Reconf_Start_rq` message to A, the providers along the message path determine the new subtree. Provider A confirms the message to the top provider with a `Reconf_Start_cf`. The second message is sent to provider B consisting of the address of the new upper provider A and an indication to move. The subtree configuration of the new

topology is updated in all providers along the reconfig message paths and B.

3.2.15.2 Establishing the New Topology

The new topology is built in parallel to the old one to avoid blocking the conference. When a provider receives a `Reconf_Start_rq` message with a move indication, it establishes a connection to the upper provider given in the message. After the connection establishment, the moved provider sends a `Reconf_Start_cf` to the top provider via the new connection to indicate the completion of the connection establishment. In our example shown in figure 3-17, provider B establishes a connection to A and sends a `Reconf_Start_cf` to the top provider using the new connection. These messages are collected in each provider if there are several outstanding `Reconf_Start_cf` in its subtree. When the provider has received all `Reconf_Start_cf`, a cumulative confirmation is given to the superior provider. After the top provider received all outstanding `Reconf_Start_cf`, the new topology establishment is finished and the update of the databases begins.

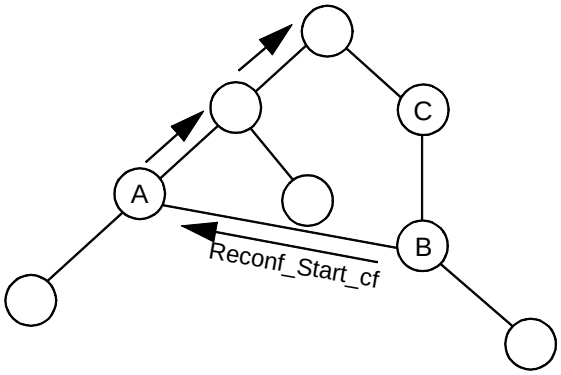


Figure 3-17: *Establishing the New Topology*

During this phase, all requests are sent using the old topology. The new one is only prepared for reconfiguration, but it is not yet used.

3.2.15.3 Updating the Databases

In SCCS, there are three different databases (see chapter 3.2.4) which have to be updated due to the reconfiguration:

- Topology database
- Conference database
- Resource database

The first one is stored in each provider containing information of the subtree below the given provider. This information is updated implicitly during the propagation of the reconfig messages. Figure 3-16 shows the topology database of provider A and C.

The second database is updated when a delete operation is invoked. For that, the multicast indication cloud is used. A *move* operation does not require the conference database to be updated (because the database stores only member and application information like network address or user name which is not changed by a move operation).

The third database is the most important one, because the resource paths of the SCCS resource

management scheme (see chapter 3.2.13) are stored in this database. The information in this database is distributed among the providers. Due to the topology reconfiguration, the resource paths may change and have to be updated which is shown with an example in figure 3-18.

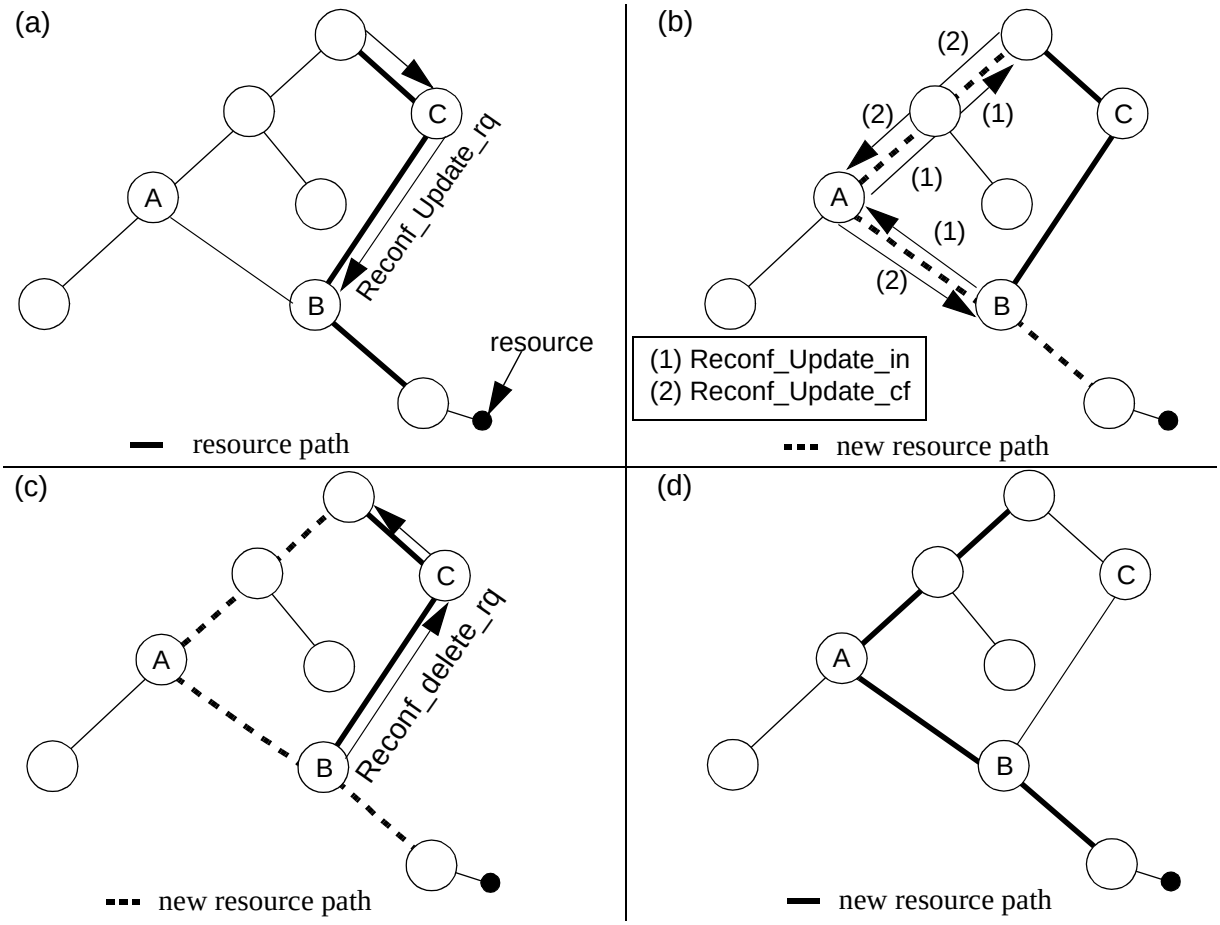


Figure 3-18: *Updating the Databases*

After the establishment of the new topology, the top provider sends a `Reconf_Update_rq` message in the subtrees which are reconfigured with the indication to update the resource database (figure 3-18 (a)). For that, the old topology is used. The providers along the forwarding path of that message switch the future routing of resource requests to the new subtree when the message is passed.

In figure 3-18 (b), the providers with a new upper node register the resources in their new subtree without an outstanding request (e.g. an outstanding token request) by sending the resource information (using `Reconf_Update_in`) upward. This message uses the new connection until a provider is reached being able to confirm the new registration. The registration cannot be denied because the resource path is only re-mapped but the resources are already allocated. When there is an outstanding request for a resource, the resource information update for that resource is forwarded after the completion of the operation. After the first forwarding of resource information, *all* further resource requests in the moved subtree are sent using the new topology for consistency with the forwarded resource information. The confirmation for the registration is sent back to the provider with a new superior node (figure 3-18 (b)).

In the third step (figure 3-18 (c)), the moved provider sends a `Reconf_Delete_rq` via the old topology containing the resources of the new subtree to mark the old resource paths for

deletion. If there are no resources to delete, a dummy message is sent upward. If there is a provider along the old resource path with newer information about that resource or another resource-branch mapping (non-exclusive resource), the provider deletes the corresponding resource from the message. The deletion (or the dummy) message is forwarded to the top provider.

Thus, a new resource-branch mapping exists (figure 3-18 (d)) in the new topology, which is directly used after forwarding the resource information using the new topology.

3.2.15.4 Switching to the New Topology

The last step in the reconfiguration is switching to the new topology. When all outstanding deletion messages for resources (including the dummy message) are received at the top provider, a `Reconf_Delete_cf` message is sent downward via the new route. This message carries the indication to delete the old connection at the moved provider (B in our example). The connection is disconnected when there are no outstanding messages on that link. The disconnection message is confirmed to the top provider using the new routes. After that, the reconfiguration is completed as shown in figure 3-19. An explicit message for the end of the reconfiguration is not necessary, because the new topology is used since the forwarding of the resource information.

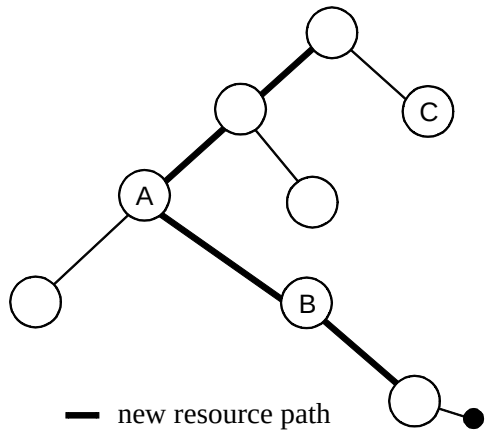


Figure 3-19: *Reconfigured Topology*

3.2.15.5 Correctness of the Reconfiguration

During the update of the databases, requests may proceed to avoid blocking the conference without leading to failures due to inconsistent identifiers or database entries. The topology database is implicitly updated during the propagation of the start messages and is enabled after the propagation of the `Reconfig_Switch_rq`. The conference database is updated similar to a normal delete or add operation in SCCS by using the multicast indication cloud.

For consistency of resource requests, the correctness of the reconfiguration scheme is shown only for token operations which may occur during the reconfiguration to simplify the proof. The proof is shown informally based on the protocol definition given in [SCCS99]. The operations to be considered are: allocation, release, pass, ask, and inquiry for owners.

Lemma 1: Token allocation, release, and pass do not lead to inconsistencies.

Only resources without pending operations are updated by the moved provider. Resource(s)

with pending operations are updated in further requests, until all resources in the subtree are updated and consistent. After the first update request, the new subtree is used. Hence, resource requests for new resources which occur after the first update request are consistent.

Lemma 2: Token ask operations do not lead to inconsistencies.

There are two cases to consider. In the first one, the token ask operation is forwarded upward and reaches a provider with a resource path marked for deletion for that resource. Instead of using the old resource path, the provider forwards the request upward until it reaches a provider with new (valid) resource information from the new resource path, which is at least the SCCS top provider. The new resource path exists at that moment, because the mark for deletion is done after the resource update. Thus, the token owner is reached.

In the second case, the token ask request is already forwarded in downward direction, because it reached a provider with still valid resource information on the old resource path. In that case, the old resource path is used until the resource owner is reached.

Lemma 3: Inquiring for owners operations do not lead to inconsistencies.

The inquiry request is routed to the (old) top provider for coordination of the operation. The top provider uses the old resource paths. Similar to lemma 2, the request is forwarded downward using the old topology regardless of the deletion mark. If the moved provider receives a `Reconf_Delete_cf` message, the disconnect operation is suspended until all pending inquiry operations in the moved subtree are confirmed and forwarded to the old upper node. Thus, valid token owner information of the moved subtree is inquired.

Lemma 1 to 3 show that the proposed scheme provides a non-blocking reconfiguration mechanism with ensured consistency.

3.2.15.6 Costs in Terms of Time

In this section, the costs of a move operation in terms of time to complete the operation are determined. For simplification, it is assumed that just one subtree is moved and that the top provider remains the same. Additionally, it is assumed that the transmission time t_t on all links and the service time t_s in all providers are the same. Additionally, t_{Co} is given as the time to connect a moved provider to a new upper provider, t_{Dis} as the time for a disconnect operation, l_m as the level of the moved, and l_n as the level of the new superior provider in the old tree (l_m and l_n range from 1,..,h with h the height of the tree).

The cost determination is divided into the same four steps which were used for the description of the reconfiguration mechanism. For the first step, two reconfig messages are sent by the SCCS top provider to the moved and the new superior node sequentially. The new superior node confirms the reconfig message. Thus, the initialization costs for that step are given by

$$t_{init} = l_m \cdot t_s + (l_m - 1) \cdot t_t + l_n \cdot 2 \cdot t_s + (l_n - 1) \cdot 2 \cdot t_t \quad (3.1)$$

In the second step, the moved provider establishes a new connection to the new superior node and confirms the establishment to the top provider using the new connection. Thus, the connection establishment costs are given by

$$t_{connect} = t_{Co} + (l_n + 1) \cdot t_s + l_n \cdot t_t \quad (3.2)$$

For the third step, it is assumed that there are no outstanding operations for any resources in the moved subtree. Furthermore, it is assumed that, in the worst case, the update has to be forwarded to the top provider. After receiving a `Reconf_Update_rq` with a confirmation from the top provider, the resource update is forwarded in one step together with the deletion of the resources along the old path. Thus, the costs are given by

$$t_{update} = 2 \cdot (l_m \cdot t_s + (l_m - 1) \cdot t_t) + 2 \cdot ((l_n + 1) \cdot t_s + l_n \cdot t_t) \quad (3.3)$$

In the last step, the moved provider is requested to disconnect from the old superior node and to confirm the operation. The costs are given by

$$t_{switch} = 2((l_n + 1) \cdot t_s + l_n \cdot t_t) + t_{Dis} \quad (3.4)$$

The total costs of the move operation are the sum of (3.1) to (3.4) given by

$$t_{reconfig} = t_{CO} + t_{DIS} + t_s \cdot (3 \cdot l_m + 7 \cdot l_n + 5) + t_t \cdot (3 \cdot l_m + 7 \cdot l_n - 5) \quad (3.5)$$

Typically, t_{CO} and t_{Dis} are much larger (several hundreds of milliseconds) than t_s (a few milliseconds) and t_t (depending on the link speed). When moving several subtrees, most operations (e.g. the connect and disconnect operations) are invoked in parallel or are sent in combined messages (e.g. the initialization of the reconfiguration). Hence, the total costs can be easily calculated by summation.

3.2.15.7 Costs in Terms of PDUs

The costs in terms of required PDUs can be determined similar to the costs in terms of time using the same steps as in chapter 3.2.15.6.

In the first step, for the initiation of the reconfiguration, a message is sent to the moved and the moving provider who confirms the message. Hence, the number of PDUs sent is given by

$$n_{init} = l_m - 1 + 2 \cdot (l_n - 1) \quad (3.6)$$

In the second step, the connection establishment which is not included in the calculation is confirmed by the moved provider using the new connection. Thus, the number of PDUs is given by

$$n_{connect} = l_n \quad (3.7)$$

In the third step, the resources are updated. Assuming the worst case that the PDUs have to be routed to the SCCS top provider, the number of PDUs is given by

$$n_{update} = 2 \cdot (l_m - 1) + 2 \cdot l_n \quad (3.8)$$

In the last step of the reconfiguration, a switch message is sent to the moved provider which is confirmed to the SCCS top provider, so the number of PDUs is

$$n_{switch} = 2 \cdot l_n \quad (3.9)$$

The total costs in terms of transmitted PDUs can be obtained by the sum of (3.6) to (3.9), which results in

$$n_{reconfig} = 3 \cdot l_m + 7 \cdot l_n - 5 \quad (3.10)$$

Some of the PDUs used in the scheme are accumulated when invoking several reconfiguration operations. Hence, the number of PDUs given in equation 3.10 multiplied with the number of operations results in an upper bound for the PDUs sent during a reconfiguration operation.

If considering the number of bytes transferred for the reconfiguration, the PDU size has to be taken into account. An upper bound for the number of transferred bytes is given by multiplying equation 3.10 with the size of the PDU. When invoking several reconfiguration operations, some of the PDUs are accumulated when forwarding upward. Hence, the number of transferred bytes is smaller than given by the upper bound.

In the performance evaluation of the dynamic reconfiguration in chapter 6, only the number of PDUs for the reconfiguration is used using equation 3.10 for the calculation.

3.2.16 Dynamic Reconfiguration of Conferences

In the last section, a scheme was presented allowing to reconfigure the SCCS tree topology during runtime of the conference without blocking (parts of) the conference and with ensured consistency of the resource identifiers. This service provides the functionality to reconfigure an entire SCCS tree. However, it does not give a motivation to do that. The next section presents an extension of this static reconfiguration service proposed in [Kl99] by providing a reason to reconfigure the tree topology for which the static reconfiguration service of SCCS is used.

As mentioned in chapter 3.2.13, the proposed resource management scheme of SCCS improves resource requests especially if they are invoked locally in the tree topology. The idea of the proposed extension, called *dynamic reconfiguration*, is to locate highly active users in a given scenario and to group them locally within the tree topology of SCCS to improve the floor control requests among these users.

First, a motivation of the problem is given before presenting the overall workflow of the algorithm. Each part within the workflow is explained, and the parameters are outlined which can be chosen to adjust the algorithm.

3.2.16.1 Motivation and Idea

The proposed resource management scheme presented in chapter 3.2.13 improves resource requests if the corresponding providers are located nearby within the SCCS topology, i.e. with a short distance in terms of hops in the tree. Normally, when creating a conference and extending it by joining users or appending other conferences, the tree topology is organized more or less randomly. As a consequence, resource requests are not routed optimally with respect to floor control operations. The reconfiguration service presented in chapter 3.2.15 allows to reconfigure the tree topology during runtime of the conference. Thus, it offers the basis to optimize the tree after creation of the conference.

As mentioned in the requirements for a conferencing service (see chapter 2.1.2.2), the response time for floor control requests is very time critical in specific scenarios. In contrast, administrating SCCS multipoint channels, like joining channels or admitting other users to join a private channel, is not that time-critical regarding the response time for the request. As a consequence, only floor control requests are taken into consideration for an improvement in terms of response time for the requests.

The idea of dynamically reconfiguring a running conference is to detect *activity* in terms of certain floor control operations and to group the corresponding providers locally within the conference tree, i.e. to shorten the distance of the involved providers. For the latter, the reconfiguration service of SCCS is used. Hence, together with the proposed resource management scheme, this leads to smaller response time for the floor control operations of locally grouped users. For that, the *dynamic reconfiguration scheme*, introduced in the following sec-

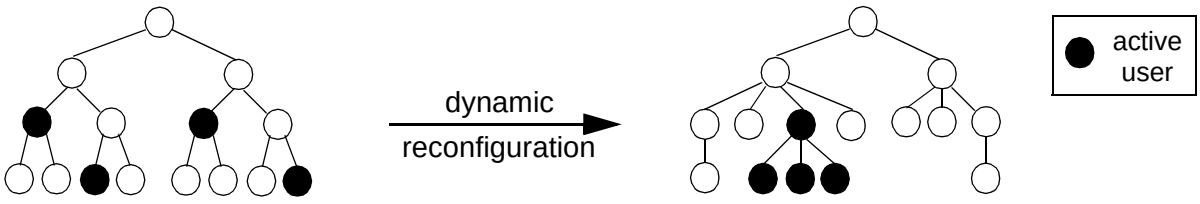


Figure 3-20: *Dynamic Reconfiguration of SCCS Topologies*

tions, defines the notion of *activity* with respect to certain floor control operations and a *quality metric* to evaluate tree topologies in SCCS. The proposed dynamic reconfiguration scheme will be able to detect activity with respect to several tokens or combinations of tokens. Furthermore, the placement algorithm is outlined to group the active users locally in the tree topology. Figure 3-20 clarifies the basic idea of the approach, i.e. grouping the black, active users locally in the tree topology.

3.2.16.2 Workflow

In the following, the workflow of the proposed dynamic reconfiguration [TrKl99] is presented to depict the different parts of the algorithm. The blocks of the workflow are explained in more detail in the following sections.

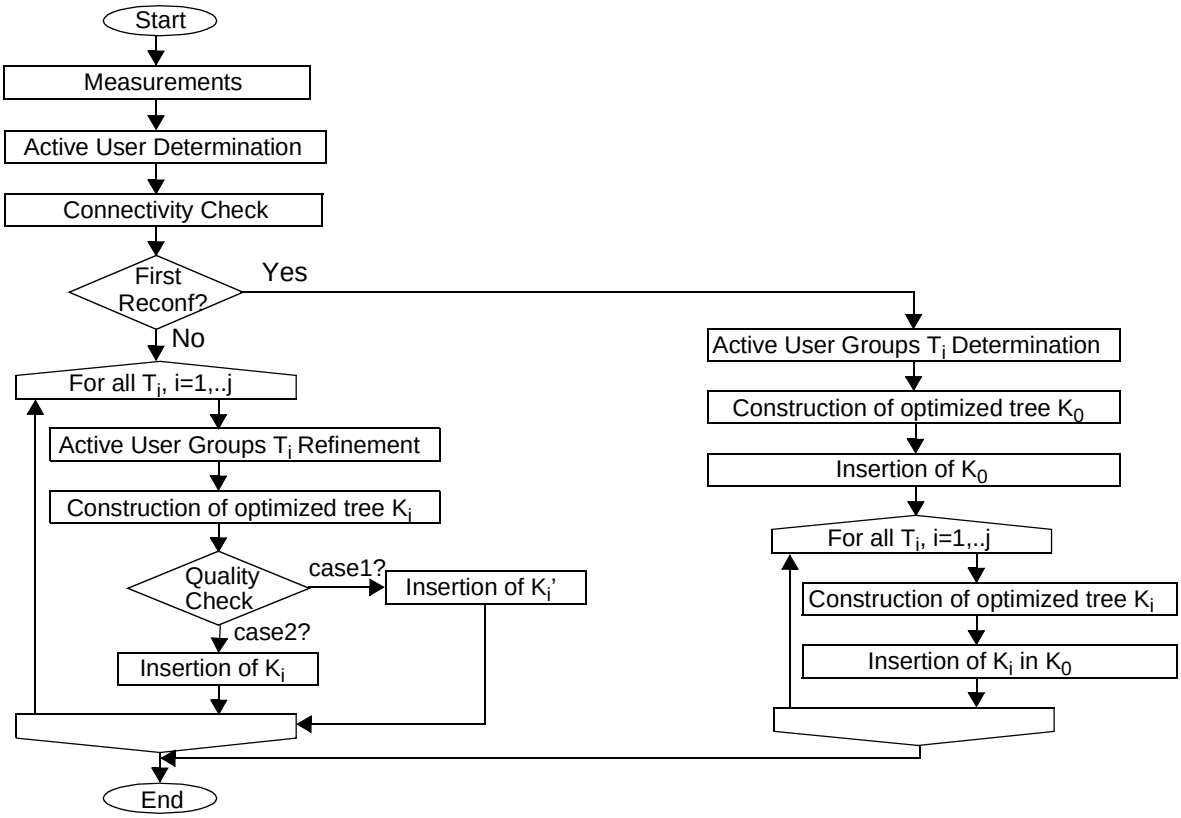


Figure 3-21: *Workflow of Dynamic Reconfiguration*

For each token which is used in SCCS the necessity to be considered in the dynamic reconfiguration has to be signaled by the application which allocates the token first. Because the semantics of a token within an application scenario is not known to SCCS, a token is not considered in the dynamic reconfiguration by default. If there is at least one token to consider, each reconfiguration is invoked after a defined reconfiguration interval t_{dr}. This interval may be set by the application which uses SCCS. By default, the reconfiguration interval is set to five minutes. In chapter 3.2.16.12, a detailed description of all parameters of the reconfiguration is given. During the conference, measurements of the token request frequencies are performed. Based on these measurements, the active users are determined with respect to a specific token. Link time measurements are used for testing the ’connectivity’ of a certain user. When invoking the first reconfiguration, the users are divided into disjoint user groups which are active with respect to

a certain combination of tokens. For each of these groups, an optimized subtree is built based on a *quality metric* which takes the token request frequency into account. The optimized subtrees are joined together in the entire conference tree.

When invoking further reconfigurations, the active user groups are refined by deleting and/or adding users to these groups based on the current measurements of the activity. For the refined groups, new optimized subtrees are constructed. The quality of these new subtrees is compared to the quality of revised subtrees to avoid heavy reconfiguration operations in SCCS. The appropriate subtree is then placed in the entire conference subtree.

3.2.16.3 Measurements

The decision of how and why to reconfigure the control tree is based on measurements of the current activity in the running conference. As already mentioned, tokens are mostly used for floor and access control. As a consequence, the measured frequency to request certain tokens is used to indicate the *activity* of an application in a scenario.

Token request frequencies are collected by the SCCS top provider by invoking a `Reconf_Frequency_rq` downward the tree. The request is forwarded and duplicated on each branch until it reaches a leaf node. The tokens for which the token request frequency information is requested are specified as parameters. The appropriate information is sent back to the top provider. The intermediate providers accumulate the responses before forwarding the PDU. The tokens which are considered in the dynamic reconfiguration have to be set within SCCS which is done during allocation of the token by the requesting application.

Additionally, the link time for each SCCS connection within the control tree is measured. This is done by invoking a `Reconf_LinkTime_rq` from the SCCS top provider downward the tree. The request is forwarded and duplicated on each branch down the tree until it reaches a leaf node. During the propagation, the request arrival timestamp is stored in each provider. When the request is sent back from the leaf nodes to the top provider, the intermediate providers cumulate the responses, compute the link time of the appropriate downward link by using the differential timing information, add their information, and forward the response upward. Thus, the link times in the control tree are determined by using a differential approach to avoid synchronized clocks [Lamp78].

The interval which is used to issue the measurement requests corresponds to the reconfiguration interval (see chapter 3.2.16.12). Thus, the overhead of obtaining the measurements depends on the settings for the reconfiguration. Within SCCS, the measurement data is sent with lower priority for a minimal influence on the running conference.

3.2.16.4 Active User Determination

Based on the measurements of the token request frequencies for each token, a set H_t is built containing all users which have used that token *t* during the previous measurement period. These sets are ordered by the measured frequencies. An *active user* with respect to a token *t* is now defined as a user belonging to the a_b percent of the most active users which have used that token. This lower bound a_b for the most active users is a parameter which can be used to adjust the active user group determination (see chapter 3.2.16.6). Figure 3-22 outlines an example for an activity distribution of a panel discussion. There are three different groups with different activity areas. The group with the lowest activity is the participant group. The second one represents the experts of the panel, while the activity of the chairman is the highest one. By order-

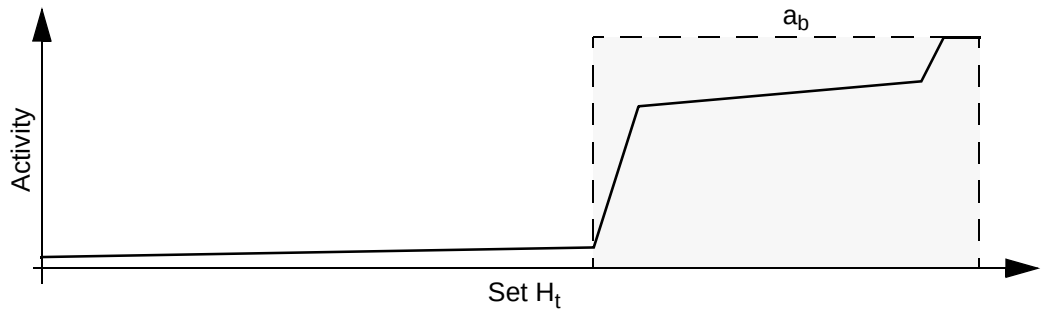


Figure 3-22: *Active User Determination*

ing the set H_t with respect to the activity and introducing the parameter a_b, the most active users can easily be separated (in the example the experts and the chairman).

3.2.16.5 Connectivity Check

Based on the measurements of the link time within the control tree, an estimation of the connectivity of each provider is performed. A provider with large link times on most of its connections is assumed to have a bad connectivity to its outer world (e.g. a mobile device). So this provider is handled separately when building the new topology by avoiding to place the provider centrally in the tree.

3.2.16.6 Active User Groups Determination

The conference users are partitioned by defining active user groups. The groups are built according to the tokens which are used actively within these groups. Hence, token operations are mainly invoked among users which belong to the same active user group.

For that, in a first step, the set of all users is partitioned into disjoint sets G_{c_i} of users which are active users (see chapter 3.2.16.4) for a combination c_j of tokens. The set G_0 contains all users being classified as *not active* on any token in the system.

In a second step, the disjoint sets G_{c_j} are grouped together by defining the relation '$G_{c_j} \cong G_{c_i}$' according to

$$G_{c_j} \cong G_{c_i} \Leftrightarrow \exists G_{c_k}; (c_k \cap c_i \neq \emptyset) \wedge (c_k \cap c_j \neq \emptyset). \tag{3.11}$$

With this transitive relation, the *active user groups* T_i are determined according to

$$T_i = \{G_{c_1},\ldots,G_{c_n}; \forall a,b \in \{c_1,\ldots,c_n\} \Rightarrow G_a \cong G_b\}, \tag{3.12}$$

with T_0:=G_0 the set of all users being not active (or marked as not active) on any token.

For instance, if there are 2 tokens in the conference and one group of users is using token number 1, a second group is using token number 2 and the third group is using both tokens. Then, there are four disjoint sets G_0, G_1, G_2, $G_{1,2}$. These sets are inserted to two separated active user groups, T_0 consisting of G_0 and T_1 consisting of G_1, G_2, and $G_{1,2}$.

3.2.16.7 Active User Groups Refinement

The separation of users in active user groups as shown in chapter 3.2.16.6 is done during the first reconfiguration. When performing further reconfigurations, new active user groups may be determined (according to chapter 3.2.16.6), because users became active on other resources.

Active user groups may be deleted because the activity in this group is below the activity limit or these groups of active users are refined. This refinement is done by inserting new members to the sets (e.g. when a user became active during the last measurement period) or deleting users from the sets. The deletion of users might occur when these users are below the activity limit (see chapter 3.2.16.4). To avoid periodic changes of the active user groups, users remain in the group for a defined period t_r after becoming non-active. For non-active user, the token request frequency is adjusted using an exponential average scheme according to

$$f_t(u) = a_r \cdot f_{act,t}(u) + (1 - a_r) \cdot f_{old,t}(u) \tag{3.13}$$

with a_r the exponential average parameter (from 0 to 1), $f_{act,t}(u)$ the actual measurement value, and $f_{old,t}(u)$ the old (adjusted) frequency. With this scheme, the users are more and more marked with lower frequencies when they became inactive, before they are deleted from the active user group after t_r seconds.

3.2.16.8 Quality Metric

The performance gain of the reconfiguration is determined by comparing the *quality* of the old and the reconfigured, new tree. For that, a quality metric is defined for a single user. This metric is extended to the *subtree quality* by summing up the quality of all users in the subtree.

As mentioned in the motivation of the approach, the resource management scheme improves floor control requests which can be handled locally within the tree topology. An intuitive approach would be to integrate the distance in terms of link time in the quality. But the problem with this approach is that the link times for all possible connections between the different providers are not known beforehand. The link time measurements presented in chapter 3.2.16.3 determine the link times only for the subset of existing connections in the tree topology.

As a consequence, the basis of the quality metric for a specific user is the distance in the SCCS tree topology to a user requesting the same token. This distance is determined in terms of *hops* within the tree topology, i.e. the number of SCCS connections when routing the request. Furthermore, the quality metric contains the measured frequencies of token requests.

Define $tok(u)$ as the set of tokens used by user u, $uses(t)$ the set of users using token t, $f_t(u)$ the token request frequency for token t at user u, and $d(u,v)$ the smallest number of hops between u and v. Then, the quality of the position of user u in the tree is defined by

$$Q(u) = \sum_{t \in tok(u)} f_t(u) \sum_{u_a \in uses(t), u_a \neq u} f_t(u_a) \cdot d(u,u_a) \tag{3.14}$$

Hence, the quality of a tree K is given by the sum of the quality of each user in the tree according to

$$Q(K) = \sum_{u \in U(k)} Q(u) \tag{3.15}$$

with $U(k)$ the set of users in tree K. The metric in (3.15) is used to determine the quality of trees. For a given measurement set of $f_t(u)$, it holds that the smaller $Q(K)$ the better the quality of the tree, i.e. the smaller $Q(K)$ the smaller the distance of active users within the tree.

3.2.16.9 Construction of Optimized Trees

After the determination (and refinement) of the active user groups, an optimized tree is constructed in the main memory of the SCCS top provider for each active user group according to

the quality metric in chapter 3.2.16.8. For that, the tree topology is built beginning with the most active user down to the user with the lowest activity in the group.

The problem in the construction of the optimized tree is the limited number of provided links in each node. Thus, it might be better to insert a user with high activity but limited connectivity in terms of supported links much lower in the tree. Therefore, the quality metric has to be applied also to the situation that there is a user in the set of not yet placed users with a higher connectivity than the current placed user. Because the complexity of that problem is *O(N!)* with *N* the number of users in the active user group [Kl99], an heuristic approach is used. Instead of only using the measured token request frequency, when determining the active users for a token (see chapter 3.2.16.4), the connectivity of the user is taken into account as well as the token request frequency. But to avoid bottlenecks in the reconfigured topology, e.g. by constructing subtrees with a star-shaped topology, not the entire connectivity is used by the dynamic reconfiguration. As a consequence, the number of provided links which is used for the construction of the subtree is restricted by the algorithm to a number of at most ten connections.

Following this reasoning, the value $f_t(u)$ in equation (3.13) is adjusted to

$$f_t(u) = a_c \cdot l_u + (1 - a_c) \cdot f_t(u) \tag{3.16}$$

with a_c the construction parameter (from 0 to 1), l_u the number of used links at user *u*, and $f_t(u)$ the measured activity of user *u* regarding token *t*.

Additionally, the quality metric is not applied to the situation, that there are users not yet placed in the new tree which have a higher connectivity. This reduces the complexity to $O(N^3)$ [Kl99]. The error compared to optimal trees is rather small (less than 7% [Kl99]) for a_c values from 0.3 to 0.7.

During the construction of the optimized subtree, problems may occur when the number of used connections in the active user group is too small. As a solution, providers are added to the active users which normally do not belong to this set to increase the number of supported links in this group.

As explained in chapter 3.2.16.6, the construction of active user groups divides the user set in disjoint groups of users being active regarding a combination of tokens. Due to the construction of the group, it is possible that there is still communication to users in other groups (especially to users in T_0), because only the most active users are inserted in the appropriate group. Thus, the activity outside the active user group has also to be taken into account when constructing the optimized tree. For that, a *virtual provider* is introduced representing the activity outside the active user group with one supported link only. This virtual provider is inserted in the optimized tree as if it would be a normal user. When the optimized tree is inserted, the provided connection of this virtual provider is used to connect the optimized tree to the rest of the topology.

3.2.16.10 Quality Check

To avoid heavy reconfiguration operations in SCCS, the quality of the optimized tree K_i for each active user group T_i is compared to the quality of a non-optimized tree K'_i . This non-optimized tree is built only by deleting users from and adding users to the active user group subtree of the previous period according to the refinement of the active user group. Thus, this tree is not optimal for the current measurement values. However, the number of reconfiguration operations might be much smaller because the tree remains unchanged in large parts. The quality of this tree is compared to the quality of the optimized tree K_i based on the new measurement values. If the performance gain in terms of quality difference is below a given parameter q_c, the non-optimized tree K'_i is used, because its reconfiguration overhead is lower. Otherwise, the

optimized tree K_i is used.

3.2.16.11 Insertion of Trees

The optimized trees constructed for each active user group (and for T_0) are joined together to the new conference tree. For that, the virtual provider in each optimized subtree is used as the connection point. The provider below the virtual provider is inserted in the conference tree beginning with providers on the second level of the conference tree to avoid that all subtrees are connected directly to the top provider. The static reconfiguration service of SCCS (see chapter 3.2.15) is used for this step.

3.2.16.12 Parameters for the Dynamic Reconfiguration

The proposed dynamic reconfiguration provides several parameters to adjust the approach to different application scenarios. First of all, the token to be considered by the dynamic reconfiguration has to be marked by the application which allocates the token first.

In table 3-11, all parameters of the dynamic reconfiguration approach are summarized together with the default settings. In a conducted conference, only the conference conductor is able to set the reconfiguration parameters. In the unconducted case, each user is able to modify the parameters.

In the performance evaluation of the dynamic reconfiguration (see chapter 6), the importance of the different parameters and reasonable settings for different scenarios are determined. Furthermore, the impact of the different parameters on the obtained performance gain of the mechanism is determined.

The reconfiguration interval t_{dr} might be used to adjust the static reconfiguration overhead of SCCS. The smaller the interval the higher the computational overhead to perform the reconfiguration algorithm. It can be seen that the duration of a static reconfiguration in SCCS (see chapter 3.2.15.6) is the lower bound for the reconfiguration. Additionally, the first reconfiguration is started after the time value twice as high as t_{dr}, i.e. 10 minutes by default. This longer time period is required to get first confident measurements.

Parameter	Description	Default
t_{dr}	reconfiguration interval	5 min
a_b	relative level (in percentage) for active user determination	5 %
t_r	time how long inactive users remain in active user group	10 min
a_r	value for exponential averaging for inactive user adjustment	0.5
a_c	value for for tree construction	0.5
q_c	quality level for insertion of optimized tree	20 %

Table 3-11: *Parameters of the Dynamic Reconfiguration*

Adjusting parameter a_b affects the size of the groups of active users regarding a specific token. In most scenarios, the adjustment of that parameter and the corresponding result exactly reflects the roles in the scenario. Consider for example a panel discussion. The most active user would be the chairman, while the panel experts are the next active ones. At last, the participants have the smallest activity. Increasing a_b means to include more users in the appropriate group. In this

example, after 'catching' the experts, more and more participants would be inserted in the active user group. In some scenarios, it is intended that the active user group consists exactly of the most 'important' users for which the reconfiguration should be done, e.g. the experts and the chairman. The less important users should not be considered. Hence, the parameter a_b should be set carefully to meet this requirement.

Parameter t_r avoids oscillations of the algorithm due to adding and deleting users at the active user group limit. Periodic token usage is taken into account by considering members in active user groups for a prolonged time instead of removing them directly. The averaging parameter a_r is highly correlated with parameter t_r controlling the adjustment of the active user groups.

Parameter q_c is very important to adjust the construction of optimized trees in each reconfiguration step. If the parameter is higher, the approach uses more of the old structure of the tree instead of rebuilding the tree completely. Thus, the static reconfiguration overhead can be controlled by this parameter as well.

The tree construction parameter a_c is used to reduce the tree optimization problem. But the parametrization of the value does not have a large impact on the resulting tree as shown in [Kl99]. An implementation of the tree construction algorithm might test different values for a_c to find a proper value. This is done for instance during the simulative evaluation of the reconfiguration algorithm in chapter 6.

It can be seen that the proposed dynamic reconfiguration scheme provides six application parameters to adjust the algorithm. The determination of appropriate parameter settings is one of the tasks of the performance evaluation presented in chapter 6.

A parameter of SCCS which is not specifically mentioned as a reconfiguration parameter is the *number of supported SCCS connections* within each provider (see chapter 3.2.2). Obviously, this parameter has a strong impact on the construction of the optimized trees (see chapter 3.2.16.9). The smaller the number of provided links the higher the depth of the resulting tree when considering a given start topology. But in contrast to the reconfiguration parameters of table 3-11 which can be set by the applications, the number of provided links mainly depends on the local implementation of SCCS. Memory restrictions might be reasons to limit the number of provided links. Furthermore, a single bottleneck can be avoided by restricting the number of supported connections. As explained in chapter 3.2.16.9, the reconfiguration algorithm only uses a bounded number of links for the reconfigured tree. Hence, in the evaluation of the reconfiguration, presented in chapter 6, the number of supported links in each provider is set fairly high to avoid a negative impact of this implementation-dependent parameter.

3.3 Implementation Design

The following section outlines the design of the SCCS implementation which is realized to demonstrate the feasibility of the protocol. A first attempt to realize the main features of SCCS is proposed in [TrSch98a]. This approach used CORBA [CORBA93] as the underlying communication infrastructure. But the implementation suffered from several problems of the chosen CORBA system. Inefficient multicast communication as well as tricky multithreading of objects led to the decision not to use CORBA for the final implementation of SCCS.

The current design of the SCCS implementation is fully object oriented. Thus, an *object model* is presented in the following as well as a *process model* being used for the implementation. For the latter, an activity model is also presented to give an impression of how the different proces-

ses corporate. The implementation is in conformance to the proposed protocol stack presented in chapter 3.2.1. MTP/SO [BOS97] is used as a reliable multicast transport protocol, while TCP/IP is used for performing the unicast transfer. The MTP/SO implementation is provided by Nils Seifert for free provision of reliable multicast in the SCCS implementation.

For demonstration of the SCCS functionality, a simple shared whiteboard is also introduced.

3.3.1 Object Model

Figure 3-23 shows the object model for a local SCCS provider in *Unified Modeling Language* (UML) notation [JEJ95].

The software is divided into five packages:

- **Control**: Functionality provided at the control service access point.
- **Attachment**: Functionality provided at the user service access point. Additionally, the interfaces of notification objects which are implemented at the client side are defined.
- **Connection**: Functionality at the transport service access point to provide unicast and multicast transport services to the SCCS layer.
- **UCTransport**: Abstraction of unicast transport protocols providing a generic interface for the unicast transport service (see chapter 3.2.3.1).
- **MCTransport**: Abstraction of multicast transport protocols providing a generic interface for the multicast transport service (see chapter 3.2.3.2).

The core of the implementation is located in the *Control* package. The central object in this package is the *Manager* object providing methods to create local conferences as well as to append and join remote conferences. The *Control* object represents the internal control service access point of SCCS and is used to provide the functionality of the services being invoked at that access point (see [SCCS98]). The *Conference* object is created by the *Control* object as the core of a conference providing methods for the conference management and control functionality. A certain *Conference* instance is responsible for creating different interfaces at runtime defined in several other packages. Furthermore, the *UCListener* object is implemented to listen on incoming connection establishment requests for the conference tree topology. The *CMarshal* object is responsible for marshaling and demarshaling PDUs for the conference to ensure transfer of control traffic among heterogeneous platforms.

In the *Attachment* package, the *Conference* object is responsible for the creation of the *MemberInterface* which is created when a new local participant enters the conference. This object provides methods to retrieve database entries like name and address of the endsystem, represented by the object. Furthermore, a *TokenInterface* is created by the *Conference* object for the provision of the token functionality of SCCS. When a local member of the conference joins a channel, the *Conference* object creates a *ChannelInterface* which provides methods for the data transfer and for the channel control. This object is associated to a dedicated multicast address of the underlying multicast transport protocol and is used to convey further send and receive operations to the underlying protocol which is chosen when creating the channel. The *TokenInterface, MemberInterface,* and *ChannelInterface* objects represent the *user service access points* (User SAPs) of SCCS, while the *Manager* and the *Control* objects interface the *control service access point* (Control SAP) of SCCS.

Different objects in the Attachment package provide corresponding notification objects (*TokenInterfaceNotify, ChannelInterfaceNotify, MemberInterfaceNotifiy*) to signal information changes like altering the state or adding new members to the conference. Furthermore, a central

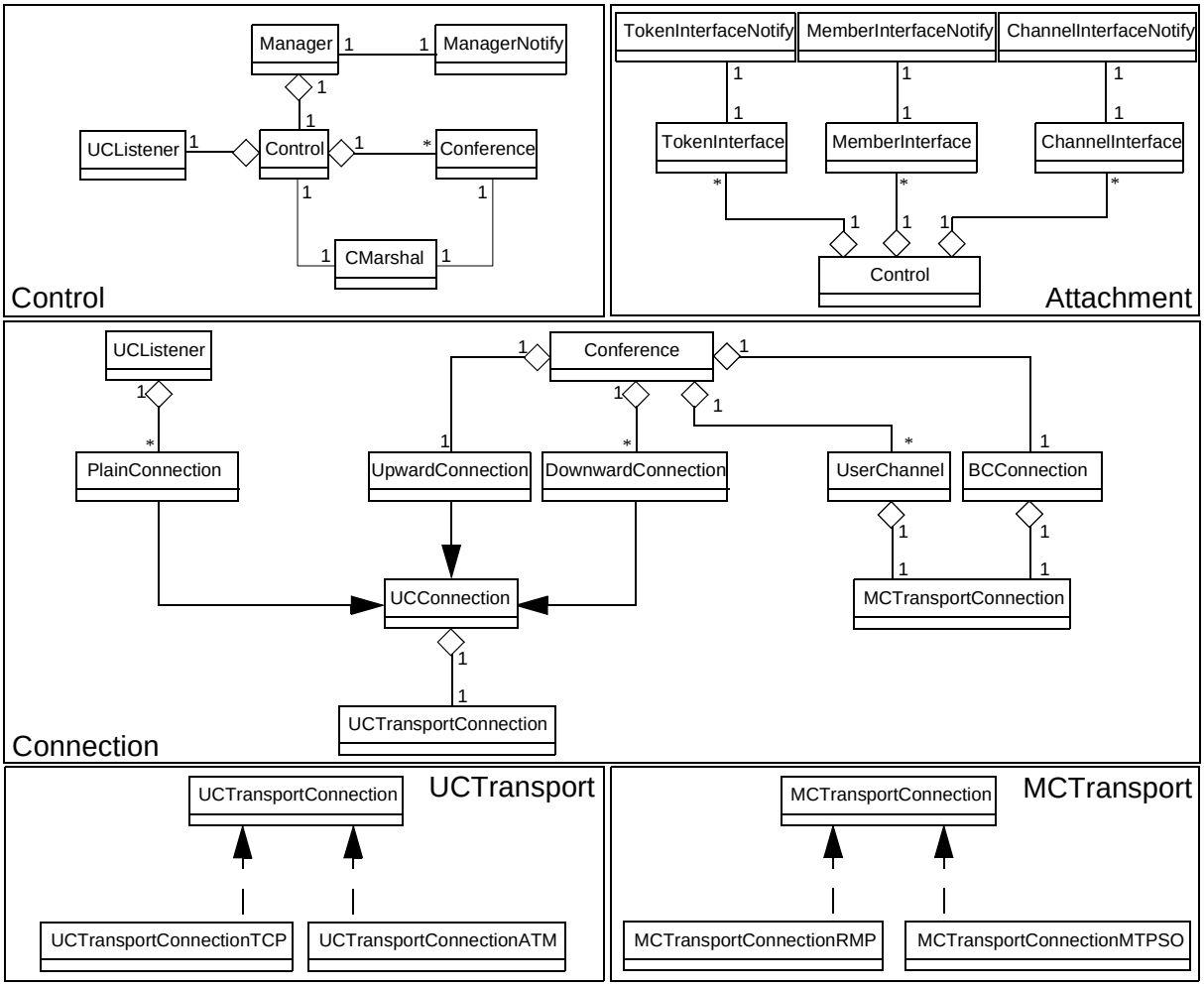


Figure 3-23: *Object Model of SCCS*

notification object for the *Manager* object is provided for control purposes. These objects have to be implemented by the application and registered during runtime to SCCS.

In the *Connection* package, the *Conference* object is responsible for creating different objects to realize the provider-to-provider communication. There may be upward and downward connections using the same *UCConnection* implementation. Both objects (*DownwardConnection* and *UpwardConnection*) are only defined as generalized interfaces to better distinguish between both directions of connections. For passively establishing connections, a *PlainConnection* object is initially created by the *UCListener*. Similar to the *UpwardConnection* or *DownwardConnection* objects, this object is only designed as a generalized interface. After determining the direction of the connection, either upward or downward, the object is casted to the appropriate object interface (*UpwardConnection* or *DownwardConnection*) and assigned to a certain *Conference* object.

Furthermore, there is a broadcast channel (*BCConnection*) for some indications of the control protocol representing the *multicast indication cloud* (see chapter 3.2.2) of the SCCS control topology. The *UserChannel* object is created as an abstraction of a multicast address when a local conference member has joined a channel to send or receive user data. Stub routines are implemented for each supported multicast transport protocol to interface send/receive operations to the application.

The *UCTransport* and *MCTransport* packages deal with the abstraction of the transport layer,

for unicast or multicast respectively, by providing methods for establishing connections, sending, and receiving octet streams, or joining multicast groups. Different implementations of the *UCTransportConnection* and *MCTransportConnection* interfaces may exist to support different protocols as illustrated in figure 3-23.

3.3.2 Process Model

The proposed object model can be easily mapped to a process model for the conferencing application and the local service provider, the application is connected to. In the following, the term *process* is used to describe a logically independent entity being able to communicate with other processes within an endsystem. In common operating systems, these logical processes are mapped to *threads* (or *light weight processes* [Tan92]) for ease of implementation. Figure 3-24 shows the proposed process model for SCCS.

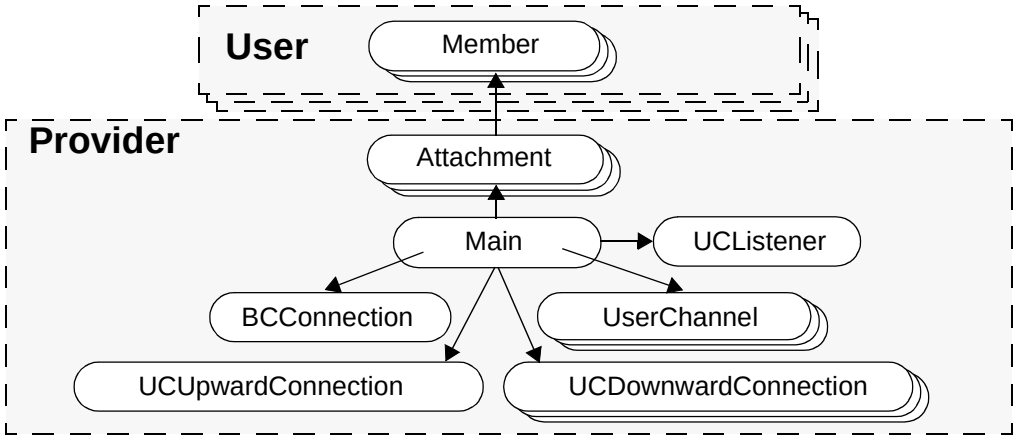


Figure 3-24: *Process Model of SCCS*

Each attachment is mapped to a different *Member* process to avoid blocking during a user service request. To each local provider, several local user entities may be connected. Each of them may consist of several Member processes, e.g. participating in several conferences or several application protocols acting as an own SCCS member.

The manager and conference object functionality is provided by the *Main* process of the provider. The *BCConnection* process is responsible for listening to the multicast indication cloud. A *UserChannel* instance processes incoming data for a certain multicast group to which a local user is subscribed. As a consequence, when a local user joins a specific multipoint channel in SCCS, a corresponding *UserChannel* instance is created if it does not yet exist locally.

The *UCListener* process is listening on incoming connection establishment requests from other providers. For that, a *PlainConnection* object and a corresponding temporary process for that connection is created. After determination of the connection direction, either upward or downward, the temporary process is declared as an *UCUpwardConnection* or *UCDownwardConnection* process. This new process is then associated to the corresponding *Conference* object (see last section) to which the connection belongs. Furthermore, *UCUpwardConnection* or *UCDownwardConnection* processes may be created actively by the *Main* process, when the provider wants to actively connect to another provider. The task of the *UCUpwardConnection* and *UCDownwardConnection* processes is to listen for incoming and to serve outgoing protocol data units respectively.

The proposed process model of SCCS can be easily implemented in a multi-threaded environment like Solaris or Windows NT. The current SCCS implementation is based on Windows NT4.0 (see chapter 3.3.4).

3.3.3 Activity Model

The following section gives an overview of the activity model used in the SCCS implementation. Only two key control activities are presented as the core of the protocol, namely *connection establishment* and *dispatching of PDUs*. The rest of the control structure of the protocol, e.g. receiving multicast data or member requests from an attachment, is rather simple from a control point of view.

Connection Establishment

The establishment of a new connection may take place in two directions within the tree topology, either in upward or downward direction. Furthermore, a connection might be established actively by a certain method which creates an *UpwardConnection* or *DownwardConnection* object and calls the *connect()* method to actively connect to a peer station.

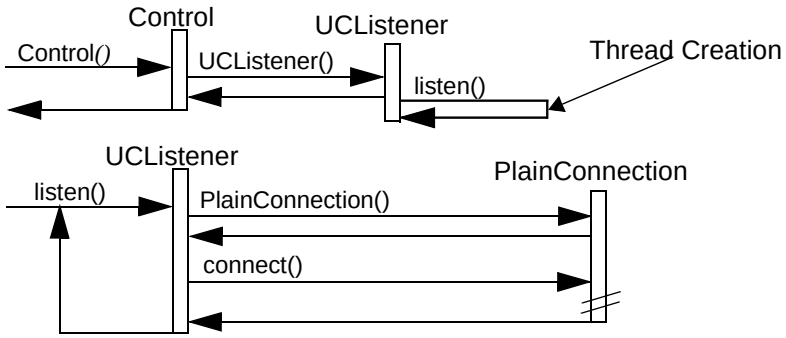


Figure 3-25: *SCCS Activity Diagram - Connection Establishment*

For the passive establishment of connections, an entity within the provider is listening on incoming connection establishment requests. This functionality is realized by the *UCListener* process (see chapter 3.3.2). In figure 3-25, the activity diagram for this type of connection establishment is presented. The *UCListener* process is created during the initialization of the *Control* object (the main object of SCCS). For that, a *UCListener* object is first created which in turn sets up an own thread running the *listen()* method of its object instance (indicated by the loop arrow in the diagram).

Within the *listen()* method, a loop is running which terminates when destroying the *Control* object. The functionality of this loop is very simple. First, a *PlainConnection* object is created, because the direction of the new connection is not known beforehand. In the second step, the *connect()* method of this object is called. This method blocks by listening on incoming connection establishment requests and successfully returns after establishing a valid connection. After that, the loop is executed again. When the connection is established, the receive loop of the *PlainConnection* is started and the incoming PDUs are dispatched (see next section).

After the connection establishment, the topology is set up (see chapter 3.2.6) by exchanging the appropriate SCCS PDUs. Within the building topology functionality (see chapter 3.2.6), the direction of the connection and the corresponding *Conference* object of the new connection are determined. As a consequence, the *PlainConnection* object is casted to an appropriate interface,

either *DownwardConnection* or *UpwardConnection*, and is assigned to the *Conference* object.

Dispatching of PDUs

When a new connection is established, the dispatching of incoming PDUs is started. The implementation of the dispatching functionality is realized in the base *UCConnection* object.

This dispatching is solved in the SCCS implementation by using a function array (or *dispatching table*) containing the corresponding member functions providing the functionality of the different PDUs. These functions are declared as private methods. The access to these functions is only defined via the *dispatch()* method provided by the *Control* and *Conference* objects as the implementation of the control and user service access point. Figure 3-26 shows the corresponding activity diagram of the PDU dispatching.

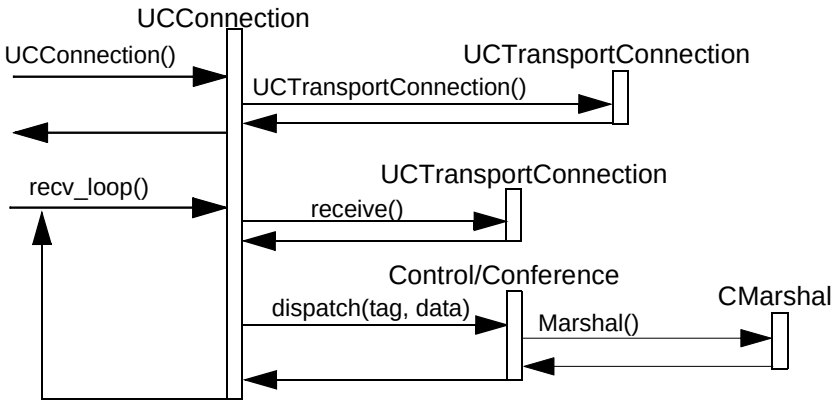


Figure 3-26: *SCCS Activity Diagram - Dispatching*

When creating a *UCConnection* object (either by creating a *DownwardConnection*, an *UpwardConnection*, or a *PlainConnection*), a *UCTransportConnection* object is created by this *UCConnection* instance as an implementation of a transport layer. Furthermore, the *recv_loop()* method of the *UCConnection* instance is set up as an own thread. This method blocks until a transport connection is established.

After connecting to a peer station, the receive loop is starting to receive incoming PDUs by calling the blocking *receive()* method of the associated *UCTransportConnection* object. This method obtains the *tag*, *length*, and the *data* content of a protocol data unit received on this connection. When the *receive()* function returns successfully, the received data is marshaled by using the common transfer syntax. The obtained *tag* is used as an index to the current dispatching table for the PDUs calling the corresponding dispatching method using the *dispatch()* function of the *Control* or *Conference* object respectively. For the first incoming PDUs, namely `Connect_rq` and `Connect_cf`, the *dispatch()* method from the *Control* object is used, because there is not yet a conference assigned. During processing of the PDUs at the control SAP, the connection is assigned to a certain *Conference* object. For further incoming PDUs, the *dispatch()* method of the Conference instance is used.

It can be seen from the chosen approach that the dispatching of the incoming PDUs is solved efficiently by using a simple table lookup improving the performance of the implementation. The dispatching loop is the main control code beside the establishment of new connections shown in the previous section.

3.3.4 Demo Application

The services of SCCS are partially implemented using the proposed object and process model presented in the previous sections. The merging functionality of SCCS is not implemented to shorten the development. Hence, appending and inviting other conferences is not possible with the current implementation. The protocol is realized under Windows NT4.0 as a static C++ library which is used to write SCCS-based applications.

For the demonstration of the SCCS functionality, a *shared whiteboard* application is implemented. This demonstration program provides simple drawing functionality in a shared workspace like placing objects (e.g. circles, rectangles), ordering objects, and assigning colors to the objects. This program is used to demonstrate floor controlled conferencing.

In addition to this application specific functionality, the program allows to use the conference management functionality of SCCS like inquiring for users or conducting conferences. It is implemented under Windows NT4.0 using the SCCS implementation for that environment. It is on-line available [Tro00c] for free download providing a setup program for ease of installation.

3.4 Assessment of SCCS

After proposing SCCS as a generic conferencing service, this section deals with the assessment of SCCS with respect to the requirements defined in chapter 2.1.2. Hence, this can be seen as a qualitative evaluation of SCCS.

Conference Management

SCCS supports the main requirements for management and control of conferences. It provides a conference database of members and applications. Furthermore, sophisticated tree management features are included in SCCS like merging, splitting, and reconfiguring running conferences. SCCS provides unconducted or conducted conferences together with conductorship management functionality and the corresponding voting policies for management functions like appending or inviting conferences. SCCS uses the underlying multicast transport functionality for conference management in form of a *multicast indication cloud* which is used for some indications instead of routing these via the tree topology.

Not included in SCCS are features like

- location service,
- conference announcement,
- network resource management,
- security, and
- accounting.

The first three features can be covered by approaches defined in the *Internet Multimedia Conferencing Architecture* (see chapter 2.3.2), providing a session announcement and initiation service as well as a guaranteed QoS provision framework. For the fourth issue, group key approaches (e.g. proposed in [WGL98]) can be applied. The first four issues can be added to SCCS as additional services without affecting the main mechanisms of SCCS. For the last issue, accounting information has to be included in the conference database. Thus, it can be seen that the missing parts of the conference management functionality can be supplemented by appropriate techniques which are currently developed.

Multipoint Transfer

The multipoint transfer model of SCCS supports all communication patterns from one-to-one to many-to-many. The multipoint transfer is abstracted by introducing a *channel* concept. Data sent via this channel is received by all joined members. SCCS supports *public* and *private* channels. The latter are conducted by the member who created the channel.

The multipoint channels are only administrated by SCCS. The transfer of the data is realized by the underlying transport layer, so efficient multicast functionality can be used. The used protocol is specified during creation of the channel. Hence, different transport protocols are supported. Also real-time transfer like audio and video can be realized by SCCS by using RTP [RFC1889] or RTSP [RFC2326] for the transfer. Thus, SCCS provides a common multipoint transport service for real-time and non-real-time data in contrast to the H.32x systems (see chapter 2.2) which separate real-time from non-real-time transfer by providing different services for both types of data. Furthermore, due to the notion of a *non-SCCS listener*, SCCS provides reception of multicast data passively by bypassing the SCCS channels. As a consequence, only these users have to join a conference who want to use SCCS services within the given scenario, e.g. for floor control. Passive reception of multicast data is possible without joining the conference. Thus, the spectrum of conferencing scenarios is extended by combining tight control for active users with loosely coupled passive entities.

Furthermore, SCCS provides the transfer of user data among heterogeneous systems by introducing a *user data marshaling layer* on top of SCCS. In the current implementation of the service, this functionality is not integrated. However, existing tools like the ASN.1 compiler SNACC [SNACC94] might be used to integrate that functionality very easily.

Distributed Applications Support

SCCS supports the synchronization and access control of distributed applications by providing a *token* concept. These tokens can be used to realize floor or access control within the application context. Several management functions for the tokens are provided like allocating exclusive or non-exclusive tokens, passing or asking for tokens, and inquiring the list of token holders. Thus, SCCS covers elaborated floor control functionality similar to approaches proposed in ([DoGLA95][T120]).

Standardized Applications

Standard conferencing applications like shared whiteboard, file transfer, or shared applications are not defined specifically for SCCS. However, due to the similar service model of SCCS compared to the T.120 standard, the application protocols ([T126][T127][T128]) defined for that protocol suite might be used to implement this functionality in an SCCS-based environment.

Robustness

The robustness issue of the requirements is divided into two aspects which have to be considered. The first one is the provision of mechanisms for *robustness on the service level*. The second one is the *robustness of the mechanisms* on the protocol level.

Robustness on the service level is very difficult to provide in tree-based approaches. This is because tightly coupled environments are inherently much less error-resilient than loosely coupled systems (see section 2.1.2.2). The main service to provide robustness on service level is covered by the provision of mechanisms for managing the tree topology of SCCS. In contrast to the T.120, SCCS provides features to reconfigure the conference tree during runtime of the

conference. Thus, simple functionality like provider deletion (e.g. due to shutting down the corresponding computer) is supported in SCCS in contrast to the T.120 which lacks of simple reconfiguration functionality. Furthermore, the merging and splitting functionality of SCCS is implemented by remapping resources and identifiers instead of releasing them. Error recovery functionality (e.g. recover from ungraceful shutdowns of providers) is not yet provided by SCCS. But in [Ei99], an error recovery functionality is proposed for SCCS which is not yet integrated in the protocol.

As mentioned in section 2.1.2.2, there are several methods to prove the robustness on protocol level in terms of *correctness* of the provided functionality using *testing and validation methods*. These methods can be used to validate a protocol specification defined as a finite-state automata description (by using SDL [Z100a] or LOTOS [ISO8807]) and generating test cases from this specification. Due to the lack of a formal finite automata specification of SCCS, this kind of validation is not suited for testing the correctness of SCCS protocol mechanisms.

The service as well as the protocol data units are specified in the service and protocol specification of SCCS ([SCCS98][SCCS99]) together with the sequences of messages to fulfil a certain action. Thus, methods like proposed in [AHP96] or [ObKe99] could be used to validate the protocol mechanisms. But this was not done for SCCS. Hence, the correctness proof on protocol level is an open issue in the development of SCCS.

Scalability

Scalability in terms of conference size and geographical distribution is a big issue in the requirements for a conferencing service as well as in the design of the protocol mechanisms of SCCS. The protocol implements mainly the following three key features to provide a high scalability of the service:

- *Usage of multicast*: Multicast is used both for control and user data. Especially the user data transfer, using more efficient transport layer facilities, shifts the aspect of scalability to lower layers.
- *Resource management scheme*: The proposed SCCS resource management scheme (see chapter 3.2.13) allows a more efficient routing of resource requests compared to the simple centralized routing in tree-based topologies, especially if the requesting entities are located locally in the tree. Thus, the resource management scheme may shorten the delays for resource requests especially in geographically distributed conferences by avoiding to forward each request necessarily to the topmost provider.
- *Dynamic reconfiguration of running conferences*: To emphasize the improvement achieved by the proposed resource management scheme, SCCS provides a dynamic reconfiguration of the control topology during the conference by grouping active users locally in the tree. This may lead to a smaller response time for resource requests among active users in the conference.

The first issue shifts the scalability aspect to the transport layer. If this layer uses efficient multicast mechanisms, the scalability of the conferencing environment can be improved. It is not necessary for the conferencing layer to implement an additional transport functionality. Hence, it can be assumed on a qualitative level that this design leads to a higher scalability compared to approaches like T.120 without giving quantitative numbers.

The second and third issue affect the control functionality provided by SCCS, especially the floor control functions, and implement new mechanisms to achieve a high scalability for the control part of the conference. As a consequence, a quantitative assessment of these mechanisms is necessary to clarify whether and to what extent the scalability aspect is achieved com-

pared to other approaches. This quantitative assessment is done in the performance evaluation of SCCS which evaluates the resource management and the usage of the dynamic reconfiguration to improve the responsiveness of the floor control.

3.5 Performance Evaluation of SCCS

The purpose of the following section is to define the goals of the performance evaluation of SCCS. For that, the specific SCCS mechanisms are depicted which are evaluated. Furthermore, the performance measures are outlined which are of interest for the evaluation. These performance measures are used to assess the evaluation facilities in chapter 4 with respect to the covered spectrum of performance measures.

Furthermore, this section deals with the system parameters which are assumed for the evaluation. These values concern application and system components. It is important to make clear to what extent these parameters influence the results of the performance measures in particular and the entire performance evaluation goal in general.

It is worth mentioning that it is not the purpose of this section to perform the evaluation of SCCS mechanisms. This is done in chapter 5 and 6 for two specific SCCS mechanisms.

3.5.1 Goal of the Performance Evaluation

In general, the goal of a performance evaluation of SCCS is to obtain performance measures when applying specific SCCS protocol mechanisms. For a more specific definition of the performance goal, the following questions have to be addressed:

1. What protocol mechanisms are evaluated?
2. What are the specific evaluation goals with respect to the considered protocol mechanism?

Concerning the first question, the protocol mechanisms have to be selected for evaluation. With respect to the requirements of a conferencing service, presented in chapter 2.1.2, *scalability* is one of the main design issues of the protocol mechanisms of SCCS. As stated in chapter 3.4, the scalability of SCCS is mainly based on two different mechanisms.

The first one is the consequent *usage of multicast* within SCCS both for conference management but also for the transfer of user data. However, the usage of multicast is just a design issue to improve the scalability of SCCS. It is not a new mechanism on the conferencing layer, because SCCS uses the services provided by the underlying transport layer. It is not within the scope of this thesis to evaluate the effectiveness of multicast mechanisms. This work has extensively been addressed in the literature (e.g. [DDC90][Erik94][HaCr97][LiPa95][MJV96] [Scho96][BOS97][HCBO99]). A very good survey of multicast mechanisms and protocols is given in [LeGLA96].

The second mechanism improving the scalability of SCCS is the proposed *resource management* scheme. Together with the *dynamic reconfiguration* of running conferences, the floor control within SCCS may be improved compared to other tree-based approaches. As a consequence, the performance evaluation of the SCCS mechanisms is divided into two parts. The first one evaluates the proposed scheme. The second part of the evaluation is dealing with the improvement to be achieved when dynamically reconfiguring the tree topology.

Concerning the second question above, the specific evaluation goal has to be considered separately for both mechanisms. For the proposed resource management scheme, it is of interest

how the conferencing system performance can be improved when applying the proposed mechanism. For that, the obtained *performance* is compared to the centralized scheme of the T.120 standard. For a fair comparison of the resource management schemes as such, the fast token give operation (see chapter 3.2.14) is not applied during the performance evaluation, because this mechanism is not included in the T.120 standard. It is not intended to compare the SCCS resource management to other management schemes in loosely coupled environments, like quorum-based or rotating-token-based schemes, because these schemes

- are not intended for a larger number of participants and
- often heavily rely on multicast transport layer functionality like message ordering, or reliability which should not be evaluated in this thesis.

For the second considered SCCS mechanism, the dynamic reconfiguration, the evaluation has to address the following topics:

- The algorithm *works correctly* in the sense that active users are detected and locally grouped in a reconfigured tree topology.
- The algorithm leads to an *improvement* of the floor control response time at least for the active users of the scenario compared to the old topology. Furthermore, the effect on other users has to be shown.
- The *impact* of the reconfiguration parameters (see chapter 3.2.16.12) on specific performance measures has to be evaluated to outline setting criterion for these parameters.

For reaching the specific performance goals mentioned above, the application and system performance measures are introduced in chapter 3.5.2 to be obtained from the system. Moreover, the parameters of the modeled system are of interest for the performance evaluation. More specifically, their settings and their impact on the specific performance goal has to be outlined, which is done in chapter 3.5.3. Furthermore, means to model groupware systems are needed to obtain the performance measures analytically or statistically. This topic is addressed in chapter 4 presenting a modeling approach to evaluate conferencing systems like SCCS.

3.5.2 Performance Measures

The performance measures for the evaluation of SCCS-based systems are basically divided into *application* and *system* measures. The first consider certain actions of the groupware applications themselves without taking the underlying system into account. Different application actions may be combined to one action, e.g. copying a file. Obviously, these performance measures are strongly related to the efficiency of the underlying conferencing system which is in our case SCCS. But they also depend on the specific implementation of the application functionality.

The second type of performance measures is obtained on system level, i.e. SCCS. Mainly, the performance of the conferencing system components providing the mechanisms of the conferencing service is obtained with these measures.

3.5.2.1 Application Performance Measures

It was stated in chapter 3.5.1 that the resource management scheme and the dynamic reconfiguration are crucial for the scalability of SCCS. Specifically, both mechanisms directly affect the efficiency of floor control requests. As a consequence, the performance measures considered for the SCCS performance evaluation are dealing with floor control operations.

Average response time for floor operations

The *average response time* for floor operations in a scenario is an important indicator for the efficiency of the floor control of the conferencing system. It can be determined for the entire conference or for specific users (by taking the location within the tree topology into account).

The response time of a floor control request is measured from the moment of invoking the request by the application until the confirmation is received by the application. To distinguish between certain classes of floor operations (testing, asking for, passing), the average response time is handled separately for the operations.

Performance Gain for floor operations

For the comparison of two schemes (e.g. the centralized T.120 scheme and the resource management scheme of SCCS), an additional performance measure is defined, called *Performance Gain G(u)*. This measure often depends on the location of the observed user *u* in the tree. For a formal definition, the following holds

Definition 3.1: Performance Gain of a Floor Operation
For a given user u located in a tree, the ***performance gain*** *when applying scheme A in relation to scheme B is defined as*

$$G(u) = \left(1 - \frac{R_u}{R'_u}\right) \cdot 100\,\% \qquad \textit{with}$$

R_u *the average response time for a floor operation when applying scheme A, and*
R'_u *the average response time for a floor operation when applying scheme B.*

Thus, *G(u)* expresses the reduction of the response time at user *u* when applying scheme A instead of scheme B. Additionally, the *overall performance gain* is determined by taking the average overall response time into account instead of using the specific response time of a single user.

Response time history for floor operations

For demonstrating the effect of the dynamic reconfiguration, the *response time history* for floor operations might be of interest. This parameter represents *instant-of-time* measures for a specific floor control operation. With this parameter, temporal overload situations of the underlying system might be detected indicated by high response times. These overload points can be cross-checked with appropriate system performance measures. Similar to the average response time parameter, this measure is separately obtained for each floor control operation.

The basic definition for the response time and performance gain assumes only one floor operation. However, it is also possible to combine several floor operations to a complex *user action*, e.g. like a file copy operation. Following this definition, the time is measured from the beginning of the first floor operation to the end of the last operation needed for the entire user action.

3.5.2.2 System Performance Measures

These performance measures are obtained from the underlying conferencing system. The measures represent values on protocol level gathered from the different entities on system level, e.g. provider or links. These values can be used to outline the system load with respect to a given

application scenario. Therefore, the following system performance measures are considered.

Average provider PDU queue length

This measure represents the average buffer needed in the different providers to service traversing floor control requests. The value may be calculated as an average for all providers or for each provider respectively. The latter is more useful to detect system bottlenecks.

In general, these values are expected to be low for most scenarios, because floor control requests which are stored in these buffers do not occur frequently in most scenarios. But when considering specific scenarios with a higher application load, e.g. shared applications, this measure reflects the robustness and the scalability of the conferencing system even in overload situations.

Provider PDU queue length history

Similar to the application measures, it might be useful to obtain instant-of-time measures to detect temporal overload situations within the providers. Hence, in addition to the average queue length, the instant-of-time measure for the queue length is obtained for each provider.

Total number of generated PDUs

As an indicator for the total system load in terms of used bandwidth, this measure is gathered from the system. When assuming, that all PDUs have the same length in bytes, the used bandwidth of the system can easily be determined.

Total number of serviced PDUs per provider

This measure is the sum of all PDUs traversing a specific provider. It can be used to determine potential bottlenecks in the system. These bottlenecks might be caused by the used mechanisms. Furthermore, this measure can be used to determine the used bandwidth per provider in the system by assuming a specific PDU length (see above).

Average link load

The average link load can be used to obtain the used average bandwidth separated for each link in contrast to the average used bandwidth of the provider determined by the previous measure. The load might be expressed in terms of PDUs or bandwidth units depending on the granularity of the simulation.

Link load history

For a time-dependent evaluation of the used bandwidth on a specific link, this measure is gathered from the system.

For all presented system performance measures, it is worth mentioning that these values should be gathered for the resource management and for the dynamic reconfiguration evaluation respectively. Furthermore, there might be additional parameters being of interest when evaluating specific parts of the system. As a consequence, a performance evaluation approach should enable to obtain a wide range of performance measures.

3.5.3 Evaluation Parameters

This section identifies the system components and their parameters being used for the SCCS performance evaluation. The purpose of this chapter is not to model the conferencing environment. For that, it is referred to chapter 4 which deals with modeling conferencing systems in general and presents a modeling approach for the SCCS performance evaluation in particular.

In a first step, the components are identified in the considered environment. In the second step, the parameters within these components are outlined before showing how the parameter could be set and to what extent the parameters influence the overall performance goal.

Starting with the first step, the following picture depicts the components of a tree-based conferencing environment like SCCS. Similar to the protocol environment description of SCCS (see

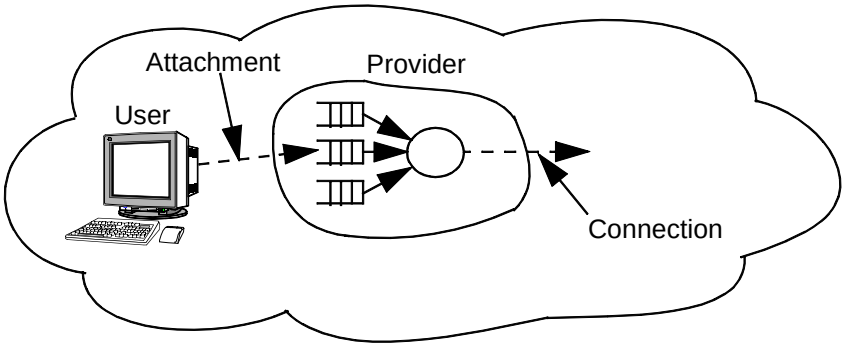


Figure 3-27: *Components of the Conferencing Environment*

chapter 3.2.2), the *conferencing environment* consists of *providers* being interconnected by *connections*. To these providers, *users* are connected by *attachments*. Note that only the control topology is evaluated, because the user data transfer system is not in the scope of the evaluation.

In the following, each component is introduced with respect to the used model in the evaluation, the parameters of the components, and the used parametrization. Furthermore, the impact of the parametrization on the performance goal is shown.

User

A *user* in the evaluation of SCCS is defined as an entity performing specific actions to reach a specific application-dependent goal. It will be seen in chapter 4 that the proposed modeling approach enables to introduce the notion of *user actions*, e.g. asking a question within a videoconference. These *user actions* lead to the invocation of correlated floor control requests which can be interpreted as a realistic workload to the conferencing system. The notion of user actions allows to define user-friendly parameters like *mean question time in a conference* which can be set for the evaluation.

For the parametrization of these user actions, measurements can be used obtained from *user tests*. The problem with these tests is that they are very costly in terms of time, network provision, and involved users. As a consequence, most of the tests are performed only for small groups caused by the costs for the tests and the bad scalability of current collaborative systems. For instance, a *shared collaboration scenario* was evaluated in [Er99] using video tapes and session protocols. Aspects like distribution among the different applications (like shared whiteboard, videoconferencing) were studied. The observations were only performed for small

groups (3 sites with up to 10 users).

As a solution of this problem, results from psychological studies ([GrSt85][KPMW94] [Walk92]) for real-life, even large scaled, but non-distributed scenarios can be used for the parametrization. Furthermore, results of user tests ([Er99][IKW98]) for small groups can be extrapolated to larger scenarios based on heuristic assumptions. Additionally, specific user actions can be set more or less intuitively. For instance, it is obvious that the mean time to raise a question within a videoconference is not within the range of several seconds but more within the range of several minutes.

The impact of the settings on the evaluation results is obvious. As mentioned above, the user actions result in a specific workload for the underlying communication system. Hence, the load of the system can be controlled by the rate of the user actions. Since the generated workload is assumed as independent from the realization of the conferencing system, the impact of the settings should be independent from different realizations of the underlying mechanisms.

Provider

A provider within a tree-based conferencing environment can be simplified as a *store-and-forward* entity [Tan96] applying the routing scheme of the conferencing protocol on incoming packets. Hence, incoming PDUs are stored in a queue and the service unit is processing the first packet in the queue according to the specific routing scheme. For a conferencing system, *first-come first-serve* (FCFS) is assumed for the request queue ordering mechanism.

A provider in the conferencing system can be modeled as a G/D/1 system. Hence, the incoming process is generally and the service time is uniformly distributed. There is a central service unit implementing the routing scheme of the environment. The incoming process is a cumulative arrival process resulting from the workload generated by the attached users within the conferencing environment. Normally, it is very hard to determine the distribution of a specific arrival process of an arbitrary provider within the tree topology. It will be seen that using the modeling approach presented in chapter 4, it is not necessary to specify the workload at a certain node, i.e. the arrival process is not an parameter within the evaluation.

It will be shown in the simulations of chapter 5 that the queues of the providers are not highly loaded which is equivalent to the measurements of the T.120-based system (see chapter 2.2.3). Obviously, this result depends on the rate of the user actions. However, for the evaluation of SCCS, it is assumed that the request queue of the providers has sufficient storage space without any overflow to deal with.

Thus, the remaining parameter for a provider is the distribution of the service time to forward the packets based on the routing decision. For this parameter, a constant service time is assumed based on the measurements of the T.120 implementation presented in chapter 2.2.4. Furthermore, the service time is chosen depending on the location of the provider within the tree (leaf, intermediate, or topmost node). For the SCCS routing scheme, an additional table lookup in the resource database is needed. First implementations of SCCS showed that there is no significant difference to the service times of the T.120 implementation. Hence, the same parameter set is used for both schemes.

Connection

The transfer of PDUs takes place using tree connections between providers causing a certain latency of the transmitted packets. Several methods can be used to determine the latency of connections on transport level. Complex methods to determine the one-way latency like proposed in [FaRu97] can be used as well as simple ping measurements which are the basis for the

Network Time Protocol (NTP [RFC1305]).

Normally, the configuration of the SCCS tree topology is more or less randomly. This is because the conferencing layer does not take the link capacity or link transfer time into account when establishing the tree topology. Thus, the impact of the link time is randomly on the results of the performance evaluation independent from the realization of the routing. Furthermore, the intention of the performance evaluation is a comparison of schemes. It is not intended to present exact figures for specific scenarios and configurations.

As a consequence, the latency of the PDU transfer on a provider-provider connection is set to the same value for all connections. For that, results presented in [FaRu97] are used for specific types of networks (LAN, MAN, or WAN).

Attachment

The transfer of service primitives from the attached user to the local provider and vice versa, either as requests or notifications, depends on the realization of the local implementation. On most modern operating systems, an efficient thread-based communication is used. This is also the case for the SCCS implementation introduced in chapter 3.3. Furthermore, there is no impact on the performance goal because the transfer via the attachments has to be realized independent from the protocol mechanisms. As a consequence, the measurements obtained from the T.120 implementation are used, which are very similar to the current SCCS implementation.

As a summary, table 3-12 presents an overview of the parameters. Note that the *mean user activity time* parameter represents an entire class of parameters describing the user behavior modeled with the approach presented in chapter 4.

Parameter	Description
mean user activity time	mean time between two specific user actions
service time provider	constant time to forward a PDU within a service provider
link time	constant time for the transfer via a single tree topology connection
attachment time	constant time for transfer from the provider to the user and vice versa

Table 3-12: *Evaluation Parameters*

In addition to the parameters mentioned above, there are several parameters to be set for the dynamic reconfiguration, presented in chapter 3.2.16.12. These parameters have standard settings shown in table 3-11.

3.6 Summary

This chapter presented the *Scalable Conferencing Control Service* (SCCS) as an approach to provide a generic conferencing service to facilitate and accelerate the implementation of conferencing applications for tightly coupled scenarios.

In the first part of the chapter, SCCS services and protocol mechanisms were presented. It was shown that the SCCS service model covers main aspects of the service requirements for conferencing environments introduced in chapter 2.1.2. The *conference management* services consi-

der aspects like the provision of a conference database and mechanisms to merge or split conferences during runtime. Furthermore, functionality is provided to reconfigure the conference during runtime of the conference without blocking the conference while ensuring consistency of the conference control data. The *multipoint transfer* services enable all kind of transfer patterns from one-to-one to many-to-many by introducing a channel concept as a transport layer abstraction. Public as well as private channels are provided with several administration facilities as a basis service. But the channels are only administrated by SCCS. The transfer of the user data is directly realized by underlying transport protocols. As a consequence, SCCS is suited to transfer real-time as well as non-real-time data. Hence, it offers a homogeneous transfer system on conference layer level for several data types in contrast to the H.323 standard. Furthermore, more efficient transport layer functionality, like multicast, might be used by SCCS. Additionally, SCCS enables the *passive multicast data reception.* This feature extends the applicability of SCCS to scenarios with only a few active entities but with a large passive auditorium as a combination of the paradigms of tightly and loosely coupled environments for the transfer of multipoint data. An additional layer is proposed for the SCCS environment to provide *user data marshaling* for the transfer of typed user data like objects. For the *support of distributed applications,* SCCS provides a token concept to implement access and floor control in group communication scenarios. Efficient administration functionality for the tokens is provided, so sophisticated floor control functionality is available.

The services of SCCS can be integrated into the *Internet Multimedia Conferencing Architecture* (see chapter 2.3.2) as a supplement for tightly coupled conferences. With this integration, SCCS might easily be extended by commonly used announcement and initiation functionality in the Internet. Furthermore, group key and key distribution techniques might be integrated to provide secure conferences.

Beside the service definition, the protocol mechanisms to realize the services were introduced. The main design issue for the protocol mechanisms was to provide the SCCS service functionality with a high scalability in terms of conference size and geographical distribution. For that, SCCS consequently uses *multicast transport layer* functionality, both for user data and control traffic. Furthermore, a *resource management* scheme was proposed to provide low response times for floor control requests even in large scaled scenarios. Furthermore, a *dynamic reconfiguration* of the tree topology was proposed by detecting activity in group communication scenarios and grouping active user locally in the tree topology by using the tree reconfiguration functionality of SCCS.

The services and protocol mechanisms were introduced in this chapter in detail. A more formal definition of the services and protocol mechanisms was not shown in this chapter due to the complexity of the specifications. However, the service and protocol specification are available as technical reports ([SCCS98][SCCS99]).

Similar to the presented conferencing environments of chapter 2, SCCS was assessed with respect to the requirements given in section 2.1.2. First, it was shown that most of the requirements are covered by SCCS. Aspects like location service, announcement, group security, and accounting are not provided by SCCS. However, they can be integrated in the approach as separate components in parallel to the control structure of SCCS by using existing or currently developed approaches. Some services to provide *robustness* on service level are provided by SCCS, but error recovery of ungraceful destruction of (parts of) the tree is not yet provided. Robustness on protocol level in terms of testing and validating the mechanisms was enabled in SCCS by specifying the services and the protocol using message sequence charts. However, the validation of the mechanisms was not performed due to the complexity of the entire service and its underlying protocol. For the *scalability* aspect, two main protocol mechanisms were propo-

sed. The usage of multicast transport layer functionality shifts the scalability aspect from the conferencing to the transport layer. The second mechanism, the proposed resource management scheme together with the dynamic reconfiguration of the tree topology allows to improve resource requests if the requesting entities are located closely within the tree topology.

In addition to the qualitative requirements assessment, a performance evaluation of the SCCS environment was introduced for a quantitative evaluation. It was not the intention of the chapter to perform this evaluation. Its purpose was to define the performance evaluation goal. For that, the mechanisms were outlined to be evaluated, namely the SCCS resource management scheme and the dynamic reconfiguration. For both mechanisms, the specific performance goals were presented. Furthermore, the components of a conferencing environment were outlined which have to be considered for the evaluation. Moreover, application and system performance measures were presented to be obtained when evaluating the SCCS mechanisms. For each component, the parameters which have to be considered for the evaluation were outlined. Furthermore, the impact of the evaluation parameter settings on the specific performance goal was shown.

It can be concluded that the proposed conferencing service fulfils a wide range of the service requirements for conferencing environments. Furthermore, it provides sophisticated protocol mechanisms being designed for large scaled conferencing scenarios. For two specific protocol functions, a performance evaluation was proposed regarding the achieved improvement when applying the mechanisms.

For the provision of means to perform this evaluation, the next chapter will present techniques to model groupware scenarios leading to an modeling approach to be used for the performance evaluation of SCCS.

CHAPTER 4

Group Communication Modeling

After introducing existing group communication systems and presenting the Scalable Conferencing Control Service (SCCS), this section deals with the problem of how to model group communication for performance evaluation purposes in general.

Different aspects are covered like structural object modeling as well as modeling the communication, user behavior, and the access control to resources in a distributed environment. The resulting approach enables to model and evaluate several aspects of group communication environments ranging from application aspects like user behavior to the evaluation of protocol mechanisms of the underlying conferencing system.

The chapter starts with a motivation in section 4.1 for the problem of modeling group communication systems. In particular, requirements for a group communication modeling approach are defined. After that, a survey of existing distributed systems modeling proposals is presented in chapter 4.2. It is shown to what extent these approaches fulfil the requirements being defined as crucial aspects in group communication modeling.

In the second part of the chapter, a description language is proposed in section 4.3. This approach is used as a modeling framework enabling the modeling of both the groupware system structure and the communication between distributed entities. For the realization of the communication and user behavior description, two approaches are presented. The first one, described in chapter 4.4, allows the simulative evaluation of the group communication system, while the second approach in chapter 4.5 enables an analytical evaluation.

The proposed modeling technique is used in chapter 5 and 6 to evaluate main mechanisms of SCCS, namely the resource management scheme and the dynamic reconfiguration of the tree topology.

4.1 Motivation

As mentioned in chapter 2.1.2, there are several requirements for a generic conferencing layer to facilitate the implementation of group communication applications. Approaches like SCCS try to fulfil these requirements by providing appropriate services and protocol mechanisms. But the provision of these services and mechanisms does not cover the realization of the specific

group communication application as such. *Toolkits* for collaborative systems ([ADH93][AnBa93][Ar92][CBV92][Cr90][HiGe98][RoUn99][SiSchu98]) might help developers and users to facilitate the specification and implementation of appropriate groupware applications. But at the end, the structural design work remains at the developer of the application. Furthermore, if the system is realized, the performance of the resulting application might be of interest. Communication aspects like resulting control and data traffic are of importance as well as user behavior aspects (e.g. mean time to invoke a specific action) and the response time of the system for this specific user action.

Simple performance evaluation approaches like testing real-life implementations are often too limited to the specific scenario to get useful results. Furthermore, such tests need a large amount of effort to run, especially in large scaled scenarios. Additionally, the system is already realized when the evaluation is done by testing. But it might be more useful to evaluate the system or even components beforehand without implementing them. As a consequence, the usage of a scenario-dependent model to obtain simulative or analytical results is crucial. Moreover, incorporating user profile data gathered from real-life scenarios might be useful to obtain more realistic results.

Hence, modeling group communication systems tries to provide means to enable both *structural design* of specific groupware applications and *performance evaluation* of the system without implementing and installing it, i.e. in the design phase of the system. Most modeling approaches which are presented in the related work section cover only some or specifically one modeling aspect. Thus, the developer of a groupware application has to deal with several approaches and tools for different aspects, he/she is interested in. As a consequence, it is crucial to provide the modeling facilities in an integrated approach from a developer's and user's point of view. It has to be emphasized that it is not intended to cover neither *software model checking* aspects like proposed in [HoSm99] or offered by tools like SPIN [Holz97] nor protocol validation and testing aspects like in ([KSCK99][ObKe99]). Thus, the modeling approach, which is outlined in this chapter, deals with modeling and performance evaluation aspects in the design phase of a groupware system.

For that, the following sections present a model to describe group communication systems both on structural and behavioral level. For clarification what is needed in such a modeling approach, the next section outlines requirements which should be fulfilled by a groupware model.

4.1.1 Requirements

In this section, requirements for a group communication description are presented which should be covered by a modeling framework for group communication systems:

- **structural distributed object modeling**: the involved objects of the group communication system should be defined by the model. The focus is not on specifying the objects in detail like specifying *interfaces* or *attributes*, but on modeling the structure of the distributed objects within the collaborative environment. For that, a detailed object description is not needed. *Inheritance* and g*rouping* of objects should be supported in order to *structure* the group communication system.
- **access modeling**: the approach should enable the modeling of the access to objects. This modeling aspect is twofold. The first part covers the *access rights* associated with an object. Many objects are only readable for certain users in a system, while other objects might be readable as well as writable. Thus, the modeling framework should provide *positive* (explicit

permission to write) as well as *negative access rights* (explicit denial to write) to objects. This may be realized by defining *write* as well as *read access sets* of users. Secondly, the access model should consider the aspect of *access control*, which means that access rights may change dynamically during runtime of the system based on *rules* which are defined in a *social protocol*.

- **user behavior modeling**: the model should allow to describe the behavior of an entity using the system in detail. For that, *timed actions* should be supported which can be parametrized based on statistical data, i.e. *user profile input data* should be supported.
- **control traffic modeling**: the access to distributed objects is realized by defined control traffic which is exchanged between the involved entities. This traffic should be abstracted and described separately on a detailed level to allow the generation of a scenario-specific *control workload*. For that, a standard access model to objects should be defined which is general enough to cover different access realizations.
- **data traffic modeling**: enabling the access to distributed objects often invokes data traffic, especially when using streaming objects like audio and video. The traffic of these objects should be modeled in detail to allow the generation of a scenario-specific *data workload*.
- **independence from realization**: the description should be independent from the realization of the communicating entities in the system. Only functional requirements for the control and data entities should be described independent from the realization of the entities. Nevertheless, realization approaches are presented as an extension.
- **simulative and analytical evaluation**: the description of the entities within the group communication applications should enable simulative as well as analytical evaluation of the system, depending on the realization of the entity description.

For the simulative and analytical evaluation of a system, it is important, what performance measures can be obtained by the evaluation. The goal is to provide a wide spectrum of measures. A set of performance measures is given in chapter 3.5 for the performance evaluation of SCCS. Nevertheless, there might be further measures in a group communication system. The proposed evaluation facilities are assessed according to the provision of performance measures they allow to obtain from the system.

It can be summarized from the requirements that a descriptive approach for group communication systems should provide a structural model of the static objects, their access rights, and the communicating entities on control and data level in a collaborative environment with functional requirements for the realization of the communicating entities. Additionally, a simulative and analytical evaluation of the system should be enabled with a wide spectrum of possible performance measures to be obtained.

4.2 Related Work

Before proposing the modeling approach used for the performance evaluation of SCCS, this section presents a brief survey of group communication system modeling approaches. It is divided into three categories. The first one considers *stochastic models* especially designed to allow for an analytical evaluation of the system. The second one is dealing with *workflow approaches* designed for group communication systems or business processes. At last, the third one covers *behavior descriptive approaches* which fit best with the existence of highly correlated actions within groupware systems.

4.2.1 Stochastic Models

The first category which is presented in this section is dealing with stochastic models of groupware systems. These models reach from simple probabilistic approaches to more complex methods like stochastic Petri nets. Common to these approaches is that they have been designed to evaluate a system analytically. Due to this design goal, some restrictions have to be made to handle the resulting system numerically.

4.2.1.1 Simple Probabilistic Approach

The first approach uses a simple model for a group communication system. The distribution of a certain event is given for each user in the system. Using simple probabilistic calculations allows to calculate certain performance measures. For the sake of simplicity, it is assumed that the events are independent from each other, which is a strong restriction, especially for groupware systems. Hence, the high correlation of groupware events and messages is not taken into account.

The main disadvantage of this simple approach is that scenarios using more complex rule-based access models are very difficult to handle, both from the modeling as well as from the numerical point of view. Furthermore, the calculations are more or less specific for a defined scenario and even for the observed user. Thus, the parameter determination has to be changed for other scenarios or other users. As a consequence, this approach is only useful to get an at least rough impression of the performance of a system. For more complex scenarios, more sophisticated models have to be applied.

In chapter 5.2, this simple approach is used to demonstrate the complexity of the calculation.

4.2.1.2 Queueing Networks

A more sophisticated approach to model communication systems is to use *queueing networks* ([Klein75][Klein76][Tak91]). Each component within the system is modeled using a *server component*, consisting of one or more *service units* and a *queue* with a given amount of storage. For the dynamic description, the notion of a *flow* is used. Its carried information, e.g. *clients*, enters a component of the system as an *input flow* according to given parameters. After being queued and serviced, it leaves the component as an *output flow*.

Most of the approaches to model queueing networks suffer from the restrictions being necessary to determine the performance measures. For instance, queueing systems like *Jackson networks* [Jack63] assume input flows with exponentially distributed interarrival and service times. This assumption simplifies the evaluation of the network, but is not at all a realistic assumption, especially for real-time services [PaFl95]. As an extension, *Fork-Join networks* were proposed in [LPM98]. But in this approach, a stream is *forked* in sub-streams which must be *joined* to a stream again after servicing. This assumption is a very strong restriction, which might be true for the proposed application (multipoint database queries [LPM98]) but not for collaborative environments in general. Additionally, the approach restricts input flows to Poisson processes similar to Jackson networks.

Another approach to evaluate a system using queueing models was proposed by Kühn [Kuehn79]. His method defines the flows and service times within the system using the first two moments of the distribution. This assumption enables an approximate solution of the system. Whitt implemented this idea in his *Queueing Network Analyzer* (QNA) approach [Whitt83]. This model enables the approximate evaluation of queueing networks with GI|G|m

nodes, i.e. the distribution of the arrival and service process are generally defined by the first two moments of the distribution. Further extensions of this approach were proposed ([Haver95][Haver98a]) even for the usage in multicast environments [SSH98]. For the multicast extension, a pre-processing step is applied to the queueing network. In this step, input parameters of a resulting QNA system are determined based on multicast routing probabilities. Thus, the multicast-based QNA network is transformed to an ordinary QNA network (see figure 4-1 [Schu99]), so that the normal QNA calculations can be applied.

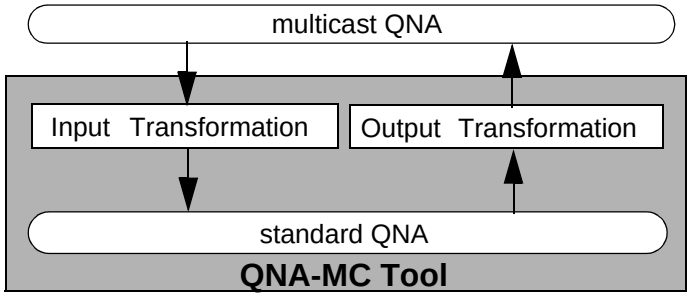


Figure 4-1: *Pre-Processing by QNA-MC*

In contrast to the Fork-Join model, the multicast extension of QNA does not require to join multicast streams together after forking them. Thus, arbitrary multicast routes may be defined with routing probabilities for each route.

Furthermore, the QNA model allows general distributions for the arrival (and the service) processes, which might be interesting when using user-profile data for generating the arrival processes. Using the techniques of QNA allows to determine the usual performance measures of queueing network, e.g. average queue length or response time.

As a main disadvantage, this multicast approach requires routing probabilities to be defined for each route within the queueing network. The problem is that these probabilities are normally not given in the scenario. Thus, the QNA model needs a further pre-processing unit (similar to the QNA multicast extension and located on top of the multicast extension) to calculate the routing probabilities from the input parameters of the system applying the routing mechanisms of the used resource management scheme. These routing probabilities have to be extracted from the considered scenario which defines the control traffic implicitly by the given protocol.

Applying the QNA method allows to reduce the computation effort and to use more generic information flows, compared to methods like Fork-Join networks and simple probabilistic approaches. However, applying QNA to scenarios with complex access models and user behavior especially in large scaled scenarios would lead to rather complex QNA systems. Thus, it can only be used for simple evaluations.

4.2.1.3 Stochastic Petri Nets

A more sophisticated approach to model finite-state automata systems (and therefore dependencies between events) and to evaluate these systems analytically are *stochastic Petri nets* ([BaKr96][BRR87][Haver98b]). This approach models a system using a directed bipartite graph with *places P* and *transitions T* which can be mapped (under certain circumstances) to a continuous time Markov chain enabling the analytical evaluation of the system. For small systems, a simulative evaluation is provided by this approach as well.

Despite their advantages on analytical level, stochastic Petri nets are not suited for the structural

design of large systems. Furthermore, the effort to determine the resulting Markov chain numerically grows rapidly with the size of the system, since they do not provide means for hierarchical structuring.

There exist a lot of extensions of the original modeling approach ([BaKr96][BRR87] [Haver98b]) for the analytical part. An interesting extension, called *Object Coordination Net* ([GGW98][WGG97]), is presented in chapter 4.2.3 introducing structural means to stochastic Petri nets for the structural design of large systems. Furthermore, behavior descriptive approaches like the *behavior network model* [Kung93] use stochastic Petri nets for the analytical evaluation of the system and provide a mapping of the original system to stochastic Petri nets. This is also the case for the proposed modeling approach presented in chapter 4.3 which uses stochastic Petri nets for the description of communicating entities within a group communication system to evaluate the resulting system analytically. As a consequence, stochastic Petri nets are introduced in more detail in chapter 4.5.1.

4.2.2 Workflow Approaches

A large field of research in the area of collaborative systems is dealing with *workflow description* and *workflow management*. The spectrum of applications covers collaborative computer systems as well as business and production processes and the dependencies between tasks and entities in these systems. The following section gives a short overview of this research topic. For that, the main weaknesses of these approaches are outlined in the context of modeling groupware systems based on the requirements defined in chapter 4.1.1.

The approach presented in [FaLo96] defines group communication models in the context of the *Reference Model of Open Distributed Processing* (RM-ODP) [X904]. The authors propose an *object-group* as a collection of *objects*. Objects organized in object groups are accessed externally by a single name. Thus, this definition allows to define a structure of the system. To each object group, a *group context* is defined describing the *objectives* of the group, the *group policy* (group management and cooperation), and the *membership specification* consisting of a list of members. The group context is created and managed by the *group administrator*. Members of an object group may join or leave the group during runtime. For the description of interaction between the different objects (or objects groups), the approach defines basic requirements of group interaction, including *distribution*, *collation*, *ordering*, *quorum*, *synchronization*, *concurrence control*, and *atomicity*. Additionally, different policies for these forms of interactions were defined. It is not defined in the approach how to specify the interaction between the objects formally. The approach only defines a high-level abstraction of group communication interaction between objects. It is very difficult to break the model down to specific scenarios for modeling aspects mentioned in the requirements of chapter 4.1.1.

Gulla and Lindland present a model in [GuLi94] based on the *Input-Process-Output* (IPO) paradigm of describing workflow within a system. This basic scheme defines the production chain within a system and is extended by aspects from the *Consumer-Supplier* paradigm (describing the consumption of services provided by service suppliers). This means that coordination facilities are integrated in the approach, defined as the *Actor-Service* model, to describe actor relationships in coordination activities. These extensions are incorporated in *Data Flow Diagrams* which, in a combination with the Actor-Service model, enable to model *production* as well as *coordination* activities. Similar to the previous approach, this model allows the high-level definition of workflow in processes, including specifically production processes. It is not within the scope of this approach to evaluate these processes analytically or simulatively.

The approach presented by Tang and Veijalainen [TaVe95] introduces the concept of *inter-task*

dependencies in transaction workflows and their enforcement. The workflow is divided into several sub-tasks with dependencies among these tasks. These dependencies are described separately for each task, so the transaction workflow is described in a modular manner. In their work, the authors extend this idea in the sense that the enforcement of inter-task dependencies may be distributed. This is a crucial requirement for distributed workflows. For that, an enforcement protocol is proposed based on automata for each task of the workflow. The method does not define how to use the automata description for evaluation of the system.

In [ShDe92], a model is presented considering the access to objects in distributed environments. It is not really a workflow approach in the common sense. It is more intended as an extension to existing workflow approaches to consider the access to objects. For that, the authors propose basically a set *S* of *subjects*, a set *O* of *objects*, and a matrix *R* of *explicit access rights*, where the rows represent the subjects and the columns the objects. The approach allows the notion of *positive* (to allow access) or *negative* rights (to deny access). Furthermore, the *inheritance of rights* is supported by defining object groups and introducing an *inference function* for access rights. The idea of *right inheritance* is extended to the notion of *multiple user roles.* This means that a role is mapped to one or more subjects with a defined access to objects or object groups. The authors explicitly do not define a social protocol to change the access dynamically after specification. This is the major drawback of the approach, because it does not support to change access rights dynamically (dynamic access control) based on rules being defined in a social protocol.

It can be summarized that models of workflow approaches usually cover the data flow but not the control flow. Hence, they do not allow to model the control traffic in a distributed environment. More specifically, the workflow approaches do not provide sophisticated access control models, e.g. by providing the notion of *roles* and *rules*. The systems are typically defined as a whole. As an exception, the approach in [TaVe95] tries to model a workflow divided in tasks but not provided by roles in a system. As a consequence, the workflow approaches model access control, user behavior, and control traffic only insufficiently.

In the following section, approaches are presented dealing more with the entities and their behavior in the system instead of describing the resulting workflow.

4.2.3 Behavior Descriptive Approaches

A lot of research effort was dedicated to the topic of behavior descriptive approaches ([ALL88][BeMa99][Chen76][GGW98][HZS95][JEJ95][JoSw99][Kung93][LZGS84][Ould95] [QPSFV97][Sche98][She87][SSM90][VaPu99][WGG97]).

Some of these approaches simply extend workflow approaches e.g. by introducing probabilities associated with transactions [VaPu99]. Others deal with introducing behavior descriptive elements like finite-state descriptions associated to objects. The major drawback of these models is that each of them is focussed on certain aspects of the requirements of chapter 4.1.1. This section tries to give an overview of a selection of these behavior descriptive models.

Object Coordination Nets

As a combination of structural design means using UML (*Unified Modeling Language* [JEJ95]) class definitions and behavior description means using an extension of Petri Nets, the *Object Coordination Nets* (OCoNs) were proposed in ([GGW98][WGG97]). In that approach, the static object description in terms of interfaces and attributes is realized using a UML class notation.

To each object, a *resource allocation net* is associated, representing the state-dependent resource allocation within the object implementation which is visible to the user. The implementation of member functions (referred to as a *service*) in an object is described as an *input-process-output* diagram, called *service net*. The process part in the service net also represents the visible state-dependent behavior of the object implementation. For both nets, an extension of Petri Nets is defined. For that purpose, *resource* and *event* places are introduced to distinguish these places from others. The event processing (using event places) is used to describe the control flow within the implementation. Resources are more fixed and normally not consumed. For this purpose, bidirectional arcs are defined for resource places to indicate the non-consuming resource. Service nets may also be *folded* to enable hierarchies of implementations. Figure 4-2 shows the class structure within the OCoN approach. The shaded gray part defines

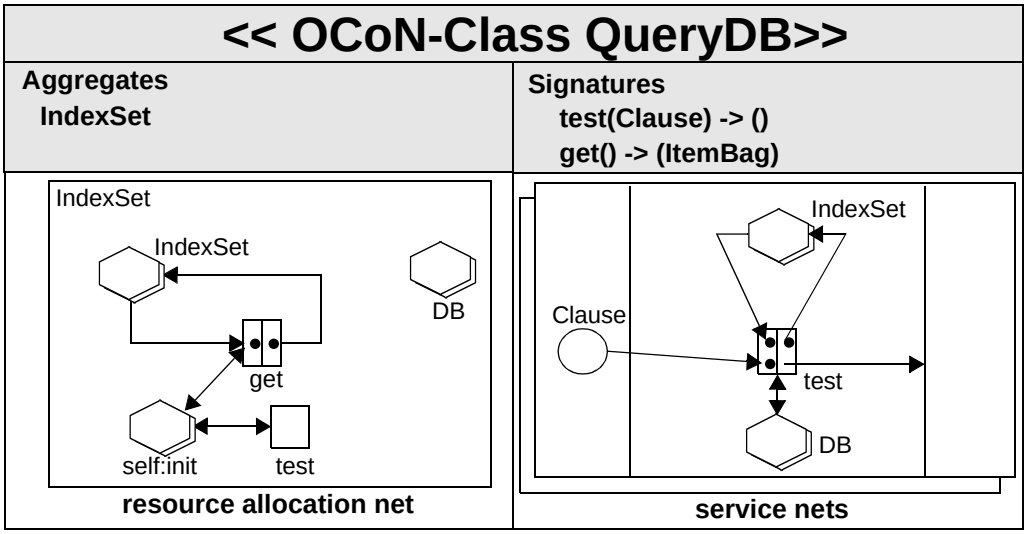


Figure 4-2: *Object Coordination Nets - Class Structure*

the static structure of the object using the UML class definition. The resource allocation net describes the resources of the object and the allocation behavior, while each member function (two in the example) has its own server net defining the visible behavior of the implementation in the input-process-output notation. The approach is based on a graphical notation, and a modeling tool for OCoNs is available.

The approach does not provide the mapping of the OCoN to commonly used stochastic Petri nets like *colored Petri nets* [BaKr96] for performance evaluation of the resulting system. It also neither offer a role-based description of the system nor an access control model.

Architectural Modeling Box for Enterprise Redesign (AMBER)

The AMBER approach, proposed in [JoSw99], is based on an actor/action/item model intended for business model description. However, it can also be applied to collaborative systems. An *actor* in the AMBER model designates a resource carrying out a specific business process. Structuring actors by grouping them and building hierarchies is also supported. The interaction between actors is realized via defined *interactions points* with a binary or ternary relation.

The *action* is the main concept in the behavior domain of AMBER. There are two attributes associated with each action, the *actors* which are involved and the *result* of the action. Causality relation is provided in the form of boolean operations which can be supplied with certain

constraints. Additionally, *triggers* can be defined as immediately enabled actions. Furthermore, actions may be grouped for structuring behavior, and they may be iterative and replicated.

The *item* in AMBER models the objects on which the actions are performed. Items may be associated with actions and actors in the model. Structural design in terms of interfaces is not yet supported. However, the notion of *profiles* is defined for specialized type definitions.

Analyzability is one of the main aspects to be covered by AMBER. It offers structure as well as semantic analysis. For performance evaluation, mappings to analytical techniques like queueing theory or graph models are used [JoSw99]. Additionally, AMBER provides a graphical modeling tool to easily design models.

The modeling approach of AMBER is basically focussed on the level of processes. Thus, communication or access control aspects, like access rights, are not supported by this approach. Additionally, the behavior description is defined as simple as possible, which is sufficient for business processes but might be a drawback when modeling groupware applications.

Manifold

The MANIFOLD approach presented in [PaAr97] defines an event-driven control-based coordination framework in contrast to the data-driven *Linda* model [ACG86]. It defines the notion of *agents* representing active entities in a collaborative environment. Two different types of agents are distinguished, namely *worker agents* and *manager agents*. The latter are responsible for the coordination of the former which are performing any kind of computational work, i.e. the worker agents are managed by a certain manager agent. In order to define management issues like computational effort, hierarchies may be built by managing manager agents like worker agents. The lowest level of computational effort cannot be further subdivided. Thus, it defines the lowest level of the hierarchy. Hence, the MANIFOLD model introduces a hierarchical *role-based* approach consisting of manager and managed agents.

The communication between agents is realized by establishing *streams* between agents dedicated to defined *input/output ports* at the agents. The description of the agents themselves is event-driven by reacting on certain *events* in the system defined as a finite-state automaton. This reaction to events may lead to new stream connections or to the disconnection of previously established streams.

MANIFOLD does not provide a structural model to describe the object data on which the actions are performed. Instead of extending the data-driven LINDA model by combining this approach with the control-based MANIFOLD framework, it just defines a control framework for activities. Therefore, the approach lacks on data aspects like structural design, access control, and access patterns in the user behavior.

Behavior Network Model

Defining the notion of *conceptual information modeling*, several approaches were proposed ([BeMa99][Chen76][Kung93][She87]) for modeling communication systems in general. A very sophisticated approach was proposed by Kung in [Kung83] introducing the *behavior network model* (referred to as BNM in the following). A BNM provides static structural description of the system as well as modeling the dynamic aspects in form of *processes* and their *interfaces*.

The static description within a BNM defines *entities*, their *relationship*, and *attributes* using the *entity-relationship model* proposed in [Chen76]. Based on these notions, complex data structures are defined which can be used to describe the static structure of a system with a fine granularity. Additionally, a graphical representation of these structures is presented to simplify the

specification.

For the dynamic description of the system, the notions of *process* and *interface* are defined. The behavior is modeled by a workflow between processes using the defined interfaces under *temporal constraints* (expressing the dependence of a certain behavior on previous actions). Additionally, Kung presents a mapping of the dynamic description to Petri Nets with a construction method to determine the reachability set of the Petri Net (see chapter 4.5.1) from the BNM specification. Thus, the BNM provides a model for structural system design as well as for behavior description with a mapping to Petri Nets for analytical performance evaluation.

The main drawback of the BNM approach is the lack of an access control model and the missing modularity in the system design, which means that the entire system has to be defined. It is not possible to build a system from a pool of modules by instantiating the modules to build certain scenarios with a given set-up.

Unified Modeling Language (UML)

As an approach to model large scaled software systems, the *Unified Modeling Language* (UML [JEJ95]) was developed. The main focus of UML is on the structural description of large software systems in high detail for software engineering. Additionally to the structural design, UML allows to describe behavior within the software system.

UML provides *class diagrams* as means to model software system structure on a class-based level. Each class contains *methods* and *attributes* which may be declared as public, private, or protected. Classes may be inherited by other classes or implementation relationships may be defined. Furthermore, relations between classes may be defined as well. Class diagrams can be grouped to *packages* to enable structural ordering of the software system into components. It can be seen that the class diagram mainly provides structural means of object-oriented languages like C++ [Strau98] or Java in a graphical manner. Tools like *Rational Rose* [Rat98] implement this graphical representation of classes as a case tool and also allow to generate code from the class diagrams to simplify the implementation steps.

In addition to the detailed structural description of the software system, UML also provides modeling behavior in the software system by defining interaction with the outside world. For that, an *actor/use-case model* is provided. *Actors* in the UML notation are not part of the software system. They represent entities interacting with the system and are normally defined in the requirement phase of the software project. *Use-cases* describe the interaction between an actor and the system, or more formally, a use-case is a "sequence of transactions performed by a system that yields a result for a particular actor" [JEJ95]. The use-case is described by a *flow of events* to perform the transactions. Additionally, use-case relations may be defined. All these descriptions are graphically represented within a *use-case diagram*. Usually, these diagrams are hierarchically ordered to enable structured modeling.

Furthermore, UML supports *sequence diagrams* enabling the description of message sequences (or sequence of transactions) within the software system to perform certain tasks. These sequence charts are very similar to *Message Sequence Charts* (MSCs [Z120]) which are used to specify exchanged messages in communication protocols. These sequence diagrams are also integrated in the use-case diagrams to describe actions needed to implement certain use-cases.

It can be summarized that UML provides sophisticated means to model large software systems. The main focus of UML is on the graphical representation to define relationships and dynamic behavior of distributed systems. Moreover, it provides structural modeling of groupware application in a very detailed manner following the object-oriented software design paradigm. Furthermore, the system behavior may be modeled based on actors and use-cases with sequence

diagrams as a supplement for transaction description.

There are several drawbacks of UML regarding the requirements defined in chapter 4.1.1. The first one is that the behavior description (use-cases) is a sequence of transactions only. Thus, for modeling more complex user behavior, the roles within a group communication scenario have to be defined as a system component itself, which then invokes actions to other system components described as a sequence chart. Moreover, the access model, consisting of access rights and access control, is only modeled insufficiently. Positive or negative rights have to be modeled with means like public and/or private properties. A dynamic change of the access is very difficult to model. Moreover, a differentiation between control and data traffic has to be modeled within the resulting software system. It is not covered by the modeling approach as such. UML was meant to provide a modeling approach for large software systems. It was not intended to provide performance evaluation means for checking the software performance without implementing it, neither analytically nor simulatively.

As an extension of the pure UML definition of a system, a combination of SDL and UML is standardized by the ITU in Z.109 [Z109]. With this approach, SDL is used within a UML system to describe component behavior and to enable software system checking, testing, and validation. Nevertheless, similar to the pure UML approach, communication aspects, like control and data traffic differentiation and access control model, are only insufficiently supported.

4.2.4 Summary

As it was shown in the last section, there are many approaches to model systems, software, its components, and the workflow between the different actors in the system. The presented stochastic approaches enable an analytical evaluation of a communication system either by using simple probabilistic or more sophisticated queueing network techniques. The main drawback of these analytical methods is that the computational effort rapidly increases even in relatively small scenarios. This effect is aggravated when defining more complex access models and assuming complex user behavior. Additionally, the assumption of independent requests is a strong restriction for groupware applications which typically generate highly correlated traffic.

Most of the workflow approaches provide the description of the (resulting) data workflow within a flow and the dependence between tasks and entities. Modularity or the notion of roles is only poorly supported by the models. Furthermore, modeling user behavior is also not provided by these approaches. The approach presented in [ShDe92] defines an access right model supporting the notion of multiple user roles and inheritance of rights, which is meant as an extension to existing approaches. However, it lacks of a sophisticated access control model to support dynamic changes of the access rights.

The proposed behavior descriptive approaches allow more complex models of the communication system, because they support additionally the notion of *behavior* in some sense. Some of the approaches use stochastic methods to evaluate the system analytically, e.g. Petri nets. Each of these approaches has one or more weaknesses, because they are concentrating on specific modeling aspects. This leads to an incomplete model of the group communication system regarding the requirements defined in chapter 4.1.1. The main weaknesses which are common to all approaches are:

- *missing detail of the communication aspect*: Only the workflow from one object to another in the system is modeled, but not the specific control and data information which is exchanged within this workflow. Aspects like control traffic to enable the data workflow

and the data itself are not described in detail. Especially streaming objects like audio and video are described insufficiently with these approaches.

- *poor access control model*: Most of the proposed approaches do not cover the access control requirement sufficiently. Some of them define access rights, but none of the approaches allows a dynamic change of access rights. But this feature is crucial for groupware systems where most of the scenarios have a defined access control scheme specific for the given scenario.
- *less detailed user behavior*: Group communication systems are used by humans which interacts with distributed objects. This interaction is often based on *timed actions* (e.g. *x* mouse movements per minute in a shared application). These timed actions create typical control and data traffic, which is not covered by the related work.

Especially the first two points are of interest when modeling groupware applications for the evaluation of group communication infrastructures like SCCS. But also the last point is very important when assuming a more complex user behavior.

As a consequence, the related work presentation underlines the need for a modeling approach which covers a wide range of requirements, especially the three mentioned above. The next section introduces a new modeling approach which fulfils these demands.

4.3 Group Communication Description Language

This chapter presents the *Group Communication Description Language* (proposed in [Tro00a]), referred to as GCDL, as a modeling approach for groupware applications. It is designed to fulfil the requirements defined in chapter 4.1.1. After outlining the idea for the model which is used within GCDL, the components of the approach are introduced. Furthermore, it is described how to use GCDL as a load model to evaluate SCCS. Finally, two examples for modeling real applications are given.

4.3.1 Idea of GCDL

A group communication system might be described statically by distributed *objects* which may also be combined to *object groups* to form the *scenario* on the highest level of abstraction. These objects may be organized centrally, e.g. the content of a database, or they may be distributed (like multicast streams for audio and video). A defined set of *access rights* is associated to each object. Both, negative and positive rights are provided. Due to the inheritance of objects, GCDL supports the inheritance of access rights as well.

The communication in collaborative environments can be interpreted as a competition to gain access to certain objects within this environment. This competition is realized according to defined *roles* (e.g. the chairman in a lecture scenario) and *rules* (when to grant object access to another user) which have to be applied to gain access to objects.

The roles which are predefined at start-up of the system and associated to objects and participants of the system interact according to defined rules which can be interpreted as a *social protocol*. This is the basis for the communication on control level and for a given *behavior* to model the user actions. The social protocol defines the control model which is used to access different objects within the system. Due to the hierarchy of objects introduced by inheritance, roles may also be structured hierarchically, because roles are assigned to objects.

The access control among the autonomous instances of the corresponding role itself is abstracted by defining *floors* [DoGLA95] (or *tokens*) which are associated to an object. Floors are often used in group communication infrastructure systems to realize access to distributed objects. Instances may ask for a floor or the current floor holder may give the floor to another instance. Floors may be exclusive or non-exclusive. Thus, the resulting *control flow* of this access coordination consists of simple *token-passing* and *token-asking* transactions [DoGLA95] based on the defined rules to grant the access to the object. With the notion of floor-based access control, the change of access control and therefore the dynamic change of access rights is supported, which is crucial for collaborative environments.

If the access is granted on control level, access to the object itself takes place based on certain traffic patterns on data level which might often be described statistically (e.g. by Markovian models or trace-driven descriptions [Rose97]). These traffic patterns, applied on the object, result in a *data flow* which is sent to the underlying system.

Thus, an instance of a role within a participant of the collaborative environment is divided in a *control* and *data* process. The former is responsible to gain access to the associated object by applying the rules of the social protocol, while the second one becomes active when the access is granted and applies the data traffic patterns on the object.

The communication itself is realized by the *environment* of the system representing underlying communication systems (e.g. a conference control protocol together with a streaming protocol for audio and video).

As a consequence of the above, the following steps have to be applied:

1. define the scenario which is considered,
2. define the objects within the scenario and merge them to object groups,
3. define the access rights associated to the objects,
4. define the roles for each object,
5. define the access model for each object by using the floor control protocol,
6. define the behavior for each role,
7. define the data traffic pattern for each role,
8. define control and data processes for each role implementing the user behavior, the access model, and the data traffic pattern.

For the provision of *implementation independence*, GCDL does not cover the implementation of the control and data processes. It can be seen from the outline of the modeling approach that the requirements defined in chapter 4.1.1 are fulfilled with GCDL, while the implementation of the access and data traffic modeling is not directly addressed to allow the use of different implementation techniques.

Example - Large Meeting

In the following, a short example is presented to clarify the modeling approach of GCDL. This is done on a textual basis, because a formal description has not yet been defined.

The scenario which is considered is a *large meeting scenario* (see figure 4-3). The objects within this particular scenario are the audio and video stream of the currently speaking person. Everybody in the scenario is allowed to read and write to these objects, i.e. to send audio and video data to other users. Although the access rights for the objects are very simple, there is nevertheless an additional *access control* model for this scenario which describes the dynamic change of the current access.

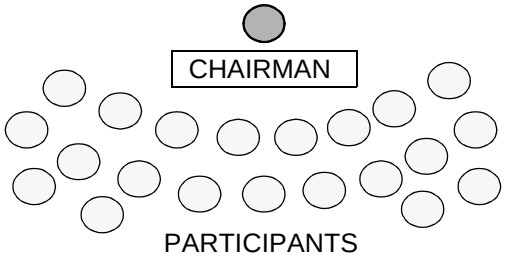


Figure 4-3: *Large Meeting Example*

For that, two *roles* are defined in the scenario, which are the *chairman* and the *participant*. Usually, there are several instances of the latter, while the chairman role has one instance only. The access control model for the audio/video stream is as follows. The chairman acts as a conductor of the meeting, which means that he usually sends audio/video data continuously. If there is a participant who wants to ask a question, the access to the audio/video stream is requested (in reality this would be done by raising a hand - in the distributed system case a floor is requested). It is the task of the chairman to decide, whether to grant the floor to the requesting participant or not. If access is granted, the participant is allowed to access the audio/video stream (i.e. send audio/video data) for a certain amount of time, representing a maximum length for the question. The participant may ask for a prolongation of the question time, which might be granted by the chairman or not. After the question time is over, the access floor is passed back to the chairman. It can be seen that the access model in GCDL consists of the specification of the access rights and the specification of the access control, defined as a social protocol.

The user behavior is fairly simple. Each participant instance generates floor asking requests based on *timed actions* which might be based on user profiling statistics, representing e.g. *mean inter-question intervals*. The probability to ask for prolongation as well as the probability to grant the prolongation by the chairman might also be set based on previously gathered statistics.

The pattern for the data traffic might be generated by using appropriate traffic generators for audio and video traffic based on e.g. Markov chains or trace files.

It can be seen from this simple example that the GCDL approach should enable modeling of arbitrary group communication scenarios both on structural and behavioral level. In fact, the presented example and further, more complex scenarios are feasible by GCDL and are used in the performance evaluation of SCCS.

In the next sections, the idea of GCDL is formalized by presenting the components of the model consisting of an object and process model as well as defining a description language to describe these scenarios formally.

4.3.2 Formal Description

In the last section, a motivation for a modeling technique was given based on roles in a distributed system applying rules on objects or groups of objects. In this section, a formal description of this idea is presented leading to the modeling framework GCDL. First, an overview of the different components within GCDL is given, before presenting each component in detail.

4.3.2.1 Components of GCDL

The GCDL model is divided into different components presented in figure 4-4 as an overview. It can be seen from this picture that GCDL provides components for structural as well as beha-

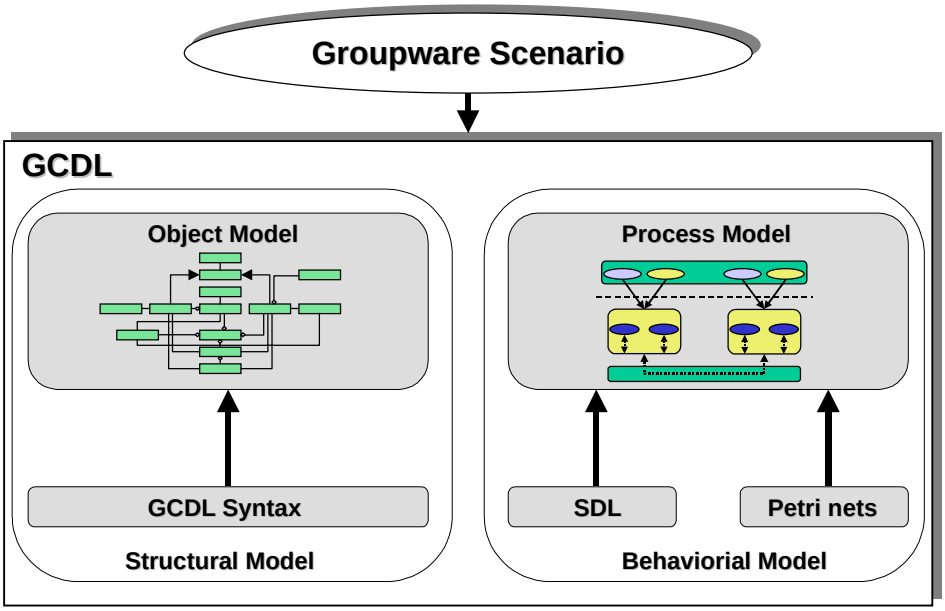


Figure 4-4: *Components of GCDL Modeling Framework*

vioral modeling of a considered groupware scenario. For the structural model of the scenario, an *object model* is provided enabling hierarchical description of the distributed objects, the roles accessing these objects, and the access rights to these objects. In chapter 4.3.2.2, this object model is presented in detail. For a formal description of the structural object model, the GCDL framework uses an *EBNF syntax* [ISO14977] defined in chapter 4.3.2.4.

Beside the structural model of the scenario, GCDL provides a behavioral model of the scenario. For that, a *process model* is proposed describing the instantiation of roles and the communication of these instances within the distributed environment. This process model is presented in chapter 4.3.2.3. The behavior description as such is not directly within the scope of the GCDL model. Only functional requirements for the behavior description are defined which are crucial to be fulfilled by the chosen description technique. As examples for such description techniques, SDL [Z100] and Petri nets approaches are presented in chapter 4.4.1 and chapter 4.5 respectively. While the first allows a simulative evaluation of the considered scenario, the latter is used to analytically evaluate a groupware scenario.

In the following, the presented components are described in more detail, starting with the object and process model of GCDL before introducing the EBNF syntax used in GCDL.

4.3.2.2 Object Model

The object model of GCDL is divided in the *object* and *roles* description (see figure 4-5). The former is built top-down from the scenario to the objects which exist in this scenario. The fine-granular description of the data is done by defining *objects* within the groupware environment describing e.g. audiovisual streams, documents, and so on. Objects may be grouped hierarchi-

cally to build complex *object groups* consisting of smaller ones to enable inheritance and structuring of objects. The uppermost level of that hierarchy forms the *scenario* in which the objects are embedded. *Access sets* are associated to each object containing a set of users within the conference which are allowed to read and write to the object. The access to the objects is abstracted by one or more *floors* representing certain actions concerning the associated object. The meaning of each floor and the realization of the read and write access to the object is not within the scope of GCDL. It is implicitly defined by the interactions of the appropriate *roles* (see below) representing the *rules* within the scenario.

The role description of the object model defines the different *roles* within the scenario. Each role is associated to one or more objects. Additionally for each role, the set of users is defined behaving according to this role. The behavior of each role is described separately for the *control* and *data* part to model the control traffic responsible for access control to the distributed objects and to model the data traffic patterns when the access is granted to a corresponding role. Both parts are realized in parallel processes. If the control part gets access to the associated object, the corresponding data part applies the data traffic patterns when accessing the object. The realization of the control and data description is not within the scope of GCDL. Only functional requirements are defined which have to be fulfilled by the realization.

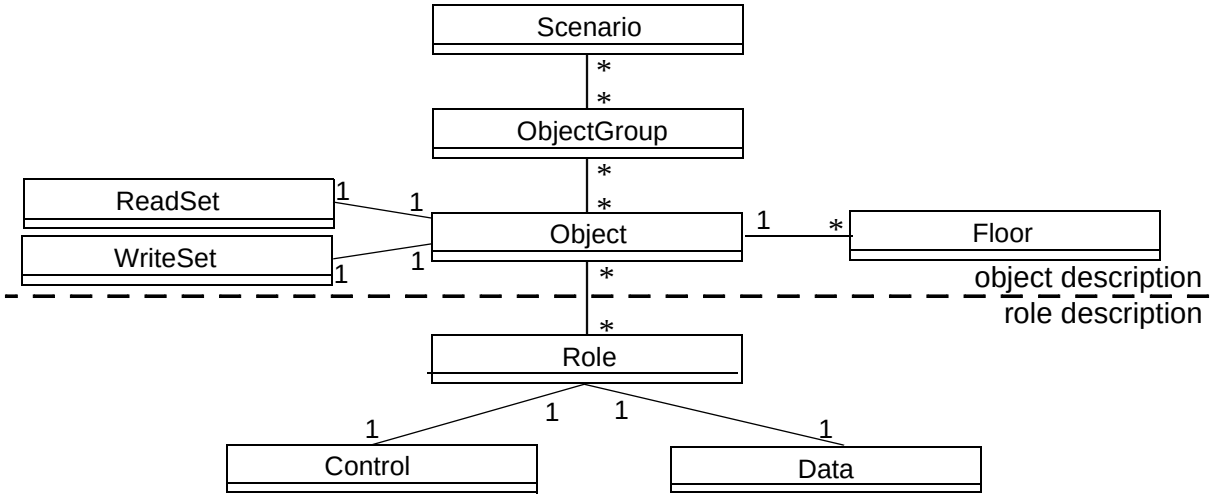


Figure 4-5: *Object Model of GCDL*

Thus, GCDL provides a structural model of a group communication system and its control and data entities in the system. This structural model is described using an own description syntax, presented in chapter 4.3.2.4.

The GCDL object model does not provide a detailed description of the control and data processes in form of automata or interfaced objects to be realized in a programming language. It is intended to be a structural framework which describes the objects and the corresponding communication entities in a collaborative environment. However, in chapter 4.4 and 4.5 a description of the communication entities with SDL ([Z100a][Z100b]) and Petri nets automata respectively is presented.

4.3.2.3 Process Model

The object model presented above gives a static description of the system consisting of objects (groups) and the appropriate roles operating on these objects. The process model which is pre-

sented in this section describes the dynamic behavior of the system and its interacting entities.

The dynamic description of the system is given in the different roles accessing objects. As stated in the object model, each role is associated to one or more objects and a set of users. For each associated user, an independent *instance* of the role exists. There may be different roles for each user in the system. This enables the multi-access of a user to different objects based on different roles. Each instance of the roles consists of two separate processes responsible for control and data traffic modeling.

The rules of accessing the object are implicitly defined by the interactions between the different instances of the roles associated to an object. Each instance of a role acts autonomously from other instances and coordinates the access to the object by exchanging *floor control* signals via the *environment* which represents the underlying communication system (e.g. a conference control protocol like SCCS). The data traffic is realized by the data process in each instance sending data traffic signals to the environment. Note that neither the realization of the environment nor the behavior description of the roles is within the scope of the GCDL framework.

Figure 4-6 shows the process model as described above. The static GCDL description of the

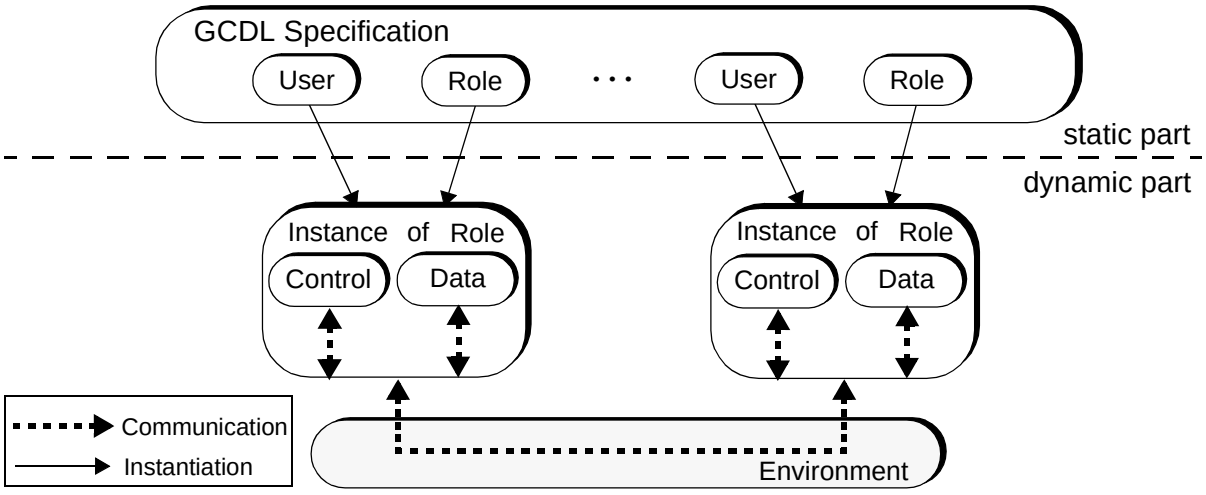


Figure 4-6: *Process Model of GCDL*

system is mapped uniquely onto instances of roles. These instances may exchange floor control signals via the environment to coordinate the access to the associated objects.

4.3.2.4 Syntax of GCDL

As stated in the component overview of chapter 4.3.2.1, the structural object model in GCDL is described by an own GCDL syntax. For that, the EBNF notation [ISO14977] is used. The syntax definition is enriched with comments for further explanation of the definition.

```
# EBNF syntax of GCDL
# simple data types:
# users are identified by integers
 <User>       ::=  <integer>;
# attribute of floor
 <attribute>  ::=  "exclusive" | "non-exclusive";
# floor to coordinate the object access (given as an identifier)
 <Floor>      ::=  F_<name> = <attribute> ; <integer>; ["..." <integer>;]
```

```
# instance activation time with start and end time in seconds
 <Time>       ::=  (<integer>;<integer>)
# instances are defined by the user and an activation time
 <Instances>  ::=  (<User>; [{ <User>; } | "..." <User>;] <Time>;)
# set of instances for a role
 <InstanceSet>::=  I_<name> = { <Instances>; }

# sets of simple data types:
# set of users with read rights
 <ReadSet>    ::=  RS_<name> = <User>; [{ <User>; } | "..." <User>;]
# set of users with write rights
 <WriteSet>   ::=  WS_<name> = <User>; [{ <User>; } | "..." <User>;]

# now the main parts of GCDL are specified: object and role
# an object consists of floor(s) to handle the access and
# sets of users being allowed to read and/or write to the object
 <Object>     ::=  O_<name> = <Floor>; { <Floor>; } <ReadSet>; <WriteSet>;
# an object may consist of other objects (object groups)
 <ObjectGroup>::=  OG_<name> = { <Object>; | <ObjectGroup>; }

# role is associated to one or more objects and described as
# an automaton, the instance set is defined
 <Role>       ::=  R_<name> = <Object>; { <Object>; } <InstanceSet>;
                   <Control>; <Data>;
```

The control and data processes for each role are only named in GCDL, the realization of the processes is not within the scope of GCDL.

It can be seen that a *logical* floor can be defined as a set of single floors by enumerating the possible values. The appropriate floor value of this logical floor is accessed by using the floor name followed by brackets, e.g. `F_AV_grant(1)` means to use the first floor value of the logical floor `F_AV_grant`. With this construct, floors might be grouped to logical units. In chapter 5.4.2, a panel discussion scenario is presented using this notation for grouping floor dedicated to a certain role in the scenario.

For the set of role instances, the following constraints hold. At least all writable users of the associated object(s) are included in the set of instances to enable the access competition of the different role instances. Furthermore, if the read access of associated objects is also controlled by floors, the readable users of the corresponding object(s) are also included in the instance set.

As mentioned in chapter 4.1.1, GCDL does not specify the description of the control and data process defining the behavior of the specific role. It just defines functional requirements for this realization:

1. Each instance of a role gets the information which is needed for correct instantiation of the role including the following:
   a. the activation time of this specific instance
   b. for each associated object the following status information:
      i. the possession status of each associated floor at start-up, e.g. the floor for the right to speak is by default located at the chairman of the conference
      ii. the access mode (read or write access)
2. To model user behavior in detail, the control and data process realization should provide *timed actions* to generate events based on statistical user profile data.
3. *External signals* which are sent to the environment are defined, but the realization is not within the scope of GCDL. The functional specification of the external signals is shown in table 4-1.

Signal	Description	Direction
FArq(floor)	request to ask for the *floor*	outgoing
FAin(floor)	indication of a request to ask for the *floor*	incoming
FPrq(floor, user)	request to pass the *floor* to a *user*	outgoing
FPin(floor, user)	indication of a floor pass request	incoming
FPrs(floor, user)	response to a floor pass request	outgoing
FPcf(floor)	confirmation of a floor pass request	incoming

Table 4-1: *Functional Specification of Floor Requests*

The data traffic signals which are sent to the environment are not within the scope of GCDL. In chapter 4.4 and 4.5, two behavior description techniques are presented to realize the control and data processes by fulfilling the functional requirements mentioned above.

4.3.3 Examples

In this section, the applicability of the description language GCDL for structural modeling is demonstrated by two examples. The control and data traffic is defined non-formally by describing the traffic without presenting an automata specification.

4.3.3.1 Scenario 1 - Shared Application

In the first example, a scenario is presented demonstrating two shared applications in a course unit, which might be a learning session. In this scenario, the first application shows the results of an experiment (controllable by all users after requesting access at the conductor), while the second one controls the experiment itself. Because only the conductor and experienced users are allowed to control the experiment, the second application is restricted with respect to the write access but is readable for all users.

Both applications are modeled as separate objects. There are three different roles (conductor, experienced user, normal user) acting on both objects. The number of users in the scenario is 10. For the sake of simplicity, the conductor has number 1, and there are 4 experienced users numbered 2 to 5. The duration of the lecture is one hour.

The GCDL specification of the scenario is given as:

```
# first the floors are defined for both applications
F_appl1_grant= exclusive; 1;
F_appl1_ask  = exclusive; 2;
F_appl2_grant= exclusive; 3;
F_appl2_ask  = exclusive; 4;
# then the instance sets are defined
I_cond       = (1; (0;3600))
I_exp        = (2;3;4;5; (0;3600);)
I_normal     = (6;7;8;9;10; (0;3600);)
# read and write sets are defined
RS_appl1     = 1;2;3;4;5;6;7;8;9;10;
WS_appl1     = 1;2;3;4;5;6;7;8;9;10;
RS_appl2     = 1;2;3;4;5;6;7;8;9;10;
WS_appl2     = 1;2;3;4;5;
# define the objects, one for each application
```

```
O_appl1      = F_appl1_grant; F_appl1_ask; RS_appl; WS_appl1;
O_appl2      = F_appl2_grant; F_appl2_ask; RS_appl2; WS_appl2;
# the objects may be grouped to a scenario
OG_scenario  = O_appl1; O_appl2;
# the roles are defined, conductor, experts, and normal users
R_cond       = O_appl1; O_appl2; I_cond; ctrl_cond; data_cond;
R_exp        = O_appl1; O_appl2; I_exp; ctrl_exp; data_exp;
R_norm       = O_appl1; O_appl2; I_norm; ctrl_norm; data_norm;
```

The processes *crtl_cond*, *ctrl_exp*, and *ctrl_normal* implement the rules for accessing the shared application objects while the processes *data_cond*, *data_exp,* and *data_normal* implement the data which is transmitted when access to the shared application is granted. The meaning of the floors is very simple. The floor *F_appl1_grant* (resp. *F_appl2_grant*) is used to grant write access to another user. The current holder of this floor is allowed to write to the object. With *F_appl1_ask* (resp. *F_appl2_ask*), a user asks for write access. This floor is owned by the conductor and is used to indicate the request only. The social protocol for both objects is the same, only the access rights differ. Therefore, the same roles can be used for both objects.

4.3.3.2 Scenario 2 - Video Conference with Whiteboard

The second example is an unconducted video conference with a whiteboard for demonstration purposes. The access model of this scenario is much simpler than the first one. There are two objects (audio/video stream and whiteboard). The access to the audiovisual stream is voice-controlled, i.e. the loudest speaker gets the access to the stream. The access to the whiteboard content is unconducted. The participant who wants to write to the whiteboard has to ask the current floor holder, who grants the floor after a certain waiting time. Again, there are 10 users in the system, but 5 of them enter the conference after 30 minutes. The duration of the conference is one hour. The GCDL description is as follows:

```
# two floors are needed only, one for each object
F_AV_grant   = exclusive; 1;
F_WB_grant   = exclusive; 2;
# there is one instance set only
I_all        =    (1;2;3;4;5;(0;3600););
                  (6;7;8;9;10;(1800;3600););
# everybody is allowed to read and write
RS_all       = 1;2;3;4;5;6;7;8;9;10;
WS_all       = 1;2;3;4;5;6;7;8;9;10;
# there are two objects, AV stream and the shared whiteboard
O_AV         = F_AV_grant; RS_all; WS_all;
O_WB         = F_WB_grant; RS_all; WS_all;
# the objects may be grouped to a scenario
OG_scenario  = O_AV; O_WB;
# now the roles are defined
R_AV         = O_AV; I_all; ctrl_AV; data_AV;
R_WB         = O_WB; I_all; ctrl_WB; data_WB;
```

The roles R_AV and R_WB are different in the process realization on control and data level due to the differences in accessing the objects and in generating data for these objects.

4.3.4 Load Modeling with GCDL

As explained in the previous sections, GCDL allows to model groupware application structurally by defining a scenario containing objects to which roles are associated. Beside the structural model aspect, a main purpose for introducing GCDL is to use it for load modeling to enable the evaluation of main mechanisms of SCCS. This section shows how GCDL can be used for *workload modeling*.

In the structural GCDL description of a collaborative system, a role which is associated to objects consists of a control and data part to enable the modeling of control and data traffic in the system. As explained in the process model, the control and data processes communicate by sending *signals* to the *environment*. The latter represents the underlying communication environment (e.g. SCCS), while the signals represent the exchanged data in the system which are sent among the processes internally, but also among the users in the system. It is the task of the environment to route signals from a role instance A to another role instance B in the system.

As a consequence, the control and data processes of each role instance in the system generate a defined workload to the environment representing the group communication system. Thus, the GCDL approach can be used as a *workload generator* for the environment according to given rules and roles which are defined by the role instances of GCDL. Because the workload is generated based on a social protocol description of the group communication system, the resulting workload is much more realistic compared to e.g. randomly generated, independent requests. Furthermore, the workload is generated for the control and data part of the system due to the description of *control* and *data* entities. Thus, the evaluation of both parts of the communication system, the control part (e.g. SCCS) and the data part (e.g. an arbitrary multicast transport service), is feasible. For the workload generation of the data part, appropriate *data traffic patterns* are applied, e.g. based on Markov models or trace-driven descriptions [Rose97].

In ([TrKa98][Tro99b]), an *automata-based load model* was proposed by the author which applies the idea of GCDL to evaluate the resource management scheme of SCCS, i.e. the control part of a collaborative system. In this approach, the detailed structural GCDL description is not used. But in the GCDL context, this load model implements the control process as finite-state automata sending floor control signals to the environment which is the unknown system to be evaluated. The data process is not implemented due to the fact that SCCS is a control protocol and not a data transfer protocol. The finite-state automata implement the rules of the *social protocol* of different scenarios generating workload for the system which is typical for the corresponding scenario. By adding *monitoring points* to the runtime system, different performance measures can be obtained. The runtime system may be a real environment (e.g. an SCCS implementation) or an environment simulating the group communication infrastructure.

Using finite-state automata for the realization of the control and data processes is very similar to specifying a communication protocol, which enables verification and testing of the protocol. For that, specification languages like SDL ([Z100a][Z100b]), Estelle, or LOTOS [ISO8807] are used. The protocol is specified as autonomous automata communicating via an environment representing the ’known’ world.

In GCDL, the autonomous automata realize the control and data processes of the autonomous role instances of each associated object. Compared to specifying and testing communication protocols, the focus in modeling group communication for workload generation is different. In the protocol specification area, the protocol is tested and validated by generating test cases (delivering the workload of the system) and running simulations in an environment (representing the ’known’ world).

When applying the GCDL modeling approach for evaluation of group communication systems, the environment is the part of the system which is actually evaluated. Thus, the autonomous automata generate the workload for evaluating the environment.

In chapter 4.4, the automata-based load model, presented in ([TrKa98][Tro99b]), is generalized using the proposed GCDL approach and realizing the behavior description (given in the control and data processes) with the specification language SDL. Additionally, a simulation framework is presented to map GCDL scenarios to an event-based simulation which is used to evaluate

main mechanisms of SCCS in chapter 5 and chapter 6. In this way, the GCDL description framework enables the workload generation and modeling of the control and data traffic in a group communication system.

4.3.5 Summary

In this section, a framework was proposed to model groupware systems. The intention of the approach was to structurally model the objects within a group communication environment and to enable performance evaluation of the system without implementing it. Thus, the approach is intended for the *design phase* of a groupware application. It does not cover software model checking aspects of realized, i.e. implemented, systems. The proposed *Group Communication Description Language* (GCDL) allows to model the objects within a group communication environment on a hierarchically structured level. Another goal of the development of GCDL was to enable the communication and behavior modeling of groupware applications. By introducing *roles* and *rules*, GCDL enables to model the control and data traffic as well as user behavior within collaborative environments. A role describes the interaction among other roles associated to the corresponding object via the environment to realize the access model for this object and to apply the data traffic patterns when accessing the object. A role is divided in two parallel processes responsible for the control and data behavior of the specific role. The realization of these processes is not within the scope of GCDL. However, functional requirements were defined which have to be fulfilled by the chosen implementation. Two examples demonstrated how the GCDL description together with a given realization of the processes can be used as a load model for the evaluation of group communication systems. In the following two sections, approaches for the behavior description within GCDL are presented.

4.4 Realization with Event-based Simulation

The Group Communication Description Language introduced a framework to model group communication applications. The realization of the role processes and the specification of the *social protocol* defining the rules is outside the scope of the GCDL framework. As it was mentioned in chapter 4.3.2.1, the behavior of the communicating entities in the GCDL approach is defined in the control and data process of the specific role. The behavior description of the control and data processes may be realized using different approaches.

Two examples of such approaches are presented in this and the following section. Both use *finite-state automata* to describe the role behavior in the system. There exist many approaches to define finite-state automata in distributed systems. Formal description techniques like SDL ([Z100a][Z100b]) or LOTOS [ISO8807] might be used to describe the automata. In this section, SDL is used for a *simulative* evaluation of the groupware system. Other techniques might be used describing the messages in the system as sequences of transactions ([AHP96][Z120]), which are then mapped to finite-state automata. In fact, the Petri net approach, presented in chapter 4.5, is another approach to describe behavior of a communicating entity. This modeling technique is used for an *analytical* evaluation of the system.

As mentioned in chapter 4.3.4, the GCDL modeling approach might be used for simulation workload generation to evaluate basic mechanisms of group communication environments. The idea for realizing the GCDL modeling approach with event-based simulation is to describe the control and data processes within GCDL as SDL finite-state automata in a first step. In a second step, these automata are mapped to an event-based simulation for which a simulation

framework is proposed to accelerate the development of simulation environments for the performance evaluation of group communication systems.

4.4.1 Using SDL for Process Description

The *Specification and Description Language* (SDL [Z100b]) is used to specify communication protocols. For that, an *SDL system* is defined consisting of concurrent *SDL processes* exchanging *SDL signals* with other processes using *SDL channels* or *SDL channel routes*. Different SDL systems may communicate by exchanging signals using the *SDL environment* which represents the known world. For further details concerning the SDL specification, see [Z100b].

In this section, the GCDL modeling approach is realized using SDL. For that, the process model of GCDL has to be mapped onto an SDL system, which is shown in figure 4-7. Each

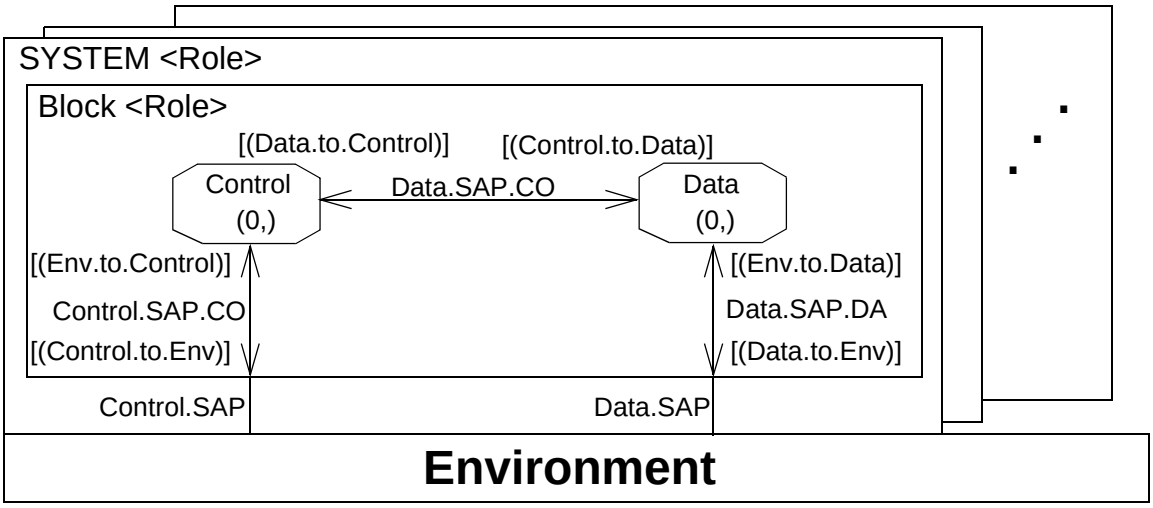


Figure 4-7: *Realizing GCDL with SDL*

role is mapped onto an autonomous SDL system which then communicates via the SDL environment. For that, the SDL channel *Control.SAP*, containing floor control signals, and the channel *Data.SAP* for the data process specific signals are used. Each SDL system consists of exactly one SDL block for the role definition. This block is divided in two SDL processes, one for the control process and one for the data process, communicating by a implementation-dependent channel *Data.SAP.CO* between both processes. These processes are instantiated according to the instance set information in GCDL (see chapter 4.3.2.1). The term <Role> in figure 4-7 represents for the name of the role defined in the GCDL specification.

In addition to the short graphical representation of the SDL system, a more detailed textural representation of the SDL system is given in appendix A of the thesis. This textural representation gives a more detailed overview of the used variables and data types implementing the functional requirements of chapter 4.3.2.4.

As stated in chapter 4.3.1, the approach of using SDL for behavior description in group communication environments is very similar to the specification of communication protocols in general to enable testing and verifying of the social protocol. However, the simulation of group communication scenarios is also one of the goals using GCDL and its realization with finite-state automata. For the simulation of the GCDL system described with SDL, approaches like

presented in ([CTK99][LaKo99]) might be used to translate the SDL specification into a C++ program to run a simulation with the resulting system. Unfortunately, most of these automatic translation tools are restricted to certain SDL constructs only and/or the resulting code is hardly extendable, e.g. it is difficult to add monitoring points for gathering performance measures. As a consequence, the specific mapping of the SDL description used here onto an event-based simulation is presented in the next section.

4.4.2 Mapping to Event-based Simulation

The second step in the event-based simulation realization is to map the SDL behavior description to a runtime system which can be evaluated. Mapping an SDL system to an event-based environment is fairly straight-forward (see figure 4-8 and [SpHo95]):

- each state of an automaton instance is realized by an internal *state* variable in the corresponding *PControl*, *PData*, or *Environment* instance
- both, internal and external, signals are mapped to events to which a defined *payload* is associated containing description data for this signal (like floor numbers, unique automaton address)
- newly generated events are inserted in an event queue (*push_event()*) which is sorted by the

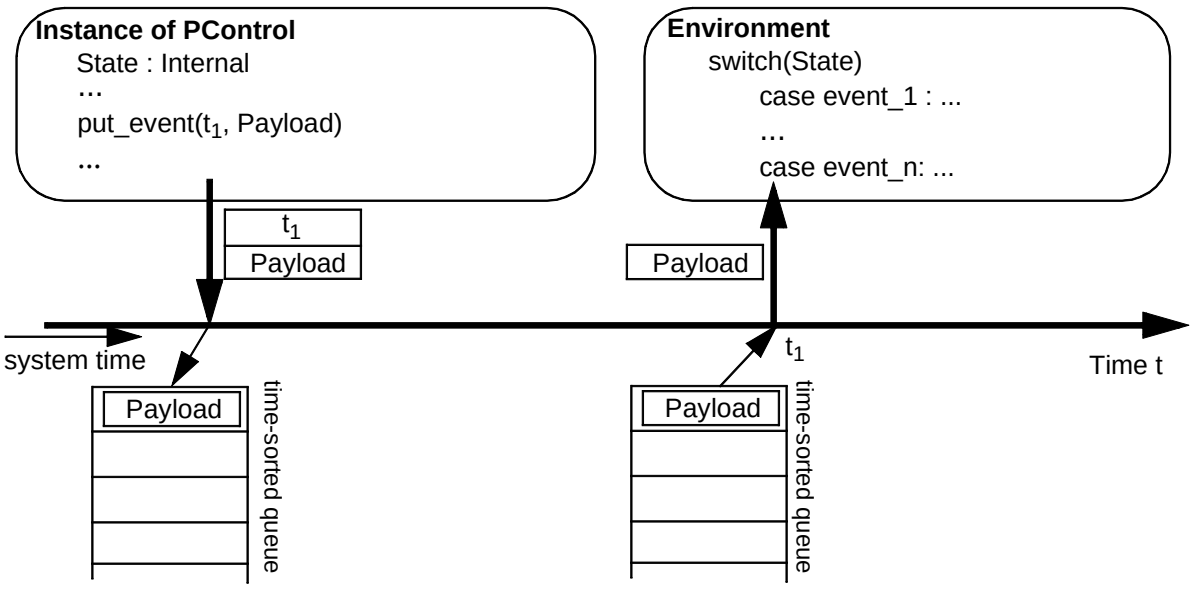


Figure 4-8: *Event-based Simulation - Event Propagation*

event execution time

- a *system time* is handled by the runtime system representing the current simulation time
- the next event (relative to the system time) is removed from the list and dispatched to the corresponding instance of process
- depending on the current state of the automaton instance, the events are handled by the appropriate process instances, e.g. by using switch-case constructs or dispatching tables

If the SDL environment representing the used communication system is also simulated, an external signal is used as an input event for this system. Otherwise, external signals are transformed to input signals for a real-life system by a *proxy environment*.

It can be seen that the mapping of the SDL description to an event-based simulation is very simple. Only simple event handling functionality is needed which is provided e.g. by systems like ATLAS [Da97]. In the next section, a simulation framework is presented providing simulative evaluation of systems which are described by GCDL.

4.4.3 Simulation Framework

The main purpose of the realization of the control and data processes with SDL is to enable the simulative evaluation of groupware scenarios. To simplify and accelerate the implementation of this evaluation, a simulation framework is crucial. This framework should enable a scenario-based performance evaluation together with an easy *plug-and-run* design. As a consequence, an object-oriented design is chosen. A previous version was published in [Tro99c], which is extended by the control and data model used in GCDL. In the next two sections, the object and activity model of that framework are presented.

4.4.3.1 Object Model of the Framework

The object model shown in figure 4-9 is specified using the *Unified Modeling Language* (UML [JEJ95]). The model is divided in a *scenario-independent* and *scenario-dependent* part.

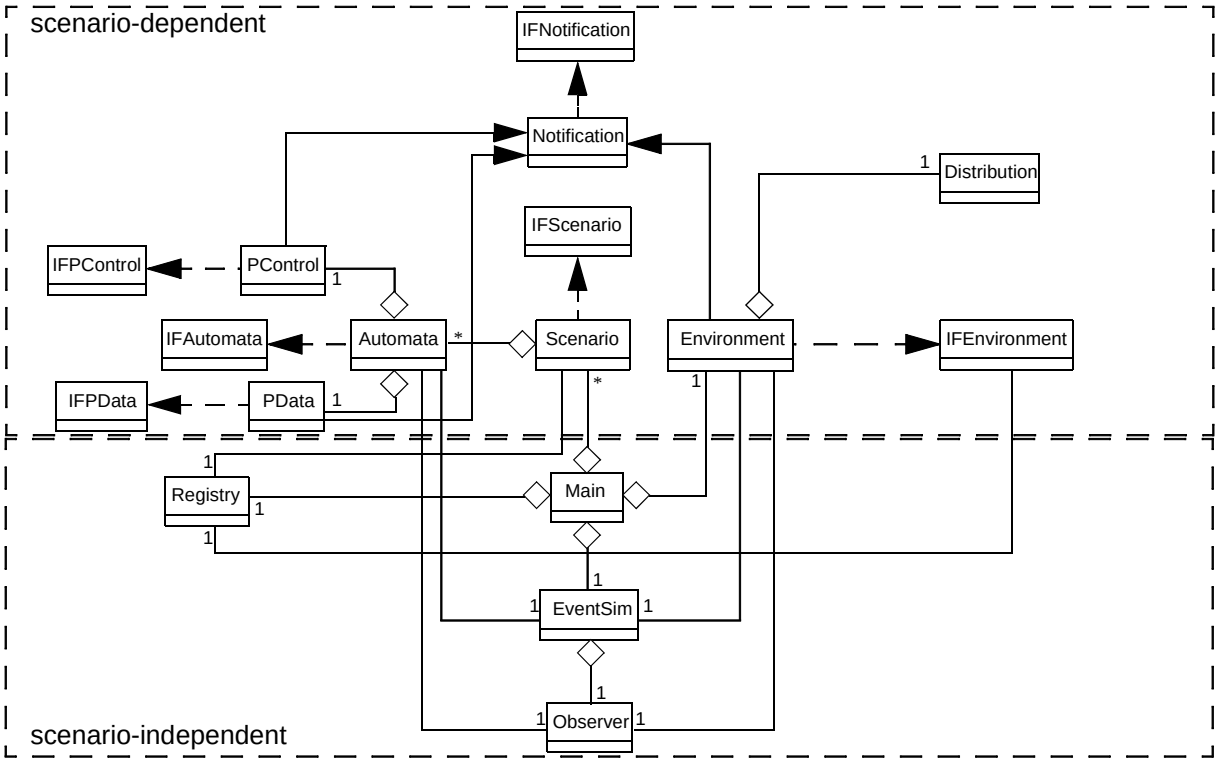


Figure 4-9: *Object Model of Simulation Framework*

The core of the scenario-independent part of the framework is the *Main* object which initializes the different *Scenario* objects according to an input file (or a graphical user interface) and starts the runtime system. For that, an object *EventSim* implements the event-based simulation techniques, providing methods for callback registration, event scheduling, and event insertion in the system. Additionally, an *Observer* object is provided to which observed values are registered.

An identifier is returned which is used to identify values pushed to that observer. For each value, a small window is created showing the history of that value. At last, a *Registry* object is provided for distributed simulation which is explained below.

The scenario-dependent part of the framework is defined by interfaces for notifications, environment, automata, scenarios, and processes (the interfaces are named with *IF* as a prefix). The interfaces are implemented by corresponding objects (denoted with the *realize* relation in the UML diagram) using the provided event-based simulation functionality.

Different *Automata* objects are grouped into *Scenario* objects implementing the different roles within a scenario. Thus, the instances of *Automata* objects represent the corresponding user instances in the GCDL framework. Each Automata object has a one-to-one relation to certain *PControl* and *PData* objects for the realization of the automata processes which realize the rules within the scenario. All these objects provide an initialization method for start-up like notification or observation value registration. The creation of certain *Scenario* objects depends on the initialization by the *Main* object. Thus, a selection of scenarios (with given parameters for that scenario) is feasible.

The *Environment* object provides an interface with a *route()* method for transferring external signals between given automata. The environment itself may be a simulation for the system to be evaluated, thus using the event callback mechanism, or it may be a proxy implementation to an existing system. This proxy is responsible for dispatching external signals to/from the existing system from/to the automata.

Due to the fact that the automata are autonomous, the framework may be extended to distributed simulations. For that, the *Registry* object is used which registers the local automata objects. The interface provides a *query()* method to ask for local or remote automata objects. The *Distribution* object is used to transfer an external signal from the local environment to a remote one using UDP with acknowledgements. There is no coordination needed for local events, because this is done implicitly by the coordination of the autonomous automata which is integrated in the definition of the automata. Thus, the framework provides an easy extension to distributed automata-based performance evaluation. However, the interactions between the autonomous automata are normally highly correlated. As a consequence, it is expected that the performance gain which can be achieved by distributing the simulation is not very high. But due to the implicit coordination of the simulation defined by the *rules* of the scenario, this distribution can easily be integrated in the framework without adding high synchronization overhead.

4.4.3.2 Activity Model of the Framework

In the following section, the activity model of the simulation framework is presented divided in three parts, the initialization of the simulation, the event loop, and the distribution of a simulation. *Activity diagrams* [JEJ95] for each of these parts are presented together with a detailed description of the used objects and their functionality.

Initialization

The simulation framework is initialized by calling the *init()* method of the Main object. During the first step, the *EventSim* and *Observer* object are created to enable the event handling and the monitoring functionality. After that, input files and graphical user input values may be parsed to set up defined simulation parameters, like the number of participants or the monitored values. In the second step, the scenario-dependent objects are created, namely the *Environment* and *Scenario* object. The *Environment* object registers event notification callbacks at the *Event-*

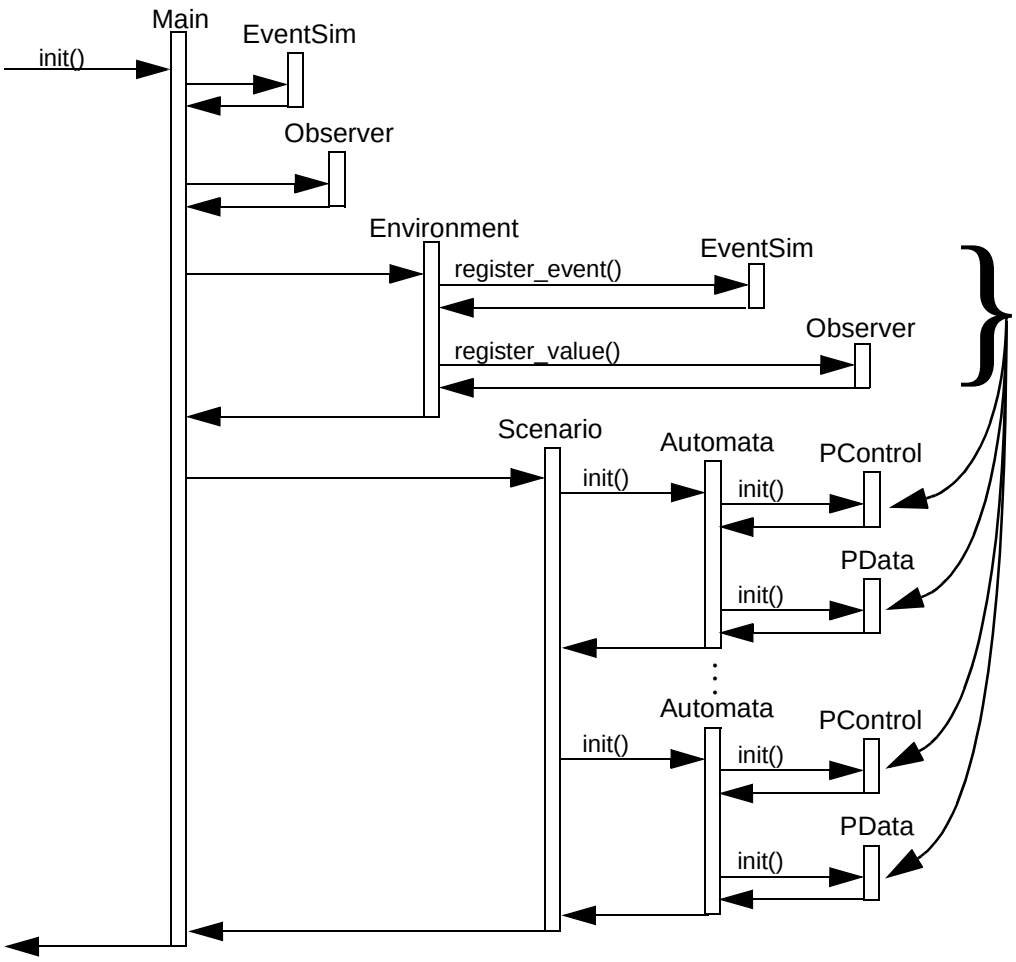


Figure 4-10: *Simulation Framework Activity Diagram - Initialization*

Sim object calling its *register_event()* method. Furthermore, monitoring values may be registered at the *Observer* object by invoking the *register_value()* object. The *Scenario* object creates and initializes instances of *Automata* objects representing the role instances of the scenario. The initialization of these instances may be based on values read from the input file or given by a user interface during the *Main* object initialization.

Each *Automata* object creates and initializes a *PControl* and *PData* object representing the control and data processes within the GCDL approach. Each instance of the process objects (*PControl* and *PData*) registers event notifications and monitoring values similar to the *Environment* initialization. As a consequence, the event *Notification* objects are instantiated within the *Environment* instance as well as in the *PControl* and *PData* instances.

Event Loop

The event loop of the framework is the core of the simulation and keeps it running. It it activated by calling the *run()* method of the *Main* object. In a loop (terminated after the pre-defined duration of the simulation or by a simulation error), the *schedule()* method of the *EventSim* object is invoked to dispatch the next event in the queue. The associated notification for this event is called by invoking the *event_callback()* method of the registered object. During handling the event in the *Notification* object (which is associated either to an *Environment*, *PControl*, or *PData* object), new events may be generated by calling the *put_event()* method of the *EventSim* object. Furthermore, monitoring values may be pushed to the *Observer* object using

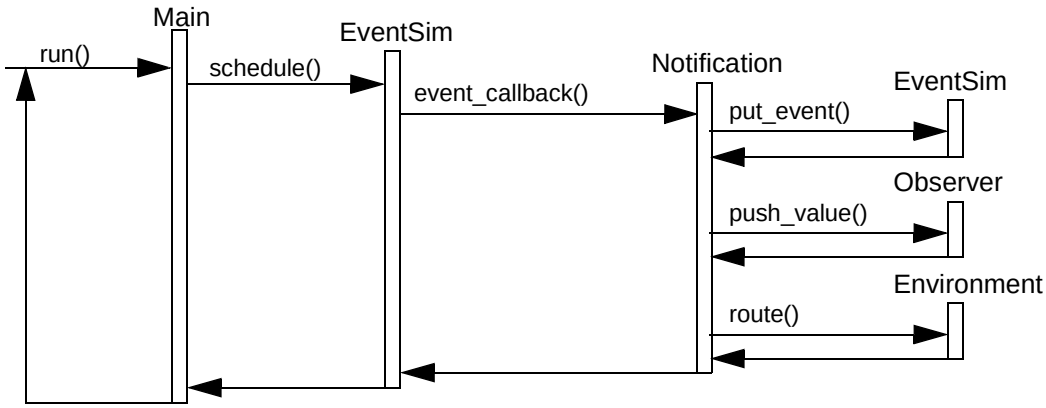


Figure 4-11: *Simulation Framework Activity Diagram - Event Loop*

the *push_value()* method. At last, external signals may be sent to other automata instances using the *route()* method of the *Environment* object. By generating new events within the Notification objects, the simulation is kept running until a special event is dispatched signaling the end of the simulation. This event is set during the initialization phase in the *Main* object.

Distribution

As mentioned in the object model of chapter 4.4.3.1, the simulation framework can easily be extended to distributed simulation due to the inherent independence of the different automata (the different automata acts autonomously from each other according to the defined rules).

However, a fundamental requirement for the distributed simulation is that the environment is able to be distributed. This might be the case if the environment is not simulated but a real-life environment (e.g. an existing communication protocol). In that case, a proxy environment in each simulation host implements the *route()* method to send the appropriate external signals to the real environment.

In the case of a simulated environment, the distribution of the environment might be much more complicated, leading e.g. to synchronization problems. Nevertheless, the distribution of a simulated SCCS environment is feasible, because each node in this protocol administrates its own state, and only messages (PDUs) have to be exchanged. Thus, the simulated SCCS environment, which only serves as a message channel, could be distributed among different hosts being synchronized by the exchange of PDUs.

The open issue to realize the distribution of the different automata objects is presented in the following. The initialization and the event loop have to be extended to provide that functionality. When initializing the local *Environment* object, it registers with the local *Registry* object by calling its *register()* method. The registration information (unique simulation-wide identifier plus location information) is sent via the network to the *Global Manager*. This centralized object is chosen at start-up of the conference and contains a mapping of the unique identifier and location information. Additionally, each local *PControl* object registers with the local *Registry* object similar to the *Environment* object. When all object instances have been registered with the Global Manager, this host notifies the other hosts about the end of initialization and distributes the content of the global database to all other hosts to minimize future read requests.

In the event loop, a local *Environment* object queries for the location information of a certain process instance when the *route()* method of the *Environment* object is called. The unique identifier of the object is given as a parameter of the *route()* method and is used in the *query()*

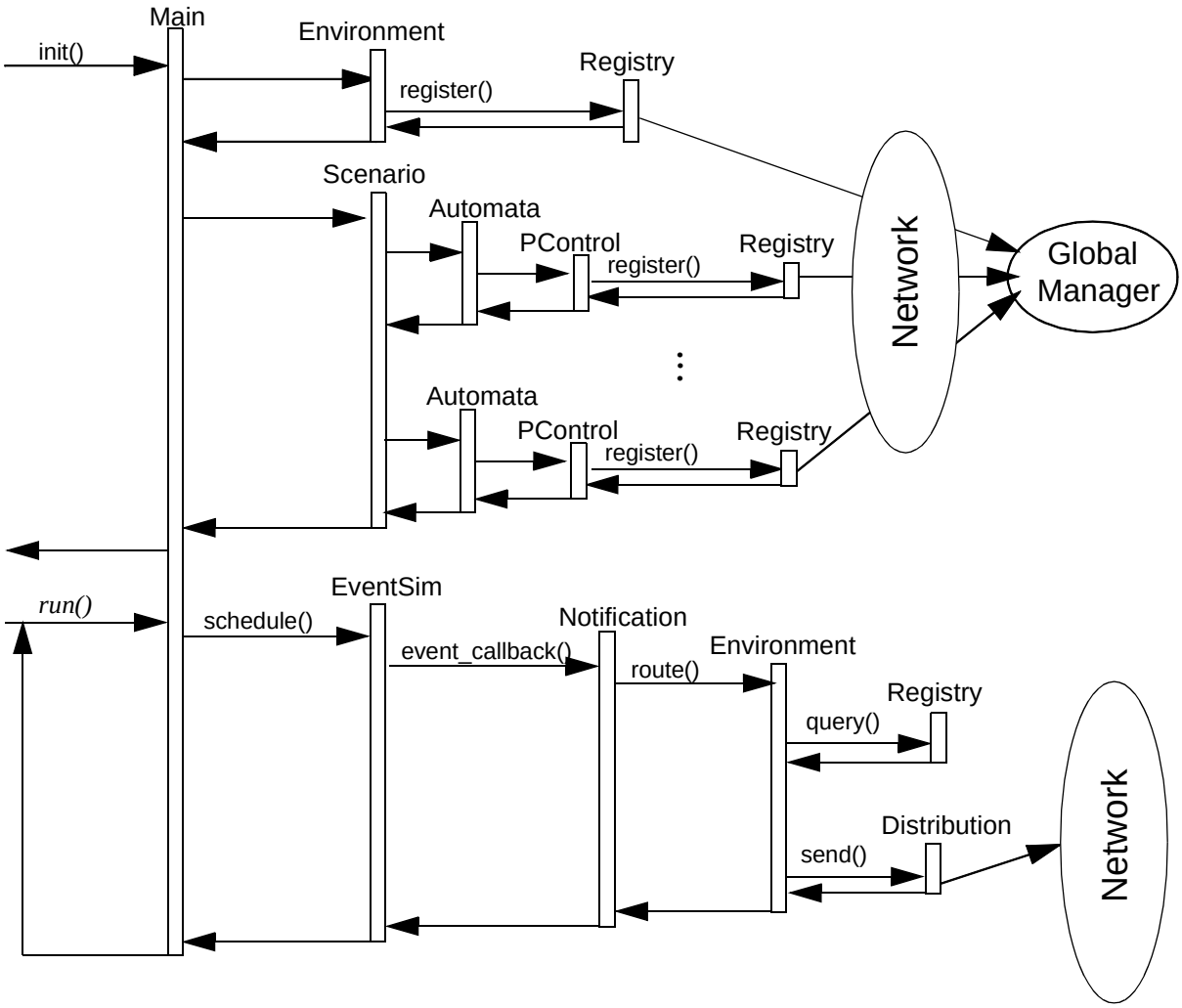


Figure 4-12: *Simulation Framework Activity Diagram - Distribution*

method of the local *Registry* object. After resolving the location, the *Environment* object uses the *send()* method of the *Distribution* object to send the external signal to the appropriate local *Environment* object which, as a result, routes the signal locally to the corresponding local process instance.

It can be seen that the simulation framework easily enables distributed simulation due to the inherent autonomy of the automata instances. However, the distribution of the environment is a task which has to be considered by the programmer and its difficulty depends on the given problem.

4.4.3.3 Group Communication Simulation Tool

The proposed simulation framework for the scenario-based evaluation was realized for the purpose of evaluating main SCCS mechanisms, i.e. the control part of a collaborative system. This section describes the main features of the resulting program, called *Group Communication Simulation Tool* (GCST). The results of the simulative evaluation in the chapters 5 and 6 will be obtained with GCST. For the realization of the program, a non-distributed version of the framework is used, thus the *Registry* and *Distribution* objects of the framework are not realized. Furthermore, the data part of the communicating entities is not realized, because only the con-

trol part of the communication system is evaluated.

The main design goals for GCST are

- *extensibility*: This is provided by following the simulation framework which allows an easy extension of the program by adding new scenarios and automata objects.
- *ease of use*: GCST provides a graphical user interface which allows an easy operation of the simulation.
- *fast*: the underlying event handling functionality is kept simple to provide high performance.

GCST provides five different scenarios:

- *large meeting*: consists of a chairman and many participants within a conducted conference
- *panel discussion*: consists of a chairman, several experts, and many participants within a conducted conference
- *expert round*: consists of several experts and many participants within a conference
- *private channels*: similar to panel discussion, but some participants open private channels for side communication (*sub-sessions*)
- *generic token*: worst case scenario for the dynamic reconfiguration simulating a uniformly distributed usage of several tokens in a conference

A more detailed description of the scenarios can be found in the evaluation chapters 5 and 6.

The environment in this context represents the underlying conferencing protocol (SCCS) and its main mechanisms like resource management and dynamic reconfiguration. Hence, the environment is simulated as well. As a consequence, the *Environment* object of the simulation framework is realized and provides the functionality of routing floor control requests based on the centralized scheme of the T.120 standard as well as using the proposed resource management scheme of SCCS (see chapter 3.2.13). Furthermore, the environment implementation provides the dynamic reconfiguration functionality of SCCS.

Therefore, five *Scenario* and the corresponding *Automata* and *Process* objects as well as the *Environment* object were implemented using the simulation framework of chapter 4.4.3. This results in a graphical user interface based simulation program (see figure 4-13), which can be easily extended by other scenarios for evaluation of the environment.

GCST provides a context-sensitive help to explain the different scenarios, their input and output files, and the topology description. The topology is displayed as a tree in the main window, and several scenario-dependent statistics can be displayed when double-clicking on node or user elements in the window. Several on-line displays (realized as *Observer* objects) may be shown during runtime for visualization of floor control response time for observed users in the scenario. Furthermore, at the end of the simulation, several statistics windows are displayed on the screen. Up to now, all performance measures of chapter 3.5 for the evaluation of the main SCCS mechanisms are provided by the current GCST implementation. Due to the simulative approach, it is very easy to integrate further performance measures in the program. For that, the corresponding performance measures have to be registered with the *Observer* object and, at a defined monitoring point, the performance values are pushed to the *Observer*. Thus, the simulation framework covers a wide spectrum of performance measures and allows an easy extension of this spectrum.

The software was implemented under Windows NT/9x using Visual C++5.0 and is available for free download under [Tro00b] with a graphical setup program.

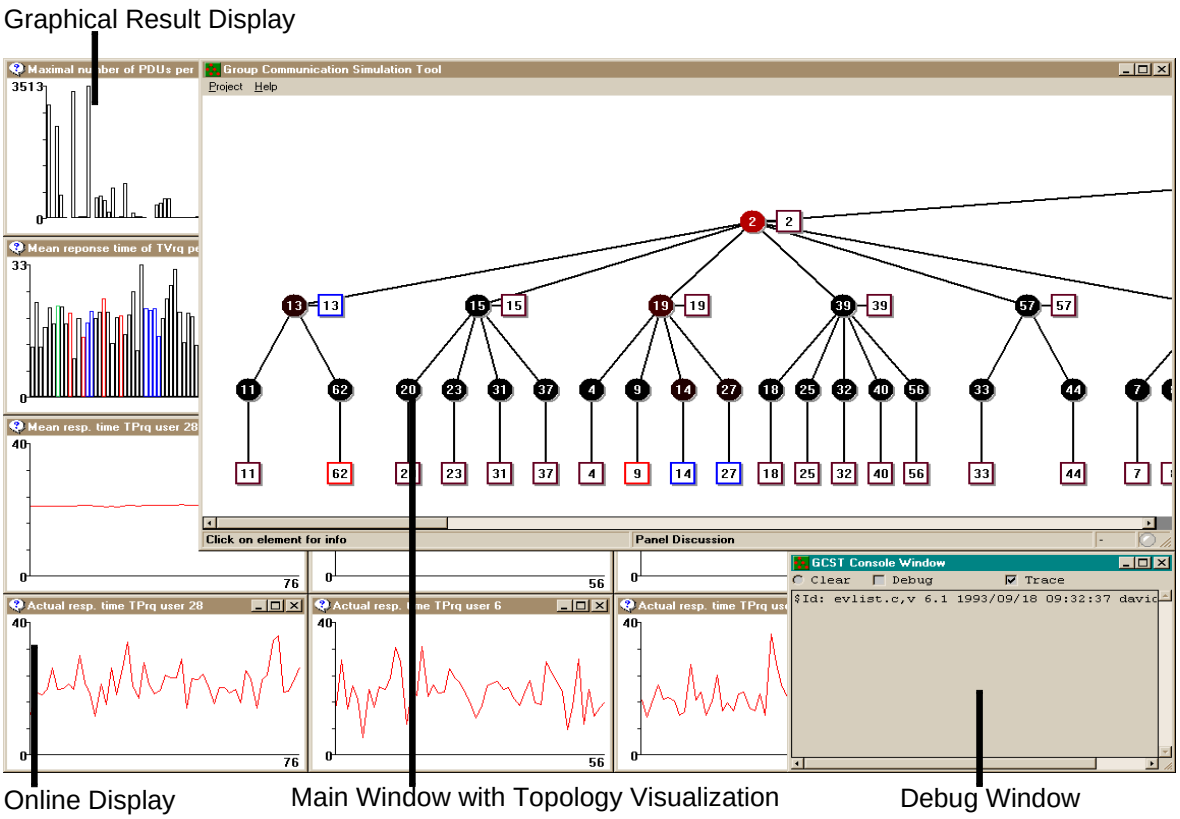


Figure 4-13: *Group Communication Simulation Tool (GCST)*

4.4.4 Summary

This section presented an approach to realize the behavior description part within the GCDL framework. For that, the specification and description language SDL was used which provides a wide range of constructs to describe finite-state automata. Therefore, it allows to sufficiently define the role and rules with GCDL describing the social protocol and the user behavior within a given scenario. A framework was presented to realize the GCDL system in SDL by defining an appropriate SDL system for each role within the scenario communicating with other roles (SDL systems) via the SDL environment using defined SDL signals. Furthermore, the mapping of the SDL system to an event-based simulation was shown, and a simulation framework was designed which facilitates the development and implementation of event-based scenario-dependent simulations in the context of GCDL. The object as well as the activity model of the framework were described. Additionally, an example implementation which is used for the performance evaluation of SCCS was outlined.

4.5 Realization with Petri Nets

In the proposed modeling framework of chapter 4.3, stochastic Petri nets can be used for the behavior description of the communicating entities. Similar to the approach given in chapter 4.4, the behavior of the control and data process is defined by Petri nets as a finite-state automata. In contrast to the SDL description, the stochastic Petri net realization enables both the analytical as well as the simulative evaluation of the resulting system. In the following section,

the focus is concentrated on the analytical evaluation of groupware systems.

The section starts with a brief introduction concerning Petri nets and the measures which can be analytically obtained. In the second part, the realization of the behavior description is presented using the Petri nets modeling technique.

4.5.1 Petri Net Introduction

In this section, basic definitions concerning stochastic Petri nets are presented for a better understanding of the further sections.

Stochastic Petri nets (Petri nets or SPNs in the following) provide a graph-based definition of a finite-state machine. It will be seen that under certain assumptions, SPNs can be mapped to an underlying *continuous time Markov chain* (CTMC) with finite state space. This resulting Markov chain can be analyzed numerically to obtain specific performance measures of the system.

SPNs are directed bipartite graphs with *places P* and *transitions T* and a subset of edges called *input arcs* (from places to transitions) and a set of *output arcs* (from transitions to places). Places are drawn as circles, transitions as bars, while input and output arcs are depicted as arrows. Transitions may be *immediate* or *timed*. The former do not consume any time for operation, while the latter require a negative exponentially distributed time for *firing*. Immediate transitions have higher priority than timed transitions. To distinguish between both types, immediate transitions are typically drawn as filled bars, while timed transitions are depicted as outlined bars. Places may contain one or more *tokens* drawn as small black circles within the places. A certain distribution of tokens is called *marking*.

In the following, some formal definitions are presented before outlining some general properties of SPNs. For further details about SPNs, see ([BaKr96][BRR87][Haver98b]).

Definition 4.1: Stochastic Petri nets
(P, T, I, O, H, W, m_0) is called stochastic Petri net (SPN) with
$P=\{P_1, P_2, ..., P_n\}$ a set of ***places***
$T=\{T_1, T_2, ..., T_m\}$ a set of ***transitions***
$I: P \times T \rightarrow N$ a function which assigns a multiplicity to each input arc
$O: T \times P \rightarrow N$ a function which assigns a multiplicity to each output arc
$H: P \times T \rightarrow N$ a function which assigns a multiplicity to each inhibitor arc
with $\aleph$ the set of numbers 1,2,3,...
$W: T \rightarrow R^+$ a function which assigns to a transition t either a ***firing rate*** *if the transition is timed or a* ***weight*** *if the transition is immediate*
$m_0 \in N^n$ the ***initial marking****, i.e. the initial number of tokens in place $P_1,...,P_n$*

Definition 4.2: Input/Output Places
$I_p(t) = \{p \in P; I(p,t) > 0\}$ is defined as the set of ***input places*** *of transition t and*
$O_p(t) = \{p \in P; O(p,t) > 0\}$ is defined as the set of ***output places*** *of transition t.*

Based on these definitions, a brief introduction to the dynamic SPN properties can be given [Haver98b], often referred to as *firing rules*. A transition *t* is called *enabled in a marking m*, when all input places $I_p(t)$ contain at least $I(p,t)$ token. When there are immediate transitions enabled in marking *m*, the following rules are applied:

- Let *W* be the sum of weights of all enabled immediate transitions.

- Enabled transition *t* fires with probability *W(t)/W*.
- Upon firing, transition *t* removes *I(p,t)* tokens from each input place *p* and adds *O(t,p)* tokens to each output place *p*. The result of the fired transition is a new marking *m'*.

A marking *m* leading to another marking *m'* under transition *t* is written as $m \xrightarrow{t} m'$, i.e. firing transition *t* leads from marking *m* to *m'*. This fire operation is an *atomic* action, i.e. it either takes place completely or not at all.

When there are no immediate transitions enabled, the following rules hold:

- An enabled timed transition *t* fires with rate *W(t)*, which is defined as an exponentially distributed time.
- When firing, remove and add operations are performed similar to the immediate case.

When there is more than one timed transition *t* enabled in a particular marking, one of these transitions is selected randomly based on the firing rates of the transitions (see [Haver98b]).

For simplification of the SPN specification, an *enabling function* $E: T \times M \rightarrow \{0, 1\}$ may be used. This function specifies for each transition *t* in marking *m* whether it is enabled or not. This means that after applying the normal firing rules for a transition *t*, the enabling function is evaluated. If this function is 1, the transition is said to be enabled.

Definition 4.3: m' reachable from m
A marking m' is defined as ***reachable*** *from m if there is a sequence of transitions* $t_1,...,t_n$ *such that*

$m \xrightarrow{t_1} m_1 \ldots m_{n-1} \xrightarrow{t_n} m'$ *with* $n \geq 1$

Definition 4.4: Reachability Set - Reachability Graph
For a given initial marking m_0*, the set of possible markings that are reachable from* m_0 *is called* ***reachability set*** $R(m_0)$*. The tuple* $(R(m_0), E)$ *is called* ***reachability graph*** *with the markings of the reachability set as nodes in the graph, and there is an edge between markings m and m' if m' is reachable from m.*

4.5.2 Performance Evaluation by SPNs

As already mentioned, SPNs are mainly used for analytical performance evaluation of a system. Simulative evaluation of a system is also feasible, but is not used in this thesis. In this section, a brief introduction is given what measures can be obtained from Petri Nets in general before presenting these measures with respect to the performance measures in chapter 3.5.

It is assumed in the following that the reachability set of the SPN is finite, i.e. $|R(m_0)| < \infty$. This means that the number of tokens in each place is finite. This restriction easily holds considering the fact, that buffer space to store token requests is finite in group communication, too.

Measures to be obtained from SPNs

There are several measures which can be obtained from Petri Nets being divided in *steady-state* and *transient measures* [Haver98b]. The former are based on the solution of the underlying Markov chain and the resulting steady-state probabilities defined as p_m for marking *m*. Using these results, the following conclusions can be obtained [Haver98b]:

- Let *A* be a subset of $R(m_0)$, then the *probability for marking set A* is determined by

$$P(A) = \sum_{m \in A} p_m \tag{4.1}$$

- The *average number of tokens in a place P* is obtained by

$$E[P] = \sum_{k=0}^{\infty} k \cdot P(A_k) \text{ with } A_k \text{ the event that } k \text{ tokens are in place } A. \tag{4.2}$$

- The *throughput* X_t of a timed transition *t* with firing rate *W(t,m)* in marking *m* is derived using the following equation

$$X_t = \sum_{m \in R(m_0), t \in E(t,m)} W(t,m) \cdot p_m . \tag{4.3}$$

- Furthermore, using Little's law, the average delay *E[R]* of a token traversing a subnet R is computed by *E[R]=E[N]/X* with *E[N]* the average number of tokens in the subnet and *X* the throughput of the subnet.

It can be seen that the steady-state measures correspond directly to the underlying Markov chain steady-state probabilities. It should be mentioned that the computational effort to calculate the steady-state probabilities grows with the size of the reachability set (corresponding to the size of the Markov chain). This computational effort is the strongest restriction for the usability of this evaluation approach.

Additionally to the steady-state evaluation, transient measures may be obtained from the system. This is of interest especially when the system life-time is too short to reach the steady state or if temporary overload occurs with no corresponding steady-state [Haver98b]. Instead of using the steady-state probabilities of the system, the instant-of-time vectors *p(t)* are used which are obtained by solving the linear differential equation

$$p'(t) = p(t) \cdot Q \text{ with given } p(0) \tag{4.4}$$

and *Q* the *generator matrix* of the Markov chain. Then, similar to the steady-state case, instant-of-time measures can be obtained which correspond directly to the instant-of-time vectors *p(t)* of equation (4.4).

For a formal description of the different measures and further information, see [Haver98b].

SCCS Performance Measures to be obtained from SPNs

With respect to the performance measures which are defined in chapter 3.5, it can be said that all these measures can be obtained from the Petri net. The average values like average response time or average queue length can be determined using the steady-state measures of the Markov chain like explained in the last section. The history values are instant-of-time measures which are determined using the generator matrix *Q* and equation (4.4).

Thus, it can be summarized that the Petri Net approach enables an analytic performance evaluation of group communication systems. Furthermore, the performance measures defined for the evaluation of the main SCCS mechanisms may be obtained. However, it is shown in the next section, that the feasibility of obtaining these measures is often restricted due to the complexity of the resulting Markov chain. Especially the evaluation of system performance measures results in a large Petri net which is very difficult to handle numerically.

4.5.3 Realization Approach with Petri Nets

This section presents the realization of the behavior description with stochastic Petri nets in general, while the next section specifically addresses the realization of the environment.

The behavior description of GCDL is realized with Petri nets by modeling each *role* in the system and the *environment* by separate *modules*. Within each module, the *internal* synchronization of the role (or the environment) is done by introducing the notion of *local places*. For the communication between application and environment modules, *interface places* are defined which are used to build the entire Petri Net based on the given configuration (number of entities). Interface places are locally unique for each module. For the synchronization of several modules, *global places* are defined which are unique within the entire system. Hence, these glo-

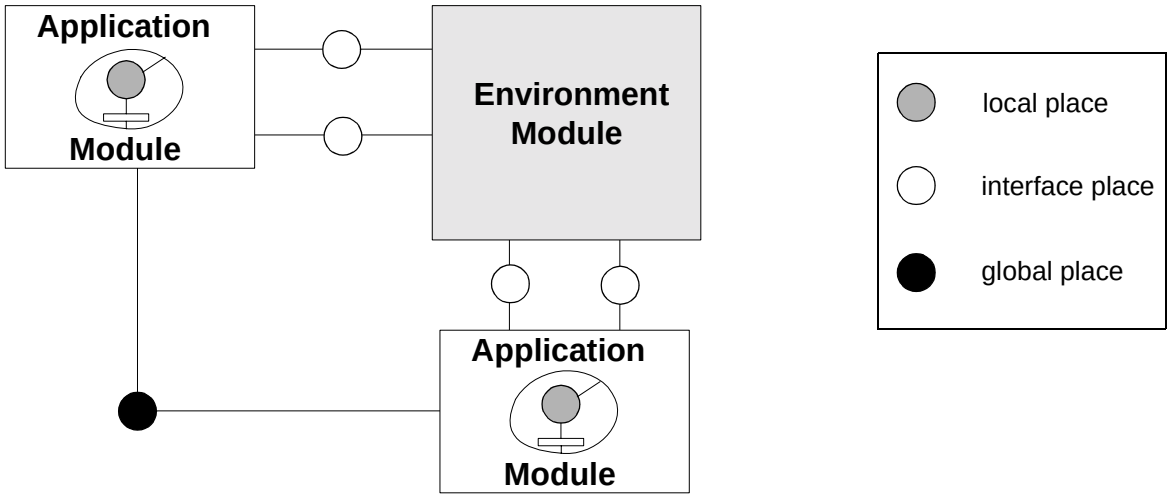


Figure 4-14: *Mapping GCDL onto Petri Nets*

bal places can be used for multipoint transfer of tokens. Figure 4-14 shows the mapping approach. For each entity of the role a separate *module instance* exists. Thus, a Petri Net for a large meeting scenario with 64 participants and the conductor of the meeting consists of 65 application modules and one environment module, which are concatenated to the entire SPN.

If the environment is fully modeled as well, it might consist of further sub-modules which might be structured similar to the entire system, using interface, local, and global places. But the environment might also be modeled using a different level of detail which is demonstrated in the next section.

4.5.4 Realization of the Environment

As mentioned in the last section, the description of a simulated environment might be realized using a different level of detail. For the case of a tree-based conferencing environment, e.g. SCCS, two levels of detail are presented in this section to demonstrate the restrictions of the Petri net realization approach for the evaluation of main SCCS mechanisms.

The first approach models the environment on a high level of detail. With this model, measures of the underlying conferencing system can be determined. The second approach models the application part only without modeling the underlying groupware environment. This is a strong restriction of the modeling approach, because it does not enable the performance evaluation of

certain environment mechanisms.

In the following, both approaches are described in more detail. Furthermore, the shortcomings as well as the advantages of both approaches are outlined.

Approach 1 - Modeling the Entire System

The first approach models the entire group communication system consisting of the underlying environment (the tree of providers) and the application using the infrastructure. In the following, a binary tree is assumed for the sake of simplicity. The environment is modeled by applying the routing scheme to the requests which are sent from one application module entity to another.

The environment is divided in three modules realizing the different types of nodes in the tree-based environment. The system is built of several instances of these modules which are the following ones:

1. routing module *top node* : providing two downward links and one application attachment
2. routing module *intermediate node* : providing two downward links, one upward link, and one application attachment
3. routing module *leaf node* : providing one upward link and one application attachment

Depending on the given topology, a corresponding number of routing module instances exist in the considered scenario. From this information, the entire SPN is built (including the application modules) by concatenating the different modules. Figure 4-15 shows the different

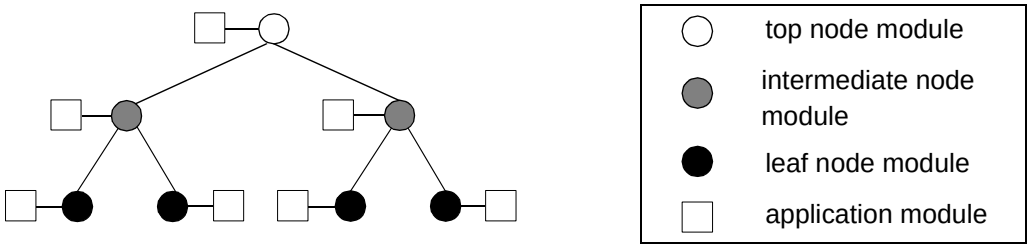


Figure 4-15: *Approach 1 - Modeling the Entire System with Petri Nets*

instances within a 3-level binary tree topology. Thus, the entire system consists of 14 submodules each of them consisting of a Petri Net as described in chapter 4.5.3.

The main advantage of modeling the entire system including the routing modules is that performance measures of the communication infrastructure can be determined.

The main shortcoming of this approach is the complexity of modeling the routing scheme proposed in SCCS with Petri Nets. Routing floor asking requests is very easy to model, because only binary information is needed (is the token allocated in a subtree ?). In contrast to that, routing floor passing requests back from the floor holder to the requesting entity is very difficult, because an explicit addressing takes place during routing the request (the floor holder sends the floor passing request to the requesting entity). To model this routing in each routing module, the information has to be stored which application in the subtree below that module has recently issued a floor asking request. The amount of data to be stored depends on the size of the subtree and of the location of the specific instance (the higher the location in the tree the higher the amount of information). Additionally, the size of the Petri net dramatically increases due to the storage of the routing information.

Approach 2 - Modeling the Application only

It can be seen from approach 1 that modeling the routing in the underlying infrastructure results in an increasing complexity of the resulting Petri net, especially when considering large scaled scenarios. Another approach is to model the application instances only. This means that the application modules communicate directly via the interface places which can be seen in figure 4-16. When using this approach, global places are mostly required to synchronize the applica-

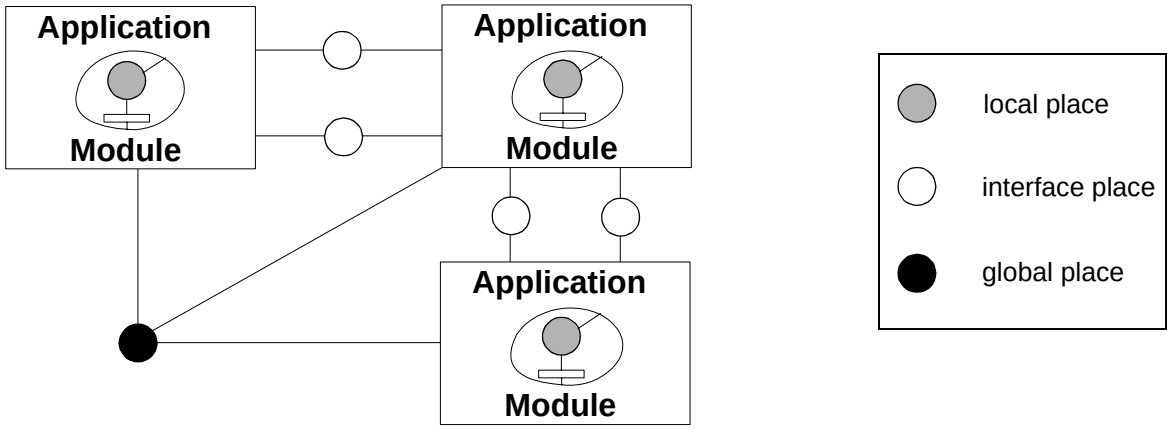


Figure 4-16: *Approach 2 - Modeling Applications only*

tion modules appropriately, which is also the case in the application example of chapter 5.3.

The main advantage of this approach is the smaller size of the entire Petri Net which depends on the number of entities only but not on the size and complexity of the infrastructure. Thus, the computation of the underlying Markov chain is simplified.

The main shortcoming of the approach is that it is not feasible to determine performance measures of the underlying communication infrastructure which might be of interest, e.g. when to obtain queue length statistics. But for an evaluation based on application level measures, this approach is sufficient enough.

4.5.5 Drawbacks when using Petri Nets

The last section showed that it is feasible to realize the behavior description within GCDL using stochastic Petri nets to get an analytical evaluation approach. However, this approach does have some drawbacks which are explained in this section.

Complexity

The complexity of the SPNs highly depends on the scenario which has to be evaluated in terms of scale and number of roles. Even the simple approach of modeling the application only leads to a large joined Petri net, which is very difficult to handle numerically. Modeling the entire system consisting of application and environment modules (and sub-modules) is not practical due to the enormous information storage which is necessary for modeling the routing scheme.

Structure

Compared to the realization of the control and data process using SDL (see chapter 4.4), it can be seen that the realization with SPNs leads to a more monolithic system while the SDL approach allows a more structured realization. Using the *module* approach, a design structure

was introduced in the Petri net realization as well. However, this design is less structured than the SDL approach. This comes from the background of SDL as a specification language of even larger protocols. Hence, means to structure the specification are defined in SDL, while SPNs do only offer a flat structure of the system.

Performance Measures

The analytical evaluation with stochastic Petri nets allows to determine steady-state probabilities of the underlying markov chain as well as instant-of-time measures using the generator matrix of the Markov chain (see chapter 4.5.2). It can be seen from chapter 4.5.2 that using these values, certain performance measures for the performance evaluation of SCCS may be obtained using the Petri net approach.

However, it highly depends on the realization of the system. If the complex realization of the entire communication system is used (approach 1), the system performance measures can be obtained in addition to the application performance measures. But it was already mentioned in the last section that determining performance measures for the communication infrastructure leads to larger Petri nets with a higher complexity of the underlying Markov chain. This effect is even worse in larger scaled scenario in terms of conference members.

If the realization approach 2 is used, the resulting Petri nets are smaller. Hence, the computational effort for obtaining performance measures is reduced. However, this approach does not allow to evaluate the underlying conferencing system which might be necessary when evaluating e.g. protocol mechanisms of the used system.

As a result, obtaining performance measures on application and system level is feasible in general, but specifically depends on the considered scenario.

Access Control

In most group communication scenarios, concurrent access control models are allowed even in simple scenarios. For instance in a large meeting scenario, participants may ask for the floor concurrently. The requests are normally stored in FIFO (First In-First Out) queues by the conductor. FIFO queues may be realized with Petri nets (see figure 4-17 [BaKr96]), but the complexity of the queue increases with the number of stages. Thus, the access control models which can be realized by Petri nets are mostly very simple on the one hand. On the other hand, the control models are difficult enough to result in complex Petri nets even in simple scenarios (chapter 5.3.1).

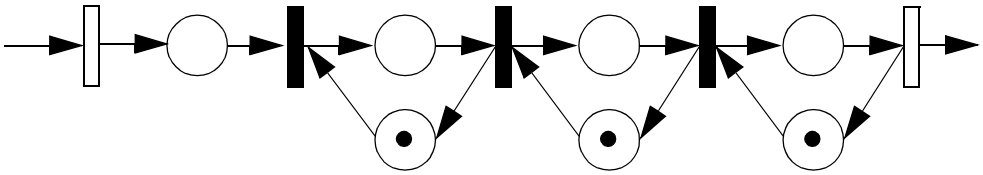


Figure 4-17: *Three-Stage FIFO Queue with SPNs*

Routing Problem

If the simple mapping approach 2 is used, which models the application only, this leads to the following problem (see figure 4-18):

Assuming that application A is invoking a floor request at time t_1, while the floor is held by application B. If there is another application C, which is closer to B, requesting the floor at the

time t_2, this request might be served earlier in the real environment. This would be the case if the time difference t_2-t_1 were less than the latency to be needed to forward A's request up to the node where user C is attached to. If A's floor request had already passed this node, its request would be served earlier. This situation is shown in figure 4-18.

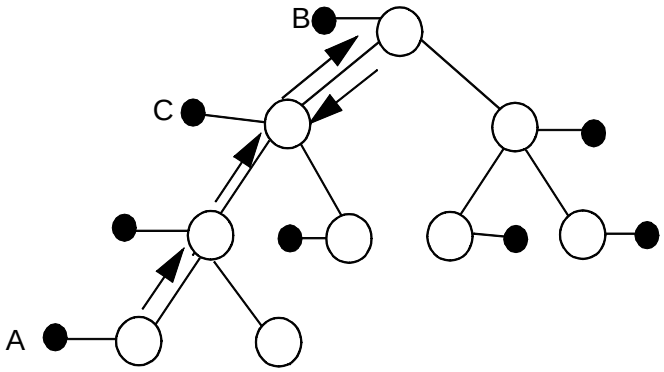


Figure 4-18: *Routing Problem*

These situations cannot be handled when using the simple mapping approach and steady-state probabilities, because the second realization approach does not model the routing in the conference environment. In chapter 5.3, an assumption is made for the access model which excludes such situations in the evaluation. But as a result, it does not include ALL floor control requests in the evaluation.

4.5.6 Summary

This section presented the realization of the GCDL behavior description using stochastic Petri nets. After a brief introduction to the basic definitions of Petri Nets, a short overview was given what measures may be obtained by Petri Nets when using this technique for an analytical evaluation of a system. It was shown that in general all performance measures which were defined for the performance evaluation of SCCS can be determined. After that, the realization of the control and data process within the GCDL context was outlined. Two approaches were depicted, presenting two different levels of detail for the realization. The first one tried to realize the applications as well as the environment in the case of a simulated environment. The second realized the applications only without taking the underlying conferencing system into account. Both realization techniques suffer from several drawbacks. The main disadvantages are the increasing complexity of the resulting SPN, especially in the first approach, and the missing realization structure compared to the SDL approach. The effect of growing Petri nets is even worse when introducing more complex access control models in the scenarios, for instance using queues to store requests.

It can be summarized that the Petri net realization approach enables the analytical evaluation of groupware systems within the GCDL context in general. But this ability highly depends on the complexity of the resulting Petri net.

4.6 Summary

This chapter dealt with the problem of modeling groupware applications in the design phase of the system to facilitate the implementation and evaluation of the system without implementing it. The focus of the work was on modeling the object structure and describing the dynamic behavior, including aspects like communication, user behavior, and access control.

The structural design aspect is mainly used to support developers in the implementation design phase of group communication systems. The behavior modeling aspect is used to cover performance aspects allowing developers to evaluate the groupware system without implementing and installing. For a more formal discussion of the modeling problem, requirements for modeling approaches were defined to be fulfilled by modeling approaches for groupware systems.

The presentation of the related work in the area of groupware modeling and evaluation depicted three main weaknesses of existing approaches, namely the *missing detail of communication aspects*, *insufficient user behavior modeling*, and a *poor access control model*. Furthermore, most approaches deal only with specific modeling problems. This forces the developer to use different methods for different aspects.

As a modeling approach, the *Group Communication Description Language* (GCDL) was proposed. The focus of GCDL is on both the structural design of groupware systems and modeling aspects like control/data traffic as well as user behavior with an independence from the realization of the implementation. The access control concept of GCDL is based on the definition of *roles* accessing *objects* within a *scenario* according to defined *rules*, referred to as *the social protocol*. Several *instances* of a role may exist representing dedicated users in a scenario. To each object within the system, *negative* as well as *positive access rights* are associated. Furthermore, the dynamic change of object access is abstracted using a *floor control* notation.

The components of the framework were presented consisting of the object and process model. For the realization of the former, the syntax definition of a description language was presented to enable structural modeling of the considered system. For the latter, the instantiation of roles and the communication among these instances were presented. But the GCDL framework does not define how to describe the behavior of the communicating entities, defined as *processes*, or how to apply the rules or the data traffic patterns within the scenario and modeling the user behavior. However, functional requirements for the description of the processes were defined. As an example, two different description approaches were presented which might be used in the GCDL framework for analytical and simulative performance evaluation.

The first approach realized the control and data processes within GCDL as SDL systems. For that, a framework definition of these SDL systems was given which can easily be mapped to an event-based simulation, which was also depicted. For an easier implementation of these event-based simulations, a simulation framework was introduced. The object and activity model were outlined, and a realization of this framework was presented. An implementation of this framework is used for the performance evaluation of SCCS, presented in chapter 5 and 6.

For an analytical evaluation of groupware systems, the description of the control/data processes with stochastic Petri nets was presented. Two particular methods were depicted, modeling the entire system (application and environment) or the applications only. For both, a *module-based* realization was used to define a structure in the realization. However, a well-defined structure as in SDL cannot be achieved with this approach. Furthermore, the main disadvantage of both methods is that the resulting Petri net grows rapidly with the size of the scenario as well as with the level of modeling details in terms of access control and user behavior.

Both behavior description approaches, using SDL or Petri Nets, enable the performance evalua-

tion of a groupware system modeled with GCDL either on simulative or analytical level. The performance measures which were defined for the performance evaluation of SCCS can be obtained from both realization approaches.

It can be summarized that the definition of the Group Communication Description Language offers an easy and convenient method for groupware systems modeling. Furthermore, GCDL covers the defined requirements and provides an integrative approach to deal with structural design of groupware systems as well as with modeling aspects like control/data traffic or user behavior. It was shown that by using SDL and stochastic Petri nets for the behavior description analytical as well as simulative evaluation is feasible.

The presented modeling framework will be used in the next two chapters for the performance evaluation of SCCS protocol mechanisms, namely the proposed resource management and the dynamic reconfiguration of the conferencing tree topology.

CHAPTER 5

Resource Management Evaluation

After proposing a scalable conferencing service to facilitate the development of groupware applications, an approach to model groupware systems was defined to enable structural modeling as well as performance evaluation of group communication systems in general. This chapter deals with the evaluation of a specific mechanism of SCCS, namely the *resource management*. This mechanism is compared to the scheme being used in the T.120 standard.

There are two goals of the performance evaluation. As a *system aspect*, performance measures for the resource management of SCCS should be obtained to outline the improvement when applying this mechanism. On the other hand, the applicability of different modeling methods are compared as a *methodology aspect* of the evaluation. It is expected that the observed improvement is very similar for all applied modeling methods. As a consequence, this chapter is focussed on the methodology aspect.

Due to the stringent timing constraints for floor control operations (see chapter 2.1.2.1) and for the sake of simplicity of the result presentation, only floor requests are considered. The evaluation is very similar when performing channel requests.

As a starting point for the evaluation, the problem is addressed in chapter 5.1 how to consider various topologies of the conferencing environment. This problem occurs due to the topology dependency of the proposed resource management in SCCS.

For performing the evaluation, three different methods are presented. In chapter 5.2, the simple stochastic approach, presented in chapter 4.2.1.1, is applied. This approach is used to give a first impression of the performance gain that can be achieved and to demonstrate the computational effort which is needed for the results, even in very simple scenarios. The other two approaches use the GCDL model, but offer different realizations for the behavior description. First, SPNs are used in chapter 5.3 to describe the communicating entities as presented in chapter 4.5. This approach allows to determine analytical results. But it is shown that the effort to obtain analytical performance measures grows rapidly with the complexity and the size of the given scenario. The third method, presented in chapter 5.4, uses the SDL realization of the behavior description, presented in chapter 4.4. It is shown that this evaluation method allows to obtain the widest spectrum of performance measures even in more complex scenarios in terms of user behavior and access control. However, only statistical results are obtained with this method.

5.1 Classification of the Tree Topology

The main problem in evaluating the proposed resource management scheme of SCCS is that the results highly depend on the tree topology. The smaller the distance between the active users in the tree, the higher the performance gain of the proposed resource management scheme. This is because the resource requests can be handled more locally in the tree.

As a consequence, the performance measures of the scheme should be determined for arbitrary topologies. As an approximation, a finite number of tree topologies is generated randomly. But this might lead to large standard variations when averaging the results. Furthermore, the average values do not properly reflect the variety of start topologies in a conference with respect to the distribution of the active users.

5.1.1 Distribution Metric $V(H_t)$

To overcome these weaknesses, the generated topologies are classified according to a metric that takes into account where the active entities reside in the tree. The evaluation results are presented according to a *tree topology spectrum* defined by the minimum and maximum metric value.

In the following, this classification metric is defined for the distribution of active entities. For that, define H_t as the set of active users with respect to the token *t*. Moreover, R_t is the set of variations which can be build by pairs of users in H_t. Furthermore, let *d(u,v)* be the shortest path from user *u* to *v* in the tree (distance metric). Then, the *distribution metric* $V(H_t)$ is defined as

$$V(H_t) = \frac{1}{|R_t|} \cdot \sum_{u,\, v \in R_t} d(u, v) \tag{5.1}$$

It can be seen from the definition that the metric averages the distances of all shortest path routes between active users with respect to a specific token *t*. The smaller $V(H_t)$, the closer are the active users located in the conference tree.

Using this definition of $V(H_t)$, the performance evaluation is now performed using the following *topology classification* approach:

A given number of topologies, defined as n_r, is randomly generated. For each of these topologies, the value for $V(H_t)$ and the performance measures are determined, using any of the proposed modeling techniques.

After the entire evaluation is finished, the minimum and maximum for all $V(H_t)$ values of the random topologies are determined, and

$$class_i = min + \frac{max - min}{n_c} \cdot i \text{ for } i=0,...,n_c\text{-}1 \tag{5.2}$$

is defined as the start value of class *i* with n_c defined as the number of considered classes. Each of the randomly generated topologies is now classified to class *i* if the $V(H_t)$ value for this topology is in the interval $[class_i, class_{i+1})$.

Within each class, the performance measure results of the topologies are now averaged. As a consequence, the results can be presented depending on the class of the topology. It can be seen that the proposed classification approach is very similar to a histogram approach.

5.1.2 Statistical Considerations

There are two parameters in the proposed classification approach. The first is the number of simulation runs to be performed, while the second one defines the number of classes in the *tree topology spectrum*. In the following, a brief statistical discussion of the distribution metric is presented to clarify how to set both parameters.

For that, define $n = |R_t|$ as the number of variations of the active user set and $X_i=d(v_i)$ with $v_i \in R_t$ and $i=1,...,n$. Then, $(X_i)_{i=1,...n}$ defines a sequence of independent and identically distributed random variables which satisfy the assumption for the *Central Limit Theorem* [Rao84]. Hence, with

$$S_n = \frac{1}{n} \cdot \sum_{i=1}^{n} X_i \tag{5.3}$$

it follows that S_n is normally distributed. Hence, $S_n \sim N(\mu, \sigma)$ with $\mu = E(X_n)$ and $\sigma = Var(X_n)$. Comparing equation (5.3) with the definition of $V(H_t)$, it can be seen that the metric $V(H_t)$ is normally distributed.

Hence, the number of topologies assigned to edge classes, i.e. close to the minimum and maximum $V(H_t)$ value, is expected to be much smaller compared to the mid range classes. As a consequence, the number of simulation runs together with the number of considered topology classes must be chosen in a way that even for edge classes the confidence of the obtained results is assured. This means that both parameters have to be chosen so that the number of topologies in the edge classes is large enough to ensure small confidence intervals of the results.

It is worth mentioning that the dependence on the topology exists for all presented performance evaluation methods. For the sake of simplicity, only binary trees are used for the first method, so the topology dependence can be solved much simpler. For the other evaluation methods, this dependence has to be taken into account.

5.2 Evaluation using Simple Probabilities

The following section presents a performance evaluation based on a simple stochastic model to determine the performance gain for simple floor testing operations.

For the evaluation, the following assumptions and simplifications hold:

- The observed user is located in a leaf of a binary tree.
- To each node in the binary tree exactly one user is attached.
- The transmission time on all links in the tree is the same.

In the following, two scenarios are presented concerning the distribution of the floor in the conference. In the first one, the floor is distributed uniformly among the users in the conference, i.e. simulating a group discussion without a conductor. For the second scenario, a dedicated conference member is defined with a higher probability of possessing the floor, i.e. simulating a conducted conference.

For both scenarios, the *performance gain* is determined (see chapter 3.5.2.1), when applying the resource management scheme proposed by the author compared to the centralized scheme of the T.120 standard. Other performance measures (see chapter 3.5), either on application or

system level, are not determined due to the complexity of the computation.

5.2.1 Scenario 1 - Uniformly Distributed Floor

The considered scenario is very simple. There is only one object which is accessed by all users for reading and writing using one floor associated to the object. A uniform distribution of the floor allocation among the users is assumed. In a first step, the performance gain is determined independent from the *mode* of the floor, namely whether it is allocated exclusively or non-exclusively. In a second step, the determination of the performance gain is applied for a exclusively/non-exclusively allocated floor respectively.

5.2.1.1 Determination of the Performance Gain

Given a binary tree with height $h>1$, and let $n:=2^h-1$. Furthermore, define X as the random variable that a request originated from a leaf node provider has to be routed up to a given level in the tree, i.e. $P(X=j)$ is the probability that a request has to be routed up to level j.

Hence, the expectation of X is given by

$$E[X, h] = \frac{\sum_{i=0}^{h-1} iP(X=i)}{\sum_{i=0}^{h-1} P(X=i)} \tag{5.4}$$

The performance gain $G(u)$ (in percentage) for a user u located at tree level $j=h$, i.e. in the leaf of the tree, can be determined as the difference of routing the request with the centralized scheme or using the proposed scheme. In figure 5-1, the performance gain is illustrated. The

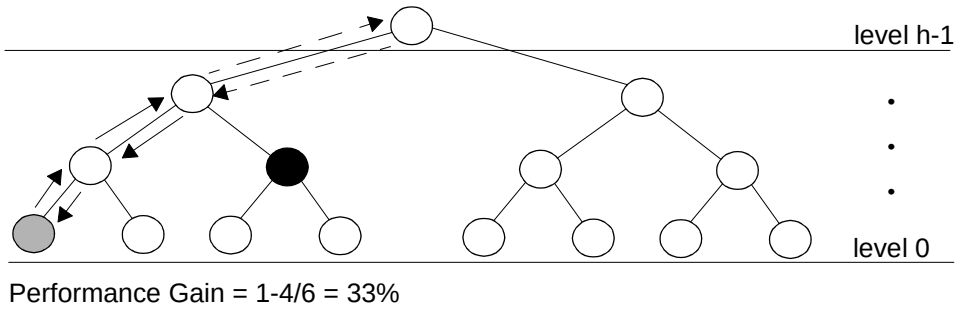


Figure 5-1: *Performance Gain for Floor Testing*

gray user sends a test request for a floor which is allocated by the black provider. The request has to be routed up to the first level in the tree, when using the proposed scheme (full arrows). In the centralized scheme, the request is always routed to the topmost provider (full plus dotted arrows). It can also be seen that the performance gain does not depend on the location of the black provider in the other subtree of the provider on level j. Thus, the performance gain can be determined by

$$G(u) = \left(1 - \frac{E[X, h]}{h-1}\right) \cdot 100\,\%. \tag{5.5}$$

It can be seen that *G(u)* is defined in a different way than in chapter 3.5.2.1, where the performance gain is defined as the ratio of two different response time values. But the definition above is equivalent, because the response time directly depends on the level to which the request is routed. It can also be seen that the performance gain *G(u)* for a user *u* only depends on the tree level *j=h*.

5.2.1.2 Case 1 - Exclusive Floor

The results of the last section are used to determine performance results for an exclusively allocated floor. For that, it is assumed that the exclusive floor is allocated by a user in a binary tree with a uniform distribution. Let $p_i := n^{-1}$ be the probability that user *i* possesses the floor exclusively. Hence, it holds

$$\sum_{i=1}^{n} p_i = 1 \tag{5.6}$$

With the definition of the previous section, *P(X=j)* can be determined by

$$P(X = j) = 2^j \cdot p \text{ for } j=1,..,h-1 \tag{5.7}$$

following the reasoning that *P(X=j)* is given by the probability that there is a floor holder in the other subtree below the provider on level j (with 2^j conference members). Hence, the expectation of X is given by

$$E[X, h] = p(h \cdot 2^h - 2^{h+1} + 2) \tag{5.8}$$

The performance gain *G(u)* is determined according to equation (5.5).

5.2.1.3 Case 2 - Non-Exclusive Floor

In the second case, a non-exclusive floor is assumed for the access to the object. In contrast to the first case, the probability to possess the floor is defined as $p \in (0,1)$.

P(X=j) is given by the probability that there is a floor holder in the other subtree below the provider on level *j* (with 2^j conference members) and simultaneously, there is no floor holder in the subtree to which the requesting provider belongs (with 2^j-1 conference members). Hence, it holds

$$P(X = j) = 2^j \cdot p \cdot (1-p)^{(2^j - 1)} \text{ for } j=0,..,h-1 \tag{5.9}$$

The expectation of *X* and the resulting performance *G(u)* can be determined according to equations (5.4) and (5.5).

5.2.1.4 Results

Figure 5-2 shows the performance gain for both cases. For the exclusive case, it can be seen that the performance gain decreases from nearly 33 percent to 12 percent, when the tree height increases. For the non-exclusive case, different values of *p* are considered. It can be seen that for a fixed value of *p* the performance starts to decrease when the trees becomes larger, but at a certain point the performance gain increases. For the height-dependent value of *p* the performance gain decreases when the tree grows, which is consistent to the results for the exclusive floor.

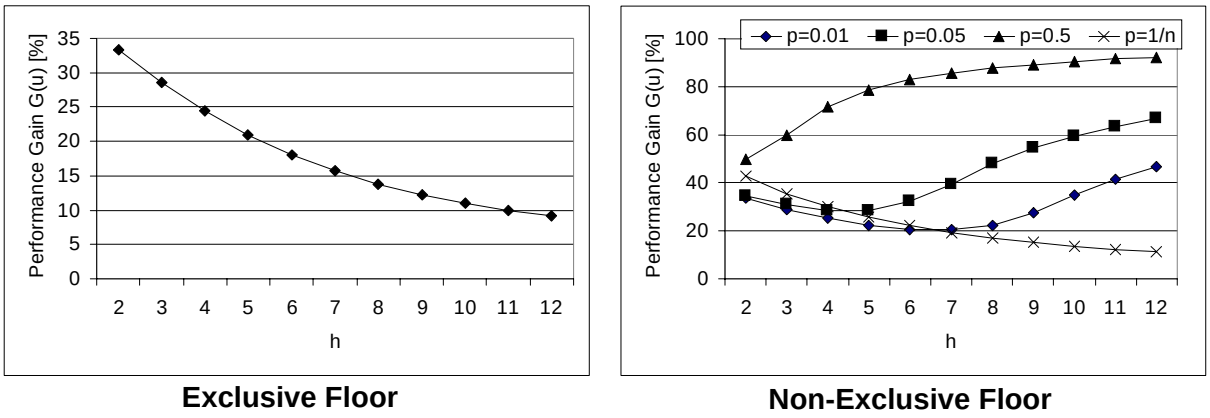


Figure 5-2: *Stochastic Approach - Performance Gain G(u) Scenario 1a (left) and Scenario 1b (right)*

5.2.2 Scenario 2 - Active User

In this second scenario, the same assumptions are valid concerning the object and floor given in the system as in the first scenario using an exclusive floor. But in contrast to scenario 1, there is a dedicated *active user* in the system with a higher probability to request an exclusive floor compared to the rest of the users. For the rest of the members, a uniform distribution is assumed similar to scenario 1.

Given a binary tree with height *h*, and let n:=2^h-1, $a \in (0, 1)$ and $p = \frac{a}{n-1}$.

It is assumed that the active user $k \in \{1, \ldots, n\}$ grabs the exclusive token with probability $p_k = 1 - a$, while the rest of the members possess the token with probability p_i:=p for all $i \in \{1, \ldots, n\} - \{k\}$. Hence

$$\sum_{i=1}^{n} p_i = 1 .$$

Define $l_k \in \{0, \ldots, h-1\}$ as the level, where the active user resides. It is assumed without any restriction that the active user resides in the same branch of the requesting user below the top provider. Otherwise, scenario 1 can be used with a higher probability in the other subtree below the top provider, i.e. adding *a* to the probability in equation (5.7).

Define *X* as the random variable that a request has to be routed up to a given level in the tree, i.e. *P(X=j)* is the probability that a request has to be routed up to level *j*. This probability is given similar to scenario 1, but divided in two cases

$$P(X = j) = 2^j \cdot p \text{ for } j \neq l_k \tag{5.10}$$

$$P(X = j) = (2^j - 1) \cdot p + 1 - a \text{ for } j = l_k \tag{5.11}$$

For the expectation of X and the performance gain *G(u)*, the equations (5.4) and (5.5) are used. Figure 5-3 shows the performance gain *G(u)* depending on *h* and l_k for different values of a. It can be seen that the performance gain decreases for increasing *a*. This is not surprising, because *1-a* denotes the activity of the active user, which means that with increasing value for *a* the activity decreases together with the performance gain. Furthermore, it can be seen that the performance gain also decreases for higher l_k. This is due to the location of the observed conference member who resides in the leaf of the tree. Higher values for l_k means that the active user reside closer to the top provider than to the observed user.

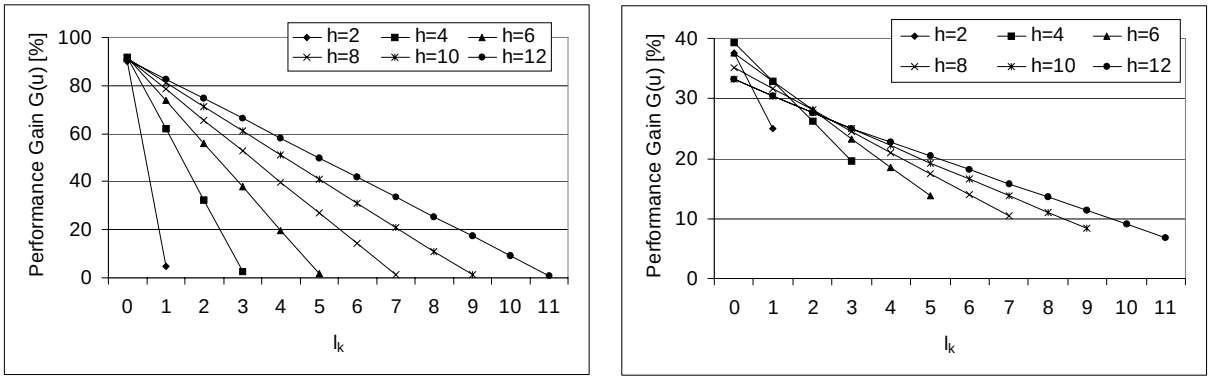


Figure 5-3: *Stochastic Approach - Performance Gain G(u) Scenario 2*

5.2.3 Summary

This section presented a performance evaluation of the SCCS resource management scheme compared to a centralized scheme using a simple stochastic approach. Two very simple scenarios in terms of user behavior and access control were used. Scenario 1 assumed a uniformly distributed usage of the floor among all conference members, while in scenario 2 one active user was added to the system, which could be used for modeling a conductor in a conference.

Considering the *system aspect* of the evaluation, the determined performance gain in both scenarios showed an improvement when applying the proposed resource management. Furthermore, the performance gain depends on the location of the user in the tree. The higher the tree, the smaller the obtained gain. With respect to the *methodology aspect*, it can be stated that in general application performance measures can be obtained analytically using the simple method. However, obtaining results for arbitrary tree topologies, arbitrary users, and more complex scenarios in terms of access control and user behavior would probably result in more complex equations. Hence, the method is mainly restricted to simple scenarios in terms of access control, user behavior, and fixed configurations.

5.3 Evaluation using SPNs

In this section, an evaluation is performed based on the SPN behavior description within the GCDL modeling approach (see chapter 4.3) using the realization techniques outlined in chapter 4.5. The main purpose of this section is to demonstrate the computational effort, which is necessary to get analytical results even in small and simple groupware scenarios. Furthermore, the applicability of the approach should be demonstrated.

In the following, the stochastic Petri net is presented which is used for the evaluation, before calculating the steady-state probabilities. The latter are used for the determination of specific performance measures. Additionally, an automatic approach and the computational effort to determine the performance measures are presented.

Assumption

To avoid the routing problem of chapter 4.5.5, it is assumed that only one floor asking request is issued within a specific time interval. For this assumption, the following definition is given:

Definition 5.1: Successful Operation
A floor asking operation which is sent to a floor holder at time t_1 and received at time t_2 is defined as ***successful*** *if there is no other floor asking operation in the time interval $[t_1,t_2]$.*

Considering that only the successful floor operations are taken into account for the performance improvement of the system (because non-successful operations are not confirmed by a floor passing operation), the assumption issuing only one request within a given time interval is made in the following to model and evaluate a very simple conferencing scenario.

5.3.1 Scenario - Uniformly Distributed Floor

The considered scenario for the performance evaluation is very simple. There is one object in the system to which an exclusive floor is associated. This floor is requested by each user *i* after ask_i seconds (ask_i is exponentially distributed). The current floor holder passes the floor to the requesting user after $pass_i$ seconds ($pass_i$ is exponentially distributed). The communication environment consists of a binary tree of height $h>1$ with $n:=2^h-1$ nodes and one user attached to each node.

5.3.1.1 Stochastic Petri Net

The SPN of the communication system is modeled using the second approach presented in chapter 4.5.4 by modeling the application level only. For each application instance, a separate module $i=1,..,n$ is defined, as explained in chapter 4.5, with *n* the number of users in the system. *Local places* are indexed with *i*, *global places* do not have any indices.

Each module is divided in two functional parts. The first one is responsible for *asking* the floor, while the second one *passes* the floor to the requesting application. Figure 5-4 shows the SPN module for a single application which is explained in the next section.

Note that this is the Petri Net for one application entity only. The entire Petri net for the scenario consists of $n(=2^h-1)$ modules. Figure 5-5 shows the topology for tree height h=3. The modules are numbered as shown in this figure. The entire combined SPN is not shown, because it is very complex due to its overlapping global places.

Initial Marking of the Complete SPN

After combining the SPN modules, the initial marking of the resulting SPN has to be defined. At the beginning, there is a floor holder *i* in the scenario, and a global request is possible. As a consequence, there is initially a token in the global place *grequest* and a token in the local place $token_i$ indicating that this user holds the floor. In the SPN modules of the other users, there is a token in local place $start_j$ with $j \neq i$.

Firing the Concatenated SPN

In the functional part *floor asking*, each application issues a floor asking request if a global request for the floor is possible (token in *grequest*), by firing the transition ask_i with exponential distribution. With the *grequest* place, the successful operation assumption is fulfilled.

The *waiting for confirmation* state in the requesting module is modeled by putting a token in the local place $lrequest_i$. The transition $mark_i$ stores the local floor holding information, firing exponentially distributed, in the local place $token_i$ if the floor is made globally available by the

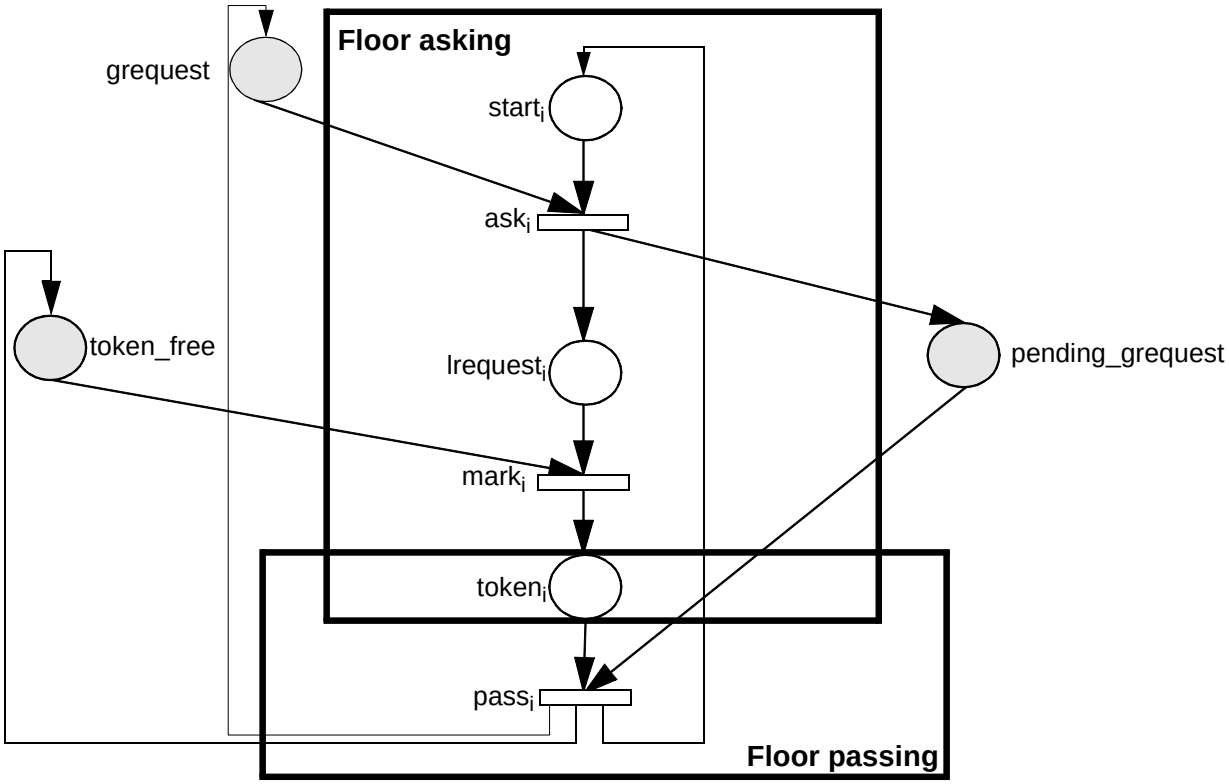


Figure 5-4: *SPN Application Module*

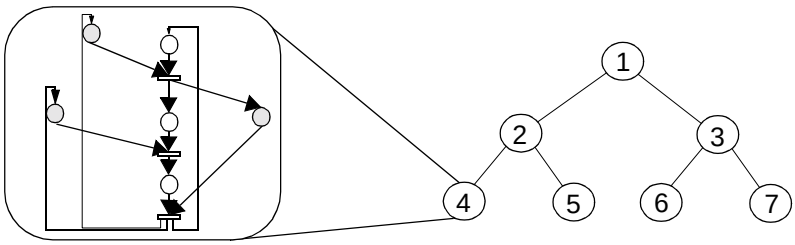


Figure 5-5: *Petri nets - Topology with h=3*

current floor holder (signaled by global place *token_free*) and if there is a local request for the floor (signaled by local place $lrequest_i$).

In the *floor passing* functional part, the transition $pass_i$ fires exponentially distributed if there is a pending global floor asking request (signaled by global place *pending_grequest*) and the module is the current floor holder (signaled by local place $token_i$). The transition $pass_i$ models the floor passing time (consisting of the sum of minimum token holding time and service time for handling the request) and places a token in global place *token_free* indicating the success of the floor passing operation, so the appropriate $mark_i$ transition of the requesting application is able to fire (see above). Additionally, issuing a global request is re-enabled by placing a token in global place *grequest*, and a new token is placed in $start_i$ to request for a floor locally in the future.

5.3.1.2 Determination of the Steady-State Probabilities

For the determination of the steady-state probabilities, the SPN is mapped to the underlying Markov chain.

Due to the *successful operation assumption*, there can be only one active floor asking/passing operation in the entire system. In the following, the sequence of floor asking by a user and the following floor passing operation of the current floor holder is called *transaction*. Such a transaction between user *i* and *j* can be described by the following sequence of markings:

1. user *j* possesses the floor (token in local place $token_j$), and a global request is possible (token in global place *grequest*)
2. user *i* has requested the floor, while user *j* possesses the floor (token in local place $lrequest_i$ and $token_j$)
3. user *i* has requested the floor and the token is free (token in local place $lrequest_i$ and global place *token_free*)
4. user *i* who has requested the floor possesses it now (token in local place $token_i$)

The latter point is equivalent to the first one in the sense that the floor holder has now changed from user *j* to user *i*. By dividing a transaction for arbitrary users as shown above, the markings of the SPN can be divided into three groups. The first group S_1 contains all markings describing that any user possesses the floor. The second group S_2 contains the markings for the situation that a user has requested the floor, while it is possessed by another one. And the last group S_3 contains the markings that a user has requested the floor which is now free. For a formal description of these three groups the following notation for the different markings is used:

Definition 5.2: Transaction Markings
Given user j as the current floor holder in the system. For a transaction between user i and j in the Petri Net of figure 5-4, the possible markings are named by $S=(\#grequest, \#pending\_grequest, \#token\_free, \ldots, \#start_i, \#lrequest_i, \#token_i, \ldots, \#start_j, \#lrequest_j, \#token_j, \ldots)$.
The local places of all users k not equal to i or j are set to $(\#start_k, \#request_k, \#token_k)=(1,0,0)$. *Without restriction, user i is written first in that notation.*

Using the notation of definition 5.2, the three marking groups can be defined as

$$S_1 = \{S_{j1}; j \in \{1,\ldots,n\}\} \quad \text{with } S_{j1}=(1,0,0,\ldots,1,0,0,\ldots,0,0,1,\ldots), \quad (5.12)$$

$$S_2 = \{S_{ij2}; i \in \{1,\ldots,n\}; j \in \{1,\ldots,n\} - \{i\}\} \quad \text{with } S_{ij2}=(0,1,0,\ldots,0,1,0,\ldots,0,0,1,\ldots), \quad (5.13)$$

$$S_3 = \{S_{i3}; i \in \{1,\ldots,n\}\} \quad \text{with } S_{i3}=(0,0,1,\ldots,0,1,0,\ldots,1,0,0,\ldots). \quad (5.14)$$

The set *S* contains all states of the Markov chain as the union of S_1, S_2, and S_3 with a size of

$$|S| = |S_1| + |S_2| + |S_3| = n + n \cdot (n-1) + n = n^2 + n \quad (5.15)$$

with $n:=2^h-1$ the number of users in a tree of height *h*. As a consequence, the number of states of the underlying Markov chain grows quadratically.

The rates between appropriate markings of the three marking groups are given by the firing rates of the transitions ask_i, $pass_i$, and $mark_i$ in each module i. So define λ_{i1} as the rate for transition ask_i, λ_{i2} as the rate for transition $pass_i$, and λ_{i3} as the rate of transition $mark_i$ in module *i*. Then, the resulting Markov chain for one transaction between user i and j within the system is given in figure 5-6. Using this part of the entire Markov chain and the definition of *S* as the union of separated groups S_1, S_2, S_3, it can be seen that the generator matrix *Q* of the entire

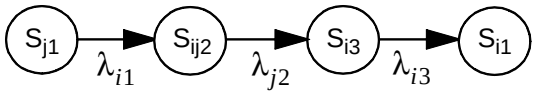


Figure 5-6: *Markov Chain for a Transaction*

Markov chain has the form

$$Q = (q_{ij})_{i,j}^{n(n+1)} = \begin{bmatrix} 0 & A & 0 \\ 0 & 0 & B \\ C & 0 & 0 \end{bmatrix} \in \Re^{n(n+1) \times n(n+1)} \tag{5.16}$$

with the sub-matrices

$$A = \begin{bmatrix} 0 & \cdots & 0 & & & \lambda_{n1} & & 0 \\ \lambda_{11} & & 0 & \cdots & & 0 & & \\ 0 & & & & & & & \lambda_{n1} \\ & & \lambda_{11} & & & 0 & \cdots & 0 \end{bmatrix} \in \Re^{n \times n \cdot (n-1)}, \tag{5.17}$$

$$B = \begin{bmatrix} \lambda_{22} & & 0 \\ 0 & & \lambda_{n2} \\ & \vdots & \\ \lambda_{12} & & 0 \\ 0 & & \lambda_{(n-1)2} \end{bmatrix} \in \Re^{n \cdot (n-1) \times n}, \tag{5.18}$$

$$C = \begin{bmatrix} \lambda_{13} & & 0 \\ 0 & & \\ & & \lambda_{n3} \end{bmatrix} \in \Re^{n \times n}, \tag{5.19}$$

and the constraint

$$q_{ii} = \sum_j q_{ij} \quad \text{for } i=1,..,n(n+1) \tag{5.20}$$

Matrix A denotes the transitions from marking group S_1 to S_2, B from S_2 to S_3, and C from S_3 to S_1. The steady-states probabilities can now be obtained by solving the equation

$$\pi \cdot Q = 0 \text{ with } \sum_i \pi_i = 1 \tag{5.21}$$

In the following, the steady-state probabilities are indexed using the same notation as in equations (5.12) to (5.14), e.g. π_{j1} is the steady-state probability that user j possesses the floor.

5.3.2 Performance Measures

System performance measures, like queue sizes, cannot be obtained from the SPN above, because the conferencing environment is not modeled. As a consequence, only the *performance gain G(u)* is obtained as an application performance measure. Furthermore, the *average floor request frequency* is determined as another application parameter of the system.

For the demonstration how to obtain the performance measures from the SPN, only the equations of these measures are presented. In chapter 5.3.2.3, a possible automatic calculation is proposed using available tools.

5.3.2.1 Computation of the Performance Gain G(u)

In equation (5.5), the performance gain *G(u)* of the proposed resource management scheme for a user *u* was calculated for floor testing requests. This section determines *G(u)* for floor asking or floor passing requests.

In figure 5-7, the routing is illustrated when performing a floor asking request, indicating the

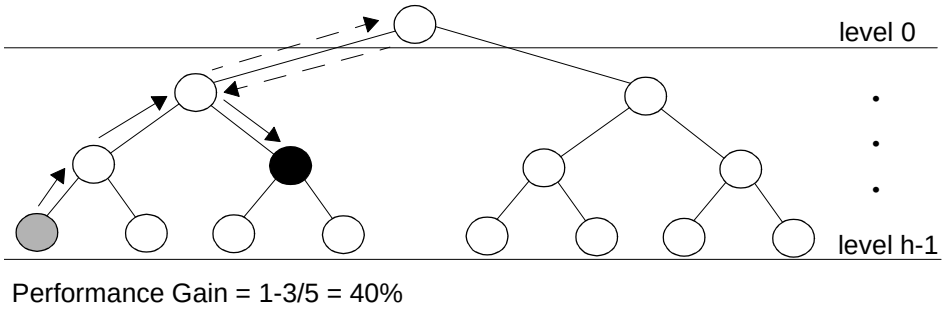


Figure 5-7: *Performance Gain for Floor Asking*

old routing scheme as an addition of the full and dotted arrows. In this example, the gray user asks for the floor which is allocated at the black provider. In contrast to the floor testing case in chapter 5.2, the performance gain for floor asking (and passing) operations also depends on the location of the current floor holder in the subtree. As a consequence, the simple approach of chapter 5.2 cannot be used, i.e. determining the mean level to which the requests are routed using the proposed routing scheme and applying equation (5.5). Instead, the exact number of hops used for routing the request to a specific floor holder has to be taken into account.

Hence, define $d_m(u,v)$ as the minimal number of hops between user *u* and *v* and $l(u)$ as the level of user *u* in the tree starting with 0 for the top provider. Thus, the performance gain $G_u(v)$ for a floor asking operation of user *u* with user *v* as the current floor holder can be determined by

$$G_u(v) = \left(1 - \frac{d_m(u,v)}{l(u) + l(v)}\right) \cdot 100\ \% \qquad (5.22)$$

Thus, the mean performance gain *G(u)* for floor asking operations is calculated by multiplying $G_u(v)$ with the probability that user *v* is the current floor holder at the time of issuing the request, and determining the sum over all possible floor holders. The probabilities are given by the steady-state probabilities π_{uv2} of the underlying Markov chain. Hence, the mean performance gain *G(u)* of user *u* is given by

$$G(u) = \frac{\sum_{v \neq u} G_u(v) \cdot \pi_{uv2}}{\sum_{v \neq u} \pi_{uv2}} \tag{5.23}$$

An advantage of the proposed SPN model compared to the simple stochastic model of chapter 5.2 is, that the Petri Net approach allows to obtain results for several topologies. The determination of *G(u)* in equation (5.22) and (5.23) is not specific for binary trees. Hence, arbitrary topologies might be used.

As mentioned in chapter 5.1, the location of the different entities in the tree topology has to be taken into account, because it affects the average response time and therefore the performance gain *G(u)*. In chapter 5.1, a classification approach for this problem is proposed. Hence, the computation of $G_u(v)$ has to be performed for each tree topology which was generated randomly. This results in a class-based presentation of the performance gain *G(u)*. But it has to be emphasized, that the classification has to be applied only on equations (5.22) and (5.23). As a consequence, the generator matrix *Q* and the steady-state probabilities does not need to be recalculated for each considered topology.

5.3.2.2 Other Measures to be Obtained

The steady-state probabilities can be used to obtain another application parameter which might be interesting to characterize the given workload of a specific user for the underlying conferencing system.

In the used SPN of chapter 5.3.1.1, the workload produced by each user is specified by the parameters ask_i, $mark_i$, $pass_i$ for each Petri net module *i*. From these parameters (by using the resulting steady-state probabilities), the *average floor request frequency* for each user can be obtained from the system. These values are determined by using the appropriate firing rates and the steady-state probabilities from marking group S_2, i.e. the probabilities that the observed user has issued a floor request, while another user is holding the floor. Thus, the *average floor request frequency F(u)* of user *u* is given by

$$F(u) = \lambda_{u1} \cdot \sum_{i \neq u} \pi_{ui2} \tag{5.24}$$

5.3.2.3 Automatic Evaluation Approach

In the previous sections, the steady-state probabilities and the performance gain *G(u)* were obtained manually by determining the generator *Q*, solving equation (5.21), and applying equation (5.22) and (5.23). But it is also possible to generate the results automatically from the SPN by using existing tools like SPNP (*Stochastic Petri Net Package* [CMT89]).

With this tool, the SPN is specified in a C-like language, and the generator matrix is determined for solving equation (5.21). But in the presented evaluation approach, the SPN was designed as a *module* and the entire Petri Net consists of several of these modules with overlapping *global places*. As a consequence, a preprocessing unit is necessary to generate the concatenated Petri net from the module description and the given topology of the conferencing environment. However, it is possible with SPNP to define arrays of places building symmetric concatenated SPNs which is very similar to the considered scenario. However, the preprocessing unit is proposed to follow the module-based SPN design on the one hand and to allow different settings

for the module instances on the other hand.

With this preprocessing step, SPNP is able to determine the steady-state probabilities from the generator matrix *Q*. But for the determination of the performance gain *G(u)*, a postprocessing step is also needed to determine the number of hops between dedicated conference members for the calculation of $G_u(v)$ (equation (5.22)). After that, the steady-state probabilities can be used to compute *G(u)*. When using the classification of the tree topology (see chapter 5.1), the post-processing step has to be applied for each topology, and the results have to be presented within each class of topology.

Both steps, preprocessor and postprocessor, can be written in C or C++ to be applied on the input and output files of SPNP respectively. In figure 5-8, the different steps of this automatic

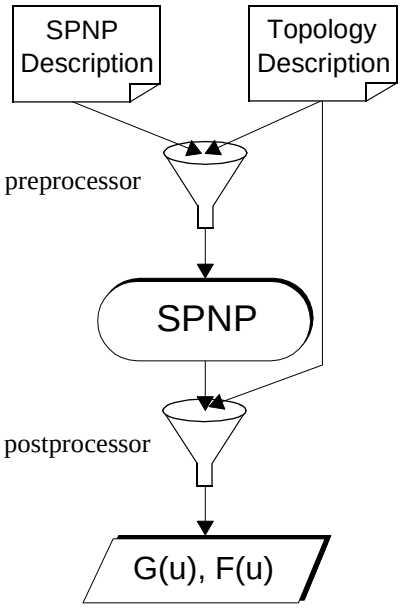


Figure 5-8: *Automatic Solution using SPNP*

solutions are shown. It can be seen that for each scenario, only the SPNP description of the SPN modules and the topology description of the underlying conferencing system is needed.

5.3.2.4 Computational Effort

Tools like SPNP enable the computation of the steady-state probabilities of the underlying Markov chain by determining the generator matrix *Q* and solving equation (5.21). This section outlines the estimated computational effort to perform this computation and furthermore to determine the performance measures *G(u)* and *F(u)* using the proposed automatic solution of the previous section. It begins with a first rough estimation of the memory and CPU consumption, before presenting some methods for decreasing the computation effort.

The time needed for the determination of the generator matrix *Q* from the C-like Petri net specification typically ranges from several seconds to a several minutes or hours, especially for a large number of transitions. Parallel and more efficient sequential solutions can be used to reduce the computation time to a few seconds or minutes, even for large Petri Nets with millions of transitions using a Linux-based PC-Cluster.

For the numerical solution of equation (5.21), iterative linear equation solution methods like

Jacobi or *Gauss-Seidel* iterations are used. According to [Haver98b], the computational effort for these methods depends on the number of non-zero elements in *Q* if the generator matrix is stored sparsely to save memory. In the current scenario, the number of non-zero elements of Q is $2n^2-n$ (see equation (5.16)). As a consequence, the memory consumption and the number of summations and multiplications grows of order $O(n^2)$ [Haver98b]. The number of iterations for the approximative methods highly depends on the used numerical approach and the required accuracy of the solution. Typically, it might range between just a few and several thousands.

In addition to the solution of equation (5.21), the pre- and post-processing steps have to be applied to either generate the C-like notation for the Petri net or to determine *G(u)* and *F(u)* from the steady-state probabilities using the topology description. From equation (5.22), (5.23), and (5.24), it can be seen that the computational effort for the postprocessing step is of order *O(n)*. The same holds for the generation of the entire Petri net description in the preprocessing step. The postprocessing step has to be applied on a given number of topologies if using the classification approach for the tree topologies.

For the computation of a chart similar to figure 5-2, i.e. the computation of the performance gain for binary tree topologies from height 1 to 12, the following maximum effort can be estimated assuming 64 bit floating point values for the entries in the matrices:

1. up to 4096($=2^{12}$) Petri net modules have to be concatenated to the entire Petri net in the preprocessing step.
2. the size of the generator matrix grows up to ($2*4096^2-4096$)*8=256MB, and the size of the steady-state probabilities vectors (two for the Jacobi method, one for Gauss-Seidel) grows up to (4096^2+4096)*8=128MB each. Hence, the entire memory consumption is 512MB without any other data structure overhead and variables, which means that more than 512MB data are transferred from and to the main memory or to the hard disk if the main memory is not large enough.
3. more than 32 million multiplications and summations have to applied for the previous step.

The presented numbers give a rough impression of the computational effort for obtaining numerical results even in a simple group communication scenario with respect to the considered access control model and user behavior. Remember that only the application level is modeled. The effort to obtain results for a fully modeled group communication system including the system level would be even larger in terms of memory and processing power.

Refinement of Computation

The rough estimation presented above outlines figures for a brute force computation of the system without taking the structure of the system into account. Several methods can be applied to reduce the computational effort in terms of memory and CPU utilization.

The first is the more efficient sparse storage of the generator matrix *Q*. Instead of storing the exact value of each non-zero value in *Q* (it is assumed that the accuracy of the values is 64 bit), an index to a table is saved only. For each user, 3 parameters are necessary. Hence, 16 bit are enough for this index for conference sizes up to 16384 users. As a consequence, the memory consumption of the generator matrix can be reduced to $2*4096^2*2$=64MB for the 4096 user case, i.e. the memory is reduced to 25 percent. The memory consumption of the index table is 3*4096*8=96kB, i.e. very small.

Another reduction of the computational effort might be realized when considering the symmetric character of the presented Petri net. When assuming, that many of the application instance parameters λ_{i1}, λ_{i2}, and λ_{i3} are equal, methods for *lumping* [Haver98b] states on Markov

chain level can be used. Approaches like *colored Petri nets* [BaKr96] can be used to model application instances with equal parameter settings, which leads to simplified Markov chains. But for applying these methods, the assumption must be valid that many parameter sets of the applications are equal. But especially when using user-profile data as input parameters for the system, a random shift of these values is often applied for creating more ’random’ behavior in the system. This shift technique would reduce the applicability of lumping or colored Petri nets approaches.

5.3.3 Summary

This section presented the application of the SPN method for the evaluation of the proposed resource management scheme in SCCS. For a simple application scenario, the SPN and the following computations for obtaining performance measures were outlined.

Regarding the *system aspect* of the evaluation goals, results for the performance gain were not presented. It was only shown how to obtain the performance gain of the proposed scheme. For that, the determination of the application performance measures was performed manually to demonstrate the different steps. But an automatic process was also proposed to facilitate the computation using available tools with additional pre- and postprocessing units. Furthermore, figures for the computational effort were estimated to obtain the numerical results for the situation to gather similar results as with the stochastic approach.

With respect to the *methodology aspect*, it can be summarized that the used SPN method allows to obtain analytical results for the considered group communication system. However, due to the chosen modeling granularity only application-level performance measures were obtained from the model. Modeling the system level would result in an even larger model increasing the complexity of the computations. Furthermore, considering more complex scenarios in terms of access control and user behavior would additionally increase the complexity.

5.4 Evaluation with Simulation

As the last evaluation method, the simulation-based approach, presented in chapter 4.4, is applied. The scenarios considered for the evaluation are dealing with more realistic situations, which are modeled using GCDL together with the SDL behavior description approach. The applicability of the technique should be demonstrated even for complex scenarios with different roles and more sophisticated access control models in large scaled environments. It is also shown that this evaluation method provides the widest spectrum of performance measures to be obtained from the system, including measures of the application and system level.

Two different scenarios are presented, namely a *lecture* and *panel discussion*, which are more complex compared to the considered scenarios of the other evaluation methods. For each scenario, the GCDL specification and the evaluation parameters are depicted before presenting the performance measure results.

5.4.1 Scenario 1 - Lecture

The first scenario, which is presented in this section, models a *lecture* in a distributed environment. It consists of two roles, namely the *chairman* of the session and the *participants*, accessing the audio/video stream of the conference.

5.4.1.1 GCDL Specification

The GCDL definition of the *lecture* scenario is given as follows:

```
# one floor is needed to access the audio/video stream
# the other one is needed for prolongation requests
F_AV_grant   = exclusive; 1;
F_AV_prolong = exclusive; 2;
# there are two instance sets
I_chair      = (1;(0;d););
I_part       = (2;...;n;(0;d););
# everybody is allowed to read from and write to the stream
RS_all       = 1;...;n;
WS_all       = 1;...;n;
# there is an AV stream object
O_AV         = F_AV_grant; F_AV_prolong; RS_all; WS_all;
# the object is grouped to a lecture scenario
OG_lecture   = O_AV;
# now the roles are defined
R_chair      = O_AV; I_chair; ctrl_chair; data_chair;
R_part       = O_AV; I_part; ctrl_part; data_part;
```

The parameter *n* in the GCDL specification denotes the number of participants in the scenario, whereas parameter *d* represents the duration of the lecture.

For the performance evaluation of the resource management within a conferencing service, the data process of the role definition does not need to be defined. Hence, only the control processes of the chairman (`crtl_chair`) and the participants (`ctrl_part`) have to be specified to model the access control for the audiovisual stream. The access control for this scenario is very simple:

The chairman is the conductor of the conference. Hence, for each access to the audio/video stream, a floor asking request has to be sent to the chairman. If the chairman is willing to be interrupted by the current requestor, the floor `F_AV_grant` is given to the asking participant, who is then allowed to sent audio/video data to the stream. After a certain amount of time, the floor has to be given back to the chairman. But a participant may ask for prolongation of the access by asking for the floor `F_AV_prolong`. If the prolongation is granted by the chairman, the floor `F_AV_prolong` is immediately given back to the chairman. A participant may ask for several prolongations. If the prolongation is denied and the maximum asking time is over, the access floor `F_AV_grant` has to be given back to the chairman. The probability for granting the prolongation is chosen based on a uniform distribution similar to the probability for asking for prolongation by a participant.

The chairman cannot be interrupted, when answering to a previously asked question of a participant. Furthermore, the chairman might directly address a participant with a question. The different parameters for maximum answering/asking time and the different probabilities can be set in the automata (see next section).

This non-formal behavior description was defined in [Ka98] as SDL specifications for both roles. As an example, figure 5-9 shows the control process definition for the participant role in the lecture scenario. It can be seen that the usage of timers enables to model profile-based user behavior. The used SDL constructs are very simple, and even for the more complex chairman description, the SDL specification is fairly short (less than two pages).

5.4.1.2 Parameter Settings

Beside the evaluation parameters which were presented in chapter 3.5.3, a set of application parameters is provided by the SDL specification of the chairman and participant role, which

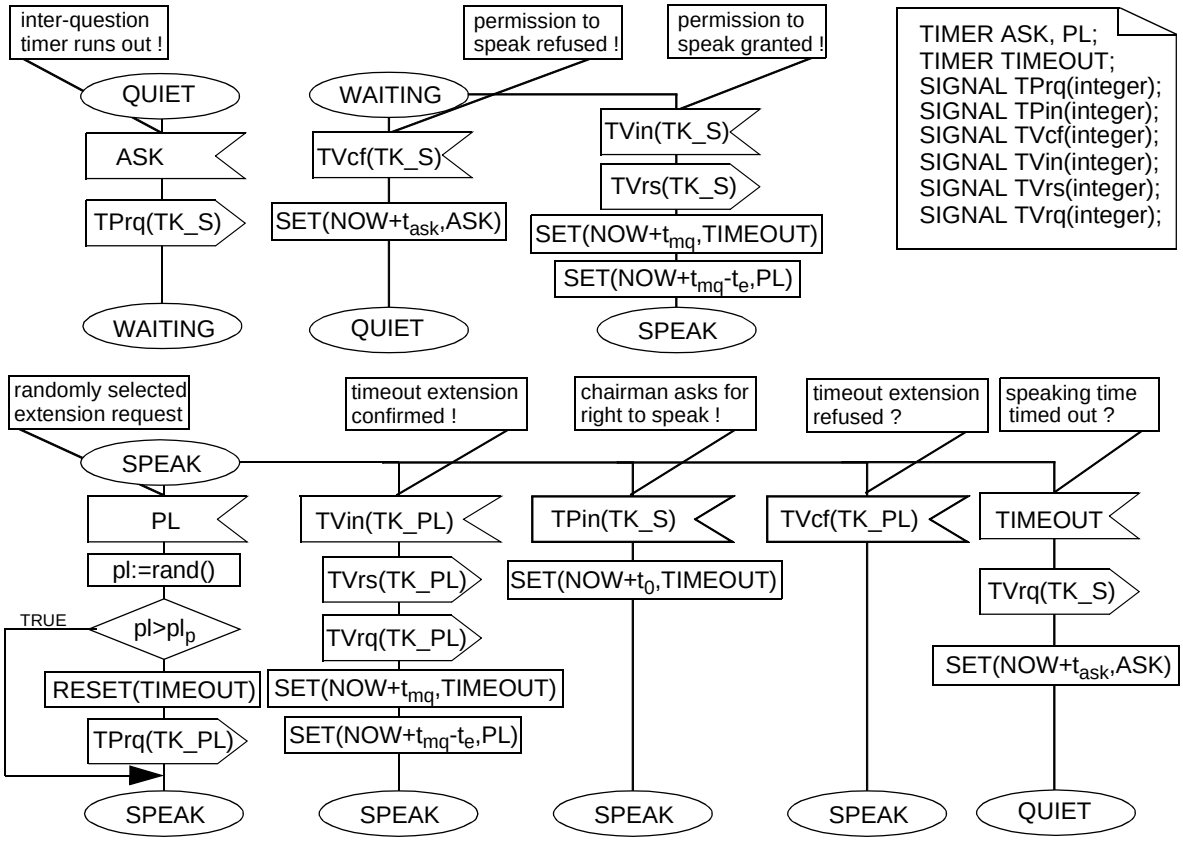


Figure 5-9: *SDL Control Process Definition of Participant Role for Scenario 1*

can be separately specified for each user.

Furthermore, the number of conferencing members is given in the GCDL specification as parameter *n*. For the presentation of performance measures in dependence on the conference size, five different values are used. Otherwise, a fixed value (bold in the table) is used.

In table 5-1, the entire application parameter set for the following simulation results is presented. For the parameters, which are specified with intervals, the following definition is given:

1. The corresponding timer values in the role specification are chosen based on a exponential distribution.
2. The expectation of the timers of each role instance is chosen randomly within the given interval using a uniform distribution during start-up of the simulation.

The prolongation parameters specify a *dice* value for the corresponding functionality. For that, a uniformly distributed random value between 0 and 1 is chosen during simulation. If the value is smaller than the appropriate prolongation parameter, the specific action is performed, e.g. a member asks for prolongation. This selection can be seen in the participant role specification of figure 5-9 (*pl:=rand()*). With this interval-based parametrization, a *random shift* of the user parameters is introduced leading to different user profiles in the scenario.

It was mentioned in chapter 3.5.3 that it is very difficult to obtain realistic user-profiles for these scenarios. Hence, the values for the user actions, like asking a question, are set to heuristic values instead of using observed values.

Parameter	Description	Value
t_{ask}	mean question interval of participant	[5-15] min
t_{mq}	maximum question length of participant	[5-15] sec
t_{ans}	mean question interval of chairman	[5-15] min
t_{msc}	maximum answer and question length of chairman	[5-15] sec
pl_p	participant probability to ask for prolongation	0.1
pl_c	chairman probability to grant prolongation	0.2
n	number of conference members	32, 64, 128, **256**, 512
d	duration of lecture scenario	90 min

Table 5-1: *Application Evaluation Parameters for Lecture Scenario*

In table 5-2, the system evaluation parameters are depicted, which are also used in the second scenario. The service times within the providers of the conferencing system are set to the values determined in chapter 2.2.4. For the link time on each provider-provider connection, a LAN connectivity is assumed [FaRu97], according to the reasoning of chapter 3.5.3. The same reasoning was considered when setting the link time of the attachment.

The topologies of the conferencing environment are generated randomly with the same start seed for the random number generator. Hence, the sequence of tree topologies is the same for both resource management schemes to guarantee a fair comparison. To each node, exactly one user is attached, except for one node (because there are 2^{x-1} nodes, but 2^x users). But note that

Parameter	Description	Value
l	link time between providers	2 ms
a_t	link time to user attachment	0.2 ms
d_l	maximum number of downlinks per node	8
s_t	service time of top provider	2 ms
s_i	service time of intermediate provider	1.8 ms
s_l	service time of leaf node provider	1.4 ms
n_r	number of simulation runs	500
n_c	number of classes for topology classification	1

Table 5-2: *System Evaluation Parameters for Lecture Scenario*

the generated trees are not binary ones, the number of downlinks per node is chosen randomly from 1 to d_l (see table 5-2). When generating the random topology, the SCCS top provider is set to provider 0 as the only fixed point in the randomly generated tree topology.

Due to the random generation of the tree topology, the *class-based classification* approach (see chapter 5.1) is used to present the performance measure results. For that, a large number of

simulation runs are performed to increase the confidence of the obtained results. However, there are only two roles in the considered scenario, the *chairman* and the *participants*. The active user set H_t in this scenario only contains the chairman leading to a constant value for the distribution metric $V(H_t)$ (equation (5.1)). Thus, there is only one class of topology to be used in this scenario.

5.4.1.3 Results

This section outlines the obtained results of the simulation scenario described above. The presentation starts with the application performance measures, before presenting the measures on system level. For all diagrams, the 95 percent confidence intervals are determined.

Application Performance Measures

In figure 5-10, the performance gain is shown in dependence on the tree height *h*. The line *total* illustrates the average performance gain for all conference members, while the other graphs show the performance gain for individual members, which are chosen randomly. The confidence intervals for the statistical results are very small due to the high number of simulation runs which were performed (less than 0.1%). Hence, they are not shown in the diagrams to improve the visibility of the graphs.

It can be seen from this picture that the average and the chairman's performance gain are very similar, while the value of *G(u)* for the participants vacillates more among the average value. However, the tendency which can be seen from all graphs is that the performance gain decreases with increasing tree height from about 19 percent down to 14 percent. This statistical

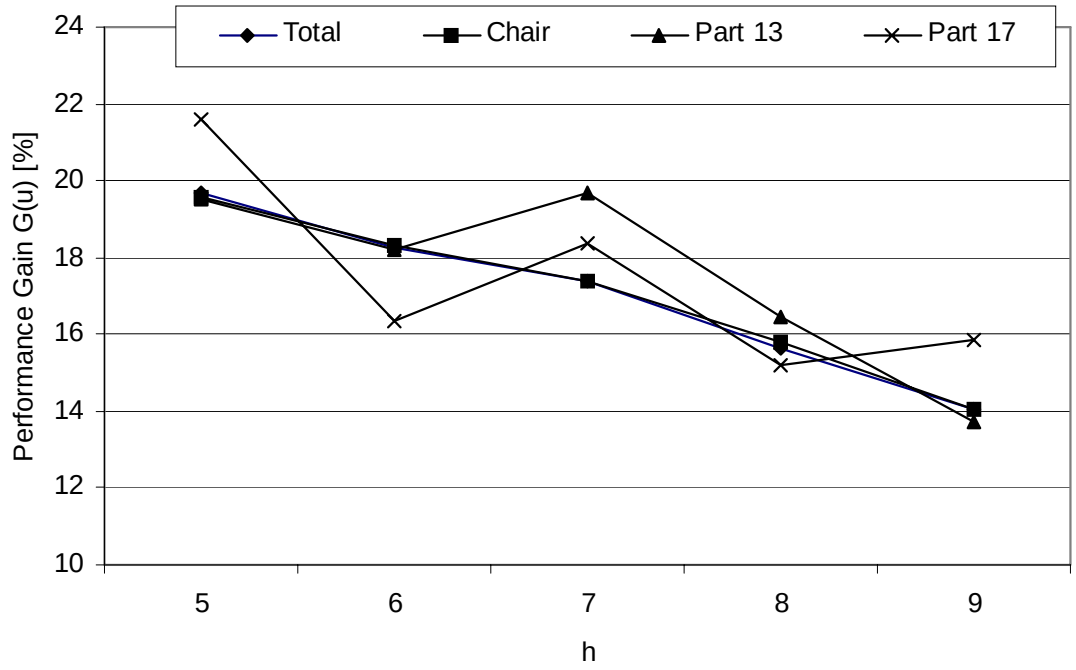


Figure 5-10: *Performance Gain Scenario 1*

result is similar to the analytical results of chapter 5.2, but for a more complex scenario in terms of user behavior and access control model.

In addition to the average performance measures, the simulation tool GCST allows to obtain *instant-of-time measures* for the response time of floor operations at specific users by defining *on-line displays* for each user. In figure 5-11, an example is shown for the current (left) and mean (right) response time for floor asking operations (`TPrq` denotes the appropriate SCC-SPDU for that operation) at user 12 which is the chairman in this scenario. The value t_s denotes

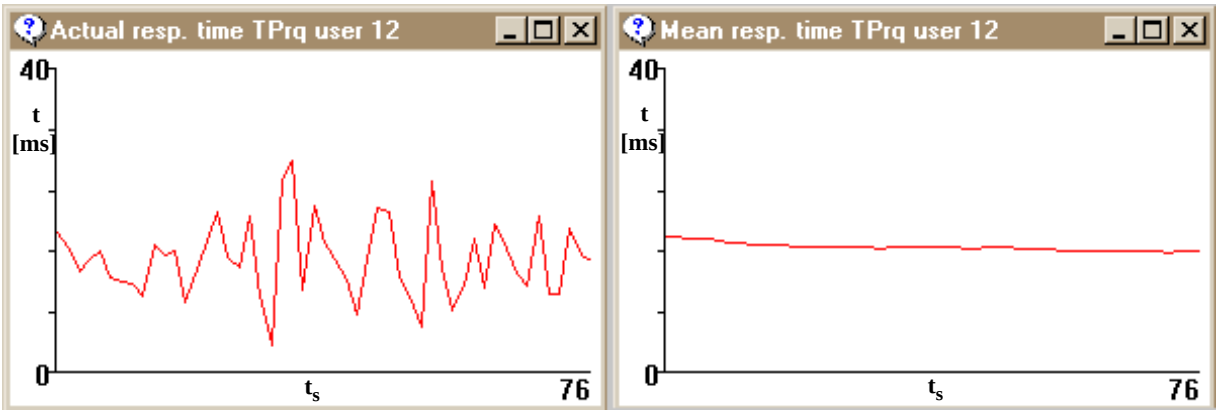


Figure 5-11: *Floor Asking History in GCST*

the time of observation of the corresponding measured value. Similar on-line displays for floor passing operations might be obtained as well.

Furthermore, the scenario implementation, which is based on the simulation framework of chapter 4.4.3, enables the integration of *monitor points* to measure the response time of an entire *user action* from the beginning of the first request to the end of the last one. However, for the current evaluation, these instant-time measures are not used.

Moreover, the simulation-based approach allows to gather further scenario-specific application measures like *number of questions* in the scenario or *mean speaking time* of individual users, either as a timeline or average statistic. But it was not in the scope of the resource management evaluation to gather these statistics.

System Performance Measures

Following the definition of the performance measures in chapter 3.5.2.2, the *number of serviced PDUs* in the conferencing system and the *queue length* of the SCCS providers are of interest, when evaluating the different resource management schemes.

For the former, the results presented in figure 5-12 were obtained from the simulations by adding *monitoring points* at each queue of the nodes in the topology. The term *reduction* of serviced PDUs per node is defined by applying the SCCS resource management scheme instead of the centralized version. The values are presented for the fixed SCCS top provider (node 0) and the chairman's node in the tree. Furthermore, the total reduction of PDUs is shown.

It can be seen from the figure that the reduction in the topmost provider is very high (more than 40 percent), and it increases slightly with higher number of users in the system, i.e. for larger tree height *h*. However, this effect might be caused by the higher confidence intervals of this graph. The total reduction of PDUs in the conference decreases with higher number of conference participants, which is consistent to the analytical results of chapter 5.2. The confidence intervals are very small (less than 0.2 percent).

For the chairman's provider, it can be stated that the number of PDUs remains almost the same,

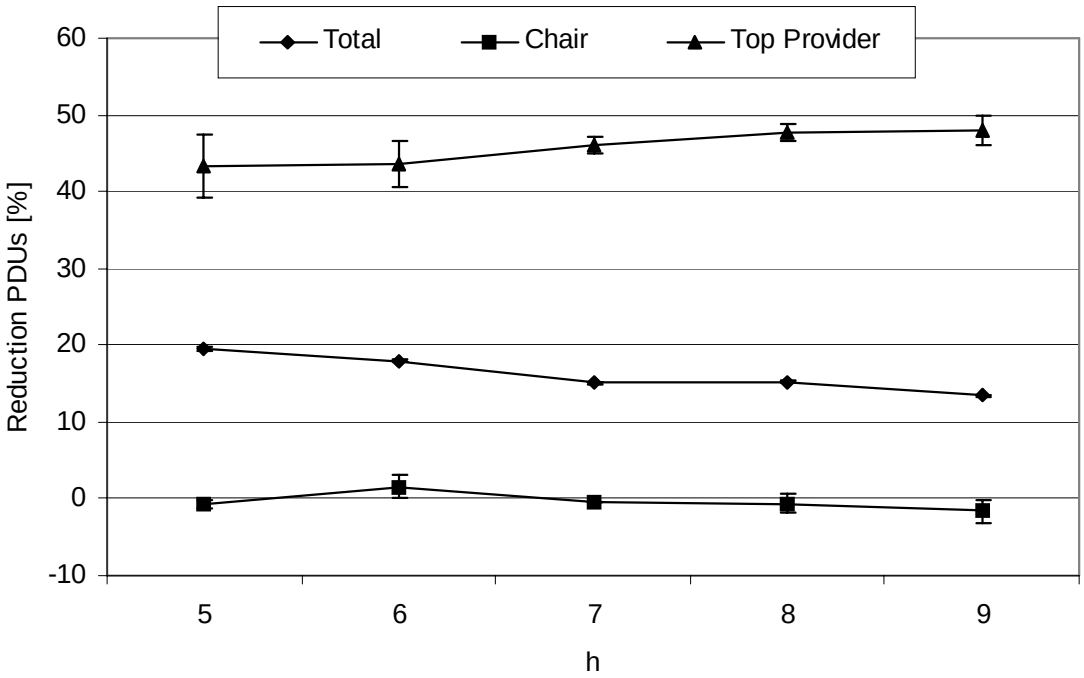


Figure 5-12: *Reduction of PDUs per Node Scenario 1*

when applying the new scheme. This can be explained by the fact, that all floor asking and passing requests are routed to the chairman anyway independent from the used routing scheme. Therefore, the number of PDUs at this node changes only slightly within the range given by the confidence intervals.

Concerning the queue length as the second performance measure on system level, the measurements showed that the queues are almost empty during the conference. This is not surprising due to the chosen parametrization of the user actions (see table 5-1) using parameters in the range of several minutes for a user action. Considering more progressive settings in terms of shorter interaction times would probably lead to higher queue lengths. These more progressive settings might be used for scenarios like shared applications where mouse movements invoking floor control requests might occur within shorter interaction times.

5.4.2 Scenario 2 - Panel Discussion

The second scenario deals with a panel discussion example adding a third role to the scenario, namely the *experts* of the panel. It is shown in this section that even for this more complex scenario in terms of user behavior and access control model, the simulation approach enables the performance evaluation of the proposed resource management scheme together with the wide spectrum of performance measures.

Similar to the first scenario, the GCDL specification is given together with a non-formal description of the *social protocol*. After that, the parameter settings for the simulations are depicted before presenting the performance measure results.

5.4.2.1 GCDL Specification

The GCDL definition of the *panel discussion* scenario is given as follows:

```
# one floor is needed to access the audio/video stream
```

```
# there is also a prolongation floor
# there is a logical floor for the experts consisting of
# m floors
F_AV_grant   = exclusive; 1;
F_AV_prolong = exclusive; 2;
F_AV_expert  = exclusive; 3;...;m-1+3;
# there are three instance sets
I_chair      = (1;(0;d););
I_panel      = (2;...;m+1;(0;d););
I_part       = (m+2;...;n;(0;d););
# everybody is allowed to read from and write to the stream
RS_all       = 1;...;n;
WS_all       = 1;...;n;
# there is an AV stream object
O_AV         = F_AV_grant; F_AV_prolong; F_AV_expert; RS_all; WS_all;
# the object is grouped to a panel scenario
OG_panel     = O_AV;
# now the roles are defined
R_chair      = O_AV; I_chair; ctrl_chair; data_chair;
R_panel      = O_AV; I_panel; ctrl_panel; data_panel;
R_part       = O_AV; I_part; ctrl_part; data_part;
```

The parameter *n* of the description above denotes the number of conference members in total, *m* defines the number of experts in the panel, and the duration of the panel discussion is given as parameter *d*.

Similar to the first scenario, only the control processes which describe the access control and user behavior are defined for the performance evaluation. In addition to the floors which are known from scenario 1, to each expert a floor is assigned, used to signal a question for the corresponding panel member. These floors are defined as a *logical* floor `F_AV_expert` (see chapter 4.3.2) to simplify the GCDL definition. The scenario is described non-formally as follows:

The *chairman* is the conductor of the conference controlling the access to the audiovisual stream object. Each *panel expert* might ask for access to the stream to issue a statement concerning a certain topic. A *participant* might ask for access as well for asking a question to the chairman or to one of the panel experts. A floor asking request for the specific expert floor (`F_AV_expert()`) indicates a question for an expert. A question to the entire panel is directed to the chairman. The asked expert (or the chairman) answers to the question. In the meanwhile, an interruption of the stream access is prohibited. The same prolongation mechanism is provided as in scenario 1. Similar to the lecture scenario, the chairman might ask one of the panel members or one of the participants a specific question.

In [Ka98], SDL specifications for the chairman, panel member, and participant roles are presented. The definition of the floors is a bit different, but the functionality of the scenario is the same. The specification for the participants is very similar to figure 5-9, with an additional panel expert handling.

5.4.2.2 Parameter Settings

A set of application parameters is provided by the SDL specifications of the different roles in the scenario. Table 5-3 shows the entire application parameter set which is used for the following simulation results. The definition of scenario 1 is used concerning the values in brackets. It can be seen that additional parameters are available for the configuration of the panel members.

Parameter	Description	Value
t_{ask}	mean question interval of participant	[5-15] min
t_{mq}	maximum question length of participant	[5-15] sec
t_{spk}	mean statement interval of panel member	[5-15] min
t_{ms}	maximum statement length of panel member	[5-15] sec
t_{ans}	mean question interval of chairman	[5-15] min
t_{msc}	maximum answer and question length of chairman	[5-15] sec
pl_p	participant probability to ask for prolongation	0.1
pl_e	panel member probability to ask for prolongation	0.1
pl_{ap}	participant probability to ask a question	0.1
pl_{ae}	panel member probability to ask a question	0.1
pl_c	chairman probability to grant prolongation	0.2
n	number of conference members	32, 64, 128, **256**, 512
m	number of panel members	6
d	duration of panel discussion scenario	90 min

Table 5-3: *Application Evaluation Parameters for Panel Discussion Scenario*

The same assumptions and settings as in scenario 1 are used for the conferencing system and the classification of the tree topology (see table 5-2). But instead of a single class for the tree topology classification, the parameter n_c is set to 15 classes. This is caused by the active user subset H_t (see equation (5.1)) which contains only a subset of all users, i.e. the chairman and the panel members.

5.4.2.3 Results

This section outlines the evaluation results divided into the presentation of the application and system level measures, including the 95 percent confidence intervals for the statistical results.

As stated in the previous section, the classification approach for the tree topologies is used in this scenario. As a starting point for the presentation of the simulation results, the classification approach is evaluated with respect to its influence on the confidence of the obtained results.

Influence of Classification Approach on Results

The main goal of introducing the classification of tree topologies was to better represent the dependence of the results on the location of the active users within the tree topology. In figure 5-13, this dependence is demonstrated for the chairman in the panel discussion. In this picture, the response time for floor asking requests serviced by the chairman is shown depending on the topology class. The constant line represents the case that there is only one class defined for all topologies. The second line shows the values for the 15 classes case, which is the chosen parameter setting in the considered scenario.

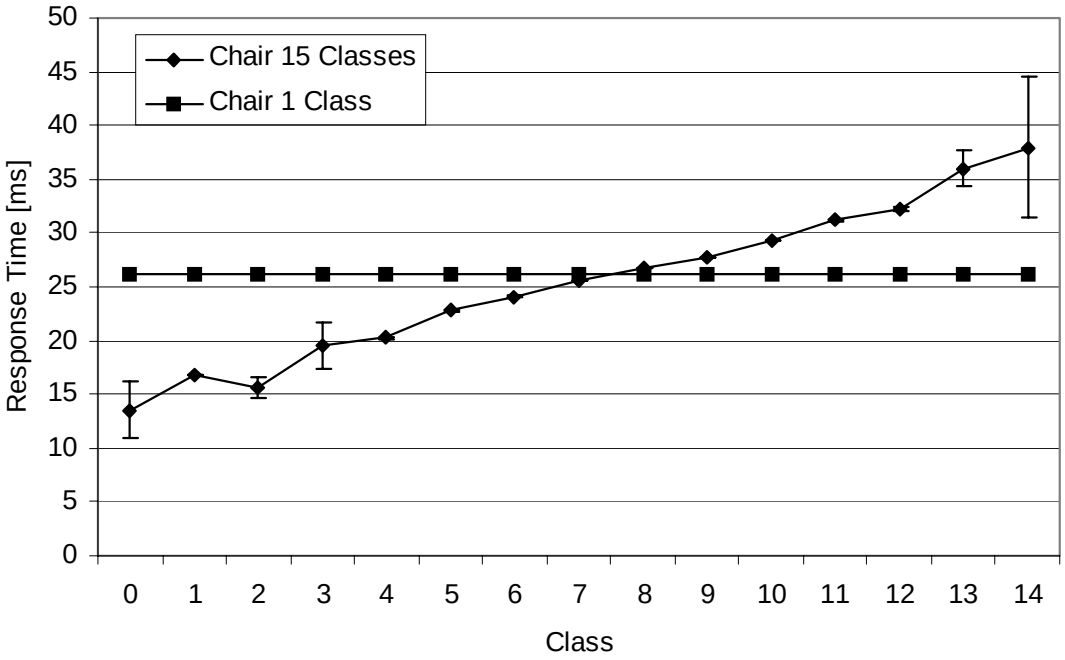


Figure 5-13: *Comparison Floor Asking Response Time for 15 and 1 Classes and 256 Users*

It can be seen from the figure that the floor control response time almost linearly increases when using the classification approach. It can also be seen that the spectrum of the values for the classification approach ranges from about 13 to 37 milliseconds. In contrary, the one class case leads to average response time in the middle of this range. Hence, it can be stated that the proposed classification approach better represents the location dependence of the proposed resource management scheme by taking the active users into account when presenting the results.

Another effect can be seen in figure 5-13. The confidence intervals for the 15 classes case are much larger at the edge of the diagram, while they are very small for the mid classes. This can also be demonstrated by depicting the standard deviation in milliseconds for the floor asking response time shown in figure 5-14 for three different role instances. The average values for the response time range from 20 to 45 milliseconds for the different classes.

Comparing the values for the different roles, it can be seen in figure 5-14 that the chairman's standard deviation values are lower compared to the expert's and participant's ones. This is caused by the higher number of floor asking requests which are generated by the chairman as the central point of the scenario. As a consequence, more response time values can be used for the determination of the mean response time in contrast to the lower active roles, expert and participant. However, it is shown for all roles that the standard deviation values at the edge of the diagram are much higher than in the mid range.

This effect can be explained by the Gaussian distribution of the metric $V(H_t)$ (see chapter 5.1.2). As a consequence from this distribution, the number of topologies in the edge classes is much smaller compared to the mid range, i.e. only a few topologies (less than 5) compared to several tens are assigned to the classes at the edge. This has a strong impact on the confidence intervals of the statistically obtained values.

With respect to figure 5-13, it can be summarized that the classification approach allows to obtain more topology-dependent results. However, figure 5-14 demonstrates the problem of minimizing the standard deviation and therefore the confidence intervals of the statistical

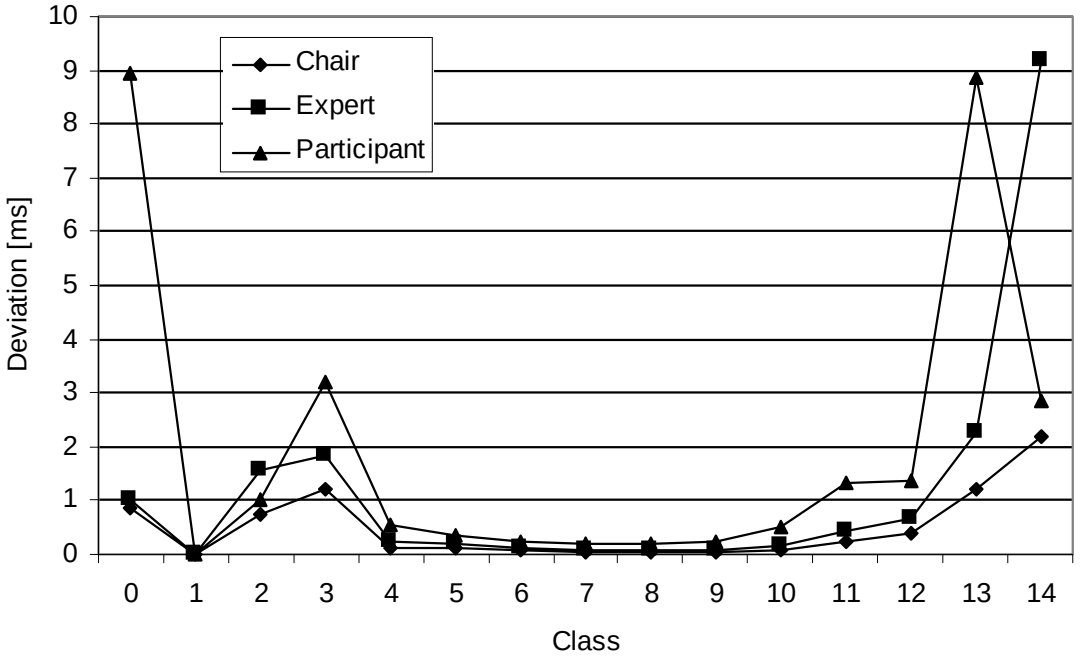


Figure 5-14: *Deviation of Floor Asking Response Time for 256 Members*

results by gathering a large number of values.

Application Performance Measures

As the most important application performance measure, the *performance gain G(u)* for the floor asking operation depending on the topology class is determined. In a first step, *G(u)* is

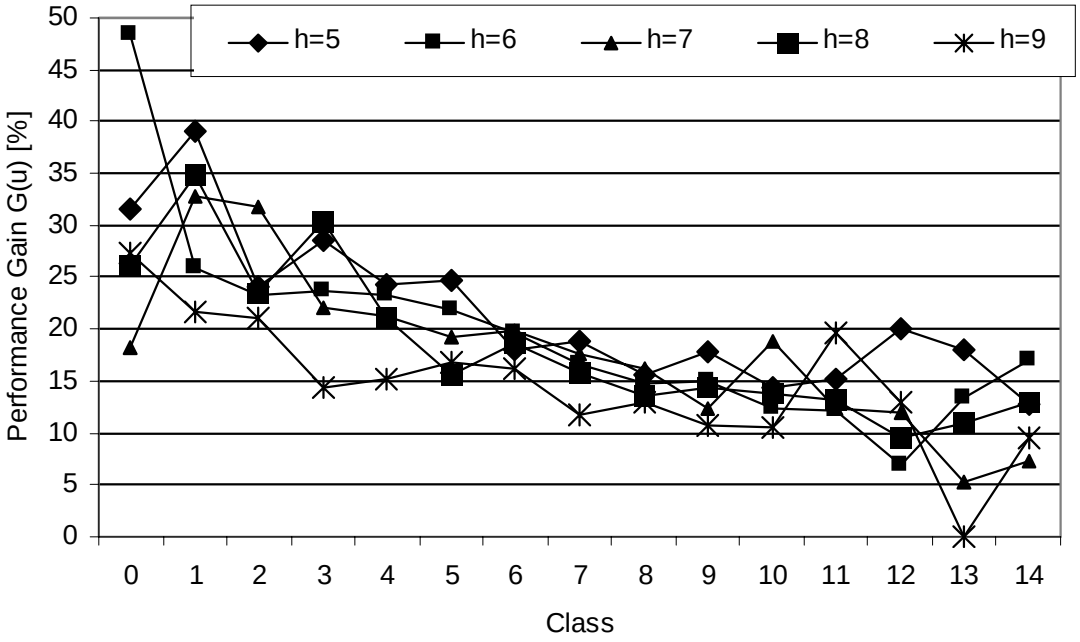


Figure 5-15: *Performance Gain G(u) Scenario 2*

determined for different conference sizes in terms of members. The second step compares the performance gain for the different roles of the scenario. For the sake of better visibility of the

results, the confidence intervals are not shown in both diagrams.

In figure 5-15, the performance gain is shown for different conference sizes in terms of members. Two tendencies can be seen from the results. The first one is the dependence of the performance gain on the class number. It can be seen that the higher the class the smaller *G(u)*. This can be explained by the definition of metric $V(H_t)$. Smaller values for $V(H_t)$ mean that the active users in the scenario, i.e. the chairman and the panel experts, are located more closely in the tree topology than for higher values. Hence, the resource management scheme of SCCS can obviously achieve a higher performance gain for closer active users.

The second tendency, seen in figure 5-15, is that the performance gain slightly depends on the number of members in the conference. While this effect is not observable at the edges of the diagram, it can be observed in the mid range of the topology classes with a smaller confidence interval (less than 0.2%). However, this dependence is not that strong compared to the results of the first scenario.

For the comparison of the performance gain which can be achieved for different roles in the conference, figure 5-16 shows the results for *G(u)* for a fixed conference size consisting of 256 members. The shown values are gathered from randomly selected role instances to illustrate the

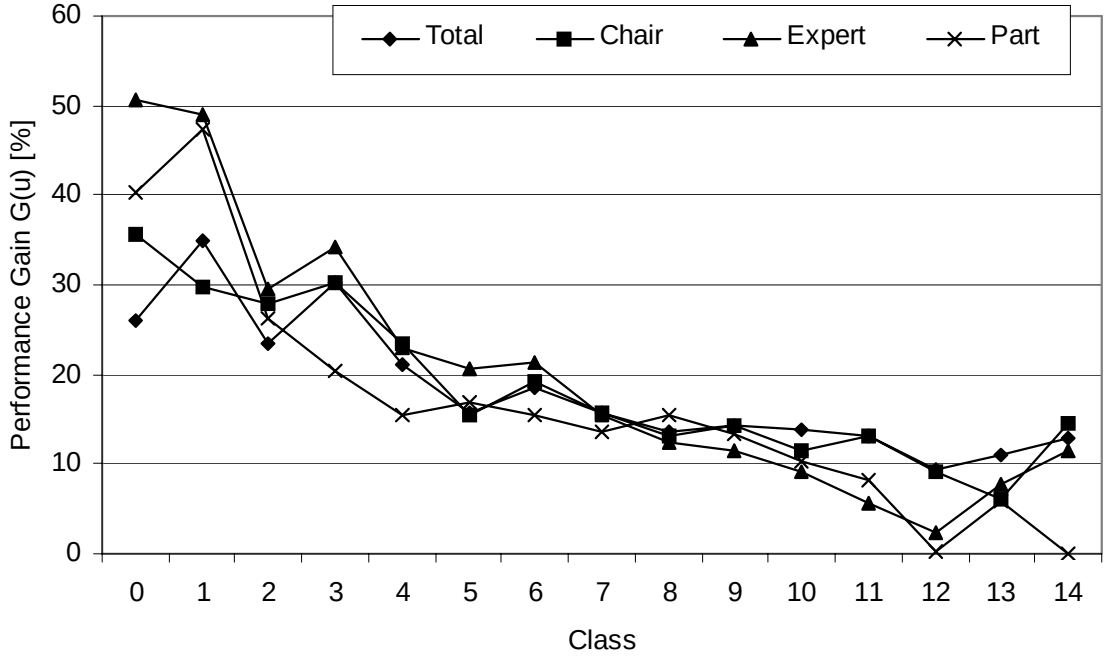


Figure 5-16: *Performance Gain G(u) Scenario 2 for 256 Members*

tendency of the values instead of averaging the values for all instances of each role. This avoids artificial average values with large standard deviations due to the different locations of the instances in the tree.

It can be seen from this picture that the performance gain values decreases for all roles for higher topology classes. This can be explained following the same reasoning as for figure 5-15 with respect to $V(H_t)$. The larger variations of the graphs for all roles at the edges of the diagram can also be explained by larger standard variations for these classes.

System Performance Measures

Similar to scenario 1, the reduction of serviced PDUs in each provider is used for depicting the

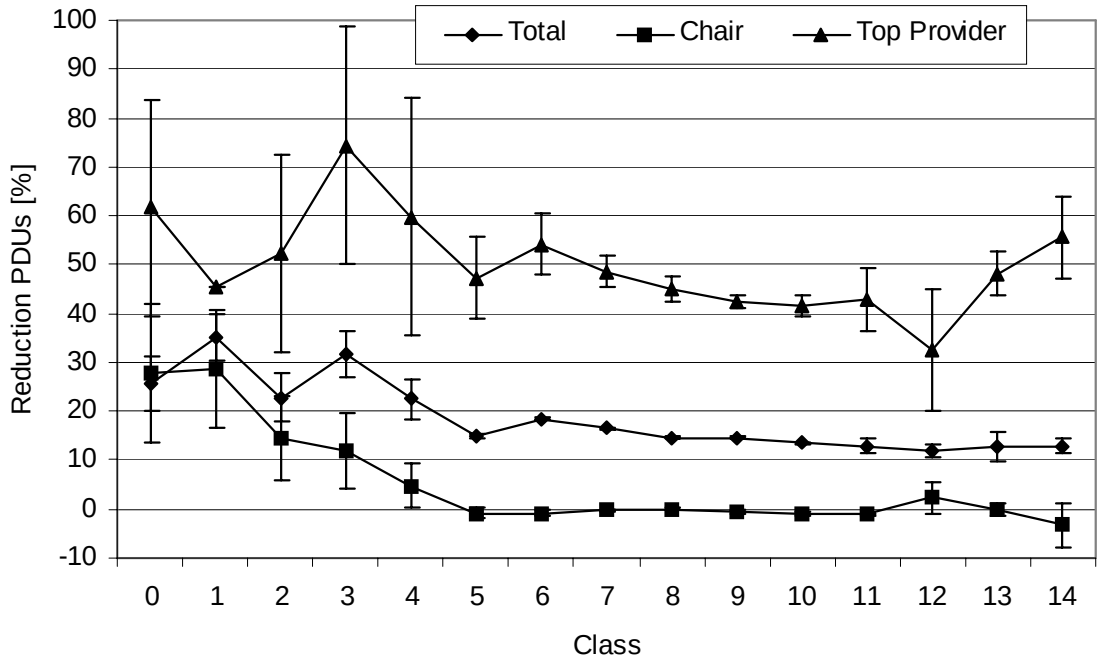


Figure 5-17: *Reduction of PDUs per Node Scenario 2 for 256 Members*

reduction of the load. In figure 5-17, the results for the chairman's node, the fixed node zero, and the total reduction is shown depending on the class of the topology. Following the reasoning concerning the influence of the classification, the large confidence intervals for the edge classes can be explained.

It can be seen that the total PDU reduction and the reduction at the top provider slightly decreases for higher topology classes. Similar to the first scenario, there is no reduction at the chairman of the panel discussion. This can be explained following the reasoning of the first scenario. Furthermore, the range of the obtained results is almost the same as in the first scenario.

Concerning the queue length within each provider, the observed results were almost zero during the simulation. At most, two PDUs were stored in a queue simultaneously. Similar to the reasoning in scenario 1, more progressive values of the application level parameters would probably lead to larger queue lengths.

5.4.2.4 Computational Effort

The computational effort for obtaining the performance measures for both scenarios is very small for a simulative method. The computation time for a single simulation run, i.e. for one topology, ranges from 0.15 seconds to approximately 1.4 seconds for a 32 node topology and a 512 node topology respectively. The values were obtained on a 500 MHz Pentium III PC running GCST (see chapter 4.4.3.3) under Windows 98. Hence for the considered settings, i.e. 500 simulation runs, the entire computation time for one conference size ranges from approximately 75 seconds to 730 seconds, i.e. in the order of several minutes. The amount of memory is less than 4 MB, which includes the graphical user interface and the topology visualization.

5.4.3 Summary

This section presented the evaluation results using the SDL behavior description within GCDL. For that, two scenarios were realized using the simulation framework of chapter 4.4.3.

The GCDL definitions, parameter settings on application and system level, and the obtained performance measures were presented for both scenarios. The first scenario simulated a large meeting based on two roles, namely the chairman and participant. In the second scenario, a third role was added leading to the most complex group communication scenario in the entire chapter in terms of user behavior and access control model.

It was demonstrated that the modeling approach provides performance measures on application and system level respectively with a moderate computational effort. Specifically, a high reduction of the floor control response time was observed when applying the proposed resource management scheme. Furthermore, the overall serviced PDUs were reduced together with a significantly lower load in the top provider caused by traversing PDUs.

5.5 Summary

In this chapter, the resource management scheme, which is used in SCCS, was evaluated compared to the centralized scheme used in the ITU T.120 standard. The purpose of this evaluation chapter was twofold. Concerning the *system aspect* of the evaluation, application and system performance measures were obtained when applying the proposed method. It was shown with the presented results, either analytically or statistically, that the SCCS resource management scheme allows to reduce the response time for floor operations like floor asking and passing. Depending on the topology and the role of the entity, this reduction ranges from several percent up to forty percent and more. Hence, a higher responsiveness of the system can be achieved even in larger scaled systems. Additionally, the simulative method showed that the load of the conferencing system may be reduced in terms of serviced PDUs per node in the underlying conferencing environment. It is worth mentioning that the *fast token give* mechanism, as proposed in chapter 3.2.14, was not applied during the evaluation. However, applying this technique would further decrease the response time of a floor asking/passing operation by 33 percent in an SCCS environment. These floor asking/passing operations are typical in conferencing scenarios.

Concerning the *methodology aspect* of the evaluation, three different modeling techniques were applied to demonstrate the computational effort as well as the applicability of the different approaches. The first method used a simple stochastic model to enable the analytical evaluation of the system. However, the obtained equations were fixed to the specific configuration in terms of topology, observed user, and access control.

The other two methods applied the GCDL modeling framework using different realizations for the behavior description. First, SPNs were used to define the behavior and the access control of the roles. For that, the module-based technique of chapter 4.5 was used to structurally design the SPN. To reduce the complexity of the concatenated SPN, only the application level was modeled, i.e. system performance measures were not provided. For demonstrating the computational effort, the generator matrix of the underlying Markov chain was determined together with the equations for the performance gain of the resource management scheme. Moreover, an automatic evaluation was proposed using available tools. A rough estimation of the effort showed that even for the simple scenario a high computational effort is necessary to obtain ana-

lytical results. This effort would be even higher, when considering more complex scenarios in terms of user behavior and access control and when modeling the system level of the conferencing environment.

The last modeling technique applied the SDL-based behavior description proposed in chapter 4.4. Two more realistic scenarios, a large meeting and a panel discussion, were considered using the simulation framework of chapter 4.4.3. It was shown that the modeling technique enables to obtain application and system performance measures statistically. Hence, this approach provides the widest spectrum of provided performance measures.

It can be summarized that the proposed SCCS resource management scheme may lead to a significant improvement of the responsiveness of a conferencing system by reducing the floor control response time. Furthermore, the load on system level might be reduced significantly. All presented performance evaluation methods allow to obtain specific performance measures either analytically or statistically for evaluating a specific mechanism of the underlying conferencing system. But with respect to the computational effort, the difference in the applicability of the methods varies widely. Compared to the SPN and simple stochastic approach, the simulative approach enables the evaluation of very complex conferencing scenarios together with a wide spectrum of obtained performance measures.

The simulative modeling technique, using the SDL-based behavior description, is also used in the following chapter to evaluate another specific SCCS mechanism, namely the dynamic reconfiguration of the tree topology.

CHAPTER 6

Reconfiguration Evaluation

The last chapter presented the evaluation of the resource management scheme in SCCS, which was compared to the centralized version of the T.120 standard. It was shown that the achieved performance gain depends on the distribution of the active entities in the tree topology.

This expected result was the motivation for proposing a dynamic reconfiguration of the tree topology during runtime of the conference. The mechanism presented in chapter 3.2.16 detects an activity pattern within a scenario and groups the appropriate participants of the conference nearby in a reconfigured tree topology using the reconfiguration service of SCCS (see chapter 3.2.15). Together with the proposed resource management, an additional reduction of the response time for further floor control requests is expected from this procedure.

In this chapter, the dynamic reconfiguration mechanism is evaluated using the simulation-based modeling technique presented in chapter 4. The purpose of this presentation is twofold. On the one hand, the *feasibility* of the evaluation using the proposed modeling technique should be demonstrated even for this complex protocol mechanism. On the other hand, the reconfiguration mechanism as such is evaluated according to three performance evaluation goals outlined in chapter 3.5.1. The first goal aims at the *applicability* in the sense that the algorithm detects active users in the scenario and groups them together in a reconfigured tree. The second goal is to outline the *improvement* which can be achieved when applying this mechanism. As the third goal, the *impact* of the reconfiguration parameters on specific performance measures is evaluated to derive setting guidelines for these parameters.

Due to the complex steps which have to be applied for the proposed dynamic reconfiguration, it seems not to be viable to model this scheme using analytical approaches like stochastic models or stochastic Petri nets. Hence, the simulation-based evaluation approach using the GCDL modeling framework is applied for the performance evaluation of the dynamic reconfiguration to obtain simulative instead of analytical results. Compared to the performance evaluation of the resource management, the scenarios which are considered in the following evaluation are more complex in terms of user behavior and access control. Furthermore, a time-dependent behavior is integrated in one of the scenarios for testing the ability of the reconfiguration scheme to adapt dynamically to the current system workload. Additionally, a worst-case scenario demonstrates that the proposed dynamic reconfiguration is not able to improve the response time for floor control requests when there are no recognizable *floor control usage patterns* in the scenario. This means that the scheme is not able to detect any dedicated activity in this sce-

nario, which underlines the importance of defining the considered floors carefully.

Similar to the resource management evaluation, the GCDL definition, the parameter settings, and the results of the simulations are presented for each scenario. Obviously, the results of the performance evaluation highly depend on the distribution of the active users in the starting tree topology. Hence, the topology classification approach, which was presented in chapter 5.1, is used for a topology-dependent presentation of the results.

6.1 Scenario 1 - Panel Discussion

The first scenario is very similar to the original panel discussion scenario of chapter 5.4.2. But for simulating a dynamic change of activity, *higher active participants* are added to the scenario. These participants dynamically change their activity within randomly selected intervals. With this behavior, the ability of the dynamic reconfiguration is tested to adapt to these dynamic changes.

Beside the determination of the performance gain which can be achieved with the default settings for the reconfiguration, this scenario is used to evaluate the impact of the reconfiguration parameters on certain performance measures, both on application and system level.

6.1.1 GCDL Specification

The GCDL specification of this scenario is very similar to the specification of the panel discussion in chapter 5.4.2:

```
# one floor is needed to access the audio/video stream
# there is also a prolongation floor
# there is a logical floor for the experts consisting of
# m floors
F_AV_grant   = exclusive; 1;
F_AV_prolong = exclusive; 2;
F_AV_expert  = exclusive; 3;...;m-1+3;
# there are four instance sets
I_chair      = (1;(0;d););
I_panel      = (2;...;m+1;(0;d););
I_ha_part    = (m+2;...;m+2+ha;(0;d););
I_part       = (m+3+ha;...;n;(0;d););
# everybody is allowed to read from and write to the stream
RS_all       = 1;...;n;
WS_all       = 1;...;n;
# there is an AV stream object
O_AV         = F_AV_grant; F_AV_prolong; F_AV_expert; RS_all; WS_all;
# the object is grouped to a panel scenario
OG_panel     = O_AV;
# now the roles are defined
R_chair      = O_AV; I_chair; ctrl_chair; data_chair;
R_panel      = O_AV; I_panel; ctrl_panel; data_panel;
R_ha_part    = O_AV; I_ha_part; ctrl_ha_part; data_ha_part;
R_part       = O_AV; I_part; ctrl_part; data_part;
```

Parameter *n* of the description denotes the number of conference members in total, *m* defines the number of experts in the panel, *ha* the number of higher active participants, and the duration of the panel discussion is given as parameter *d*.

Remember that only the control processes, describing the access control and user behavior, are defined for the performance evaluation. As mentioned above, the scenario is very similar to the panel scenario of chapter 5.4.2. The access control model is exactly the same. The roles of the higher active and the "normal" participants are defined equally. The difference between both is

that the parametrization of the higher active participant is changed dynamically by introducing a *high active interval* of length t_{ha}. During this period, the participant issues more questions than normal, i.e. the activity of this participant is increased by a given factor f_{ha}. After this period, the old parameter set is used again for the interval of length t_{na}.

In [Kl99], the revised SDL specification of the participant role is presented. The higher active and normal participants are realized with the same SDL automaton parametrized during initialization of the automaton instance.

6.1.2 Parameter Settings

Due to the similarity of the considered scenario with the panel discussion of chapter 5.4.2, the application parameter set is very similar. Table 6-1 shows the entire application parameter set

Parameter	Description	Value
t_{ask}	mean question interval of participant	[5-15] min
t_{mq}	maximum question length of participant	[5-15] sec
t_{spk}	mean statement interval of panel member	[5-15] min
t_{ms}	maximum statement length of panel member	[5-15] sec
t_{ans}	mean question interval of chairman	[5-15] min
t_{msc}	maximum answer and question length of chairman	[5-15] sec
pl_p	participant probability to ask for prolongation	0.1
pl_e	panel member probability to ask for prolongation	0.1
pl_{ap}	participants probability to ask a question	0.1
pl_{ae}	panel member probability to ask a question	0.1
pl_c	chairman probability to grant prolongation	0.2
n	number of conference members	256
m	number of panel members	6
ha	number of high active participants	30
t_{ha}	high activity interval	[5-10] min
t_{na}	normal activity interval	[30-35] min
f_{ha}	factor of high activity	8
d	duration of panel discussion scenario	90 min

Table 6-1: *Application Evaluation Parameters for Panel Discussion Scenario*

which is used for the following simulation results using the definition of chapter 5.4.2 concerning the values in brackets. It can be seen that additional parameters are available for the configuration of the high active members. But in contrast to the resource management evaluation, the considered conference size *n* is set to a fixed size of 256 users for the sake of simplicity regarding the presentation of the results.

The system evaluation parameters are almost similar to the scenarios for the resource management evaluation (see chapter 5.4). Only the value of d_l is set to 12 supported downlinks to avoid a negative impact of this parameter in the scenarios with a higher number of groups, following the reasoning given in chapter 3.2.16.12. Table 6-2 shows the system parameters again.

Parameter	Description	Value
l	link time between providers	2 ms
a_t	link time to user attachment	0.2 ms
d_l	maximum number of downlinks per node	12
s_t	service time of top provider	2 ms
s_i	service time of intermediate provider	1.8 ms
s_l	service time of leaf node provider	1.4 ms
n_r	number of simulation runs	500
n_c	number of classes for topology classification	15

Table 6-2: *System Evaluation Parameters for Panel Discussion Scenario*

The active user subset H_t (see equation (5.1)) contains only a subset of all users, i.e. the chairman and the panel members. Hence, the number of classes for the topology classification (parameter n_c) is set to 15, which was also used in the previous simulations. The higher active participants are not included in the active users subset H_t.

The last parameter set which has to be defined for the simulations specifies the default settings used for the reconfiguration parameters. In table 6-3, these values are shown which are similar to the ones presented in chapter 3.2.16.12. Remember, that the first reconfiguration is invoked after 2*t_{dr} minutes.

The last two parameter sets, i.e. for the system level and the reconfiguration algorithm, are also used for the following scenarios to allow a better comparison of the obtained results.

Parameter	Description	Value
t_{dr}	reconfiguration interval	5 min
a_b	relative level (in percentage) for active user determination	5 %
t_r	time how long inactive users remain in active user group	10 min
a_r	value for exponential averaging for inactive user adjustment	0.5
a_c	value for for tree construction	0.5
q_c	quality level for insertion of optimized tree	20 %

Table 6-3: *Default Parameters of the Dynamic Reconfiguration*

6.1.3 Results

The parameter sets presented in the last section are used to perform a simulative evaluation of the reconfiguration using the simulation tool GCST (see chapter 4.4.3.3).

The result presentation starts with the presentation of application and system level performance

measures which are obtained for the default settings of the reconfiguration parameters (see table 6-3). Furthermore, reconfiguration parameters are varied to determine the impact of specific parameter settings on certain performance measures. In the following graphs, the confidence intervals are not shown to increase the visibility of the diagrams. However, the obtained results show the same characteristics concerning the dependence of the results on the topology class, i.e. larger derivation values for the edge classes. Hence, the results are mainly used to outline the tendency and the range of the improvement which can be achieved.

6.1.3.1 Default Reconfiguration Parameter Set

Similar to the evaluation of the resource management scheme presented in chapter 5, the performance gain which can be achieved when applying the proposed dynamic reconfiguration is the main performance measure on application level. According to definition 3.1 of the performance gain, the response times of floor requests are compared when applying the dynamic reconfiguration (scheme *A* in definition 3.1) or not (scheme *B*).

In the considered scenario, there is exactly one active user group accessing the audiovisual object. The performed simulations showed that the proposed dynamic reconfiguration mechanism detects exactly this active user group and optimizes the tree topology appropriately.

Application Performance Measures

In figure 6-1, the statistical results for the performance gain are shown in dependence on the class of the start topology. Beside the values for the total performance gain, the results for different role instances are also depicted, namely for the chairman, an expert, a high active participant, and a normal participant. Similar to the results of the resource management evaluation, the graphs are only shown for specific role instances instead of averaging the values of all instances of a role following the reasoning of chapter 5.4.2.3.

The major result which can be seen from the graphs is that the proposed dynamic reconfiguration drastically improves the response time of floor requests for the active users in the scenario, while the response time for the normal participants in the scenario remains almost the same. Furthermore, it can be seen from the picture that the overall performance gain as well as the performance gain for the most active roles (chairman, expert, and high active participant) slightly increases for higher topology classes. The performance gain for the normal participant vacillates around zero percent improvement even for higher topology classes.

The increasing performance gain can be explained by the response time of the active users, when applying the dynamic reconfiguration and the results for the resource management evaluation. In figure 6-2, the response time values are depicted when performing the dynamic reconfiguration. It can be seen that the duration of the floor requests does not depend on the class of the start topology. This results from the fact that the dynamic reconfiguration rebuilds the conference topology to an optimized tree independent from the current start topology. Together with the decreasing performance gain which was observed in the evaluation of the resource management scheme (figure 5-16), the increasing performance gain is obvious for higher topology classes when applying the dynamic reconfiguration mechanism.

System Performance Measures

On system level, mainly two performance measures are of interest. The first depicts the number of PDUs needed to perform the reconfiguration of the tree topology. For that, the calculation of chapter 3.2.15.7 is used to determine an upper bound of required reconfiguration PDUs when

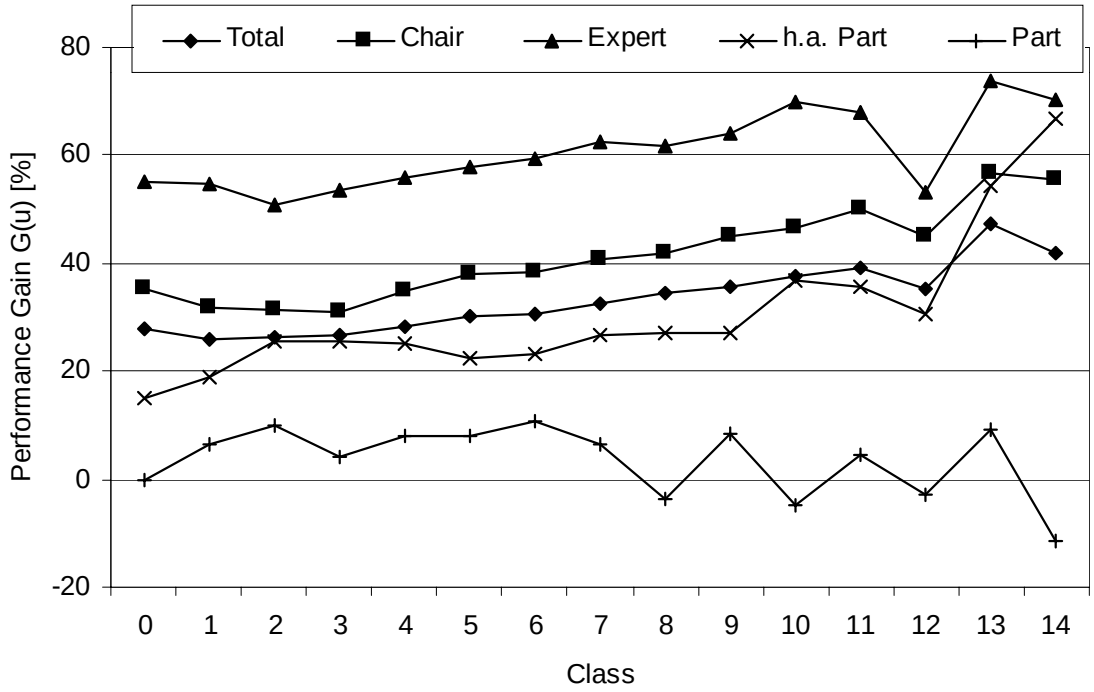


Figure 6-1: *Reconfiguration Performance Gain Scenario 1*

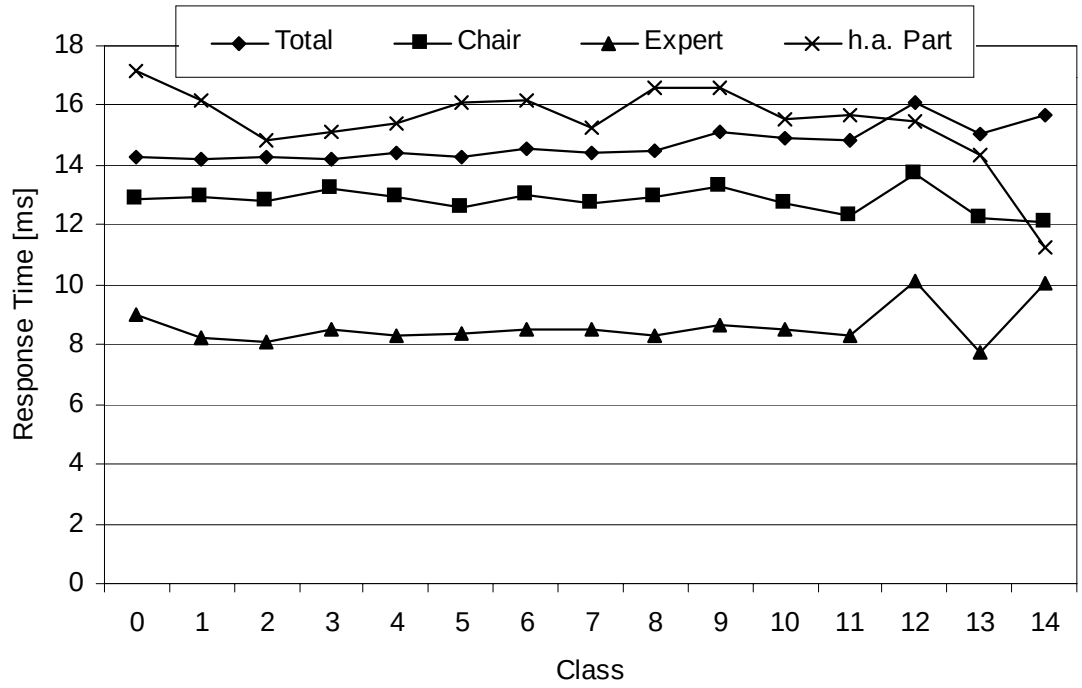


Figure 6-2: *Reconfiguration Response Time Floor Asking Scenario 1*

moving only one provider. A summation of these PDUs results in the number of required PDUs to perform the entire reconfiguration. This approach is used to avoid the implementation of the SCCS reconfiguration service in the simulation tool which would lead to an enormous overhead due to the additional PDU handling.

As the second system performance measure, the reduction of PDUs in the conferencing system is determined, when applying the reconfiguration mechanism. The required PDUs to perform

these reconfiguration operations are added to the 'normal' conferencing PDUs to allow a fair comparison of the serviced PDUs in the system.

The left graph of figure 6-3 shows the reduction of the PDUs for the entire conference as well as for the top provider. It can be seen that for increasing classes the total PDU reduction

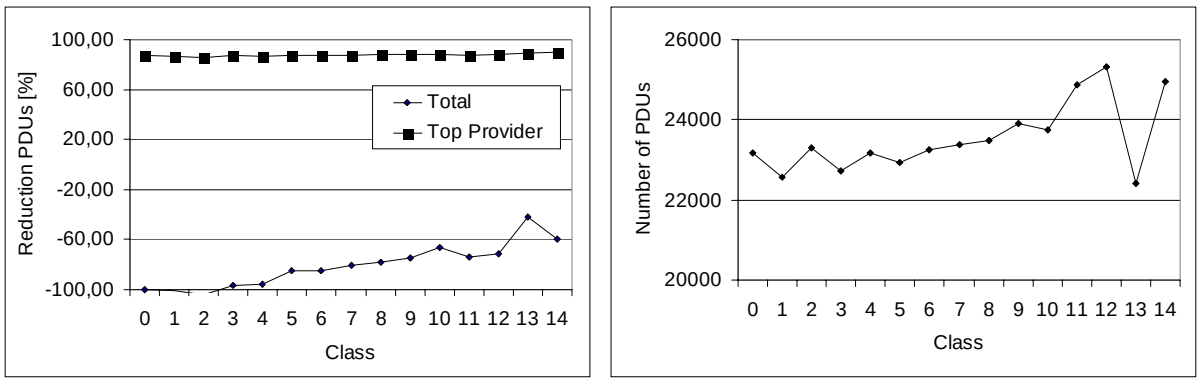


Figure 6-3: *Reduction PDUs (left) and Reconfiguration PDUs (right) Scenario 1*

increases. This is caused by the lower distribution of active users in the system for smaller topology classes (see definition of topology classes in chapter 5.1). However, it can be seen that the total amount of PDUs in the conference increases when applying the reconfiguration. This is caused by the enormous overhead needed to reconfigure the tree topology. Obviously, the total reduction of PDUs highly depends on the extent of the activity defined by the appropriate application evaluation parameters (table 6-1). The smaller the activity in the conference, the smaller the reduction in terms of PDUs due to the constant overhead of reconfiguration PDUs which are needed anyway to perform the reconfiguration of the tree topology.

For the top provider, the reduction of PDUs is very high (around 90 percent) caused by the construction of active user subtrees below the top provider. Hence, most of the traffic can be handled in this subtree, which drastically reduces the load at the top provider. Similar to the reasoning in chapter 5.4.2.3, the PDU reduction at this provider does not depend on the topology class.

The right picture in figure 6-3 shows the number of reconfiguration PDUs required to perform all reconfigurations in the entire scenario. It can be seen that this number slightly increases for higher topology classes. This is caused by the higher distribution of active users in the topology following the definition of the topology classes (see chapter 5.1). Hence, a higher number of reconfiguration operations is initially necessary to group the active users in a common subtree.

However, it is worth mentioning that the determination of the reconfiguration PDUs presented in chapter 3.2.15.7 results in an upper bound for these PDUs. Especially when moving several providers in a common subtree as it is done in the dynamic reconfiguration, this number would be far less than the presented figures resulting in a smaller number of conferencing PDUs. But for getting more exact figures for this measure, it would be necessary to implement the entire reconfiguration functionality in the simulation program.

Similar to the resource management evaluation, the simulation tool GCST enables to obtain *instant-of-time* performance measures from the scenario, e.g. the floor control response time history. With these measures, the effect can be illustrated depending on the current simulation time. Hence, the effect of the reconfiguration can be seen in these charts by a decreased response time. However, any examples for these graphs are not shown.

In addition to the presented performance measures, the simulation tool determines specific reconfiguration measures like *number of active user groups* (exactly one in the considered scenario) or *average number of active users*. These numbers are independent from the topology classes. This is obvious, because the detection of the activity does not depend on the location of the appropriate conference participants in the starting topology.

It can be summarized that the proposed reconfiguration mechanism meets the general goals defined in chapter 3.5.1, i.e. it *detects* active users in a scenario and *groups* them together in subtrees. Furthermore, the application of the algorithm leads to faster floor control requests by reducing the response time. Unfortunately, the used default reconfiguration parameters lead to a higher number of total PDUs in the conference due to the large amount of PDUs needed to perform the reconfiguration. It will be seen in the next section that adjusting specific reconfiguration parameters might reduce this overhead.

6.1.3.2 Variations of Reconfiguration Parameters

Concerning the third performance evaluation goal, the *impact* of the reconfiguration parameters on specific performance measures, the following section presents results for variations of specific reconfiguration parameters. This evaluation is performed isolated for each parameter. However, for the configuration of the reconfiguration parameter set (see table 6-3), it is of special interest how the different parameters should be combined to reach specific application-dependent goals. From the presented results in this section, setting guidelines for the reconfiguration parameters are extracted, presented in chapter 6.5.

Due to the obvious relation of topology classes and performance parameters like response time and PDU reduction (see figure 6-1 and figure 6-3), the following results are only presented for one class to simplify the presentation of the graphs. For that, a mid range class was chosen to obtain confident results according to the results in chapter 5.4.2.3. The confidence intervals for the values are very small (less than 0.2%). Hence, they are not shown in the diagrams.

Variation of Parameter t_{dr}

The first parameter to be varied is the interval length between two reconfigurations. Remember that $2*t_{dr}$ also defines the moment of the first reconfiguration. In figure 6-4, the impact of t_{dr} on

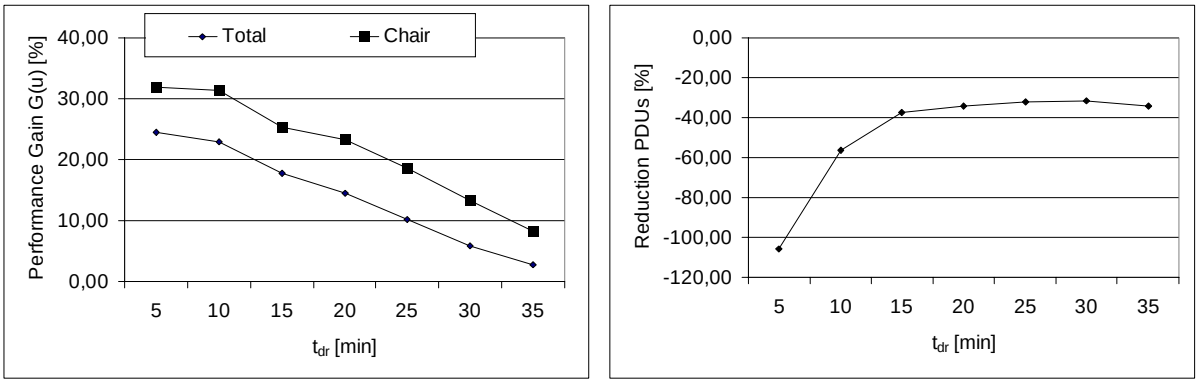


Figure 6-4: *Variation of t_{dr}: Performance Gain (left) and PDU Reduction (right)*

the performance gain *G(u)* and the reduction of serviced PDUs is presented for increasing values of t_{dr}. It can be seen that the performance gain decreases from almost 25 to 3 percent. This

is due to the fact that with increasing t_{dr} the starting point of the first reconfiguration also increases. As a consequence, the conference runs in a non-optimal configuration for a longer time period, i.e. for $2*t_{dr}$ seconds, which directly affects the overall performance gain.

In contrast to the performance gain, the PDU reduction increases for the first three values of t_{dr}. For t_{dr} larger than 15 minutes, the reduction remains almost constant. This is caused by the higher number of normal conferencing PDUs which outweighs the lower number of reconfiguration PDUs which are needed due to the lower number of performed reconfigurations.

The lower number of reconfigurations also affects the size of the active user group and the number of providers which are moved during the reconfiguration. This can be seen in figure 6-5. The presented results are obvious considering the fact that increasing t_{dr} leads to corresponding reductions of the performed reconfigurations. Using the application parameters of table 6-1, the number of performed reconfigurations is reduced from initially 16 (for t_{dr}=5 minutes) to 1 for values larger than 30 minutes.

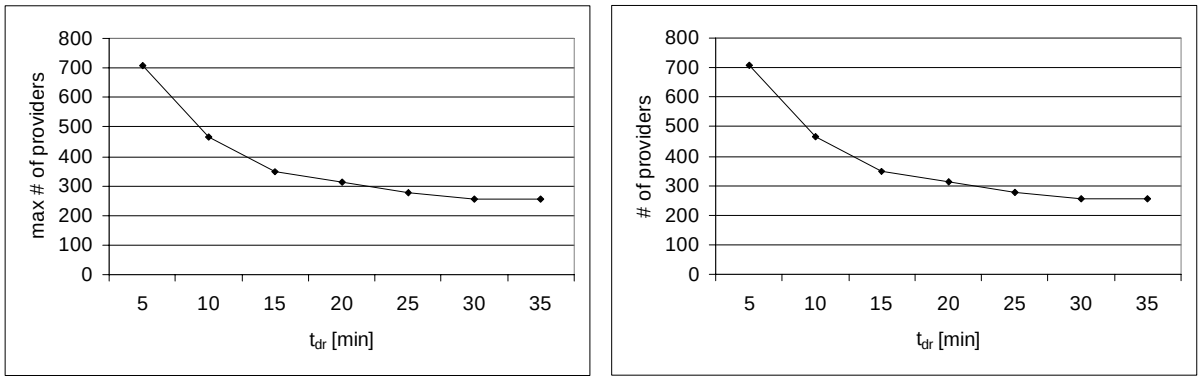


Figure 6-5: *Variation of t_{dr}: Size of Sub-Group (left) and moved Providers (right)*

As it can be seen from the definition of the parameter t_{dr}, this parameter includes two functions. On the one hand, it represents the starting point of the first reconfiguration, i.e. invoked after $2*t_{dr}$ seconds. On the other hand, the parameter defines the time at which the *i*-th reconfiguration is performed, i.e. after $2*t_{dr}+i*t_{dr}$ seconds. Hence, increasing t_{dr} to invoke less reconfigurations also shifts the starting point of the first reconfiguration. This starting point has a strong impact on the results. Before performing the first reconfiguration, the conference is running in a non-optimal start configuration. Hence, the overall performance gain is reduced because it is determined averaging all response times including this non-optimal phase. As a consequence, it is proposed to decouple the reconfiguration interval from the start point for the first reconfiguration. This can be realized by introducing another reconfiguration parameter t_{start} defining the reconfiguration start point.

This solution is evaluated in the following by setting a fixed value for t_{start} and varying the reconfiguration interval t_{dr} for further reconfigurations. For that, figure 6-6 shows the performance gain *G(u)* and the PDU reduction again, but with 10 minutes for parameter t_{start}.

It can be seen that the performance gain *G(u)* remains almost the same, while the reduction of the serviced PDUs increases with larger values for t_{dr}. This can be explained with the elimination of the variable non-optimal phase. With the fixed start point for the first reconfiguration, the time interval for finding a first reconfigured tree topology is the same for all t_{dr} settings. The reconfigured topology is more or less optimal which can be seen from the results in figure 6-6. Further reconfigurations are only used to adjust the first reconfiguration, i.e. by adding less active users (like the higher active participants) to the active user group. Obviously, this result

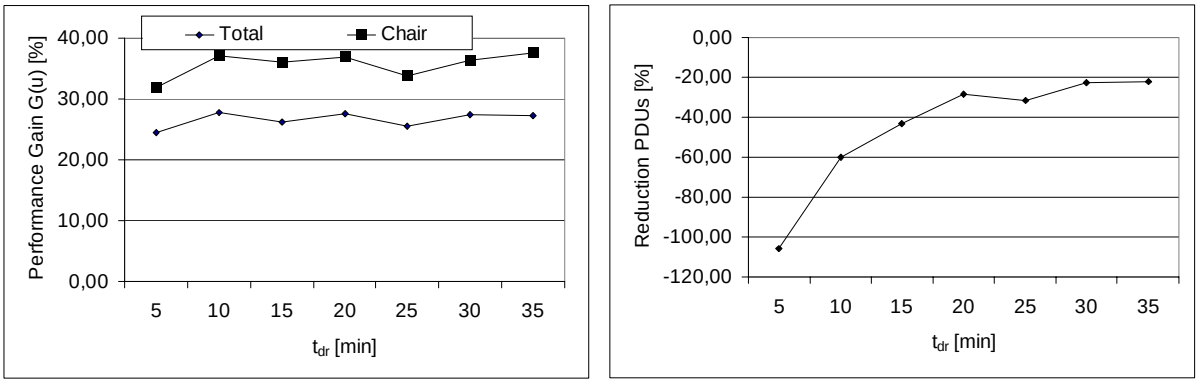


Figure 6-6: *Variation of t_{dr}: Performance Gain (left) and PDU Reduction (right) for fixed Start Point*

highly depends on the activity in the considered scenario, especially for the obtained PDU reduction. If the members of the active user group highly fluctuate, the achieved PDU reduction is expected to be much lower than in the considered scenario. However, it can be concluded that introducing a fixed start point allows to obtain an almost constant performance gain and to control the number of serviced PDUs in the conferencing system.

Variation of Parameter a_b

In the following, the parameter a_b is varied. This parameter is defined to select the users with an activity within the top a_b percent of all active users (see chapter 3.2.16.4). Hence, the parameter controls the size of the active user groups for the dynamic reconfiguration.

Following the definition of a_b, it is obvious that the size of the active user subtree and the num-

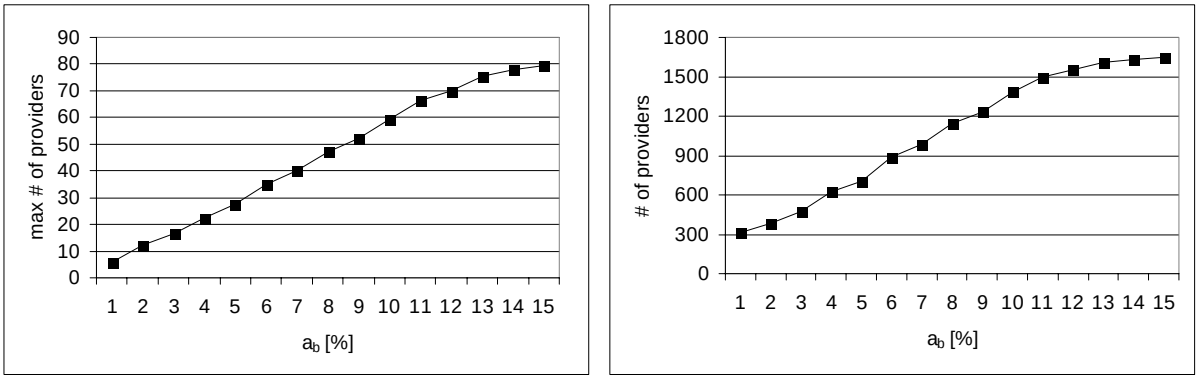


Figure 6-7: *Variation of a_b: Size of Sub-Group (left) and moved Providers (right)*

ber of moved providers for the reconfiguration increase with higher values of a_b, as it is illustrated in figure 6-7. It can be seen from the presented results that the size of the active user subtree is larger compared to the size of the subtree derived by the corresponding value for a_b. For instance, a value of 5 percent for a_b means that the 13 most active users are inserted in the active user group for a conference size of 256 members. However, according to figure 6-7, for a_b=5, an active user group consisting of 29 users is obtained. This can be explained by the addition of users who become inactive after a certain time frame. Hence, when performing the next reconfiguration, these users are not longer within the top a_b percent of active users. However, they still belong to the active user group for at most t_r minutes (see chapter 3.2.16.7).

Obviously, the variation of parameter a_b also has an effect on the achieved performance gain *G(u)* and the PDU reduction. In the left part of figure 6-8, this effect is presented. It can be seen

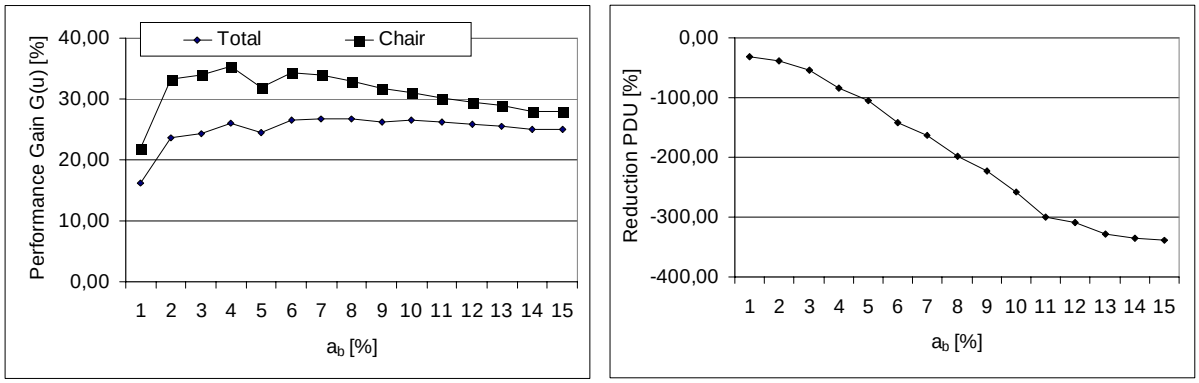


Figure 6-8: *Variation of a_b: Performance Gain (left) and PDU Reduction (right)*

that the performance gain *G(u)* first starts to increase for smaller values of a_b, while it decreases slightly for the chairman with larger values of a_b. The explanation for this result is the increasing number of users in the active user group resulting in a larger optimized subtree for this group. Due to the fact, that only the current activity measurements are used for the optimization of the active users, sporadic peak values for higher active participants might distort the long term activity pattern of another user, e.g. the chairman. Hence, considering these higher active participants might lead to a reduction of the performance gain of other users who are more active with respect to their long-term behavior.

The major drawback of increasing a_b is the enormous number of additional PDUs which are needed to perform the corresponding reconfigurations. This effect can be seen in the right chart of figure 6-8.

It can be concluded that increasing this parameter leads to an improvement of the performance gain up to a certain *break-even point*. Beyond this value, the performance gain slightly decreases. In parallel, the number of reconfiguration PDUs dramatically increases leading to a higher system load. With respect to the considered scenario, the break-even point for a_b represents an active user group containing the most active users, namely the chairman and the experts, and some of the highest active participants.

Variation of Parameter t_r

The third reconfiguration parameter to be varied is the time after which an active user is removed from the group when becoming inactive. Similar to the first variations, the achieved performance gain *G(u)* and the reduction of the serviced PDUs in the conferencing system are of interest when changing that parameter.

In figure 6-9, the results for these two performance measures are shown. It can be seen that with increasing t_r, the total performance gain *G(u)* only slightly increases, while the performance gain for the higher active participants increases significantly. The latter result is not surprising, because they become only sporadically active. Hence, these users are especially affected by the variation of t_r in the sense that with increasing t_r, more and more higher active participants remain in the active user group after being inserted in the group.

From the right graph in figure 6-9, it can also be seen that the number of PDUs in the system increases even more. This is caused by the increasing number of reconfiguration PDUs required

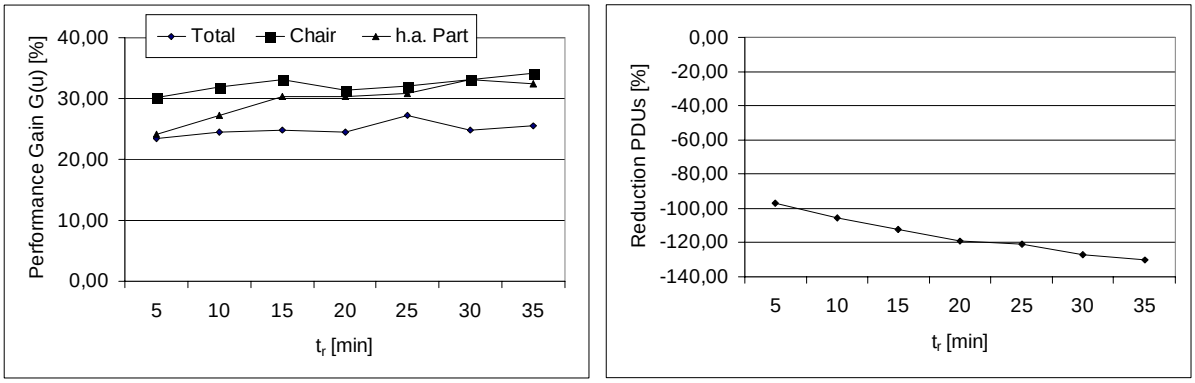


Figure 6-9: *Variation of t_r: Performance Gain (left) and PDU Reduction (right)*

for the dynamic reconfiguration. The reason for these additional PDUs compared to the default parameter setting can be seen from the graphs in figure 6-10. As explained above, more and

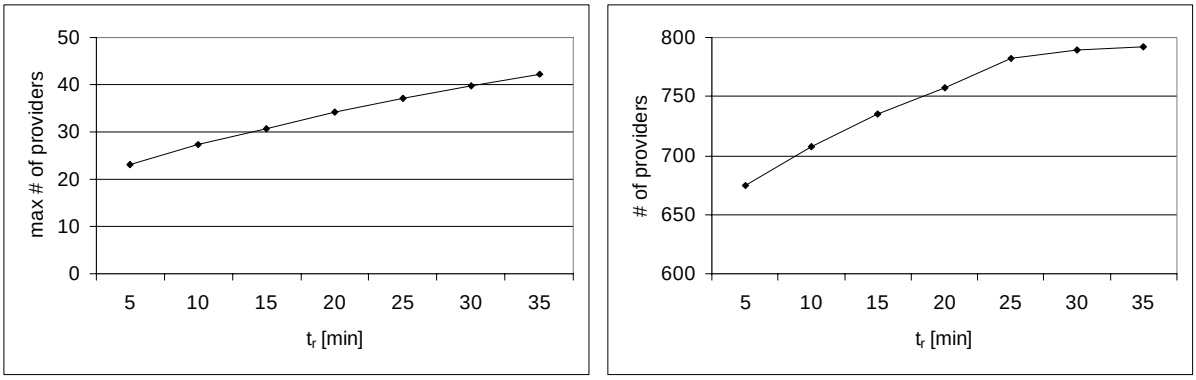


Figure 6-10: *Variation of t_r: Size of Sub-Group (left) and moved Providers (right)*

more higher active participants remain in the active user group, which is shown in the left graph of figure 6-10 illustrating the maximum number of providers in the active user group. Obviously, this leads to a higher number of moved, i.e. reconfigured, providers in the conference shown in the right part of figure 6-10.

Hence, it can be summarized that increasing the duration of inactive members in the active user group leads to an improvement for less important users, i.e. the higher active participants in the considered scenario. However, this improvement is paid by an increasing number of reconfiguration PDUs due to the higher number of members in the active user group.

Variation of Parameter q_c

The reconfiguration parameter q_c is defined to decide whether to use the optimized subtree *K* of the active user group instead of just rebuilding the current one *K'* (see chapter 3.2.16.9). If the quality difference of both subtrees is below the limit defined by q_c, the revised subtree *K'* is used instead of inserting the completely rebuilt tree *K*. Hence, this parameter can be used to control the number of reconfigurations which are performed by the proposed algorithm.

In figure 6-11, the results for the performance gain *G(u)* (left) and the reduction of the PDUs (right) are presented. It can be seen from these figures that the achieved performance gain decreases for increasing values of q_c, because more optimized subtrees *K* are not used. Instead,

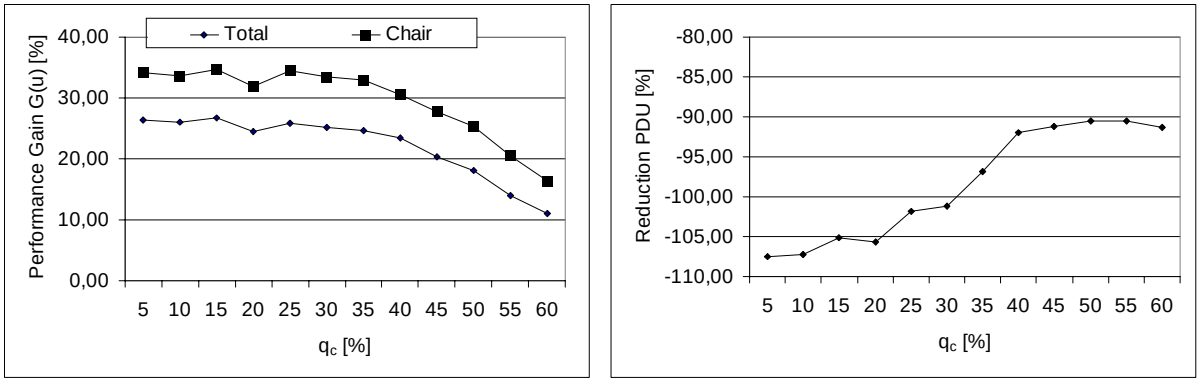


Figure 6-11: *Variation of q_c: Performance Gain (left) and PDU Reduction (right)*

only the revised subtrees *K'* are used, which are only suboptimal resulting in a decreased performance gain. In contrast to parameter t_r which especially increases the performance gain of less active user, the degradation of the performance gain affects all users in the conference when changing q_c.

The right graph of figure 6-11 shows that the number of conference PDUs also decreases with increasing q_c. However, for values above 40 percent, no further reduction of the serviced PDUs can be observed. This is caused by the higher number of 'normal' conferencing PDUs being issued due to the higher usage of revised subtrees *K'* instead of the optimal ones. In parallel, the number of reconfiguration PDUs decreases, because a smaller share of complete subtree reconfigurations is observed for a higher value for q_c. Both effects cancel out each other leading to

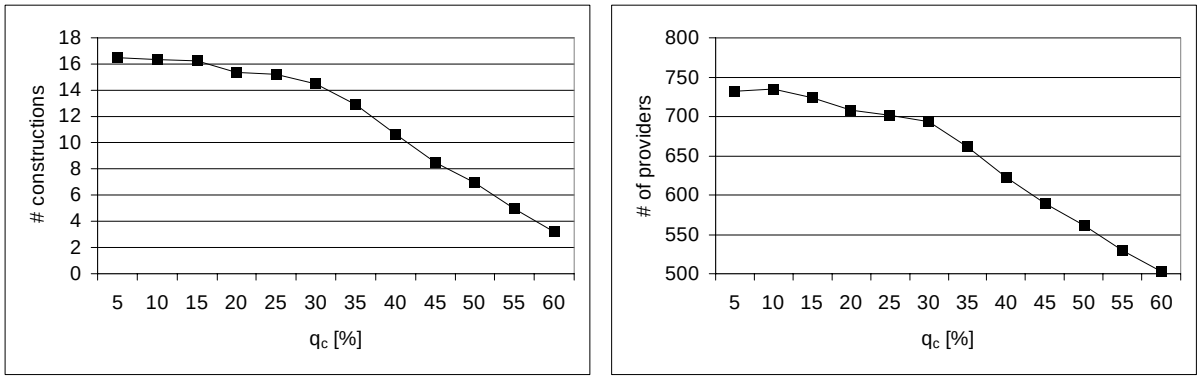


Figure 6-12: *Variation of q_c: Number of constructed Trees (left) and moved Providers (right)*

the obtained results.

The obtained PDU reduction can also be illustrated with the graphs in figure 6-12. It can be seen that the number of newly constructed trees (left graph) and therefore the number of moved providers in the conference decreases with higher values for q_c.

Considering the constant PDU reduction for q_c above 40 percent, it is worth mentioning that the reconfiguration PDUs are calculated only as an upper bound using the equations of chapter 3.2.15.7. Thus, it is expected that the number of PDUs is smaller compared to the obtained results. Especially when removing and adding active users to a subtree, cumulative reconfiguration PDUs would result in a lower number of packets to be sent.

It can be summarized that the variation of parameter q_c leads to less reconfigurations in the con-

ference. This results in smaller values for the performance gain *G(u)* and the number of reconfiguration PDUs. The reduction of the performance gain affects all users in the conference. Hence, this parameter can be used to control the load caused by the reconfiguration with a uniform reduction of the responsiveness of the conferencing system.

Variation of Parameter a_r

Parameter a_r is defined to slightly reduce the activity of a member within an active user group after becoming inactive with respect to the considered activity. This parameter is highly correlated to the parameter t_r which defines the remaining time of an inactive member of an active user group. Hence, with a given parameter t_r, the activity values of the inactive user are getting smaller when remaining in the group. As a consequence, this inactive user is set more and more at the edge of the active user subtree due to its lower activity.

In the considered scenario, the conference members who are added to an active user group and are becoming inactive after a certain time are typically higher active and normal participants. The activity of these users is normally smaller compared to users like experts or chairman. Thus, these members are set at the edge of the active user subtree, regardless whether they are active or they became inactive for a while. As a consequence, it is obvious that the impact of this parameter is not very high in terms of response time improvement, which was also shown in [Kl99]. Hence, it is recommended to use the default value for this parameter, namely 0.5.

Variation of Parameter a_c

The parameter a_c is introduced to reduce the complexity of the optimized tree determination by weighting the activity measurements with the connectivity of the corresponding node. The impact of this parameter can be easily determined by constructing trees *K* as explained in chapter 3.2.16.9 with different values for a_c and calculating the quality *Q(K)* following equation (3.15).

In [Kl99], these calculations were performed, and the results showed that the impact of this parameter is fairly small (less than 7% difference). In the current implementation of the dynamic reconfiguration within GCST, different values for a_c between 0 and 1 are used for the tree construction and the best tree with respect to *Q()* is chosen by the algorithm.

Because this tree construction is only performed in the memory of the top provider, it does not affect the number of PDUs needed to perform the reconfiguration. As a consequence, this parameter is not implemented as an application parameter in the current realization of the algorithm but as an internal on-line optimization parameter.

6.2 Scenario 2 - Separated Sessions

As an extension of the first scenario, the second one adds a dynamic aspect to the scenario. This is realized by dynamically changing the scope of some participants based on a defined timeline. This behavior is characterized with the notion of *separated sessions*, because some of the participants in the panel discussion join a sub-conference during runtime of the conference. Within these sub-sessions, a dedicated activity, e.g. a discussion or a shared application, is started using specific floors controlling these activities by accessing specific objects in these sub-sessions. This behavior results in a changed *activity pattern* of the participating members of these sub-sessions. For the sake of simplicity of the scenario description, the notion of *higher active participants* is removed from the scenario.

With this scenario, the ability of the dynamic reconfiguration is tested to detect these different activity patterns caused by the access to separated objects. It is expected from the algorithm that the different members are sorted in different *active user groups* (see chapter 3.2.16.6), which are handled appropriately. Moreover, the scenario is used to consider the impact of specific reconfiguration parameters on the achieved performance gain and the active user group determination.

6.2.1 GCDL Specification

Due to the timed behavior for creating the separated sessions, the GCDL specification of this scenario is more complex compared to the specification of the plain panel discussion:

```
# one floor is needed to access the main audio/video stream
# there is also a prolongation floor and several floors for
# the panel members similar to the plain panel discussion
# Furthermore, there are three additional floors for the sub-sessions
F_AV_grant   = exclusive; 1;
F_AV_prolong = exclusive; 2;
F_AV_expert  = exclusive; 3;...;m-1+3;
F_AV_ss1     = exclusive; m+3;
F_AV_ss2     = exclusive; m+3+1;
F_AV_ss3     = exclusive; m+3+2;
# there are several instance sets with time values for the timed behavior
I_chair      = (1;(0;d));
I_panel      = (2;...;m+1;(0;d));
I_part       = (m+2;...;n;(0;d));
I_part_ss1   = (s;...;s+s1;(tss1,tss2));
I_part_ss2   = (s+s1+1;...;s+s2;(tss1,tss2));
I_part_ss3   = (s+s2+1;...;s+s3;(tss1,tss2));
I_part_ss4   = (s;...;s+s2;(tss2,tss3));
I_part_ss5   = (s+s2+1;...;s+s3;(tss2,tss3));
I_part_ss6   = (s;...;s+s1;(tss3,tss4));
I_part_ss7   = (s+s1+1;...;s+s2;(tss3,tss4));
I_part_ss8   = (s+s2+1;...;s+s3;(tss3,tss4));
# there are different access sets for the different streams
RS_all       = 1;...;n;
WS_all       = 1;...;n;
RS_ss1       = s;...;s+s1;
WS_ss1       = s;...;s+s1;
RS_ss2       = s+s1+1;...;s+s2;
WS_ss2       = s+s1+1;...;s+s2;
RS_ss3       = s+s2+1;...;s+s3;
WS_ss3       = s+s2+1;...;s+s3;
RS_ss4       = s;...;s+s2;
WS_ss4       = s;...;s+s2;
RS_ss5       = s+s2+1;...;s+s3;
WS_ss5       = s+s2+1;...;s+s3;
# there are different AV stream objects
O_AV         = F_AV_grant; F_AV_prolong; F_AV_expert; RS_all; WS_all;
O_AV_ss1     = F_AV_ss1; RS_ss1; WS_ss1;
O_AV_ss2     = F_AV_ss2; RS_ss2; WS_ss2;
O_AV_ss3     = F_AV_ss3; RS_ss3; WS_ss3;
O_AV_ss4     = F_AV_ss1; RS_ss4; WS_ss4;
O_AV_ss5     = F_AV_ss3; RS_ss5; WS_ss5;
# the objects are grouped to the entire panel scenario
OG_panel     = O_AV; O_AV_ss1; O_AV_ss2; O_AV_ss3; O_AV_ss4; O_AV_ss5;
# now the roles are defined
R_chair      = O_AV; I_chair; ctrl_chair; data_chair;
R_panel      = O_AV; I_panel; ctrl_panel; data_panel;
R_part       = O_AV; I_part; ctrl_part; data_part;
R_part_ss1   = O_AV_ss1; I_part_ss1; ctrl_part; data_part;
R_part_ss2   = O_AV_ss2; I_part_ss2; ctrl_part; data_part;
R_part_ss3   = O_AV_ss3; I_part_ss3; ctrl_part; data_part;
R_part_ss4   = O_AV_ss4; I_part_ss4; ctrl_part; data_part;
R_part_ss5   = O_AV_ss5; I_part_ss5; ctrl_part; data_part;
```

```
R_part_ss6   = O_AV_ss1; I_part_ss6; ctrl_part; data_part;
R_part_ss7   = O_AV_ss2; I_part_ss7; ctrl_part; data_part;
R_part_ss8   = O_AV_ss3; I_part_ss8; ctrl_part; data_part;
```

Parameter *n* of the description above denotes the number of conference members in total, *m* defines the number of experts in the panel, and the duration of the panel discussion is given as parameter *d*. Furthermore, the parameters s_1, s_2, s_3 denote the number of participants in the different separated sessions, and the time values t_{ss1},...,t_{ss4} define the different starting points of the corresponding actions (see explanation below).

Similar to the first scenario, only the control processes which describe the access control and user behavior are defined for the performance evaluation. In addition to the plain panel discussion, a *timed action* behavior is added to the scenario illustrated in figure 6-13. The scenario

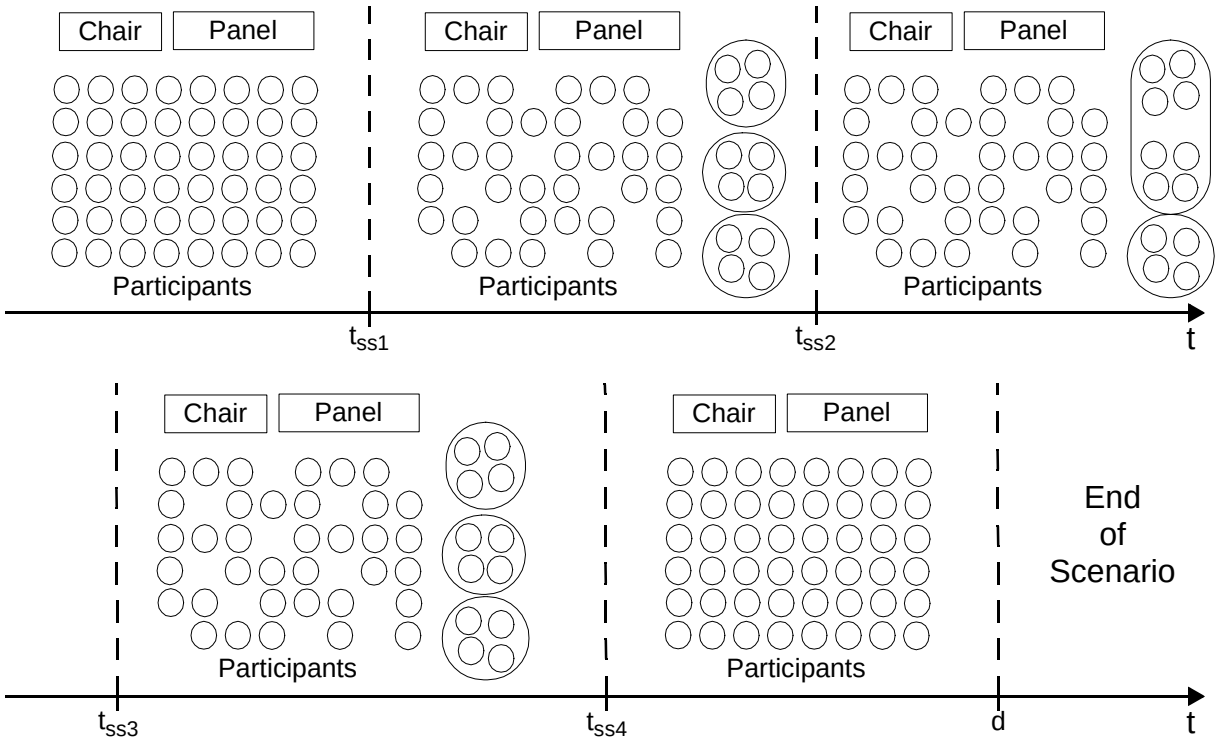


Figure 6-13: *Timed Action Behavior Scenario 2*

starts similar to the panel discussion scenario with three different instance sets, namely for the chairman, the experts, and the participants accessing the main audiovisual stream (`O_AV`). After t_{ss1} minutes, three sub-sessions are created (instance sets `I_part_ss1, I_part_ss2, I_part_ss3`) by defined participants simulating a private channel discussion between these conference members. After t_{ss2} minutes, two sub-sessions are merged into one (instance sets `I_part_ss4, I_part_ss5`), while after t_{ss3} minutes the three sub-sessions are restored again containing the same members as before (instance sets `I_part_ss6, I_part_ss7, I_part_ss8`). The sub-sessions are released after t_{ss4} minutes, and the normal panel discussion continues until it ends after *d* minutes.

Within each sub-session, the appropriate participants access an own audiovisual stream (`O_AV_ss1` to `O_AV_ss5`) which is controlled using a simple floor for access control. When three sub-sessions are created, the participants of each sub-session generate requests with parameter q_{ss1}, while in the two sub-session case, requests are generated every q_{ss2} minutes.

Despite the different instance sets for the timed behavior, the same participant role specifica-

tion (`ctrl_part` in the GCDL specification) is used, presented in [Kl99].

It can be seen from the specification above that the GCDL modeling approach allows to specify even more complex scenarios in terms of time-dependent behavior.

6.2.2 Parameter Settings

In addition to the application parameters which were defined for the panel discussion in chapter 5.4.2.2, several parameters have to be specified for the definition of the sub-sessions and the starting points of the timed actions. Table 6-4 shows the entire application parameter set which

Parameter	Description	Value
t_{ask}	mean question interval of participant	[5-15] min
t_{mq}	maximum question length of participant	[5-15] sec
t_{spk}	mean statement interval of panel member	[5-15] min
t_{ms}	maximum statement length of panel member	[5-15] sec
t_{ans}	mean question interval of chairman	[5-15] min
t_{msc}	maximum answer and question length of chairman	[5-15] sec
pl_p	participant probability to ask for prolongation	0.1
pl_e	panel member probability to ask for prolongation	0.1
pl_{ap}	participant probability to ask a question	0.1
pl_{ae}	panel member probability to ask a question	0.1
pl_c	chairman probability to grant prolongation	0.2
n	number of conference members	256
m	number of panel members	6
s	user identifier of first sub-session member	50
s_1	number of participants in first sub-session	10
s_2	number of participants in second sub-session	10
s_3	number of participants in third sub-session	10
t_{ss1}	start of sub-session creation	30 min
t_{ss2}	start of merging two sub-sessions	60 min
t_{ss3}	time when restoring old sub-sessions	90 min
t_{ss4}	time when releasing sub-session	120 min
q_{ss1}	question interval when three sub-sessions are used	[3-6] min
q_{ss2}	question interval when two sub-sessions are used	[2-4] min
d	duration of panel discussion scenario	150 min

Table 6-4: *Application Evaluation Parameters for Panel Discussion Scenario*

is used for performing the evaluation using the definition of chapter 5.4.2 concerning the values in brackets. It can be seen that additional parameters are available for the configuration of the timed action behavior.

For the parameters on system level and for the default settings of the dynamic reconfiguration, the parameter sets of the first scenario (see chapter 6.1.2) are chosen.

6.2.3 Results

Similar to the first scenario, the settings of the last section are used to perform an evaluation of the dynamic reconfiguration mechanisms. The results of this evaluation are shown in the following. According to the reasoning of scenario 1, the confidence intervals are not shown for the default reconfiguration set results to improve the visibility of the diagrams.

6.2.3.1 Default Reconfiguration Parameter Set

The obtained performance measures for this scenario are similar to the ones gathered for the first presented in chapter 6.1.3. Hence, the achieved performance gain is shown in figure 6-14

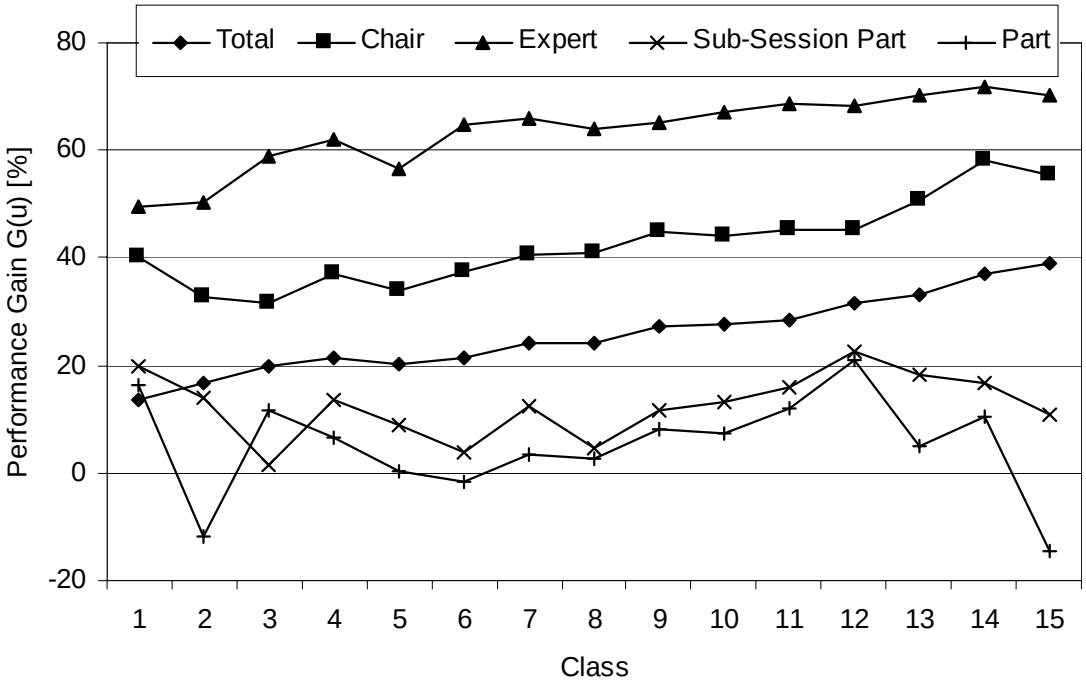


Figure 6-14: *Performance Gain G(u) Scenario 2*

for some role instances, namely the chairman, an expert, a sub-session participant, and a normal participant. It can be seen that the performance gain for the active users (chairman and expert) increases for higher topology classes similar to the results of the first scenario. The performance gain for a normal participant, who is normally not a member of an active user group, vacillates around zero percent independent from the topology class. But it can also be seen from the picture that the performance gain for the sub-session participant is slightly higher. This can be explained by the special handling of these participants due to their dedicated activity. The dynamic reconfiguration algorithm detects the (time-dependent) activity of the sub-session participants and group them temporarily in a separate subtree. Hence, all requests con-

cerning the sub-session activity are improved. However, the requests for the normal conference activity are not improved, because the sub-session participants are not high active concerning this activity. As a result, the overall performance gain is only slightly higher compared to a normal participant.

Figure 6-15 (left) shows the reduction of conferencing PDUs for the entire conference as well

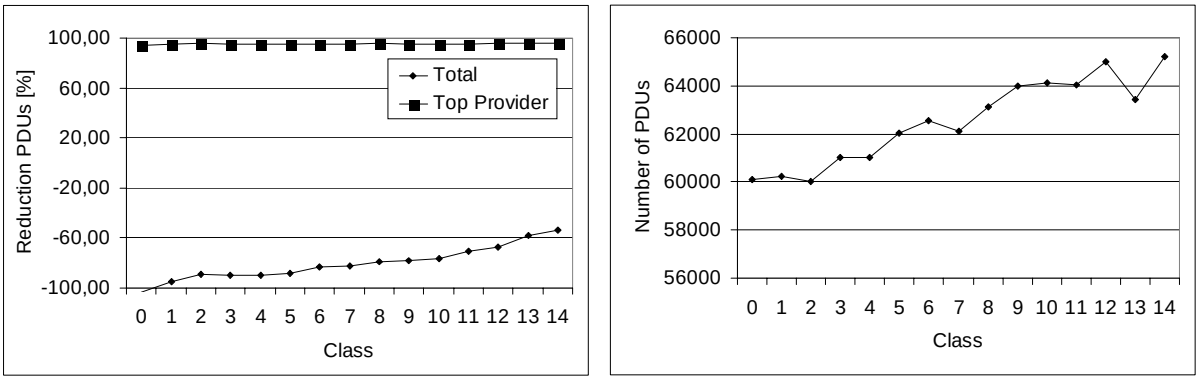


Figure 6-15: *Reduction PDUs (left) and Reconfiguration PDUs (right) Scenario 2*

as for the fixed top provider. It can be seen that the reduction for the top provider is higher than in scenario 1. However, the shape of the total reduction curve is very similar to the first scenario. The differences to the first scenario can be explained by the higher number of active users taken into account for the reconfiguration.

The negative impact of this larger active user group can be illustrated by the right graph of figure 6-15. It can be seen that the shape of the curve is similar to the first scenario. However, the number of reconfiguration PDUs is higher being caused by the higher number of users to be moved during the reconfiguration.

It can be concluded from these results that even for this more complex time-dependent scenario, the proposed dynamic reconfiguration scheme is able to detect users being active with respect to certain objects. Grouping these users nearby in the tree topology leads to a reduction of the response time for floor control requests. Hence, the responsiveness for the active user group in the conference is increased.

Moreover, the additional reconfiguration performance measures which are gathered by the simulation tool show that the number of active user groups follows the definition of the timed action behavior of the scenario. Hence, with a specific delay, the algorithm builds active user groups with respect to the scenario timeline. This delay obviously depends on the parameter t_{dr}, the interval for performing reconfigurations, and the correlation between the scenario timestamps t_{ss1}, ..., t_{ss4}.

6.2.3.2 Variations of Reconfiguration Parameters

Similar to the first scenario, variations of specific reconfiguration parameters can be applied to determine the impact on specific application and system performance measures. Especially the timed action behavior of the considered scenario is detected by the appropriate parameter variations. For instance, the variation of parameter a_b allows to include more sub-session participants in the active user groups, while changing parameter t_r leads to fewer changes of the active users after they have initially been included in the appropriate subgroup.

However, the obtained results for this scenario are generally very similar to the ones gathered in the first scenario. Hence, the graphs of these variations are not shown.

6.3 Scenario 3 - Worst Case Scenario

In the last scenario of this chapter, a worst case for the dynamic reconfiguration mechanism is demonstrated. As presented in chapter 3.2.16.1, the application has to specify the floors which are considered for optimization by the dynamic reconfiguration. A wise configuration of these floors is crucial for the success of the mechanism. In the previous scenarios, the floors for controlling the access to the audiovisual streams indicated a specific *activity pattern* which was detected by the dynamic reconfiguration for grouping the *active users* in certain *activity groups*. If this activity pattern is not recognizable, the mechanism of the dynamic reconfiguration should fail.

With the following scenario, this missing activity pattern is simulated by uniformly distributing the access activity among all participants of the conference. Hence, there is no dedicated subset of active users which can be grouped together for optimizing their response time with respect to the floor control requests. As a consequence, this worst case scenario underlines the importance to carefully define the floors which are considered for reconfiguration.

6.3.1 GCDL Specification

The GCDL specification for the worst-case scenario is very simple:

```
# there are f floors indicating any kind of access
F_grant      = exclusive; 1,...,f;
# there is only one instance set
I_part       = (1;...;n;(0;d););
# everybody is allowed to read from and write to the object
RS_all       = 1;...;n;
WS_all       = 1;...;n;
# there is a single object only
O_Object     = F_grant; RS_all; WS_all;
# the object is grouped to a scenario
OG_generic   = O_Object;
# now the role is defined
R_part       = O_Object; I_part; ctrl_part; data_part;
```

Similar to the other scenarios, parameter *n* denotes the number of conference members, *f* defines the number of used floors (specified as a *logical* floor as defined in chapter 4.3.2.4), and the duration of the scenario is given as parameter *d*.

The control process `ctrl_part` is rather simple. Each conference member asks for a randomly selected floor after a defined interval t_{ask}. This interval varies by a randomly selected offset t_{diff} for adding a random shift to the values. If a floor holder is asked for the floor, it is given immediately to the asking participant. It can be seen from this simple access control model that the activity in terms of floor control is uniformly distributed among all participants.

The SDL specification of the participant role is presented in [Kl99] for further information.

6.3.2 Parameter Settings

Due to the fact, that there is only one role in the scenario with a very simple access control

model, only a few application parameters have to be specified for performing the evaluation. In

Parameter	Description	Value
t_{ask}	mean access interval of participant	[5-30] min
t_{diff}	random offset for t_{ask}	0.1 min
n	number of conference members	256
f	number of floors	6
d	duration of worst case scenario	180 min

Table 6-5: *Application Evaluation Parameters for Panel Discussion Scenario*

table 6-5, the application parameter set is presented using the definition of chapter 5.4.2 concerning the values in brackets.

Similar to the second scenario, the parameter sets of the first scenario (see chapter 6.1.2) are used for the parameters on system level and for the default settings of the dynamic reconfiguration. Only for the topology classification, another parameter is used. As described above, there are no specific active users in the scenario. Hence, the active user subset H_t (see chapter 5.1) for the topology classification consists of all members, which result in one topology class only.

6.3.3 Results

As expected, the dynamic reconfiguration does not lead to an improvement of the responsiveness of the system with respect to the floor control response time. The results showed that the total response time of the conference is increased by approximately 9 percent. Moreover, the number of PDUs in the system is increased as well. This is not only caused by the high number of reconfiguration PDUs. In contrast to the other scenarios, also the normal conferencing PDUs increase. Hence, the application of the dynamic reconfiguration in this specific case leads to a degradation of the overall performance in terms of response time and PDU load.

Due to the uniform distribution of the activity among all conference members, the maximum size of the active user groups is very high (96 users), which results in a large number of moved providers when reconfiguring the tree topology. In table 6-6, a summary of the results for this

Parameter	Value
Total Performance Gain *G(u)*	-8.84%
Conferencing PDUs without Reconfiguration	363829
Conferencing PDUs with Reconfiguration	376489
Reconfiguration PDUs	68059
Reduction PDUs	-22%
Maximum Group Size	96

Table 6-6: *Summary of Results for Scenario 3*

third scenario is given.

It can be summarized from this short overview of the obtained results that the proposed dynamic reconfiguration mechanism does not lead to an improvement of any performance measure. On the contrary, important performance measures like the response time or the serviced PDUs in the system are increased after applying the reconfiguration, i.e. the system's performance is getting worse when applying the dynamic reconfiguration in this specific case. These results underline the importance of the definition of a floor which have to be considered by the reconfiguration. If there is no clear activity which is dedicated to the floor, the proposed reconfiguration cannot lead to any improvement. Hence, the application has to set the considered floors for reconfiguration very carefully to obtain improvements of the responsiveness instead of deteriorating the system performance.

6.4 Computational Effort

The evaluation of the dynamic reconfiguration mechanism of SCCS is performed using the simulation tool GCST. Similar to the resource management evaluation, a large number of simulation runs has to be performed to get statistical confident numbers for the results. The basic computational effort for obtaining these results is very similar to the resource management evaluation. Of course, this effort also depends on specific characteristics of the considered scenario, e.g. the number of active users or the extent of activity in terms of generated requests.

In addition to this basic computational effort, performing the dynamic reconfiguration adds an enormous overhead to each simulation run depending on the settings for the reconfiguration parameters. While one simulation for the described panel discussion scenario for a 256 members topology without dynamically reconfiguring takes about 0.7 seconds on a 500MHz PentiumIII Windows PC, the same scenario runs about 8 times longer when applying the dynamic reconfiguration with the default settings. Hence, a proposed simulation run with 500 iterations takes about 40 minutes to finish. This effort is even higher when changing specific reconfiguration parameters which affect the resulting size of active user groups, e.g. parameter a_b. But it is worth mentioning that this high computational effort for obtaining the evaluation results is not caused specifically by the proposed simulation method but by the application of the proposed reconfiguration algorithm.

However, it is demonstrated with the presented results that the proposed modeling method, based on the simulation framework presented in chapter 4.4.3, enables the evaluation of fairly complex mechanisms in group communication environments with a reasonable effort.

6.5 Setting Guidelines for the Reconfiguration

One of the goals to evaluate the proposed dynamic reconfiguration is to find guidelines for setting the provided reconfiguration parameters appropriately. In chapter 6.1.3.2, variations of the different reconfiguration parameters were performed, and the results of the impact on specific performance measures, both on application and system level, were obtained. In table 6-7, a brief summary of these results is presented describing the impact on the performance measures. Furthermore, to each parameter an *importance* scale is assigned concerning its impact on the performance measures.

Parameter	Importance	Impact for higher values			
		Performance Gain G(u)	Reduction PDU	moved Providers	Size of Group
t_{dr} variable	HIGH	decrease	increase	decrease	decrease
t_{dr} fixed	HIGH	constant	increase	decrease	decrease
a_b	HIGH	increase first then decrease	decrease	increase	increase
q_c	HIGH	decrease	increase	decrease	decrease
t_r	MID	- increase for low active users - constant for high active users	decrease	increase	increase
a_r	LOW	slightly increase	slightly decrease	slightly increase	slightly increase
a_c	LOW	almost constant	almost constant	almost constant	almost constant

Table 6-7: *Impact of Reconfiguration Parameters*

One of the results of the parameter variations is that decoupling the reconfiguration interval from the start point of the first reconfiguration (depicted as t_{dr} *fixed* in the table) allows to control the reduction of serviced PDUs without degrading the achieved performance gain. Hence, it is proposed to add another reconfiguration parameter, named as t_{start}, which defines the start point of the first reconfiguration. Furthermore, in the current realization of the reconfiguration algorithm, parameter a_c is not implemented as an application parameter. Due to its small impact on the performance measures, different values for a_c are tested internally in the implementation to choose the optimal setting for each constructed tree.

In general, it can be concluded from the obtained results of the variations that the proposed reconfiguration algorithm allows to control the main performance measures, namely performance gain and PDU reduction, by providing several reconfiguration parameters.

However, for a conferencing application which uses the proposed mechanism it might be useful to define a more user-friendly *reconfiguration goal*, e.g. "detect high active users only". This goal has to be translated in an appropriate reconfiguration parameter set which fulfils this reconfiguration goal satisfactory. This can be achieved by defining *profiles* of reconfiguration parameter sets for specific situations, which are chosen by the application.

In the following, two examples of such user-friendly reconfiguration goals are presented and appropriate settings for the reconfiguration parameters are outlined to reach these goals. As the basic scenario, the modified panel discussion of chapter 6.1 is assumed.

6.5.1 Case 1: Detecting High Active Users only

The first case which is considered aims at the detection of high active user only, i.e. the experts and the chairman of the panel discussion should be grouped nearby in the tree topology. In table 6-8, the proposed reconfiguration parameter set is presented to reach the application specific goal.

For the start point of the first reconfiguration, t_{start}, a short time is chosen to avoid a longish non-optimal phase of the conference. Due to the assumption, which is consistent to the obtained

Parameter	Description	Value
t_{start}	start of first reconfiguration	10 min
t_{dr}	reconfiguration interval	25 min
a_b	relative level (in percentage) for active user determination	3 %
t_r	time how long inactive users remain in active user group	10 min
a_r	value for exponential averaging for inactive user adjustment	0.5
q_c	quality level for insertion of optimized tree	50 %

Table 6-8: *Dynamic Reconfiguration Parameter Set for Case 1*

results in chapter 6.1.3, that the dynamic reconfiguration is able to properly detect the higher active users, the reconfiguration interval t_{dr} is set fairly high to reduce the reconfiguration overhead of the system. For the active user determination level a_b, a value is chosen which almost represents the number of higher active users with respect to the scenario settings presented in table 6-1. For this setting, it is assumed that the number of higher active users is known beforehand, which is a valid assumption for the case of a panel discussion (panel members are normally invited). As the last important parameter, the quality level for insertion of the optimized tree q_c is set to a fairly high value to reduce the reconfiguration overhead due to minor changes of the active user group.

For the last two parameters, t_r and a_r, the default settings are used due to their minor importance for the considered application specific goal.

Simulation results show that the higher active users, namely the chairman and the experts of the panel, are included in the active user group, while other conference users are only very sporadically inserted in this group, mainly due to peaks in their activity. For the active user group, a high performance gain *G(u)* is obtained, while the PDU reduction almost reaches -10 percent, i.e. the additional reconfiguration overhead is very small. Hence, the considered reconfiguration goal can be reached by the proposed reconfiguration parameter set.

6.5.2 Case 2: Adding Mid Active Users Sporadically

In contrast to the first case, it might be useful to consider less active users in the reconfiguration. For instance in the panel discussion scenario of chapter 6.1, a sporadic insertion of higher active participants might lead to improvements for this group of users in the conference in addition to the high active user group.

In table 6-9, the proposed reconfiguration parameter set for this case is presented. It can be seen that this set of parameters allows a more dynamic change of the active user group caused by different parameters. First, the default value of 5 minutes is used for the reconfiguration interval t_{dr} to allow frequent changes of the active user group. For the activity level a_b, a 4 percent value is used. This results in a group of 10 users that are inserted in the active user group (remember, that there are 7 high active users, namely one chairman and six experts according to the application parameters of table 6-1). As a consequence, a certain backlog of lower active

users is added to the active user group. Furthermore, more dynamic changes of the reconfigured

Parameter	Description	Value
t_{start}	start of first reconfiguration	10 min
t_{dr}	reconfiguration interval	5 min
a_b	relative level (in percentage) for active user determination	4 %
t_r	time how long inactive users remain in active user group	20 min
a_r	value for exponential averaging for inactive user adjustment	0.5
q_c	quality level for insertion of optimized tree	20 %

Table 6-9: *Dynamic Reconfiguration Parameter Set for Case 2*

tree topology are enabled by defining the quality level q_c with a smaller value compared to the first case. Extending parameter t_r to 20 minutes increases the size of the active user group by lower active users that became inactive after a while.

The obtained simulation results for this case show, similar to the first case, that the considered reconfiguration goal is reached with the proposed parameter set. However, this goal is paid by a higher number of reconfiguration PDUs leading to a larger reconfiguration overhead in the conferencing system.

Hence, it can be summarized that the considered, more user-friendly, reconfiguration goals can be reached with the proposed reconfiguration parameter sets.

6.6 Summary

The mechanism to dynamically optimize the tree topology within SCCS was evaluated in this chapter. The purpose of this chapter was twofold. On the one hand, it should have been demonstrated that the proposed modeling technique of chapter 4 enables the evaluation of rather complex mechanisms on system level even for complex application scenarios. On the other hand, the proposed dynamic reconfiguration mechanism was evaluated as such with respect to three evaluation goals.

Due to the complex mechanism being necessary to perform the dynamic reconfiguration, a simulative evaluation was used instead of applying analytical models. Three different scenarios were presented, each of them modeled using the *Group Communication Description Language*, proposed in chapter 4.3. For the realization of the user behavior and the access control, the SDL description technique and the simulation framework of chapter 4.4 were used, even for the more complex second scenario with a timed action behavior.

Considering the demonstration of the *feasibility* of the evaluation, it was shown that the proposed group communication description technique allows to obtain statistical evaluation results. These results were gathered from the simulation tool even for complex time-dependent scenarios. Furthermore, it was shown that the approach enables to evaluate fairly complex protocol mechanisms with a moderate computational effort.

With respect to the first goal of the performance evaluation, namely the *applicability* of the

reconfiguration mechanism, the obtained results showed that the algorithm detects active entities in the scenarios even when using the default parameter set. However, the third scenario, which was defined as a worst case for the proposed mechanism, showed that the floors to be considered by the dynamic reconfiguration algorithm have to be defined carefully by the applications. If there is no recognizable activity pattern in the conference, the algorithm does not lead to an improvement of the responsiveness of the conferencing system. In contrary, due to the high overhead caused by the reconfiguration, the system load gets even worse when applying the dynamic reconfiguration.

For the second evaluation goal, simulations with the default reconfiguration parameter set showed that the achieved performance gain is very high, especially for higher active conference members. *Improvements* up to 90 percent were obtained. Unfortunately, invoking the reconfiguration of the tree topology adds an overhead to the system leading to a higher number of serviced PDUs in the conference. However, it was shown by the evaluation that the responsiveness of the conferencing system is significantly improved when applying the proposed dynamic reconfiguration.

Isolated variations of reconfiguration parameters were also performed to outline their *impact* on specific performance measures. It was shown that specific performance measures like performance gain, PDU reduction, or number of active users can be influenced by appropriate settings for specific reconfiguration parameters. As an improvement of the mechanism presented in chapter 3.2.16, the introduction of another reconfiguration parameter was proposed defining the start point of the first reconfiguration. The presented results for the parameter variations led to the presentation of setting guidelines for these parameters. To find appropriate parameter sets, a *profile-based* approach was proposed which defines a user friendly reconfiguration goal, e.g. "detect high active users only". These goals are mapped to appropriate setting profiles for the reconfiguration parameters. These profiles can be provided by an additional application program which allows to select different settings.

However, it can be seen from the evaluation that specific problems of the used mechanisms are not solved. For instance, the proposed algorithm does not take the link time of a connection into account when applying the reconfiguration. As a consequence, the same link time was used for all SCCS connections, which is not a realistic assumption for real-world conferences. Furthermore, adaptive techniques might be used to change specific reconfiguration parameters dynamically to reach a defined performance gain. For that, monitoring facilities for the current response time of requests have to be added to SCCS when addressing that problem.

It can be summarized from the presented results that the purpose of this chapter is fully reached. It was shown that the proposed modeling technique allows to evaluate this complex mechanism even for the demanding timed action scenario. Furthermore, the achieved performance gain as well as the impact of certain reconfiguration parameters on specific performance measures were presented leading to setting profiles for specific application-dependent reconfiguration goals.

CHAPTER 7

Conclusions and Future Work

The topic of this thesis was on *scalable group communication systems*. The presented work dealt with the design of a generic conferencing service for large scaled scenarios as well as the evaluation of basic mechanisms by providing an appropriate modeling technique.

The main goal to provide a generic and scalable conferencing service is to accelerate and facilitate the development of conferencing solutions for synchronous, collaborative cooperation among distributed users. Typical conferencing scenarios are, among others, distance learning, large virtual meetings, or even distributed games using audiovisual or shared workspace means. Enabling these scenarios lead to the expectation of decreasing costs and the provision of new applications, which additionally pushed the research effort in this area from a user's point of view. A generic provision of conferencing services is useful due to the common needs of these applications independent from the specific scenario. The development of such services has been pushed recently due to the increasing availability of high-speed Internet connections from a technical point of view.

Unfortunately, recently proposed and developed group communication solutions mainly suffer either from the insufficient scalability of the used protocol mechanisms or from the provided services to some extent. As a consequence, the design and the prototype development of a generic and scalable conferencing control service was the first goal of the research work in this thesis. The proposed solution was designed on service as well as on protocol level providing means for a higher scalability of the environment. For that, mechanisms were developed leading to a higher responsiveness of the conferencing system compared to other solutions even in large scaled scenarios. Furthermore, the object and process model of a prototype implementation, including activity diagrams for the core protocol functionality, were presented to demonstrate the feasibility of the proposed service.

The second goal of the thesis was to provide a modeling technique for groupware scenarios and applications. The proposed method was used to evaluate two protocol mechanisms of the proposed service, namely the resource management scheme and the dynamic reconfiguration of the conference tree topology. On the one hand, a quantitative evaluation of the mechanisms was intended to show the improvement when applying the proposed mechanisms. On the other hand, the applicability of the proposed modeling technique should be demonstrated even in complex, realistic, and large scaled scenarios with respect to the user behavior, the access control model, and the number of participating users in the conference.

As an introduction to the research topic and as the basis to compare different conferencing solutions, the thesis started with the presentation of typical group communication scenarios. These scenarios were used to define service as well as technical requirements for a generic provision of conferencing functionality. From a service point of view, these requirements mainly cover aspects for *conference environment management*, e.g. provision of a conference database, functionality for *multipoint transfer* of user data, and means for synchronization of distributed objects and applications, i.e. *floor control services*. From a technical point of view, it was outlined that the services as well as the used protocol mechanisms have to be provided in a *robust* and *scalable* manner.

In addition to the service requirements, two paradigms were introduced with respect to the realization of the conference system, namely *tightly* and *loosely coupled* environments. On the one hand, it was outlined that the need for strong synchronization among distributed entities is much harder to provide in loosely coupled environments. On the other hand, the issue of scalability becomes much more important when following the tightly coupled paradigm for the realization of the conference system.

This conclusion was underlined by a detailed review of recently proposed environments for a generic provision of conferencing functionality. Especially the comparison of the tightly coupled approach of the *International Telecommunication Union* and the loosely coupled services of the *Internet Multimedia Architecture* proposed by the IETF showed, that scalability and full support of the introduced service requirements are contradictory in these proposed solutions. Either scalability is improved by restricting certain functionality like floor and membership control or the entire spectrum of the requirements is supported with poor scalability of the proposed services.

This dilemma led to the proposal for a generic and *Scalable Conferencing Control Service* following the paradigm for tightly coupled environments using a tree-based approach for the interconnections between service providers. SCCS supports sophisticated conference management functionality like provision of a conference database, full membership control, and reconfiguration functionality like merging, splitting, and completely reconfiguring the conference during runtime. For the support of multipoint transfer capabilities, an abstraction of the underlying transport layer was proposed using the notion of *channels* for multipoint addressing. This functionality is directly mapped to efficient multicast functionality, if supported, of the underlying transport layer. Furthermore, the notion of *non-SCCS listeners* allows to extend the scope of the user data reception to entities which have not joined the conference to improve the scalability of the considered scenario. For the provision of a scalable floor control service within SCCS, a resource management scheme was introduced by the author to improve the responsiveness of the conference system compared to other commonly used centralized techniques for this purpose. For a further improvement of the benefit of the proposed management scheme, a dynamic reconfiguration of the conference topology was introduced. This mechanism groups active conference members nearby in a reconfigured tree to further improve the responsiveness of the system in terms of shorter floor control response times.

The services as well as the used protocol mechanisms were defined in terms of service and protocol data units. Due to the complexity of the protocol, the message sequence charts, defining the interaction of the involved entities in the system, were excluded from this thesis being available as technical reports. However, the main protocol functionality was presented in textual form.

Finally, the proposed conferencing service was assessed to the defined service and technical requirements for conferencing environments. It can be concluded from this assessment that SCCS provides a wide spectrum of the demanded services including membership and floor

control. In parallel, the scalability and robustness of the proposed service was addressed by introducing services and mechanisms for this purpose like the proposed resource management scheme. Moreover, it was shown that SCCS allows the integration in the Internet Multimedia Architecture of the IETF, which additionally increases the applicability of the approach.

In addition to the proposal of SCCS, a prototype implementation of the service was presented. For that, the object and process model was introduced following the object oriented design paradigm. Furthermore, the main control activity diagrams were presented illustrating the core control functionality of the implementation. As an demonstration application, a shared white-board with main conferencing functionality was introduced, which is available on-line for free download.

As a starting point for the second part of the thesis, the performance evaluation goals were presented addressing two protocol mechanisms of SCCS, namely the *resource management* scheme and the *dynamic reconfiguration* of the tree topology. For the first, it was intended to determine the improvement of the floor control responsiveness when applying this scheme instead of a centralized version. For the second mechanism, the applicability should be outlined with respect to the ability to recognize specific activity patterns. Furthermore, the improvement of the responsiveness was of interest when applying the reconfiguration. Additionally, the impact of the reconfiguration parameters on specific performance measures was considered to be of special interest. In addition to the goal of the evaluation, the measures were introduced as well as the evaluation parameters of the system. For the latter, their impact on the performance goal was shown and the used parametrization was presented.

Before performing the evaluation of the SCCS protocol mechanisms, the issue of modeling groupware scenarios and applications in general was addressed. Similar to the protocol design issue of the thesis, requirements were defined which should be covered by modeling frameworks for groupware applications. Beside the structural description of the applications, modeling the communication and user behavior within these scenarios was outlined as one of the main points to be addressed. Furthermore, describing the access to distributed objects including the definition of read/write access rights and the time-dependent assignment of these rights was pinpointed as a crucial issue when modeling groupware applications.

These requirements were used for a review of related work in this area which was separated in *stochastic*, *workflow management*, and *behavioral* approaches. It was shown that especially the aspect of modeling the dynamic access control and the communication between the distributed entities is only poorly supported by recently proposed models. As a solution to overcome these weaknesses, a modeling framework was presented, called *Group Communication Description Language*. The object as well as the process model of GCDL was outlined to show the static and dynamic description of a groupware scenario. Additionally, the syntax of the language was defined to describe the structure of the scenario. Finally, it was shown how to use GCDL for workload generation to evaluate conferencing systems by means of realistic input data.

As GCDL mainly covers all defined requirements for modeling frameworks, its main advantage is the independence from the realization of the entity behavior description. Hence, simulative as well as analytical evaluation of the system should be enabled with this independence. As examples, two different behavior descriptions were presented in this thesis. To enable an analytical evaluation, *Stochastic Petri Nets* were used to describe the user behavior and access control respectively. For that, a module-based approach was introduced allowing to describe the different entities separately and combining them in a final step to the entire system. For this kind of realization, major drawbacks were outlined mainly caused by the increasing complexity of the described systems.

For performing simulative evaluations of conferencing systems, a realization of the behavior description was presented based on the protocol specification language SDL. It was shown how to realize the GCDL framework as SDL systems outlining a graphical as well as a textual presentation for this realization. Furthermore, the mapping to an event-based simulation was presented. For that, a simulation framework was introduced based on an object oriented design for which the object model as well as main activity diagrams were presented. This framework was implemented for the demonstration of the feasibility of GCDL leading to the *Group Communication Simulation Tool*, which realizes five different group communication scenarios following a fairly complex user behavior and access control model.

After proposing appropriate techniques to model group communication systems and scenarios, the performance evaluation of two protocol mechanisms of SCCS, namely the resource management and the dynamic reconfiguration, was taken through.

For the performance evaluation of the proposed SCCS resource management scheme, three different modeling techniques were applied. The first used a very simple stochastic model which was fixed to the considered binary tree topology and to the simple access control model. The second implemented the proposed Petri net behavior description, while the last method applied the specification language approach for a simulative evaluation of the system using the simulation tool GCST.

Considering the first goal of the evaluation, the demonstration of the applicability, it was impressively shown that the complexity of the analytical approaches increases rapidly. Applying the simple stochastic model but also the Petri net approach rapidly led to a certain point of complexity with a huge amount of computational effort for obtaining results especially in larger scaled scenarios. But also increasing the complexity in terms of user behavior and access control led to complex computations which are hard to handle numerically. Furthermore, the spectrum of obtained performance measures is very small for both methods, namely only measures on application level were determined due to the higher complexity when considering the system level as well.

For the simulative method, it was shown that even for more complex scenarios in terms of user behavior the computational effort for obtaining performance measures was moderate using state-of-the-art personal computers. Furthermore, the spectrum of obtained performance measures was much wider. Application as well as system level performance measures were gathered from the simulation tool in contrast to the analytical methods.

The obtained results themselves showed for all applied methods that the proposed resource management scheme significantly increases the responsiveness of the conferencing system with respect to the response time for floor control requests compared to the centralized management approach. The latter is used for instance in the ITU conferencing solution. Furthermore, typical bottlenecks like the top node of the tree topology are avoided using the proposed method. It was also shown that the achieved performance gain depends on the conference size in terms of participating users. The higher the conference tree, the smaller the overall performance gain of the new method. Additionally, the simulative method outlined that the performance gain also depends on the distribution of active entities in the conference. For obtaining this result, a *topology classification* approach was proposed which classifies a randomly generated topology according to the location of the active users within the tree.

The evaluation of the second considered protocol mechanism, the dynamic reconfiguration of the conference tree topology, demonstrated that the algorithm is able to detect active entities in the conference and groups them locally in a reconfigured tree. This grouping of active entities led to a further improvement of the response time for floor control requests, which was the pur-

pose of proposing the dynamic reconfiguration. These general results were obtained using two fairly complex scenarios, one with changing user activity and the other one with a time-dependent behavior. A third scenario was performed as a worst case for the algorithm in which floors were requested without a specific, recognizable *activity pattern*. This scenario showed no improvement of the system and was used to demonstrate the importance to define the floors carefully which are considered for reconfiguration. All scenarios were modeled using the SDL behavior description approach within the GCDL framework due to the complexity of the dynamic reconfiguration mechanism.

Additionally, the performance evaluation of the dynamic reconfiguration outlined the impact of the different reconfiguration parameters on specific performance measures. As a result, setting guidelines were presented proposing *profile-based parameter sets* to be chosen by the application for reaching specific reconfiguration goals.

It can be concluded that this thesis provides a scalable approach for a generic conferencing service covering the main aspects of the service and technical requirements. The scalability of two basic protocol mechanisms was evaluated using a proposed modeling methodology enabling both simulative and analytical evaluation of the considered system. The presented results depicted that the developed mechanisms improve the responsiveness of the conferencing system in terms of faster floor control requests to ensure the acceptance of the system by the user. Furthermore, it was shown that the proposed modeling technique enables the evaluation of distributed systems with more realistic, scenario-dependent workload.

Future Work

In this thesis, some protocol design and modeling aspects were not covered. As a consequence, some open issues remain for future research, namely:

- *Integration into the Internet Multimedia Architecture*: The conferencing architecture proposed by the IETF can be seen as a framework to integrate different conferencing solutions for specific applications. Despite the wide range of conferencing scenarios covered by SCCS, the integration in this framework would enable the usage of common services provided in this system. Especially the missing aspects of accounting, location service, announcing, and network resource allocation can be supplemented to SCCS-based applications by simply using state-of-the-art services of the proposed Internet architecture. Hence, SCCS has to be integrated in the appropriate conference control part of the architecture to enable the usage of SCCS by applications being implemented following this architecture and vice versa.
- *Extensive user tests*: The need for user tests is twofold. First, the applicability of SCCS and the robustness of the provided services and mechanisms can be evaluated additionally using user tests. Second, these tests can be used to extract more realistic parameters settings from the considered scenarios to be used in evaluations as performed in this thesis.
- *Automation of modeling*: Modeling specific scenarios and performing the corresponding evaluations was performed more or less manually in this thesis. For a wider applicability of the proposed modeling method, the need for automation arises especially from a user's point of view. This automation might begin with a frontend component for the more user-friendly graphical specification using the GCDL notation. As one backend component, an SDL specification tool might be used for the behavior description, ending in an automated code generation of appropriate C++ classes following the proposed simulation framework. Another backend solution might implement the proposed automation for the SPN behavior description enabling the analytical evaluation of the system.

- *Investigation of other scenarios*: Beside the considered scenarios in this thesis, other more sophisticated scenarios might be of interest. Especially, application scenarios like distance learning, shared working, or even distributed games might be modeled using the proposed technique.

The increasing demand for conferencing solutions on scientific but especially on commercial level will dramatically increase the need for sophisticated conferencing systems in the near future. As a consequence, it is expected by the author that the research effort in this topic will be further extended in the next couple of years.

Appendix A

SDL Template for GCDL

The following appendix presents a textural template for the behavior realization in GCDL using SDL as the specification language. This textural representation of the system gives a more precise definition of the different types and variables being necessary for the description. The realization based on the graphical definition of the system was presented in chapter 4.4.1.

In the SDL system, type and variable declarations are pre-defined for each role in the scenario. The FLOOR and USER types are defined beside some status information types. Furthermore, the first functional requirement in chapter 4.3.2 is fulfilled by defining corresponding input parameters to the appropriate processes and defining the corresponding FLOOR variables and their possession status. In the control and data process, a special initialization routine is created to set-up the automata by initializing the floors and setting the possession status correctly. Three main states are defined in the control process: ACCESS, NO_ACCESS, and WAITING. Further states for more complex control may be defined which is left to the specific implementation of the scenario. *Internal* signals are used for coordination of the control and data process. These signals might be *SDL timer* signals to enable certain states (e.g. like asking a question), e.g. based on statistics (e.g. *mean time to ask a question*) of user profile data.

In the following, the textual SDL definition of a role within GCDL is given. Note that the realization of channels for the data traffic process is not shown but it is similar to the floor control channel. The term <Floor> or <Role> in the specification stands for the name of the corresponding definition in the GCDL specification of the scenario.

```
SYSTEM <Role>;

    SYNTYPE   FLOOR     =    Integer CONSTANTS 1:65535;
    SYNTYPE   USER      =    Integer CONSTANTS 0:65535;
    NEWTYPE FloorStatus
         LITERALS
              Grab,
              NotGrabbed,
              Invalid;
    ENDNEWTYPE;

    NEWTYPE AccessMode
         LITERALS
              Read,
              Write;
    ENDNEWTYPE;
```

```
    NEWTYPE ObjectStatus
         STRUCT
              floorStatus   SetOF(FloorStatus);
              accessMode    AccessMode;
    ENDNEWTYPE;

    BLOCK <Role> REFERENCED;

    CHANNEL Control.SAP
         FROM ENV to <Role> WITH
              (Env.to.Control);
         FROM <Role> to ENV WITH
              (Control.to.Env);
    ENDCHANNEL;
    CHANNEL Data.SAP
         FROM ENV to <Role> WITH
              (Env.to.Data)
         FROM <Role> to ENV WITH
              (Data.to.Env)
    ENDCHANNEL;

    SIGNALLIST Env.to.Control =
         FArq,
         FAin,
         FPrq,
         FPin,
         FPrs,
         FPcf;
    SIGNALLIST Control.to.Env =
         FArq,
         FAin,
         FPrq,
         FPin,
         FPrs,
         FPcf;
    SIGNALLIST Env.to.Data =
# depends on implementation
    SIGNALLIST Data.to.Env =
# depends on implementation
    SIGNALLIST Data.to.Control =
# depends on implementation
    SIGNALLIST Control.to.Data =
# depends on implementation

    SIGNAL    FArq(FLOOR, USER);
    SIGNAL    FAin(FLOOR, USER);
    SIGNAL    FPrq(FLOOR, USER);
    SIGNAL    FPin(FLOOR, USER);
    SIGNAL    FPrs(FLOOR, USER);
    SIGNAL    FPcf(FLOOR, USER);

ENDSYSTEM;

BLOCK <Role>;

    CONNECT   Control.SAP AND Control.SAP.CO;
    CONNECT   Data.SAP AND DATA.SAP.DA;

    SIGNALROUTE   Control.SAP.CO
         FROM ENV TO Control WITH
              (Env.to.Role);
         FROM Control TO ENV WITH
              (Role.to.Env);
    SIGNALROUTE Data.SAP.DA
         FROM Env TO Data WITH
              (Env.to.Data)
         FROM Data TO ENV WITH
              (Data.to.Env)
    SIGNALROUTE Data.SAP.CO
```

```
         FROM Control TO Data WITH
              (Control.to.Data)
         FROM Data TO Control WITH
              (Data.to.Control)

ENDBLOCK;

# process for role. The parameters are to set the activation time
# and to set the floor possession correctly
PROCESS Control;
FPAR     start_activation   Duration,
         end_activation     Duration,
         objectStatus       ObjectStatus;

# floors associated with objects are defined as variables
    DCL <Floor>             FLOOR;
# the possession of the floor is set at start-up
    DCL <Floor>_possess     FloorStatus;
# variable to initialize the floor possession set
    DCL  f                  Integer;
# Timer is defined for activation of automaton instance
    TIMER start_activation, end_activation;
# Initialization function to be called at start-up
    PROCEDURE Init;
         START
# set automaton correctly according to object status
         TASK ...
# set timer for activation
         SET(NOW+<start>, start_activation);
         SET(NOW+<end>, end_activation);
    RETURN;
    ENDPROCEDURE;

    START

         CALL Init;

    NEXTSTATE Starting;

    STATE Starting;
         INPUT start_activation;
         DECISION <Floor>.possess = true;
         (true)   NEXTSTATE ACCESS;
         (false)  NEXTSTATE NO_ACCESS;
         ENDDECISION;

STATE ACCESS ...

STATE NO_ACCESS ...

STATE WAITING ...

ENDPROCESS;

PROCESS Data.<Role>;

ENDPROCESS;

# process for the data part. This process is controlled by the
# appropriate control process. The data process is instantiated
# by a control process to form a certain instance of a role.
PROCESS Data;
...
ENDPROCESS;
```

It can be seen from the textual description of the role system that SDL allows to extensively describe the role behavior in a group communication scenario. *Timed actions* are also provided

by using *timer signals* in SDL. The communication is realized by the SDL environment representing the communication infrastructure. Defined *floor control signals* are used for object access among the autonomous instances, while the data traffic realization depends on the actual implementation of the data traffic process. The control processes are instantiated according to a defined setup of the scenario. Each control process creates an associated data process being controlled by the control process. For that, data process-dependent signals are exchanged among both processes using the `Data.SAP.CO` channel. In the control and data process, a specific initialization routine is created to set-up the automaton instance by initializing the floors, setting the possession of the floors correctly, and setting some of the timers to enable the instance at a defined time.

With the approaches presented in chapter 4.4.2 and chapter 4.4.3, the presented SDL system can easily be mapped to an event-based simulation tool.

Appendix B

Abbreviations

A

AMBER	Architectural Modeling Box for Enterprise Redesign
API	Application Programming Interface
APE	Application Protocol Entity
ARM	Application Resource Manager
ASE	Application Service Element
ASN.1	Abstract Syntax Notation No. 1
ATM	Asynchronous Transfer Mode

B

BER	Basic Encoding Rules
B-ISDN	Broadband Integrated Service Digital Network
BNM	Behavior Network Model
B-WiN	German Research Network (in german Breitband-Wissenschaftsnetz)

C

CDR	Common Data Representation
CORBA	Common Object Request Broker Architecture
CSCW	Computer Supported Collaborative Work
CSDN	Circuit-Switched Digital Networks

D

DVMRP	Distance Vector Multicast Routing Protocol

E

EBNF	Extended Backus Naur Form

F

FCFS	First Come First Serve
FIFO	First In First Out
FTP	File Transfer Protocol

G

GCC	Generic Conference Control
GCDL	Group Communication Description Language
GCST	Group Communication Simulation Tool
GIOP	General Interoperability Protocol
GUI	Graphical User Interface

H

HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol

I

IANA	Internet Assigned Numbers Authority
IDL	Interface Description Language
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IID	Implicit Information Distribution
IIOP	Internet Interoperability Protocol
IMTC	International Telecommunication and Multimedia Consortium
IP	Internet Protocol
IPR	Intellectual Property Rights
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
IRTF	Internet Research Task Force
ISDN	Integrated Service Digital Network
ISO	International Standards Organization
ITU	International Telecommunication Union

L

LAN	Local Area Network

M

MAC	Medium Access Control
MAN	Metropolitan Area Network
MBone	Multicast Backbone
MC	Multipoint Controller
MCS	Multipoint Communication Service
MCU	Multipoint Communication Unit
MMUSIC	Multiparty Multimedia Session Control
MP	Multipoint Processor
MPEG	Motion Pictures Expert Group
MSC	Message Sequence Chart
MTP/SO	Multicast Transport Protocol - Self Organizing

N

NTE	Network Text Editor

O

OCoN	Object Coordination Nets
OMA	Object Management Architecture
OMG	Object Management Group
ORB	Object Request Broker
OSF	Open Software Foundation
OSI	Open Systems Interconnection

P

PCM	Pulse Code Modulation
PDU	Protocol Data Unit
PER	Packed Encoding Rules
PGP	Pretty Good Privacy
PIM-SM	Protocol Independent Multicast - Sparse Mode
POTS	Plain Old Telephone System
PSTN	Public-Switched Telephone Network
PSDN	Public-Switched Digital Network

Q

QNA	Queueing Network Analyzer
QNA-MC	Queueing Network Analyzer for Multicast
QoS	Quality of Service

R

RAS	Registration - Admission - Status
RFC	Request For Comments
RMTP	Reliable Multicast Transport Protocol
RMP	Reliable Multicast Protocol
RPC	Remote Procedure Call
RSVP	Resource Reservation Protocol
RTCP	Real-time Control Protocol
RTP	Real-time Transport Protocol

S

SAP	Session Announcement Protocol
SCCP	Simple Conferencing Control Protocol
SCCS	Scalable Conferencing Control Service
SCCSSAP	SCCS Service Access Point
SCCSSDU	SCCS Service Data Unit
SCCSPDU	SCCS Protocol Data Unit
SDL	Specification and Description Language
SDP	Session Description Protocol
SDU	Service Data Unit
SIP	Session Initiation Protocol
SPNP	Stochastic Petri Net Package
SRM	Scalable Reliable Multicast
SRMT	Scalable and Reliable Multicast Transport Protocol

T

TC	Transport Connection
TCP	Transmission Control Protocol
TOS	Type of Service
TSAP	Transport Service Access Point
TSDU	Transport Service Data Unit
TTCN	Tree and Tabular Combined Notation

TTL	Time To Live

U

UDP	User Datagram Protocol
UML	Unified Modeling Language
URL	Uniform Resource Locator

V

vat	Voice Audio Tool

W

WAN	Wide Area Network
WWW	World Wide Web
wb	Whiteboard

X

XTP	Xpress Transfer Protocol

Appendix C

List of Figures and Tables

Appendix D

References

D.1 Books and Papers

[AF91] H. ABDEL-WAHAB, M. FEIT: XTV: *A Framework for Sharing X Windows Clients in Remote Synchronous Collaboration*, Proceedings of IEEE Tricomm, Chapel Hill, 1991

[ACG86] S. AHUJA, N. CARRIERO, D. GELERNTER: *Linda and Friends*, IEEE Computer, Vol.19 No.8, pp. 26-34, August 1986

[ADH93] M. ALTENHOFEN, J. DITTRICH, R. HAMMERSCHMIDT, T. KAPPER, C. KRUSCHEL, A. KUCKES, T. STEINIG: *The BERKOM multimedia collaboration service*, Proceedings of the 1st ACM Conference on Multimedia, pp. 457-463, 1993

[AHP96] R. ALUR, G. J. HOLZMANN, D. PELED: *An Analyzer for Message Sequence Charts*, Software Concepts and Tools, Vol. 17, No. 2, pp. 70--77, 1996

[AnBa93] V. ANUPAM, C. L. BAJAJI: *Collaborative multimedia scientific design in SHASTRA*, Proceedings of the 1st ACM Conference on Multimedia, pp. 447-456, 1993

[Ar92] M. ARANGO ET. AL.: *Touring Machine: a software platform for distributed multimedia applications*, Proceedings of IFIP Conference on Upper Layer Protocols, Architectures and Applications, pp. 3-15, June 1992

[ALL88] E. AURAMAKI, E. LEHTINEN, K. LYYTINEN: *A speech-act-based office modelling approach*, ACM Transactions on Office Information Systems, Vol.6 No.2, pp. 126-152, April 1988

[BaKr96] F. BAUSE, P. S. KRITZINGER: *Stochastic Petri Nets: An Introduction to the Theory*, Vieweg Publishing, 1996

[BeMa99] B.R. BEBEE, G. A. MACK: *Evaluation of System Processes using an Executable Conceptual Model*, Proceedings of Symposium for Performance Evaluation of Computer and Telecommunication Systems (SPECTS'99), pp. 211- 215, July 1999

[BeKo98] I. BEIER, H. KÖNIG: *GCSVA - A Multiparty Videoconferencing System with Distributed Group and QoS Management*, Proceedings of 7th IEEE International Conference on Computer Communication and Networks (IC3N'98), pp. 594-598, October 1998

[BiRe94] K.P. BIRMAN, R. VAN RENESSE: *Reliable Distributed Computing with the ISIS Toolkit*, IEEE Computer Society Press, Los Alamitos, CA, 1994

[BGL96] F. BODENDORF, R. GREBNER, C. LANGENBACH: *Die Virtuelle Universität: Multimediales Tele-Teaching im B-Win*, DFN-Mitteilungen 41 (in german), June 1996

[BOGTS94] C. BORMAN, J. OU, H.-C. GEHRCKE, T. KERSCHAT, N. SEIFERT: *MTP-2: Towards achieving the S.E.R.O. properties for Multicast Transport*, Proceedings of 3th IEEE International Conference on Computer Communication and Networks (IC3N'94), September 1994

[BRR87] W. BRAUER, W. REISIG, ROZENBERG (EDS.): *Petri Nets: Central Models (Part I)*, Springer Lecture Notes on Computer Sciences 254, 1987

[BrRe98] T. BRAUN, A. REISENAUER: *Implementation of an Audio/Video Conferencing Application over Native ATM*, Proceedings of 5th International Workshop on Interactive Multimedia Systems and Telecommunication Services (IDMS'98), Lecture Notes on Computer Science 1483, pp. 342-350, September 1998

[CaDe92] S. CASNER, S, DEERING: *First IETF Internet Audiocast*, Computer Communication Review, pp. 92-97, July 1992

[Chen76] P.P.S. CHEN: *The entity-relationship model: toward a unified view of data*, ACM TODS Vol.1 No.1, pp. 9-36, 1976

[CBV92] M. CHEN, T. BARZILAI, H.M. VIN: *Software architecture of DiCE: a distributed collaboration environment*, Proceedings of the 4th IEEE ComSoc International Workshop on Multimedia Communication , pp. 172-185, 1992

[CMT89] G. CIARDO, J. MUPPALA, K. S. TRIVEDI: *SPNP: Stochastic Petri Net Package*, Proceedings of International Workshop of Petri Net and Performance Modeling (PNPM), pp. 142-151, 1989

[CSZ92] D. CLARK, S. SHENKER, L. ZHANG: *Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism*, Proceedings of ACM SIGCOMM, pp. 14-26, July 1992

[Coll88] W. T. MCLEOD (ED.): *The Collins Paperback English Dictionary*, Collins Publishing, 1988

[CTK99] C. CSEH, D. TROSSEN, R. KOGAN: *Framework for Automatic SDL to C++ Translation*, Proceedings of Conference on Protocol Specification, Testing and Validation (FORTE/PSTV'99), In J. Wu, S. T. Chanson, Q. Gao (Eds.) "Formal Methods for Protocol Engineering and Distributed Systems", Kluwer Publishing, pp. 95-115, October 1999

[CrRe99] S. J. CREESE, J. REED: *Verifying End-to-End Protocols Using Induction with CSP/FDR*, Proceedings of the Workshop on Formal Methods for Parallel Programming: Theory and Applications (FMPPTA'99), April 1999

[CrRo99] S. J. CREESE, A. W. ROSCOE: *Formal Verification of Arbitrary Network Topologies*, Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'99), June 1999

[Cr90] T. CROWLEY ET. AL.: *MMConf: An architecture for building shared multimedia applications*, Proceedings of CSCW'90, pp. 329-342, October 1990

[Da97] P. DAVIDS: *Analysis Tools for Local Area Simulation (ATLAS) V6.0*, Department of Computer Science 4, University of Technology Aachen, 1997

[DDC90] C. DIOT, W. DABBOUS, J. CROWCROFT: *Multipoint Communication: A Survey of Protocols, Functions, and Mechanisms*, Journal on Selected Areas of Communications, Vol.15 No.3, pp. 277-290, April 1990

[DoGLA95] H.-P. DOMMEL, J.J. GARCIA-LUNA-ACEVES: *Floor Control for Activity Coordination in Networked Multimedia Applications*, Proceedings of 2nd Asian-Pacific Conference on Communications (APCC'95), June 1995

[DoGLA98] H.-P. DOMMEL, J.J. GARCIA-LUNA-ACEVES: *Comparison of floor control protocols for collaborative multimedia environments*, Proceedings of SPIE International Symposium Voice, Video & Data Communications, Conference on Multimedia and Applications I, pp. 307-318, November 1998

[Ei99] W. C. EICKHOFF: *Specification and Development of Mechanisms for Reconfiguration of a Conference Control Protocol*, Diploma Thesis (in german), Department of Computer Science 4, University of Technology Aachen, 1999

[Er99] R. ERAßME: *Development of a Software Simulator for Testing Design Criterias for Data Connections in Tele-Working Scenarios*, Diploma Thesis (in german), Department of Computer Science 4, University of Technology Aachen, 1999

[Eff95] W. EFFELSBERG: *Das Projekt TeleTeaching der Universitäten Heidelberg und Mannheim*, Praxis in der Informationsverarbeitung und Kommunikation PIK (in german), No.18 Vol.4, pp. 205-208, Saur Publishing, 1995

[Erik94] H. ERIKSSON: *MBONE: The Multicast Backbone*, Communications of the ACM, Vol.37 No.8, pp. 54-60, August 1994

[FR97] A. FASBENDER, I. RULANDS: *On Assessing Unidirectional Latencies in Packet-Switched Networks*, Proceedings of IEEE International Conference on Communication (ICC'97), June 1997

[FaLo96] K. FAROOQUI, L. LOGRIPPO: *Group communication models*, Computer Communcations Vol.19, pp. 1276-1288, Elsevier Publishing, 1996

[FJC97] S. FLOYD, V. JACOBSON, S. MCCANNE: *A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing*, IEEE/ACM Transactions on Networking Vol.5 No.6 pp. 784-703, December 1997

[Fred92] R. FREDERICK: *NV - X11 video-conferencing tool*, Unix Manual Page, XEROX PARC, 1992

[GGW98] H. GIESE, J. GRAF, G. WIRTZ: *Modeling Distributed Software Systems with Object Coordination Nets*, Proceedings of the IEEE Symposium on Software Engineering for Parallel and Distributed Systems, April 1998

[Go91] A. GOSCINSKI: *Distributed operating systems: the logical design*, Addison-Wesley, New York, 1991

[GoS96] A. GOKHALE, D. C. SCHMIDT: *Measuring the Performance of Communication Middleware on High-Speed Networks*, Proceedings of ACM SIGCOMM, 1996

[GrSt85] J. G. GRUBER, L. STRAWCZYNSKI.: *Subjektive Effects of Variable Delay and Speech Clipping in Dynamically Managed Voice Systems*, IEEE Transactions on Communucations, Vol.33, No.8, pp. 801-808, August 1985

[GuLi94] J. A. GULLA, O. I. LINDLAND: *Modeling Cooperative Work for Workflow Management*, Proceedings of 6th International Conference CAiSE'94, pp. 53-65, 1994

[GuPl93] T. GUTEKUNST, B. PLATTNER: *Sharing Multimedia Applications among Heterogeneous Workstations*, Proceedings of 2nd European Conference on Broadband Islands, pp. 103-114, 1993

[HaCr97] M. HANDLEY, J. CROWCROFT: *Network Text Editor (NTE) : A scalable shared text editor for the MBone*, Proceedings of ACM SIGCOMM, 1997

[HWC95] M. HANDLEY, I. WAKEMAN, J. CROWCROFT: *The Conference Control Channel Protocol (CCCP): a scalable base for building conference control applications*, Proceedings of ACM SIGCOMM, 1995

[Hand97] M. HANDLEY: *On Scalable Internet Multimedia Conferencing Systems*, Ph.D. Thesis, University of London, 1997

[HKS93] M. HANDLEY, P KIRSTEIN, A. SASSE: *Multimedia Integrated Conferencing for European Researchers (MICE): Piloting Activities and the Conference Management and Multiplexing Center*, Computer Networks and ISDN Systems, Vol.26, pp. 275-290, 1993

[Haver95] B. R. HAVERKORT: *Approximate Analysis of Networks of PH/PH/1/K Queues: Theory and Tool Support*, in 'Quantitative Evaluation of Computing and Communication Systems', Beilner H., Bauser F. (Eds.), LNCS 977, pp. 239-253, 1995

[Haver98a] B. R. HAVERKORT: *Approximate analysis of networks of PH/PH/1/K queues with customer losses: Test results*, Annals of Operations Research, Vol. 79, pp. 271-291, 1998

[Haver98b] B. R. HAVERKORT: *Performance of Computer Communication Systems*, Wiley Publishing, 1998

[HBB99] B. R. HAVERKORT, A. BELL, H. BOHNENKAMP: *On the Efficient Sequential and Distributed Generation of Very Large Markov Chains from Stochastic Petri Nets*, Proceedings of International Workshop of Petri Net and Performance Modeling (PNPM), pp. 12-22, 1999

[HeTr97] T. HELBIG, D. TROSSEN: *The ITU T.120 Series of Standards as Basis for Conference Applications*, Proceedings of SPIE International Symposium Voice, Video & Data Communications, Conference on Multimedia Networks, pp. 190-201, November 1997

[HTT97] T. HELBIG, S. TRETTER, D. TROSSEN: *Combining CORBA and ITU-T.120 to an Efficient Conferencing Service*, Proceedings of the 4th International Workshop on Interactive Multimedia Systems and Telecommunication Services (IDMS'97), Lecture Notes on Computer Science 1309, pp. 296-307, September 1997

[HZS95] A. HELTON, E. ZULAYBAR, P. SOPER: *Business Process Engineering and Beyond*, IBM Redbook, Publication No. SG24-2590-00, 1995

[Her97] O. HERMANNS: *Multicast Communication in Cooperative Multimedia Systems*, Ph.D. Thesis (german), Department of Computer Science 4, University of Technology Aachen, 1997

[Herr92] R. G. HERRTWICH: *The HeiProjects: Support for Distributed Multimedia Applications*, IBM Technical Report No.43.9206, 1992

[HiGe98] V. HILT, W. GEYER: *A Model for Collaborative Services in Distributed Learning Environments*, Proceedings of 5th International Workshop on Interactive Multimedia Systems and Telecommunication Services (IDMS'98), Lecture Notes on Computer Science 1483, pp. 364-374, September 1998

[Holz91] G. J. HOLZMANN: *Design and Validation of Computer Protocols*, Prentice Hall, 1991

[HoSm99] G. J. HOLZMANN, M. H. SMITH: *Software Model Checking*, Proceedings of Conference on Protocol Specification, Testing and Validation (FORTE/PSTV'99), In J. Wu, S. T. Chanson, Q. Gao (Eds.)

"Formal Methods for Protocol Engineering and Distributed Systems", Kluwer Publishing, pp. 481-497, October 1999

[Holz97] G. J. HOLZMANN: *The model checker SPIN*, IEEE Transactions on Software Engineering, Vol.23 No.5, pp. 279-295, May 1997

[IKW98] T. INGVALDSEN, E. KLOVNING, M. WILKINS: *A Study of Delay Factors in CSCW Applications and Their Importance*, Proceedings of 5th International Workshop on Interactive Multimedia Systems and Telecommunication Services (IDMS'98), Lecture Notes on Computer Science 1483, pp. 159-170, September 1998

[Iona99] IONA INC.: *Iona Orbix(tm) 2000 Beta White Paper*, online available on http://www.iona.com

[Jack63] J. R. JACKSON: *Jobshop-like Queueing Systems*, Management Science, Vol.10 No.1, pp.131-142, 1963

[JEJ95] I. JACOBSON, M. ERICSSON, A. JACOBSON: *The Object Advantage - Business Process Reengineering with Object Technology*, ACM Books, 1995

[JaCa98a] V. JACOBSON, S. MCCANNE: *vat - LBNL Audio Conferencing Tool*, Lawrence Berkeley Laboratory, online available on http://www-nrg.ee.lbl.gov/vat , October 1998

[JaCa98b] V. JACOBSON, S. MCCANNE: *Using the LBL Network Whiteboard*, Lawrence Berkeley Laboratory, online available on http://www-nrg.ee.lbl.gov/wb , October 1998

[JoSw99] H. JONKERS, M. VAN SWELM: *Queueing Analysis to Support Distributed System Design*, Proceedings of Symposium for Performance Evaluation of Computer and Telecommunication Systems (SPECTS'99), pp. 300-307, July 1999

[Ka98] A. KATONA: *Modeling Multicast Data Traffic in Distributed Applications*, Diploma Thesis (in german), Department of Computer Science 4, University of Technology Aachen, 1998

[Kilk99] K. KILKKI: *Differentiated Services for the Internet*, Macmillan Publishing, June 1999

[KSCK99] M. KIM, J. SHIN, S. T. CHANSON, S. KANG: *An enhanced model for testing asynchronous communicating systems*, Proceedings of Conference on Protocol Specification, Testing and Validation (FORTE/PSTV'99), In J. Wu, S. T. Chanson, Q. Gao (Eds.) "Formal Methods for Protocol Engineering and Distributed Systems", Kluwer Publishing, pp. 337-356, October 1999

[KPMW94] A. MCKINLAY, R. PROCTER, O. MASTING, R. WOODBURN: *Studies of turn-taking in computer-mediated communications*, Interacting with Computers, Vol.6 No.2, pp. 151-171, June 1994

[Klein75] L. KLEINROCK: *Queueing System Volume 1 - Theory*, Wiley-Interscience, 1975

[Klein76] L. KLEINROCK: *Queueing System Volume 2- Computer Applications*, Wiley-Interscience, 1976

[Kl99] P. KLIEM: *Development and Evaluation of Mechanisms for Dynamic Reconfiguration of Tightly-Coupled Environments*, Diploma Thesis (in german), Department of Computer Science 4, University of Technology Aachen, 1999

[Kuehn79] P. J. KÜHN: *Approximate Analysis of General Queueing Network by Decomposition*, IEEE Transactions on Communications, Vol.27 No.1, pp. 113-126. January 1979

[Kung93] D. C. KUNG: *The Behavior Network Model for Conceptual Information Modeling*, Information Systems Vol.18 No. 1, pp. 1-21, 1993

[Lamp78] L. LAMPORT: *Time, Clocks and the Ordering of Events in a Distributed System*, Communications of the ACM, Vol.21 No.7, pp. 558-565, July 1978

[LaKo99] P. LANGENDÖRFER, H. KÖNIG: *Deriving Activity Thread Implementations from Formal Descriptions using Transition Reordering*, Proceedings of Conference on Protocol Specification, Testing and Validation (FORTE/PSTV'99), In J. Wu, S. T. Chanson, Q. Gao (Eds.) "Formal Methods for Protocol Engineering and Distributed Systems", Kluwer Publishing, pp. 169-184, October 1999

[LZGS84] E.D. LAZOWSKA, J. ZAHORJAN, G. GRAHAM, K. SEVCIK: *Quantitative System Performance: Computer System Analysis using Queueing Network Models*, Prentice-Hall, 1984

[LeGLA96] B. LEVINE, J. J. GARCIA-LUNA-ARCEVES: *A Comparison of Known Classes of Reliable Multicast Protocols*, Proceedings of International Conference on Network Protocols, October 1996

[LiPa95] J. C. LIN, S. PAUL: *RMTP: A Reliable Multicast Transport Protocol*, Department of Computer Science, Purdue University, AT&T Bell Laboratories, Holmdel, NJ, 1995

[LiP98] C. LINNHOFF-POPIEN: *CORBA - Kommunikation und Management* (in german), Springer Publishing, 1998

[LPT99] C. LINNHOFF-POPIEN, D. THIßEN: *Assessing Service Properties with Regard to a requested QoS: The Service Metric*, Proceedings of Third International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS), pp.273-280, February 1999

[LPM98] C. LINNHOFF-POPIEN, B. MEYER: *Modelling Multicast Queries in Distributed Systems*, Proceedings of International Networking Conference'98, pp. 143-150, July 1998

[Lub95] H. P. LUBICH: *Towards a CSCW Framework for Scientific Cooperation in Europe*, Lecture Notes in Computer Science 889, Springer Publishing, 1995

[Lyy90] K. LYYTINEN: *Computer Supported Cooperative Work (CSCW) - Issues and Challenges - A Structured Analysis*, Department of Computer Science, University of Jyvaskyla, SF-40 100, Finland, September 1990

[MASH99] MASH CONSORTIUM: *The MASH Project*, online available on http://www-mash.cs.berkeley.edu/mash/ , 1999

[Maff95] S. MAFFEIS: *Adding Group Communication and Fault Tolerance to CORBA*, Proceedings of the 1995 USENIX Conf. of Object Oriented Technologies, June 1995

[MaRo97] R. MALPANI, L. A. ROWE: *Floor control for large-scale MBone seminars*, ACM Multimedia, November 1997

[MJV96] S. MCCANNE, V. JACOBSON, M. VETTERLI: *Receiver-driven Layered Multicast*, Proceedings of ACM SIGCOMM, August 1996

[MCa98] S. MCCANNE: *vic - video conference*, Lawrence Berkeley Laboratory, online available on http://www-nrg.ee.lbl.gov/vic , October 1998

[Nutt96] M. NUTTAL: *Survey of Systems Providing Process or Object Migration*, Imperial College Research Report DoC 94/10, Department of Computing, Imperial College, London, 1996

[ObKe99] J. OBER, A. KERBRAT: *Specification and execution of tests using tMSC*, Proceedings of Conference on Protocol Specification, Testing and Validation (FORTE/PSTV'99), In J. Wu, S. T. Chanson, Q. Gao (Eds.) "Formal Methods for Protocol Engineering and Distributed Systems", Kluwer Publishing, pp. 453-468, October 1999

[OnSch93] L. Y. ONG, M. SCHWARTZ: *Centralized and Distributed Control for Multimedia Conferencing*, Proceedings of IEEE International Conference on Communication (ICC'93), pp. 197-201, 1993

[OAB99] J. ORVALHO, T. ANDRADE, F. BOAVIDA: *CONCHA: Conference system based on Java and CORBA Event Service*, Proceedings of SPIE International Symposium Voice, Video & Data Communications, Conference on Multimedia and Applications II, pp. 403-412, November 1999

[Ould95] M.A. OULD: *Business Process: Modelling and analysis for re-engineering and improvement*, Wiley Publishing, 1995

[Part93] C. PARTRIDGE: *Gigabit Networking*, Addison-Wesley, 1993

[PSLB97] S. PAUL, K. K. SABNANI, D. J. C. LIN, S. BHATTACHARYYA: *Reliable Multicast Transport Protocol (RMTP)*, IEEE Journal on Selected Areas in Communications, Vol.15, No.3, pp. 407-421, April 1997

[PaAb97] G. A. PAPADOPOULOS, F. ARBAD: *Control-Based Coordination of Human and Other Activities in Cooperative Informatik Systems*, 2nd Intl. Conference on Coordination Languages and Models (COORDINATION'97), pp. 422-425, 1997

[PaFl95] V. PAXSON, S. FLOYD: *Wide Area Traffic: The Failure of Poisson Modeling*, IEEE/ACM Transactions on Networking, Vol.3 No.3, pp. 226-244, June 1995

[QPSFV97] D. A. C. QUARTEL, L. FERREIRA PIRES, M. J. VAN SINDEREN, H. M. FRANKEN, C. A. VISSERS: *On the role of basic design concepts in behavior structuring*, Computer Networks and ISDN Systems, Vol.29 No.4, pp. 413-436, March 1997

[Rao84] M. M. RAO: *Probability Theory with Applications*, Academic Press, 1984

[Rat98] RATIONAL SOFTWARE COOPERATION: *Using Rational Rose 98*, 1998

[RBM96] R. RENESSE, K. BIRMAN, S. MAFFEIS: *Horus: A flexible Group Communication System*, Communication of the ACM, Vol.39 No.4, April 1996

[Rose97] O. ROSE: *Traffic Modeling of Variable Bit Rate MPEG Video and its Impact on ATM Networks*, Ph.D. Thesis, Bayerische Julius-Maximilians-Universität Würzburg, 1997

[RoUn99] J. ROTH, C. UNGER: *Group Rendevouz in a Synchronous, Collaborative Environment*, Proceedings of the Conference Kommunikation in Verteilten Systemen (KIVS'99), pp. 114-126, March 1999

[SNACC94] M. SAMPLE, G. NEUFELD: *High-Performance ASN.1 compiler*, Computer Communications, Vol.17 No., pp. 156-171, March 1994

[Scha98] K.-H. SCHARER: *Development and Performance Evaluation of a Conferencing Service using CORBA*, Diploma Thesis (in german), Department of Computer Science 4, University of Technology Aachen, 1998

[Sche98] A. W. SCHEER: *ARIS - Business Process Frameworks*, Springer Publishing, 1998

[SGHP97] D. C. SCHMIDT, A. GOKHALE, T. HARRISON, G. PARULKAR: *A High-Performance Endsystem Architecture for Real-time CORBA*, IEEE Communications Magazine, Vol.35, No.2, February 1997

[SSH98] G. SCHNEIDER, M. SCHUBA, B. HAVERKORT: *QNA-MC: A Performance Evaluation Tool for Communication Networks with Multicast Data Streams*, in 'Computer Performance Evaluation - Modelling Techniques and Tools', Puigjaner et. al. (Eds.), LNCS 1469, pp. 63-74, Springer Publishing, 1998

[Scho96] E. M. SCHOOLER: *Conferencing and collaborative computing*, ACM Multimedia Systems Vol.4, No.5, pp. 210-225, 1996

[Scho93] E. M. SCHOOLER: *Case study: multimedia conference controling packet-switched tele-conferencing system*, Journal of Internetworking: Research and Experience, Vol.4, No.2, pp. 99-120, October 1993

[SchoCa92] E. M. SCHOOLER, S.L. CASNER: *An Architecture for Multimedia Connection Management*, ACM SIGCOMM Computer Communication Review, Vol.22, No.3, July 1992

[Schu98] M. SCHUBA: *SRMT- A Scalable and Reliable Multicast Transport Protocol*, Proceedings of IEEE International Conference on Communication (ICC'98), June 1998

[Schu99] M. SCHUBA: *Scalable and Reliable Multicast Communication in the Internet*, Ph.D. Thesis (german), Department of Computer Science 4, University of Technology Aachen, 1999

[ShDe92] H. SHEN, P. DEWAN: *Access Control for Collaborative Environments*, Proceedings of CSCW'92, pp. 51-58, November 1992

[She87] I. SHEMER: *System analysis: a systematic analysis of a conceptual model*, Communication of ACM Vol.30 No.6, pp. 506-512, 1987

[SiSchu98] D. SISALEM. H. SCHULZRINNE: *The Multimedia Internet Terminal*, Journal on Telecommunication Systems, Vol. 9, No. 3, pp. 423-444, 1998

[SpHo95] O. SPANIOL, S. HOFF: *Event-based Simulation: Concepts and System Realization*, Thomson Publishing (in german), 1995

[StNa95] R. STEINMETZ: K. NAHRSTEDT: *Multimedia: Computing, Communications & Applications*, Prentice Hall Innovative Technology Series, 1995

[Strau98] B. STRAUSTRUP: *The C++ programming language*, Addison-Wesley, 1998

[SSM90] T. SUZUKI, S. M. SHATZ, T. MURATA: *Protocol Modeling and Verification Approach Based on a Specification Language and Petri Nets*, IEEE Transactions on Software Engineering, Vol. 16, No. 5, pp. 523-536, May 1990

[Tak91] H. TAKAGI: *Queueing Analysis - A Foundation of Performane Evaluation - Volume 1: Vacation and Priority System Part I*, North-Holland, 1991

[TaVe95] J. TANG, J. VEIJALAINEN: *Enforcing Inter-task Dependencies in Transactional Workflows*, Proceedings of 3rd International Conference on Cooperative Information Systems, 1995

[Tan96] A. S. TANENBAUM: *Computer Networks*, Third Edition, Prentice Hall, 1996

[Tan92] A. S. TANENBAUM: *Modern Operating Systems*, Prentice Hall, 1992

[TrRao99] M. M. TRIVEDI, B. D. RAO: *Camera Networks and Microphone Arrays for Video Conferencing Applications*, Proceedings of SPIE International Symposium Voice, Video & Data Communications, Conference on Multimedia and Applications II, September pp. 384-390, 1999

[Tro99a] D. TROSSEN: *Combining CORBA and ITU-T.120 to a Conferencing Service - Implementation and Results*, Technical Report TR002-99, Department of Computer Science 4, University of Technology Aachen, 1999

[TrSch98a] D. TROSSEN, K.-H. SCHARER: *CCS: CORBA-based Conferencing Service*, Proceedings of 5th International Workshop on Interactive Multimedia Systems and Telecommunication Services (IDMS'98), Lecture Notes on Computer Science 1483, pp. 71-76, September 1998

[TrSch98b] D. TROSSEN, K.-H. SCHARER: *SCCS: Scalable Conferencing Control Service*, Proceedings of 7th IEEE International Conference on Computer Communication and Networks (IC3N'98), pp. 698-706, October 1998

[TrPH98] D. TROSSEN, P. PAPATHEMELIS, T. HELBIG: *Improved Resource Management for the ITU T.122 Standard*, Proceedings of the 3rd IEEE Workshop on Systems Management (SMW'98), pp. 48-56, April 1998

[TrKa98] D. TROSSEN, A. KATONA: *Conference protocol evaluation: a load model to develop conference control protocols*, Proceedings of SPIE International Symposium Voice, Video & Data Communications, Conference on Multimedia and Applications I, pp. 295-306, November 1998

[TrKl99] D. TROSSEN, P. KLIEM: *Dynamic reconfiguration in tightly-coupled conference environments*, Proceedings of SPIE International Symposium Voice, Video & Data Communications, Conference on Multimedia and Applications II, pp. 391-402, September 1999

[Tro99b] D. TROSSEN: *Performance Evaluation of Conference Control Protocols*, Proceedings of the Communication Networks and Distributed Systems Modelling and Simulation Conference (CNDS'99), pp. 3-8, January 1999

[Tro99c] D. TROSSEN: *Simulation Framework for Automata-based Performance Evaluation*, Proceedings of Symposium for Performance Evaluation of Computer and Telecommunication Systems (SPECTS'99), pp. 273- 276, July 1999

[Tro00a] D. TROSSEN: *GCDL: Group Communication Description Language for Modeling Groupware Applications and Scenarios*, Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference, pp. 33-38, January 2000

[Tro00b] D. TROSSEN: *Group Communication Simulation Tool*, online available on http://www-i4.informatik.rwth-aachen.de/sccs/software/gcst.zip , May 2000

[Tro00c] D. TROSSEN: *Shared Whiteboard based on SCCS*, online available on http://www-i4.informatik.rwth-aachen.de/sccs/software/wb.zip , June 2000

[VaPu99] N. N. S.VÁZQUEZ, R. PUIGJANER: *Functional Specification of Scenarios for Performance Evaluation by Reusing Object-Oriented Simulation Performance Models*, Proceedings of Symposium for Performance Evaluation of Computer and Telecommunication Systems (SPECTS'99), pp. 223- 226, July 1999

[Walk82] M. B. WALKER: *Smooth transitions in conversational turn-taking: Implications for theory*, Journal of Psychology, 110(1), pp. 31-37, 1982

[WMK95] B. WHETTEN, T. MONTGOMERY, S. KAPLAN: *A High Performance Totally Ordered Multicast Protocol*, in 'Theory and Practise in Distributed Systems', Birman et. al. (Eds.), LNCS 938, pp. 33-57, 1995

[WGG97] G. WIRTZ, J. GRAF, H. GIESE: *Ruling the Behaviour of Distributed Software Components*, Proceedings of PDPTA'97, June 1997

[Whitt83] W. WHITT: *The Queueing Network Analyzer*, The Bell System Technical Journal, Vol. 62 No. 9, pp. 2779-2815, November 1983

[WGL98] C. K. WONG, M. GOUDA, S. S. LAM: *Secure group communication using key graphs*, Proceedings of ACM SIGCOMM'98, 1998

[ZDE93] L. ZHANG, S. DEERING, D. ESTRIN, ET. AL.: *RSVP - A New Resource Reservation Protocol*, IEEE Network, pp. 8-18, September 1993

D.2 Standards and Recommendations

[ASZ96] D. ATKINS, W. STALLING, P. ZIMMERMAN: *PGP Message Exchange Formats*, RFC 1991, August 1996

[BBCD98] S. BLAKE, D. BLACK, M. CARLSON, E. DAVIES, Z. WANG, W. WEISS: *An Architecture for Differentiated Services*, RFC 2475, December 1998

[BOR96] C. BORMAN, J. OTT, C. REICHERT: *Simple Conference Control Protocol*, Internet Draft, ftp://ftp.ietf.org/internet-drafts/draft-borman-mmusic-sccp-00.txt , June 1996

[BOS97] C. BORMAN, J. OTT, N. SEIFERT: *MTP/SO: Self-Organizing Multicast*, Internet Draft, ftp://ftp.ietf.org/internet-drafts/draft-ietf-mtp-so-01.txt , November 1997

[BCS94] R. BRADEN, D. CLARK, S. SHENKER: *Integrated Services in the Internet Architecture: An Overview*, RFC 1633, June 1994

[X292] CCITT: *OSI conformance testing methodology and framework for protocol Recommendations for CCITT applications – The tree and tabular combined notation (TTCN)*, CCITT Recommendation, X.292, 1992

[Z100a] CCITT: *Specification and Description Language (SDL)*, CCITT Recommendation Z.100, 1993

[RFC1112] S. DEERING: *Host Extensions for IP Multicasting*, RFC 1112, August 1989

[RFC2362] D. ESTRIN, D. FARINACCI, A. HELMY, D. THALER, S. DEERING, M. HANDLEY, V. JACOBSON, C. LIU, P. SHARMA, L. WEI: *Protocol Independent Multicast-Sparse Mode (PIM-SM)*, RFC 2362, June 1998

[RFC2236] W. FENNER: *Internet Group Management Protocol Version 2*, RFC 2236, November 1997

[RFC2616] R. FIELDING, J. GETTYS, J. MOGUL, H. FRYSTYK, L. MASINTER, P. LEACH, T. BERNERS-LEE: *Hypertext Transfer Protocol - HTTP/1.1*, RFC 2616, June 1999

[HPW99] M. HANDLEY, C. PERKINS, E. WHELAN: *Session Announcement Protocol V2*, Internet Draft, ftp://ftp.ietf.org/internet-drafts/draft-ietf-mmusic-sap-v2-03.txt , October 1999

[HSSR98] M. HANDLEY, H. SCHULZRINNE, E. SCHOOLER, D. ROSENBERG: *Session Initiation Protocol*, Internet Draft, ftp://ftp.ietf.org/internet-drafts/draft-ietf-mmusic-sip-11.txt , November 1998

[HaJa98] M. HANDLEY, V. JACOBSON: *SDP: Session Description Protocol*, RFC 2327, April 1998

[HCBO99] M. HANDLEY, J. CROWCROFT, C. BORMANN, J. OTT: *The Internet Multimedia Conferencing Architecture*, Internet Draft, ftp://ftp.ietf.org/internet-drafts/draft-ietf-mmusic-confarch-02.txt , October 1999

[ISO14977] ISO/IEC SC 22: *Information technology - Syntactic metalanguage - Extended BNF*, ISO/IEC 14977, 1996

[ISO8807] ISO/IEC: *Information Processing Systems - Open Systems Interconnections - LOTOS - A Formal Description Technique based on the Temporal Ordering of Observational Behavior*, ISO/IEC 8807, 1989

[G711] ITU-T: *Pulse Code Modulation (PCM) for Voice Frequencies*, ITU-T Recommendation G.711, 1988

[G722] ITU-T: *7kHz Audio-Coding within 64 kBit/s*, ITU-T Recommendation G.722, 1988

[H221] ITU-T: *Frame structure for a 64 to 1920 kBit/s channel in audiovisual teleservices*, ITU-T Recommendation H.221, 1997

[H225] ITU-T: *Media Stream Packetization and Synchronization for Visual Telephone Systems on Non-Guaranteed Quality of Service LANs*, ITU-T Recommendation H.225.0, 1995

[H235] ITU-T: *Security and encryption for H-Series (H.323 and other H.245-based) multimedia terminals*, ITU-T Recommendation H.235, 1998

[H242] ITU-T: *System for establishing communication between audiovisual terminals using digital channels up to 2 Mbit/s*, ITU-T Recommendation H.242, 1997

[H245] ITU-T: *Control Protocol for Multimedia Communication*, ITU-T Recommendation H.245, 1995

[H261] ITU-T: *Video Codec for Audiovisual Services at p x 64kBit/s*, ITU-T Recommendation H.261, 1993

[H263] ITU-T: *Video Codec for Narrow Telecommunications Channels at < 64kBit/s*, ITU-T Recommendation H.263, 1995

[H320] ITU-T: *Narrowband Visual Telephone Systems and Terminal Equipment*, ITU-T Recommendation H.320, 1993

[H321] ITU-T: *Adaption of H.320 Visual Telephone Terminals to B-ISDN Environments*, ITU-T Recommendation H.321, 1995

[H322] ITU-T: *Visual Telephone Systems and Equipment for Local Area Networks Which Provide a Guaranteed Quality of Service*, ITU-T Recommendation H.322, 1995

[H323] ITU-T: *Visual Telephone Systems and Equipment for Local Area Networks Which Provide a Non-Guaranteed Quality of Service*, ITU-T Recommendation H.323, 1996

[H332] ITU-T: *H.323 extended for loosely coupled conferences*, ITU-T Recommendation H.332, 1998

[T120] ITU-T: *Data Protocols for Multimedia Conferencing*, ITU-T Recommendation T.120, 1996

[T121] ITU-T: *Generic Application Template*, ITU-T Recommendation T.121, 1996

[T122] ITU-T: *Multipoint Communication Service - Service Definition*, ITU-T Recommendation T.122, 1998

[T123] ITU-T: *Network Specific Data Protocol Stacks for Multimedia Conferencing*, ITU-T Recommendation T.123, 1996

[T124] ITU-T: *Generic Conference Control*, ITU-T Recommendation T.124, 1998

[T125] ITU-T: *Multipoint Communication Service Protocol Specification*, ITU-T Recommendation T.125,1998

[T126] ITU-T: *Multipoint Still Image and Annotation Protocol*, ITU-T Recommendation T.126, 1997

[T127] ITU-T: *Multipoint Binary File Transfer Protocol*, ITU-T Recommendation T.127, 1996

[T128] ITU-T: *Multipoint Application Sharing*, ITU-T Recommendation T.128, 1998

[T134] ITU-T: *Text Chat Application Entity*, ITU-T Recommendation T.134, 1998

[T135] ITU-T: *User-to-reservation system transactions within T.120 conferences*, ITU-T Recommendation T.135, 1998

[X224] ITU-T: *Information technology - Open Systems Interconnection - Protocol for providing the connection-mode transport service*, ITU-T Recommendation X.224, 1995

[X680] ITU-T: *Information technology - Open systems interconnection - The directory: Overview of concepts, models and services*, ITU-T Recommendation X.500, 1993

[X680] ITU-T: *Information technology - Open systems interconnection - Abstract Syntax Notation One (ASN.1) - Specification of Basic Notation*, ITU-T Recommendation X.680, 1993

[X690] ITU-T: *Information technology - Open systems interconnection - ASN.1 Encoding Rules - Specification of Basic Encoding Rules (BER)*, ITU-T Recommendation X.690, 1994

[X691] ITU-T: *Information technology - Open systems interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1) - Part 2 : Packed Encoding Rules (PER)*, ITU-T Recommendation X.691, 1993

[X904] ITU-T: *Basic Reference Model for Open Distributed Processing - Part1 - 5*, ITU-T Recommendation X.901 - X.904, 1997

[Z100b] ITU-T: *Specification and Description Language (SDL)*, ITU-T Recommendation Z.100, 1993

[Z109] ITU-T Study Group 10: *Languages for telecommunications applications - SDL combined with UML*, ITU-T Draft Recommendation Z.109, 1999

[Z120] ITU-T: *Message Sequence Charts - MSC*, ITU-T Recommendations Z.120, 1996

[RFC1305] MILLS D.: *Network Time Protocol, Version 3*, RFC 1305, April 1992

[CORBA93] OPEN MANAGEMENT GROUP: *The Common Object Request Broker: Architecture and Specification*, 1993

[COSS97] OPEN MANAGEMENT GROUP: *Common Object Services Specification*, 1997

[DCE97] OPEN GROUP: *DCE 1.2.2 Documentation - Full Set*, Open Group Document T151, 1997

[RFC791] J. POSTEL: *Internet Protocol*, RFC 791, January 1981

[RFC793] J. POSTEL: *Transmission Control Protocol*, RFC 793, January 1981

[RFC768] J. POSTEL: *User Datagram Protocol*, RFC 768, August 1980

[RFC2212] S. SHENKER, C. PARTRIDGE, R. Guerin: *Specification of Guaranteed Quality of Service*, RFC 2212, September 1997

[RFC1889] H. SCHULZRINNE, S. CASNER, R. FREDERICK, V. JACOBSON : *RTP: A Transport Protocol for Real-Time Applications*, RFC 1889, January 1996

[RFC2326] H. SCHULZRINNE, A. RAO, R. LANPHIER: *Real Time Streaming Protocol (RTSP)*, RFC 2326, April 1998

[RFC1057] SUN MICROSYSTEMS: *RPC - Remote Procedure Call*, RFC 1057, June 1988

[SCCS98] D. TROSSEN: *Scalable Conferencing Control Service - Service Specification*, Technical Report TR001-98, Department of Computer Science 4, University of Technology Aachen, 1998

[SCCS99] D. TROSSEN: *Scalable Conferencing Control Service - Protocol Specification*, Technical Report TR001-99, Department of Computer Science 4, University of Technology Aachen, 1999

[RFC1075] D. WAITZMAN, C. PARTRIDGE, S. DEERING: *Distance Vector Multicast Routing Protocol*, RFC 1075, November 1988

[RFC2211] J. WROCLAWSK: *Specification of the Controlled-Load Network Element Service*, RFC 2211, September 1997

Appendix E

Curriculum Vitae

August, 16 1969	born in Bernkastel-Kues
1975 to 1979	Primary school in Bernkastel-Kues
1979 to 1985	Secondary school in Bernkastel-Kues
1985 to 1989	Secondary school in Wittlich
May, 31 1989	Abitur
1989 to 1990	Military service in Gerolstein
1990 to 1996	Studies in Mathematics and Computer Science at University of Technology Aachen
March, 21 1996	Diploma in Mathematics at University of Technology Aachen
1996 to 2000	Research Assistant with Department of Computer Science at University of Technology Aachen
since July 2000	Research Engineer with Nokia Research Center Boston, USA