A Multipoint Data Communication Infrastructure for Standards-based Teleconferencing Systems

vorgelegt von
Diplom-Informatiker
Jörg Ott

Vom Fachbereich 13 − Informatik −
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor-Ingenieur
− Dr.-Ing. −

genehmigte Dissertation

Berlin 1997
D 83

A Multipoint Data Communication Infrastructure for Standards-based Teleconferencing Systems

vorgelegt von
Diplom-Informatiker
Jörg Ott

Vom Fachbereich 13 − Informatik −
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor-Ingenieur
− Dr.-Ing. −

genehmigte Dissertation

Promotionsausschuß:

Vorsitzender:	Prof. Dr. rer. nat. Peter Pepper
Berichter:	Prof. Dr.-Ing. Sigram Schindler
	Prof. Dr.-Ing. Ute Bormann, Universität Bremen

Tag der wissenschaftlichen Aussprache: 28. April 1997

Berlin 1997

D 83

If, as it is said to be not unlikely in the near future, the principle of sight is applied to the telephone as well as that of sound, earth will be in truth a paradise, and distance will lose its enchantment by being abolished altogether.

Arthur Mee, 1898

Zusammenfassung

Ott, Jörg:

A Multipoint Data Communication Infrastructure for Standards-based Teleconferencing Systems

Eine Infrastruktur zur Mehrpunkt-Datenkommunikation in standardbasierten Telekonferenzsystemen

Jüngere Entwicklungen im Bereich der Telekonferenzsysteme scheinen dieser Technik nach bisher mangelnder kommerzieller Akzeptanz doch noch zum Durchbruch zu verhelfen. Dabei sind besonders die Tendenzen zur Unterstützung von Mehrpunkt- und Multimediakommunikation, zur Realisierung von Produkten auf der Basis internationaler Standards und zur Nutzung weitverbreiteter Telekommunikationsnetze hervorzuheben. Die wohl größte Bedeutung kommt jedoch der nahtlosen Integration der Telekonferenztechnik in den Arbeitsplatzrechner zu, die zur Entstehung von *Desktop-Multimedia-Konferenzsystemen (DMC-Systemen)* geführt hat.

Gegenüber dem intensiv bearbeiteten Gebiet der *Echtzeitkommunikation* ist die Unterstützung für in Telekonferenzen zu integrierende Anwendungskooperation und die dafür erforderliche *Daten-* oder *Nicht-Echtzeit-Kommunikation* vergleichsweise unterentwickelt. Gerade der Datenaustausch zwischen verteilten Anwendungssystemen im Rahmen einer Multimedia-Telekonferenz ist jedoch für viele Kooperationsszenarien von grundlegender Bedeutung, etwa zur gemeinsamen Bearbeitung eines Textes oder als Visualisierungshilfe in Diskussionen.

Die vorliegende Arbeit entstand im Kontext des EURO.VISION-Projekts, das die Entwicklung eines DMC-Systems inklusive einer Entwicklungsplattform für Telekonferenzanwendungen zum Ziel hat. Gegenstand dieser Arbeit ist die Entwicklung einer Infrastruktur für Mehrpunkt-Datenkommunikation in Telekonferenzen — *Multipoint Communication Layer (MCL)* —, die in das EURO.VISION-System eingebettet ist und die Integration verteilter Anwendungssysteme ermöglicht. Das MCL-Konzept beschreibt eine auf die speziellen Anforderungen der Anwendungskooperation in Telekonferenzen zugeschnittene Architektur, die aus einer umfassenden Betrachtung der Kooperationsvorgänge in (Tele-)Konferenzen heraus entwickelt wurde und sowohl reine Gruppenkommunikationsdienste bereitstellt, als auch an verschiedene Konferenzszenarien anpaßbare Konferenzsteuerungsfunktionalität umfaßt. Wegen der zentralen Bedeutung der internationalen Standardisierung für die globale Entwicklung der Telekommunikationstechnik hat während der Entwicklung des MCL-Konzepts eine enge Zusammenarbeit mit den relevanten Standardisierungsgremien stattgefunden. So wird insbesondere die T.120-Serie von ITU-T-Empfehlungen als Basis für die Implementierung des MCL-Konzepts verwendet, gleichzeitig wurde aber auch signifikant zu deren Weiterentwicklung beigetragen.

Abstract

Ott, Jörg:

A Multipoint Data Communication Infrastructure for Standards-based Teleconferencing Systems

More recent developments in the area of teleconferencing systems seem to finally promote this technology's breakthrough after the past lack of widespread commercial acceptance. In particular, trends towards support of multipoint and multimedia communications, towards implementation of products based upon international standards, and towards utilization of widely available communication networks are to be emphasized. The probably largest importance, however, is attached to the seamless integration of teleconferencing technology with the personal workstation leading to the emergence of *Desktop Multimedia Conferencing systems (DMC systems)*.

In contrast to the well advanced research area of real-time communications, support for applications for collaboration as integrated part of teleconferencing systems and the *data or non-real-time communication* required for those types of applications are far less developed. However, it is specifically the data exchange between distributed application system entities in the context of a multimedia conference that is of crucial importance to many teleconferencing scenarios, e. g. for joint editing of a text or as visualization aid in discussions.

The present thesis evolved in the context of the EURO.VISION project which aims at the development of a DMC system including a development platform for teleconferencing applications. The subject of this thesis is the development of an infrastructure for multipoint data communication in teleconferences — *Multipoint Communication Layer (MCL)* — that is embedded in the EURO.VISION system and that enables the integration of distributed application systems. The MCL concept defines an architecture that is specifically tailored to the needs of cooperating applications in teleconferences. The MCL was developed based upon a comprehensive review of cooperation processes in (tele-)conferences and provides group communication services as well as conference control services adaptable to different conferencing scenarios. Because of the outstanding importance of international standardization for the global development of telecommunication technology, a close collaboration with the relevant standardization bodies took place during the entire development of the MCL concept. In particular, the T.120 series of ITU-T Recommendations was taken as a basis for the implementation of the MCL concept and, in parallel, the further development of T.120 was significantly influenced by the author.

Acknowledgments

This thesis reflects the results of the research I have carried out from 1992 to 1997 at the Technische Universität Berlin in the Communications and Operating Systems Research Group created by Prof. Dr.-Ing. Sigram Schindler. An undertaking like this thesis is, of course, accompanied by frequent interactions — discussions, idea exchanges, and so forth — with many others, who thereby have contributed to this thesis as a whole. I would like to take this opportunity to thank all those who have supported me in this effort.

First and foremost, I would like to express my highest appreciation for the contributions of Prof. Dr.-Ing. Sigram Schindler to this thesis. It was his visionary engagement in the research area of multimedia teleconferencing in the late 1980s that gave me the opportunity to participate in research about and the development of various prototypes of teleconferencing systems, starting in 1990 — at a time when I still was a student of computer science at the TU Berlin. He did not only provide organizational and technical foundations for the past five years of intense research and development; I particularly appreciate the many conceptual discussions we had over the years: about the system architecture of EURO.VISION, strategies for application sharing, scenarios for integrating teleconferencing and telephony, to name just a few. Furthermore, he has early recognized the importance of international standardization for any research and development efforts in the area of telecommunications including teleconferencing. He enabled my regular participation in the most important standardization bodies for the area of teleconferencing systems (the ITU-T and the IETF), which are the only places where research results can impact the development of widely accepted communication technology at all. Finally, the early design and implementation experience in this field, running the EURO.VISION project in 1991 and early 1992, and the demanding conceptual work in this context, together with the involvement in international standardization were an unparalleled experience for me I would never want to miss — and which I probably would not have been able to obtain anywhere else.

Furthermore, I would like to thank Prof. Dr.-Ing. Ute Bormann and Dr.-Ing. Carsten Bormann for the last five years of very fruitful collaboration in virtually any sort of activities related to teleconferencing and (multipoint) communications. The countless hours we have spent together in discussions of concepts, writing papers and standardization contributions, preparing presentations, and attending a total of seven IETF meetings have significantly contributed to all my research efforts. Finally, with respect to my teaching at TU Berlin, I would like to thank Ute for the ISIS script and both of them for approximately one thousand(!) transparencies for ISIS and other lectures as well as their virtually permanent availability for any kind of questions I might have had. All of this has facilitated the work a lot and has allowed me to get, easier than expected, into

teaching our introductory lecture on international standards and telecommunication systems (ISIS) which I have been organizing for three and a half years now.

I also would like to thank Prof. Dr.-Ing. Sigram Schindler for his willingness to supervise this thesis and Prof. Dr.-Ing. Ute Bormann for taking the role of the second thesis advisor. My apologies to both of them that this thesis finally has many more pages than was originally intended. I am also very thankful to Prof. Dr. rer. nat. Peter Pepper who was kind enough to chair the review committee on a very short notice.

The last seven years of research and development for multimedia conferencing systems have been a hard working but entirely enjoyable time. When learning something new from scratch, the environment one does this in is of primary importance: the advanced and productive environment of the TELES AG has been the perfect place for doing this. This paragraph is dedicated to all those with whom I have spent the first years of engineering on the DIDAMES integrated videoconferencing system which eventually evolved to become the EURO.VISION system in the context of which this research has been carried out. The initial crew I have been working in on the DIDAMES system consisted of Jan Bastian, Ulrich Beyer, Barnim Dzwillo, Peter Reimers, Thorsten Thiele, Dr.-Ing. Stephan Wenger, and Hauke Witt and was later joined by Alexander Berresheim and Rochus Wegener. I would like to thank all of them for the fascinating time we spent together engineering on the DIDAMES prototypes. At this time and during later phases in the development — when I was already employed at the university — I also appreciated the collaboration with Stefan Braun, Andreas Illg, Karsten Lüdtke, Frank Paetsch, and Peter Schönberger in conceptual discussions, the writing of papers, and their willingness to answer all my questions.

This thesis does not only describe conceptual work but also engineering efforts which are reflected in some two hundred thousand lines of source code that have been written and re-written altogether. The prototype implementation of the T.120 infrastructure described in this thesis has been done in the context of many diploma theses, I owe thanks to all those who put more than above-average efforts into their respective parts. Andreas Schmidt has done the initial design of the MCS provider with later extensions by Henning Düwel, Arne Melcher, and Boris Nikolaus. It was also Henning who did the initial port from the UNIX platform to the 16-bit MS Windows environment, which was then ported by Arne Melcher to Windows'95 and Windows NT. Jörg Simm did the initial implementation of the T.123 protocol stacks based upon the STREAMS emulation provided by Peter Kratz. Jutta Degener designed and implemented the initial version of the GCC provider and took the risk of being one of the first persons in the world to take a deep look into ASN.1 PER encoding which she implemented for GCC. Hans-Christian Gehrcke and Matthias Kerkhoff implemented the prototypes of the two currently standardized T.120 application protocols. Boris Nikolaus and I spent hours and hours on the details of the MMAP protocol for the MCS extensions, and he implemented the protocol and performed the integration into the MCS provider. The most important foundation for this undertaking — the multicast transport protocol MTP-2 — has been implemented by Nils Seifert and Torsten Kerschat who also designed the protocol in collaboration with Dr.-Ing. Carsten Bormann and myself.

The way from an initial implementation of a working prototype to a completely standards-compliant and interoperable infrastructure takes a lot of time and effort. The newly created T.120 project group at TELES has taken on much of these efforts and created a product version of this infrastructure that has proven its standards-compliance in various interoperability tests. Those

involved in these efforts are: Henning Düwel and Arne Melcher who improved the MCS provider and implemented the local MCS message passing API, Stefan Radig who redesigned and completed the GCC provider, Jörg Simm who continued his work on T.123, Peter Kratz and André Barnitzke who implemented the GPCI environment, and Bernd Birkicht who did the local GCC message passing API. Finally, Dr. Horst Kiesling and Heinz-Peter Scharf were responsible for the interoperability testing. Altogether, the aforementioned and many other colleagues at TELES have performed the full integration of the T.120 infrastructure into the EURO.VISION system. In particular, I would like to thank Henning Düwel and Stefan Radig for their extraordinary efforts and our frequent design discussions.

I also have to thank my colleagues within the Communications and Operating Systems Research Group at the TU Berlin for their direct and indirect contributions to this thesis. Dr.-Ing. Stephan Wenger and I have had many discussions on standards-related issues and on design aspects of DMC systems. Our collaboration in standardization has intensified recently as we are now both attending the same standardization meetings and are together writing contributions on various subjects which is of mutual benefit. Dr.-Ing. Stefan Edlich with whom I was sharing the office has collected a huge pile of literature on groupware systems which significantly helped in obtaining an overview of this research area. The two of us also spent many hours in discussions, and together we have run the telecooperation project at the university for three years. I would like to thank the aforementioned colleagues as well as Volker Lausch and Achim Stindt who have taken over my lecturing responsibilities whenever I had to be on standardization meetings.

There is one friend and former colleague I would like mention for his indirect contributions to this thesis: Dr.-Ing. Dirk Buchta is the one who stimulated my decision to specialize in communications technology and to do so in the Communications and Operating Systems Research Group during my first year of studying computer sciences. Following his advice turned out to be a very wise thing to do.

Finally, I would like to sincerely thank my parents for their encouragement and their invaluable support in many aspects of daily life over the last years which allowed me to fully concentrate on my research and thus significantly contributed to the successful completion of this thesis.

Table of Contents

List of Figures

List of Tables

1

Introduction

Since the late 1950s, computer technology has been in use in enterprises to support various types of business processes. During the last forty years, the fields of computer application have widened and computer technology use in companies has become much more important. Along with its evolution, the deployment of computer technology has changed in several ways: from centralized computing divisions to the decentralized availability of computing resources in each division, later at each workplace; and from pure automation of processes to the support (rather than the substitution) of humans doing their daily work.

Computer technology started with financial applications (payroll, accounting, banking, and insurance applications) in which the computer was primarily regarded as a means for automated calculations. Then, computer use expanded into production plants. There, the first field of application was the numerical control of production machinery from which, finally, in the late 1980s, Computer Integrated Manufacturing — the control of the entire production process (R & D, design, manufacturing, quality management) — has evolved. From the mid-1970s, office applications have become important: this started with word processing; in the 1980s, spreadsheet applications took on. Furthermore, the support for design processes has emerged with the advent of powerful computers with graphics displays which also led to the development of graphical user interfaces. The inclusion of animations as well as continuous media such as audio and video (e. g. for presentations) mark the latest step in the development of computer technology.

In all these areas of application, computer technology was primarily used to substitute or simplify tasks previously carried out (manually) by humans: In manufacturing plants, its major application has been the automation of the production processes as far as possible (e. g. in assembly lines), in the office environment, previous office tools have been replaced by certain computer applications that allow employees to perform their tasks more efficiently. For example, text systems simplify writing business letters, reports, etc., thereby replacing typewriters; and spreadsheet programs make the evaluation and presentation of data, creation of statistics and charts much more efficient, compared to the former usage of calculator, pencil, and paper. For the subject of this thesis, the office environment is the focus of interest.

Particularly in the office environment, further support of business processes has appeared with the integration of telecommunication with computer technology. Telecommunication links between

enterprises are used to exchange documents — orders, invoices, and other highly structured trade documents — without requiring the physical transport of paper, floppy disks, or other media, and with reduced involvement of humans. Important applications include the automatic placement of orders with suppliers to implement just-in-time delivery of goods (as is done e. g. in the automobile industry), banking transactions, trading of stock, and flight reservation systems. The advent of local area networks has (together with the availability of personal computers) contributed to the decentralization of computing, resulting in shared information bases in enterprises, distributed applications (e. g. client-server computing), etc.

The use of computer and communication technology simplifies information sharing between persons. For example, database systems allow comfortable storage and retrieval of information with faster and easier access and complement or even obsolete traditional (paper-based) filing methods. Electronic data interchange speeds up information exchange by orders of magnitude, partially eliminating error-prone manual processing for sender and receiver, thereby providing a substitute for postal mail and maybe even facsimile.

Until recently, however, virtually all approaches to computer support in the office environment have had in common that they have improved parts of business processes independently and/or that the respective (software) systems have been deployed isolated from one another — despite the availability of communication technology. While these kinds of computer support in enterprises have led to enormous gains in productivity — and today's offices cannot be pictured without computer technology — it has become apparent that *further* efficiency improvements in the office environment can only be achieved if 1) business processes are considered in their entirety (and this not only with respect to computer support) and 2) besides individual employees also their (work-related) relationships as members of a group are considered. What is needed is the integration of communication and computer technology with explicit aim of supporting collaboration in work groups (refer e. g. to [Schindler 92a] [Kremar 92], [Jakab 93], [Bullinger *et al.* 93], [Schindler 94a], [Steinle 94]).

Throughout this thesis, the term *work group* is used in very broad sense: a work group refers to a group of persons that are collaborating at a given point in time to achieve some common goal. A work group is not necessarily reflected in an organizational structure, nor need its members belong to the same enterprise or institution. A work group may exist for as long as many years or for as short as the duration of a single phone call. A group may consist of as few as two persons or as many as several thousands. However, for the purpose of this thesis — which focuses on supporting business work groups — the upper bound for the group size is for practical reasons somewhere around hundred group members.

Two obvious observations about work groups and group processes in the office environment shall be used as a starting point for approaching the subject of computer support for work groups:

1) A group is formed to achieve a common goal which requires its members to perform a set of tasks (work items). The group members work individually and/or collaboratively on the respective work items.
2) In order to finally accomplish the group's overall tasks the group members need to communicate with one another and coordinate their individual efforts.

New types of (software) systems for the office environment are required to support work-related interactions between members of a work group, complementing those systems designed for

individually performing tasks that are already in place. The class of (software) systems that explicitly takes into account group collaboration semantics and provides the required functionality is introduced in the next section.

1.1. Teleconferencing: Support for Work Groups

Work groups in organizations are being investigated in many research disciplines including sociological sciences (of groups and organizations), psychology, management sciences, and linguistics among many others [Baecker 93]. These disciplines make important contributions to the research area of *Computer Supported Cooperative Work (CSCW)*[1] that was established in the mid-1980s [Greif 88] [Greenberg 91] [Schmidt / Bannon 92] [Baecker 93] [Grudin 94a].[2]

Within the area of CSCW, the term *groupware* (e. g. [Johnson-Lenz / Johnson-Lenz 82], [Johansen 88], [Ellis *et al.* 91], [Baecker 93]) subsumes all types of software systems that are designed to support the work processes in groups. The term itself was introduced by Trudy and Peter JOHNSON-LENZ [Johnson-Lenz / Johnson-Lenz 82, p. 47] before the establishment of CSCW. They defined groupware as

> *intentional GROUP processes and procedures to achieve specific purposes + softWARE tools designed to support and facilitate the group's work.*

This definition explicitly includes the social processes of the work group around the computer-based system into the scope of groupware. JOHANSEN restricts this scope to the computer-based systems only [Johansen 88]. Following JOHANSEN'S approach groupware can be defined as (refer to [Ellis *et al.* 91, p. 40])

> *computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment.*

This thesis follows this latter definition. The term *groupware* is used to refer to the class of (software) systems designed to support collaboration between humans in work groups. The term *groupware system* denotes a particular (software) system implementing certain functionality to provide this support.[3]

Many different types of groupware systems have been developed in the past (refer to section 2.1). The type of systems probably having received the most increasing attention during the last years — from researchers as well as from computer and telecommunication industry — is the class of *teleconferencing systems*. Teleconferencing systems constitute the subject of this thesis (see section 1.3 below). For an introduction to this subject, various terms have to be defined, most of

---

[1] From today's computer science perspective, support of office work usually implies use of computer technology so that this term appears to be somewhat pleonastic.

[2] The term *Computer Supported Cooperative Work* was created by Irene GREIF and Paul CASHMAN in 1984 (stated in [Schmidt / Bannon 92] and [Baecker 93]).

[3] As BAECKER puts it: Groupware systems are intended to support the variety of coordination processes between the members of a work group. ''Groupware represents a paradigm shift for computer science, one in which *human-human* rather than human-machine communications and problem solving are emphasized.'' [Baecker 93, p. xi]

which are commonly used in the research area of groupware systems, sometimes with slightly different meanings though. Hence, a brief definition of the most important terms is given in the following paragraphs to avoid any confusion about their respective meaning.

For now, an intuitive understanding of a *conference* and of *(conference) participants* (also called *conferees*) is considered sufficient and therefore no definition is given at this point. This understanding of a ''conference'' is extended to explicitly include discussions between two persons as well.[5] The terms *physical* and *face-to-face conference* (or meeting) refer to a conference in which all participants gather at the same location (e. g. in a conference room) for the purpose of holding the conference. The term *teleconference* refers to all types of conferences where the physical colocation of participants is not necessary. In teleconferences, geographical distance is overcome by means of telecommunication technology. Systems that implement the required functionality are termed *teleconferencing systems*.

Early teleconferencing systems have been providing nothing but audio(visual) communication capabilities to the conference participants. Examples are videoconferencing rooms, video phones, and the so-called ''bridges'' for audio conferences. More recent teleconferencing systems increasingly incorporate functionality of other types of groupware systems (e. g. shared editors) in addition to audio(visual) communication to enhance the system's applicability. In the context of such teleconferencing systems the terms *groupware application* and *(tele)conferencing application* are used interchangeably throughout this thesis to refer to a groupware system that is an integrated part of such a teleconferencing system.

Along with the enrichment of functionality, the implementation of teleconferencing systems based upon personal computers or other workstations[6] has received increasing attention. Workstation-based systems provide audio(visual) communication as core functionality and integrate groupware applications such as telepointers, whiteboards, distributed cooperative editors, and/or application sharing systems (see chapter two) to augment the interactions among the group members during the teleconference. A workstation-based teleconferencing system that supports audiovisual communication between two or more persons, that integrates various types of joint collaboration functionality (e. g. joint editing), and that provides elaborate mechanisms for running and controlling the teleconference is termed a *Desktop Multimedia Conferencing system (DMC system)*.[7] DMC systems constitute the background of this thesis.

---

[5] These interactions constitute a significant portion of communication among members of work groups and are actually little different from conferences with three or more persons. The latter type of conference may emerge from the former when two communicating partners notice that they need to involve a third person in their discussion.

[6] Within this thesis, the more general term *workstation* is used exclusively to refer to all kinds of computer systems that are intended primarily for use by a single person. This includes (among others) all kinds of UNIX workstations, IBM-compatible personal computers (PCs), Macintosh systems as well as any type of portable computers.

[7] ''Die Welten von Datenverarbeitung, Datenkommunikation und Telefonie verschmelzen zunehmend miteinander. (...) bei den Endgeräten vollzieht sich die Synthese in den allgegenwärtigen PCs oder Arbeitsplatzsystemen — das Ergebnis sind Desktop-Multimediakonferenz-Systeme.'' (S. Schindler, Technische Universität Berlin, in an interview of the ''Tagesspiegel'' [Paszkowsky 94].)

1.2. Motivation for Teleconferencing

In work groups, all kinds of collaboration processes require (intense) communication between the group members to coordinate their activities, collaborate in problem solving, perform planning and revision tasks, exchange status information, and so on. Typically, work group members either complain about too much time spent on routine communications preventing them from getting their actual work done. Or the members worry about insufficient coordination of their group work due to a lack of communication. Consequently, mechanisms that make at least a subset of the necessary communication processes less time consuming, more efficient, and easier to carry out are highly desirable.

The advent of fax machines and electronic mail has significantly improved asynchronous[9] cooperation between (distant) group members. Today, however, the primary means for synchronous interaction between group members still are physical meetings and telephone calls. Which of these synchronous means for interactive communication is chosen in a given situation depends a) on the matter to be discussed and b) on the possibility to make use of the respective means in this situation:

- In most cases, synchronous collaboration is best supported by face-to-face interactions (i. e. conferences). This is particularly true if discussion topics are more complex (e. g. in design discussions, brainstormings), if visualization is needed for mutual understanding, if personal presence is required, or sometimes even just if more than two persons are involved.

- Phone calls are useful for routine discussions and maybe information updates. However, they are not the better solution with respect to person-to-person interactions. If phone calls are used, this is done because they are the more convenient and quicker way (since phone calls can be placed ad-hoc) or the single possible way at all (because the partner to talk to is too far away) to establish a face-to-face interaction.

Obviously, audio communication via the telephone is insufficient for many occasions. However, a physical conference is more expensive to set up, in terms of time and possibly money: ad-hoc scheduling is not always possible; date, time, and venue have to be agreed upon; and, finally, some kind of ‘‘traveling’’ may be needed prior to the physical meeting: to another floor, to the next building, across the city, or to another country maybe on a different continent. While for certain types of discussions (such as initial negotiations on a contract and employee reviews) face-to-face meetings are inevitable and their drawbacks have to be accepted, many other meetings may also be held using teleconferencing technology as an alternative [Ott 95a]. Various studies have found that from ten up to ninety per cent of the meetings are potential candidates for substitution by teleconferences (refer in particular to the overview of several surveys given in [Schulte 93, p. 83], but also to the examples provided in [Egido 88], [Tonnemacher 88], and [Kolrep *et al.* 90]).

[9] A cooperation process is referred to as *asynchronous* if the involved parties are not required to be available at the same time for their interactions while *synchronous* cooperation refers to those interactions for which simultaneous availability of the persons is a prerequisite. For a precise definition of asynchronous and synchronous cooperation refer to section 2.1.2.

So the main motivation for (multimedia) teleconferencing stems from the idea that this technology in principle allows to conduct meetings without requiring the participants to travel to a common location. Instead, telecommunication channels are provided between all participants for the (in a given conference context) most important means of interaction (e. g. gestures and speech). Multimedia teleconferencing systems but also videoconference rooms or even simple videophones are expected to be at least an adequate alternative to face-to-face conferencing for certain types of meetings, if not a superior means for collaboration. The confidence of this technology's advocates in its ''problem resolution potential'' is best indicated by giving a few representative quotes from literature.[11]

- *"The next best thing to being there is a Picturephone call."* (Julius P. Molnar, executive vice president of Bell Laboratories, in 1969, quoted in [Egido 88]).
- Videoconferences are *"the next best thing to actually 'being there'"* [Tagyos 85, p. 63].
- *"Why travel when you can call?"* [Douglas 89].
- *"what automation has meant for increasing the productivity of the labour force of an enterprise, professional videoconferencing will mean for increasing its management efficiency, and hence its business success."* [Schindler 92a, p. 12]
- For some applications, suitable *"media space technologies can potentially allow us to go 'beyond being there'"* [Gaver 92, p. 23] also referring to [Hollan / Stornetta 92]).
- Using desktop video teleconferencing can be *"better than being there"* [Yager 93].

Moreover, desktop multimedia conferencing systems bring teleconferencing functionality to each employee's desk and thus increase the potential of teleconferencing tremendously in contrast to traditional ways of videoconferencing (e. g. using videoconferencing rooms). In particular, DMC systems allow ad-hoc teleconferences to be held as well as to jointly process computer-based working documents in a teleconference. These and further features enable employees to coordinate much more efficiently with co-located as well as with distant colleagues.

Based upon the aforementioned expectations, the (economic) motivation for the deployment of teleconferencing technology from the user's point of view is basically twofold:

- *Substitution of face-to-face meetings* promises savings of traveling budget and costly time. If participants of a meeting do not need to travel but can participate from their desk, they have all their documents at hand, can avoid the (often extensive) travel preparation, are not removed from their social environment, etc. [Schindler 91]. Teleconferencing also reduces risks inherent to traveling — due to violence[12], accidents, sickness, etc. — as well as the physical strain on traveling conferees — due to different climates, time zones, food, etc. (e. g. [Maciejewski 91], [Ott 95a]).

[11] Note, however, that as many critics have doubted the usefulness or at least the (broad) applicability of this type of technology for various reasons (see the end of this section).

[12] It is reported that the interest in teleconferencing has increased significantly during the time of perceived decrease in the security of air travel caused by the Gulf war in 1991 [Funkschau 91] [Pagani / Mackay 93].

- *Complementing face-to-face meetings* is a key to improving cooperation between work group members by having more frequent meetings; in particular, meetings can be convened spontaneously. Also, the quality and efficiency of meetings can be increased: experts can be consulted during a teleconference; computer-based teleconferencing systems provide tools for joint problem solving; etc.[13]

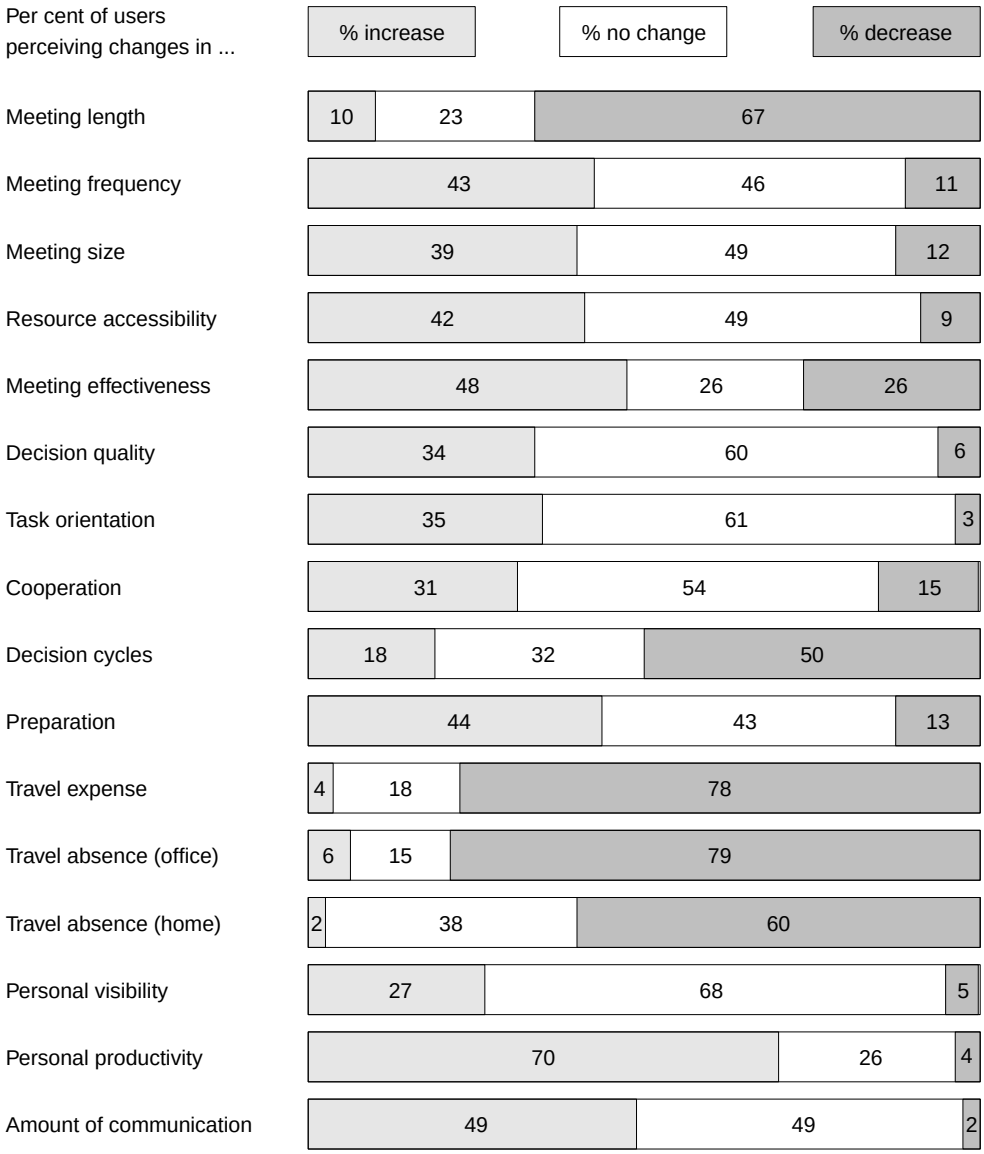


Figure 1.1: Effects of videoconferencing technology [Green / Hansell 84, p. 59]

[13] For a more detailed presentation of advantages and disadvantages of using teleconferences instead of as well as complementary to face-to-face meetings, refer among many others to [Egido 88], [Schindler / Heidebrecht 90], [Clarke 90], [Schindler 91], [Maciejewski 91], [Schindler *et al.* 93], [Schulte 93], and [Ott 95a].

A survey that has been published by GREEN AND HANSELL in 1984 emphasizes many of the effects of (primarily room-based) videoconferencing technology on meetings and participants (Figure 1.1) and confirms the validity of the aforementioned expectations.[14]

Despite the opportunities that teleconferencing technology creates, it should be noted that neither teleconferencing systems in general nor specifically DMC systems do constitute generally applicable technology. They are not suited to all kinds of group interactions: neither to all types of meetings nor to all groups. The author has investigated applicability issues and side conditions for the successful deployment of DMC systems as well as expected benefits and potential shortcomings of this technology extensively in [Ott 95a]. He has concluded that many areas in enterprises are promising candidates for use of DMC technology (provided that organizational factors are taken into account) and that desktop multimedia conferences are likely to complement rather than substitute physical meetings. Subsection 2.2.3 provides a brief overview of the most relevant distinctions between face-to-face meetings and teleconferences and the implications on applicability of teleconferencing technology.

While theoretical considerations of the usefulness of desktop multimedia conferencing form a precondition for exploiting this technology, at the time of writing, however, it is difficult to assess the actual effects of DMC technology deployment in practice: On one hand, existing DMC systems are still imperfect (and not widely deployed in business) so that the basis for a substantial DMC-specific analysis is missing. On the other hand, room-based and rollabout videoconferencing systems differ too much from DMC systems with respect to functionality, user interface, quality, etc. so that previous research results as depicted in figure 1.1 cannot be adopted unreflectedly; such earlier investigations give valuable hints though. Obviously, in order to provide a solid foundation for assessing the usefulness of DMC technology, such systems have to be built in the first place. The aforementioned *expected benefits* indeed do warrant to undertake the implementation of a fully functional DMC system, (not only) as a basis for future investigations. The author's contribution to the research area of teleconferencing and specifically DMC systems is the design and implementation of parts of such a DMC system as is detailed in the following section.

1.3. Scope of this Thesis

It has been stated above that DMC systems integrate many (types of) groupware applications and that it is this integration of services which makes DMC systems more valuable for collaboration compared to earlier (audio or audiovisual) teleconferencing systems. In order to achieve the necessary integration, a suitable communication infrastructure needs to be in place upon which all the groupware applications can be based.

The objective of this thesis is the design and implementation of a *multipoint data communication infrastructure for teleconferencing systems* that is *based on international standards* and provides the basis for groupware applications and their integration into DMC systems. In detail, the scope of the research and engineering work of this thesis is defined as follows:

[14] Note that, as the data for this survey was inquired with regard to room-based videoconferencing systems more than ten years ago, observations concerning desktop systems may differ: on one hand, availability and richer functionality of DMC systems at any time might lead to increased utilization thus emphasizing the advantages and savings. On the other hand, reduced quality of the video images and audible signals might decrease teleconferencing activity leading to contrary observations.

- *Multipoint data communication*

  Teleconferences are expected to often involve more than two participants and several locations and therefore inherently require multipoint communication facilities — in addition to conventional point-to-point communication services. Ideally, no artificial upper bound shall be technically imposed on the number of systems and participants engaged in a teleconference (however, there may well be pragmatic limits due to the subject of a teleconference). In teleconferences, the participating sites may be located in different networks. Consequently, communication in a heterogeneous network environment (concerning topology, transmission characteristics, quality of service, and other properties of the various networks) — *internetworking* — is to be supported. All the services provided by the multipoint communication infrastructure need to be independent of the (mixture of) underlying networks.

  Furthermore, with respect to requirements on communication services, the information exchanged in teleconferences can roughly be classified into what is commonly referred to as *real-time* and as *non-real-time information*.[15] Real-time information in teleconferences comprises mainly audio and video flows. Non-real-time information — also referred to as *data* — consists of (telematic) application data and control information.[16] Within this thesis, only transmission of non-real-time information types — application data and control information — is considered, i. e. the infrastructure developed is not intended to convey real-time information.

- *Infrastructure*

  This thesis aims at providing a generally applicable infrastructure in order to relieve groupware applications from dealing with group communication and control aspects in teleconferences and allow their simple integration into a desktop multimedia conferencing system.[17] The aim to develop a communication infrastructure rather than a specific (groupware application for a) teleconferencing system has several implications. First of all, no assumptions about the environment can be made. The infrastructure has to be portable across different hardware and operating system platforms and has to provide the foundation to achieve interoperability between teleconferencing systems that employ it. Furthermore, the infrastructure has to be complete with respect to the (generic) multipoint communication functionality needed by groupware applications that are used in the context of a teleconference. Finally, the infrastructure should not impose artificial constraints — as far as possible — on the types of teleconferences that can be held using a DMC system based on this infrastructure. Therefore, the *mechanisms* to implement a teleconference have to be provided by the infrastructure while the ways how these mechanisms are employed — i. e. the *policies* — are left to the specific implementations of teleconferencing systems and groupware applications.

---

[15] This very rough distinction is imprecise, and in different research areas of computer science different meanings are associated with the respective terms. Nevertheless, for the purpose of this introduction, this distinction is considered sufficient. A comprehensive discussion of communication characteristics as far as required for this thesis is given in subsection 3.3.1.

[16] This terminology follows the precedence of various international standards including G.701 [ITU-T G.701] as well as the H.300 and the T.120 series of recommendations.

[17] The value of an infrastructure is motivated more elaborately in chapter three.

- *Teleconferencing systems*

  As motivated before, the multipoint communication infrastructure developed in this thesis is targeted at teleconferencing systems. This focus restricts the types of applications that are expected to utilize the infrastructure. This restriction in turn has an impact on the group communication functionality to be provided. Typically, teleconferencing applications are interactive applications operated by humans. Humans are capable of dealing with (transient) system failures or conflicts created by operations carried out by several conferees but they need immediate feedback about the operations they have performed. Therefore, the infrastructure services do not have to address the whole range of consistency requirements (which are often achieved at the expense of interactivity) known from distributed operating and data base systems, but rather aim at functions that allow highly interactive applications to be built. In addition to group communication facilities, functionality specific to teleconferences needs to be covered by the infrastructure. This functionality includes services to configure, initiate, run, and tear down teleconferences.

- *International standards*

  Finally, teleconferencing systems have to operate in an open environment, i. e. systems from different vendors have to be interoperable. As a consequence, the protocols to be used in the infrastructure should not simply be created from scratch. Rather, they have to follow an international consensus, i. e. conform to international standards or other international agreements. Where compliance with standards is not possible — because such standards either do not (yet) exist or do not cover all the required functionality — extensions have to be defined. Service and/or protocol extensions have to be designed such that they do not affect interoperability with non-extended systems. Furthermore, specifications of extensions should be contributed to the standardization process to validate that they address commonly perceived problems and encourage that an international consensus about the respective problem's solution will be established in the future.

Summarizing the above, the subject of this thesis can be defined as follows:

> *Development of an architecture and implementation of this architecture as an infrastructure that*
>
> – *enables the interconnection of virtually any number of teleconferencing systems;*
> – *allows the integration of groupware applications within a teleconferencing system;*
> – *provides the necessary mechanisms for non-real-time group communication, coordination, conference control, and interoperability within a teleconference;*
> – *is easily portable to different hardware and operating system environments; and*
> – *is built on and compliant to existing and emerging international standards.*

The approach taken in this thesis is to define an architectural framework for multipoint data communication in teleconferences that fulfills the aforementioned requirements: the *Multipoint Communication Layer (MCL)*. The MCL consists of four functionally complementary sublayers that cover multipoint internetworking, synchronization, conference administration, and application-specific functionality, respectively. This functional subdivision is based on the analysis of architectures for teleconferencing systems — from the research arena as well as from international

standardization —, on requirement studies for group cooperation systems, and on the experience that has been gained in the past development of DMC system prototypes.

The research and engineering work of this thesis is carried out in the context of the development of a particular DMC system, namely the EURO.VISION system, which has been steadily revised from an initially mostly proprietary teleconferencing system[18] to a fully standard-compliant DMC system. The role of the EURO.VISION system for this thesis is twofold: During the past development of EURO.VISION, a lot of experience in multipoint communication and groupware application design has been gained from early EURO.VISION prototypes. And the EURO.VISION system is the primary target for the deployment of the infrastructure developed in this thesis as one important constituent of an encompassing software development kit for desktop multimedia conferencing systems (DMC SDK).

Finally, it should be noted that the work described in this thesis has been tightly coupled with international standardization. Standardization activities are closely followed and concepts as well as implementation experience of the research presented here are contributed to the international standardization process for teleconferencing systems through the author's active participation in all relevant meetings: in the ITU-T and the IETF[19] as well as in the International Multimedia Teleconferencing Consortium (IMTC) and the MERCI[20] project of the European Union.

1.4. Structure of this Thesis

This thesis contains a conceptual part (chapters two and three) that provides the necessary background and outlines the MCL concepts; an implementation description of the MCL based on international standards (chapters four through seven); and an outlook on future developments in the area of teleconferencing (chapter eight).

In chapter two, an overview of groupware systems is given, along with the most important classification criteria for groupware systems. Based on these criteria, a first characterization of DMC systems is provided. Afterwards, a model for teleconferences is defined from which the range of teleconference types as well as the functionality to be supported by a DMC system are derived. In chapter three, the Multipoint Communication Layer (MCL) is introduced as an architecture for multipoint data communication in teleconferences. Following an overview of related work on communication architectures for teleconferencing, the MCL architecture is outlined and its four sublayers are introduced. Finally, the services provided by the generic sublayers are described in detail. This concludes the conceptual framework of this thesis.

---

[18] This is due to the fact that at during the development of the first iterations of EURO.VISION from 1988 to early 1991, virtually none of the relevant international standards has been in existence. For descriptions of the respective prototypes refer e. g. to [Schindler / Heidebrecht 89], [Schindler / Heidebrecht 90], [Kratz 90], [Beyer 91], [Dzwillo 91], [Modarressi 91], [Ott 91a], [Ott 91b], [Beyer 92], [Bastian 92], [Berresheim 92]. The more recent developments are described in [Schindler *et al.* 94] and [Wenger 95].

[19] The ITU-T and the IETF form the two most important standardization bodies concerned with teleconferencing protocols. Other important organizations such as the ISO (in particular with the DSM-CC work), DAVIC, and the ATM Forum are well recognized but have different focuses for their work.

[20] MERCI — *Multimedia European Research Conferencing Integration* — is project #1007 of the Telematics Applications Programme of the European Union. Among other targets, the MERCI project aims at achieving interworking between DMC systems based on the (currently incompatible) ITU-T and IETF concepts for teleconferencing.

The subsequent chapters deal with the implementation of the MCL infrastructure based on international standards starting with an implementation outline and then elaborating on the implementation of the various MCL sublayers. In chapter four, the international standards upon which the MCL implementation is based — the ITU-T T.120 series of recommendations — are described and the components comprising the MCL services are identified. Moreover, the system architecture of the EURO.VISION DMC SDK is presented and the interfaces necessary for integrating the T.120 infrastructure components are described. This description includes the implementations of the transport protocol hierarchies to access physical networks as well as those of the interfaces to other toolkit components.

Chapter five addresses the implementation of the two lower sublayers of the MCL both of which are functionally covered by the ITU-T Multipoint Communication Service (MCS). The concepts of this recommendation as well as design of the *MCS provider* are presented. Chapter six is devoted to enhancements to the MCS provider which enable it to make efficient use of underlying multicast-capable networks. The general approach and the resulting protocol architecture are outlined and the required protocols as well as their implementation and integration into the MCS provider are described. The implementation of the MCL sublayer for conference control — which is based on the ITU-T recommendation for Generic Conference Control (GCC) — is dealt with in chapter seven. Again, the concepts of the respective recommendation are outlined followed by a description of the design of the *GCC provider*. This chapter completes the implementation aspects of the multipoint data communication infrastructure developed in this thesis.

Chapter eight concludes this thesis with a summary of its research and engineering results. Furthermore, the open issues subject to ongoing and future standardization activities, particularly related to the T.120 series of the ITU-T, are outlined. In this context, a general critique of teleconferencing standardization is given, particularly addressing the (lack of) coordination between standardization groups. Finally, a brief outlook on the future of (desktop) multimedia teleconferencing as rated by the author is given.

Typesetting Conventions

Within this thesis, parts of the text are laid out differently compared to the main text body to indicate a specific meaning. This is done according to the following typesetting conventions:

- Important definitions and results are indented left and right, surrounded by a box, and set using an italic font.
- Quotations that span several lines and quoted definitions are separated from the main text body. They are indented left and right and are set in an italic font at a reduced point size.
- Remarks, additional explanatory text, and examples are separated from the main text body, indented left and right, and are set in a roman font at a reduced point size.

2

Groupware and Desktop Multimedia Conferencing Systems

The introduction has provided a motivation for the support of work groups and has also introduced a category of (software) systems that are designed to provide this support: groupware systems. The support of collaborative processes may take a variety of shapes for which different types of groupware systems are needed. One of these types has already been outlined in the introduction: the class of teleconferencing systems that include Desktop Multimedia Conferencing systems which are the subject of this thesis. This chapter provides an encompassing definition of DMC systems in terms of the functionality they are expected to provide and the types of conferences they are intended to support.

In section 2.1, an overview of the functional types of groupware systems available today is given, followed by a more detailed overview of teleconferencing systems. Then, further categorization schemes for groupware systems are presented according to which the functional types are classified. DMC systems — constituting a hybrid of several types of groupware systems — are characterized with respect to these schemes.

Section 2.2 provides the basis for the description of the functionality to be covered by a DMC system. In this section, a definition of a teleconference along with the teleconferencing terminology used throughout this thesis is provided. Furthermore, an overview of types of conferences is given and with respect to this classification, the target range of conferences to be supported by DMC systems is defined. Finally, limitations restricting the use of teleconferencing technology as well as usability aspects of teleconferencing systems are addressed.

Following this, section 2.3 outlines the design requirements on a DMC system. The functional requirements presented are based on the results of the first two sections; additional design requirements are derived from the necessity to integrate DMC systems with the workplace and from considerations of environmental factors.

2.1. Classification Schemes for Groupware Systems

As noted in the previous section, a multitude of ways can be followed to support group processes. The different approaches towards providing this support have led to the evolution of a variety of groupware systems. Different taxonomies have been developed to categorize groupware systems (refer e. g. to [DeSanctis / Gallupe 85], [Ellis *et al.* 91], [Kaplan *et al.* 92], [Schindler *et al.* 93]):

- First of all, the functionality offered by a groupware system is discussed as the primary categorization criterion in order to provide an overview of what kinds of support is available to which types of collaborative group processes.

Two further classification schemes are included in this section in order to differentiate teleconferencing and in particular DMC technology from other (types of) groupware systems. A classification according to these schemes allows to derive which parts of business processes a groupware system is capable of supporting:

- The categorizations according to the temporal and spatial relationship of the cooperating group members are the most commonly used criteria for classifying groupware systems. The respective attributes allow to delineate the areas of application with respect to the the possible kinds of interactions between the cooperating persons and the geographical settings of these persons.
- The classification following the phases of the collaboration process supported by a groupware application allows to determine which types of groupware applications can be used complementary within a collaboration phase as well as to identify the necessary interfaces to groupware systems used in adjacent phases.

While the first categorization scheme identifies the various functions provided by groupware systems, the other two schemes allow to derive which types of groupware systems are candidates for the integration into a teleconferencing system to enrich its functionality.

- Finally, a categorization of teleconferencing systems is presented that reflects the spectrum of systems developed in the past and shows the functional range the respective system types do cover.

In the following, these four classification schemes are introduced and, at the end of this section, the characteristics of a DMC system are defined with respect to these taxonomies.

2.1.1. Functional Classification

Section 1.1 has already introduced DMC systems as a specific type of teleconferencing systems and has provided a rough outline of the functionality offered by a DMC system. Teleconferencing constitutes only one aspect out of many functional categories of groupware systems. These functional categories are formed by grouping the systems according to the kind of group work they are intended to support and by which means this collaboration is achieved. This subsection presents the functional categories of groupware systems that have evolved besides pure teleconferencing systems. As will be described in subsection 2.1.5, DMC systems incorporate functionality from various of these groupware system classes.

The commonly accepted functional groupware categories to be distinguished comprise those that are briefly described in the following (refer e. g. to [Greif 88], [Ellis *et al.* 91], [Petrovic 92]):[1] [2]

a) *Message systems*

Message systems [Borenstein / Thyberg 88] [Malone *et al.* 89] [Borenstein / Thyberg 90] provide mechanisms for the exchange of messages among individuals (within groups). Messages can be addressed to either a single person or a group. A target group can be selected by an explicit list of persons specified by the sender or an (anonymous) set of recipients of all those interested in a certain topic. Message systems include electronic mail (e-mail)[3] ([RFC 0821], [RFC 0822], [ITU-T X.400]) and bulletin board systems such as the USENET NetNews[4] [RFC 1036]. While early e-mail systems provided solely for the exchange of purely text-based messages, recent extensions to e-mail message formats allow the inclusion of graphics, images, other structured content types, and even audio/video information (e. g. voice annotation) in e-mail messages (for extensions to implement the so-called multimedia mail — the *Multipurpose Internet Mail Extensions, MIME* — refer among others to [RFC 1521], [RFC 1522], [RFC 1911], for privacy extensions to [RFC 1421]).

b) *Group knowledge bases*

Group knowledge bases [Yakemovic / Conklin 90] [Schatz 92] are used for collecting, filing, and retrieving information concerning topics of interest to a working group (the course of a project, rationales for decisions, minutes of meetings, etc.). In their simplest form they are some kind of database application with a focus on the support of groups. The members of the group enter information independently; they are able to retrieve information about certain topics by means of keywords. In more elaborate systems, entries of a knowledge base may contain cross references (in the form of hypertext links) to other entries but also to documents and information ‘‘external’’ to a group’s knowledge base (e. g. literature for further reading, specifications, software packages). These are called hypertext/hypermedia systems. An example for an infrastructure for a large hypertext system is the *World Wide Web (WWW)* [Berners-Lee *et al.* 92]. Within WWW, references in a standardized format — the so-called *Uniform Resource Locators, URLs* — are used to describe in a machine-readable fashion where a particular piece of information can be obtained and which access protocol is required for its retrieval [RFC 1736] [RFC 1737] [RFC 1738].

c) *Workflow management systems*

Workflow management systems [Flores *et al.* 88] [Medina-Mora *et al.* 92] are intended to provide support for (highly) formalized work procedures that require only rare handling of (predictable) exceptional situations. Business processes supported by a workflow management

---

[1] The references provided in following overview refer only to examples of the respective system types; they do not necessarily include the originator of a particular category. For a more complete list of groupware systems of the various types refer to [Malm 94].

[2] There is no particular order imposed on the presentation of the functional types of groupware systems.

[3] E-mail has been considered to be the most widely used groupware system [Bullen / Bennett 90]. In some business areas — primarily in those concerned with computer science — e-mail has become a ubiquitous medium for information exchange, the most obvious indication of which is that e-mail addresses have even made their way onto many business cards in the computer science industry.

[4] USENET NetNews are a widespread means for the impersonal exchange of messages relating to certain topics.

system are usually based on forms; moving and processing of these forms is planned in advance step by step. The resulting work procedures are then automated by the system: forms are forwarded automatically after processing by a specialist. The key ideas behind this are streamlining the business process and minimizing the delays between adjacent processing steps.

d) *Project management systems*

Collaborative project management systems [Kedzierski 82] [Sathi *et al.* 86] are used for monitoring progress of work, adherence to milestones, and so on. Unlike single-user project management systems, in which the data base is maintained solely by the project manager, progress report, exception reports, and so on are directly entered by all the project members. The system compiles the input and provides concise information on the project status. These systems often put emphasis on presentation of the current status to the managers, on automated guarding of deadlines, and on enforcing proper use of the system.

e) *Meeting scheduling systems*

Meeting scheduling systems and electronic (group) calendars [Sarin / Greif 85] [Lange 92] are used to simplify finding a time slot for a meeting that is convenient for all the participants. All members of a group are required to maintain an electronic calendar and keep it up to date with their actual time planning. To convene a meeting, its initiator specifies the intended participants, the time constraints, the duration, and maybe the priority of the meeting to the scheduling system. This searches for a convenient time slot and informs all participants — which then have to confirm the meeting. Conflicts are also reported and require manual intervention by some or all participants.

f) *Group (decision) support systems*

Group support systems [Stefik *et al.* 87b] [Vogel *et al.* 88] have been designed to support meetings in specifically equipped meeting rooms (so-called *electronic meeting rooms*): each place is supplied with a computer, monitor, and input devices; all the systems are interconnected via a local area network; a large main monitor (at the front) visible to all participants is also provided and shows e. g. commonly achieved results. Group decision support systems include cooperative tools for management of the conference, brainstorming, discussion, decision making, voting, etc.; they also provide tools for individual usage such as calendars and memo pads. Of course, much of the functionality of group support systems may also be provided as add-on software for any off-the-shelf workstation.

g) *Co-authoring systems*

Co-authoring systems [Leland *et al.* 88] [Neuwirth *et al.* 90] support cooperative writing and reviewing of documents by two or more persons. Central functions are access control, version management, and avoidance of conflicting modifications in documents. Also, annotations to current revisions and markers to indicate changes from one revision to another are often provided.

h) *Cooperative editors*

Cooperative editors [Dourish / Bellotti 92] [Karsenty *et al.* 93] [Kirsche *et al.* 93] [Peng 93] and other cooperative applications are designed for simultaneous (synchronous) use by two or more persons. They allow sharing information — text, spreadsheets, graphics, CAD designs,

etc., all of which are subsequently referred to as *documents* — and concurrently modifying this information. The contents of a jointly processed document are kept consistent across all participants' systems; changes made by one participant are propagated and visualized to all others, so that they actually see the document in the same state — as if all of them were co-located in front of a single system. This feature is called *What You See Is What I See (WYSIWIS)* [Stefik *et al.* 87a]. Applications that are explicitly designed for the support of collaboration (in a distributed environment) are called *collaboration-aware* applications.

i) *Remote control and application sharing systems*

Remote control and application sharing systems[5] are used to provide remote access to an application or the entire workstation. These systems have been originally developed to support remote maintenance of application and computer systems. However, due to their nature of providing remote access to virtually any kind of application, these systems can be used to implement joint editing with existing applications as well. Because of this property, application sharing systems have been increasingly integrated into computer-based audio(visual) teleconferencing systems. The main potential of this groupware system type is that existing single user applications (that are termed *collaboration-transparent* or *collaboration-naïve*) can be used for collaborative work in teleconferences without having to be modified.

j) *Audio and video communication*

Groupware systems that cover only audio [Swineheart 91] [Jacobson / McCanne 92] [Intel 96] [Handley 96], only video [McCanne / Jacobson 95], or audiovisual [Turletti 93] [TELES 95] communications need no further explanation. It should be noted, however, that such systems are rarely found as stand-alone applications — if not intended as substitute for a telephone. Instead, they are typically part of virtual office environments or teleconferencing systems (see below).

The previously described groupware system categories are elementary in that none of these categories is contained within another. Systems of the respective categories may be used as stand-alone applications, but may as well serve as building blocks for more complex groupware systems that integrate several types of groupware functionality as described in the following:

k) *Office information systems*

Office information systems [Ellis / Nutt 80] combine the functionality of some of the above systems — data base access, simple calendars, simple co-authoring tools, shared access to files, procedures for processing and forwarding information — and add simple communication functions such as e-mail, file transfer, telefax, telephony support (e. g. address book), answering machine, Internet access, etc. The goal is to provide the user with a seamless integration of the various office tools.[6]

---

[5] For an overview of remote control systems refer to [Wilde 92] and [Schmitt 94]; examples of screen/window sharing systems are desribed in [Stefik *et al.* 87a], [Lauwers / Lantz 90], [Lauwers *et al.* 90], [Ott 91a], [Ott 91b], and [Romano 97].

[6] Note, however, that many of today's widely available office information systems are mainly collections of largely independent programs that lack the desirable integration [Schindler *et al.* 93].

l) *Virtual (office) environments*

Virtual environments [Root 88] [Kraut *et al.* 90] [Mantei *et al.* 91] [Dourish / Bly 92] [Dourish 93] are intended to improve social interaction, particularly among those employees that otherwise would only meet rarely or not at all (e. g. due to being located on different floors, in different buildings, at different sites). Such systems provide the functionality to establish audiovisual communication links for an ad-hoc conversation among any two or more persons. In addition, they may provide indicators for the accessibility of persons showing if someone is absent from the office, busy, willing to accept an interruption, and so on (e. g. using the ''office door metaphor'' [Buxton 94]) to allow certain social conventions to be retained.[7]

While the categories k) and l) already include functionality of some groupware system types, the most encompassing integration of groupware functionality is found within some system types belonging to the class of teleconferencing systems:

m) *Teleconferencing systems*

As already briefly introduced in chapter one, teleconferencing systems [Schindler / Heidebrecht 90] [Ishii 90] [Watabe *et al.* 90] [Ishii *et al.* 93] [Schindler *et al.* 94] [Wenger 95] [Microsoft 96a] are intended as a technical means that allows to conduct meetings without requiring participants to be physically co-located. Teleconferencing systems often provide audio communications as the basic functionality. Depending on the type of teleconferencing system, video communication facilities may be integrated into such a system as may other of the aforementioned groupware application categories. The various teleconferencing system types developed so far are introduced in the subsection 2.1.4.

From the above considerations, it has become apparent that different types of groupware systems are designed to be applied in different settings and to accommodate different needs of the group members. The target application areas of groupware systems are typically classified according to two criteria that are described in the subsequent two subsections: temporal and spatial co-location of users as well as the collaboration phase(s) the respective groupware systems intend to support.

2.1.2. Time-Space-Matrix

In the previous subsection, a functional classification scheme for groupware systems has been introduced. In this scheme, the groupware systems have been grouped according to the type of tasks they do support and the means provided to achieve this support, i. e. the target area of application. The various functional types of groupware systems designed to accommodate different needs of the group members have been described. The needs of the group members, however, do not only depend on the type of task they want to accomplish, but also on the current work group setting, including which work group members are in the office, which of them are available at a given point in time for a discussion, etc. Different ways of collaboration may be pursued — which in turn may require different or additional groupware systems — to perform the same task

[7] Virtual office environments (such as CAVECAT [Mantei *et al.* 91] and Cruiser [Root 88]) are very similar to teleconferencing systems concerning the technology in use. Often, such systems incorporate teleconferencing functionality because their primary function — e. g. creating awareness about the activities of colleagues — may be used to determine when other persons are available for ad-hoc teleconferences.

if the work group settings differ. Teleconferencing systems, for example, are intended for other group settings than are electronic meeting rooms, and even the various teleconferencing system types are designed to accommodate different situations.

This subsection addresses those two parameters of work group settings that are most fundamental from the groupware technology point of view: time and space. That is, collaboration processes in work groups and groupware systems supporting them are classified with respect to the temporal and spatial relations of the involved individuals. The temporal relation reflects whether or not the cooperating partners interact simultaneously with one another while the spatial relation addresses the physical proximity of the group members.

Simultaneity of interaction

This classification refers to whether or not group members do interact at the same time and immediately with one another during collaboration. In essence, two modes of interaction can be distinguished:

- *Synchronous* cooperation (*same time*) means interaction among the collaborators at the same time, e. g. by holding a conference, placing a phone call, or talking to one another face-to-face.

- *Asynchronous* cooperation (*different times*) refers to cooperation processes where simultaneous presence of the collaborators is not of importance because intermediate means are used to convey the information to be exchanged. Examples are exchanges of internal memoranda, letters, and facsimile messages.

Both forms of cooperation are complementary: synchronous cooperation allows for immediate feedback among the participants and thus provides a larger potential for (fast) problem resolution; asynchronous cooperation allows — by not requiring simultaneous participation of the involved persons — interactions that otherwise would not be possible at all.

Synchronous groupware systems provide means for simultaneous interaction of the group members and in general require immediate feedback to be most effective. Asynchronous groupware systems are designed to be applied if synchronous interactions are not possible and/or for performing tasks that do not require synchronous interaction. *Mixed* systems [Rodden / Blair 91] refer to hybrid groupware systems that combine both asynchronous and synchronous groupware functionality within a single system.

Physical proximity of the collaborators

This attribute classifies work groups according to the distance between its members. Although a much more fine-grained distinction is possible and sometimes useful, the following two cases are commonly distinguished:

- In *physically co-located* groups the collaborators are located at the *same place* (i. e. in the same room rather than the same building) and can interact directly with one another (without the elementary need for technology support of some sort). Examples are conferences as well as notes left on a bulletin board or on a colleague's desk.

- *Distributed / remote* groups have its members located at *different places*. Therefore, the group members have the fundamental need for (technical) support to be able to establish a cooperation relationship at all. The geographic distance between members can be overcome e. g. by phone calls and letters or facsimile transmissions.

Groupware systems designed for physically co-located groups are tailored to be exclusively used by persons gathered in the same room. In contrast, groupware systems for distributed groups intend to (temporarily) overcome physical distance and establish a collaboration environment as close as possible to that of co-located group members. For several types of groupware systems, the issue of physical proximity is irrelevant; either because the groupware system is asynchronous in nature (such as e-mail), or because it is of no interest to the groupware system whether the common medium for discussion is established through physical proximity or by means of other groupware systems such as teleconferencing systems (as is the case with cooperative editors).

Combining Time and Space Aspects

The most widespread model that combines both temporal and spatial relationships among the cooperating partners is the *Time-Space-Matrix* [DeSanctis / Gallupe 85] [Johansen 88] as depicted in figure 2.1 which distinguishes four system categories. *Face-to-face interaction* refers to physical meetings of persons while *synchronous distributed interaction* denotes collaboration by means of telecommunication technology (e. g. teleconferences). In both cases, the group members interact at the same time with one another. If simultaneity of interaction is not provided, the terms *asynchronous interaction* and *asynchronous distributed interaction* are used to refer to collaboration between co-located and remote group members. However, the distinction between distributed and co-located asynchronous systems is somewhat artificial[8] as the type of interaction between users is — due to the asynchronous nature of these systems — basically independent from their location. For some system types (e. g. e-mail) no distinction between co-located and distributed groups can be made at all. Today, however, certain system types (e. g. project management systems) are more likely to be deployed in co-located groups. This is mainly due to shortcomings of current synchronous groupware systems that do not yet allow distributed groups to work as efficient as co-located ones. In the future, these differences are expected to decrease.

Figure 2.1 also includes a possible assignment of the functional groupware categories presented above to (one or more of) the four quadrants of the Time-Space-Matrix. The functional categories are assigned to the quadrants according to their primary intention. If a groupware system type may be used asynchronously then it is assigned to the asynchronous interactions. In the ‘‘same time’’ quadrants only those categories show up that do require synchronous interaction for meaningful collaboration. Of course, asynchronous groupware systems may be used during synchronous collaboration but they are not *primarily* intended to be used that way. With respect to the space dimension, as the figure shows, for asynchronous collaboration, no distinction is made between ‘‘same place’’ and ‘‘different places’’ applications for the aforementioned reasons. For synchronous interaction, groupware system categories that *enable* collaboration at different places are listed only for the ‘‘synchronous distributed interaction’’ quadrant. All others may be used in co-located as well as in distributed settings and are therefore included in both synchronous quadrants.

[8] And, from the user’s point of view, de-facto non-existent!

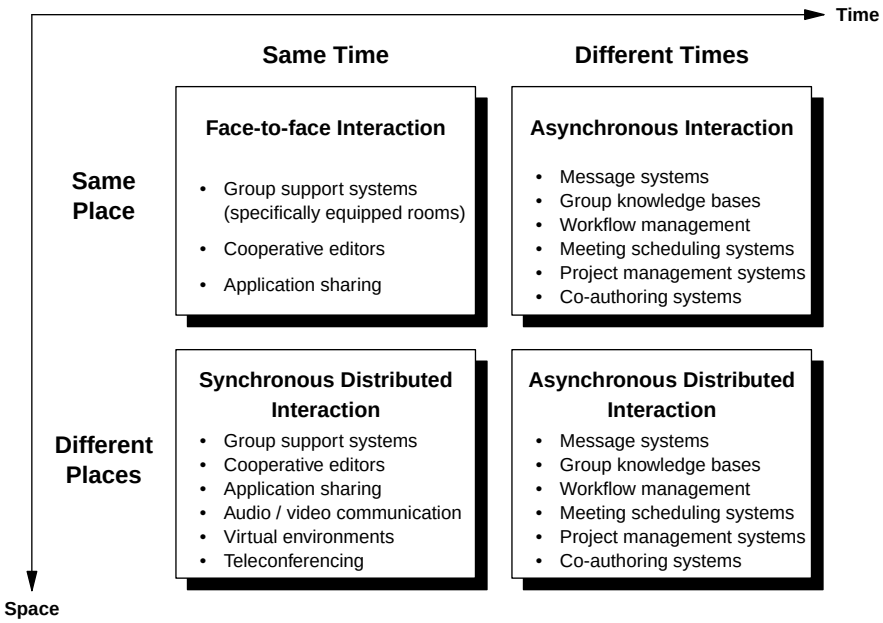


Figure 2.1: The Time-Space-Matrix

2.1.3. Phases of (Tele)Cooperation

In work groups, synchronous and asynchronous collaboration are not independent of one another (and neither should be the tools). Rather the group work can be subdivided in several phases as depicted in figure 2.2 [Schindler *et al.* 93].[9] These four phases constitute a continuous process throughout the lifetime of a group. Depending on a group's tasks the asynchronous or the synchronous collaboration phases may be predominant.

i) During the *asynchronous cooperation phase*, all group members are working individually. They occasionally exchange information about their work with other group members: results, progress reports, problems, etc. This is done either using conventional means for office communication (letters, memoranda, facsimile) or by means of asynchronous groupware systems. The asynchronous phase is the usual form of work within a work group, and most of the actual work (in terms of quantity) is carried out during this phase.

   This phase continues until either a date for a regularly scheduled conference approaches or one individual discovers a need for direct interaction with the others to clarify some issues, resolve problems, etc. that cannot be dealt with in an asynchronous fashion. The latter case requires a conference to be scheduled (on a short-term basis). Either event marks the transition to the next phase.[10]

[9] A comparable model has been described independently by [Bostrom *et al.* 91]. Their model distinguishes *pre-meeting*, *meeting*, and *post-meeting* with the meeting phase subdivided into *startup*, *during*, and *windup*.

[10] Note that in this model, placing a phone call — which is a synchronous means for collaboration — forms one loop through all four phases. This indicates that switching from one form of collaboration to another and back is done frequently and that in many cases the involved persons are not even aware of a switch taking place.

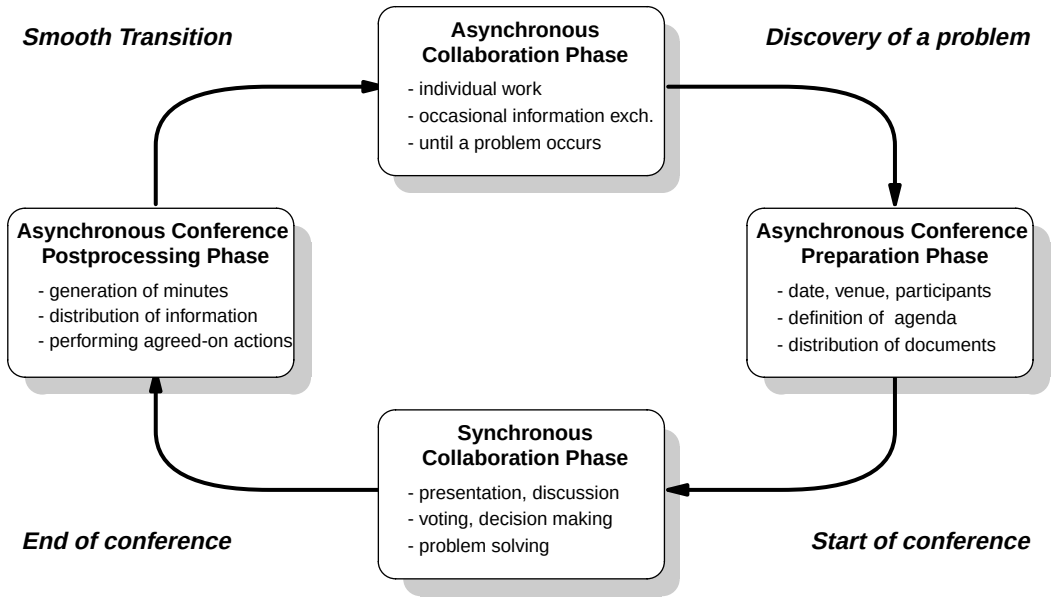


Figure 2.2: The four phases of group collaboration

ii) The *asynchronous preparation phase for the synchronous phase* is used to make all the necessary arrangements for the conference itself. In this phase the problems to be solved are identified and the persons required to take part are determined. Date and venue of the conference are set, an agenda is defined. The participants are informed about the conference. The working documents required for preparation of the conference are distributed, too.[11]

   This phase ends with the commencement of the conference leading to the synchronous collaboration phase.

iii) The *synchronous collaboration phase* covers the entire course of a (tele)conference with two or more participants. This phase may include presentations, discussions, reviews, votings, cooperative work sessions (e. g. joint editing), etc. The synchronous cooperation phase contains the largest problem resolution potential of all phases due to the direct interaction among many or all of the group members: during this phase the group addresses those problems that could not be solved individually or by asynchronous means. Due to the immediate interactions, the synchronous collaboration phase is also best-suited for updating group members about latest developments, current project status, etc. Therefore, this cooperation phase is crucial to the entire process of group collaboration.

   When the meeting is closed the conference post processing phase is entered.

iv) The *asynchronous conference post processing phase* after the end of a (tele)conference deals with information distribution, assessment of results, and preparation for the execution of decisions. During the post processing phase, the conference results are fixed and agreed upon in

[11] Note that in case of an ad-hoc telephone call, the asynchronous preparation phase mainly consists of the decision whether and whom to call, determining the callee's phone number, and having an ''agenda'' in one's mind (namely what to discuss). Also, the matter under discussion is often well-known to both partners or is easily explained so that distribution of documents may not be needed.

the minutes to achieve a coherent view of the outcome. This information is distributed to all participants as well as to other persons that need to be informed about the results of the conference. These results may include in particular assignments of work items, work schedules, etc. that determine the future work within the group.

The individuals continue their respective parts of the work (partially based on the conference results). This seamlessly leads back to the asynchronous cooperation phase.

In this idealized model, each activity of any of the group member(s) may be assigned to one of the four phases, and the phases are passed through in a round-robin fashion. In reality, the borders between these phases are soft: there need not be a unique assignment of a particular activity to exactly one of the phases, individual phases may overlap with others. Depending on the situation, phases may also be very short or even left out entirely.

Table 2.1 relates the functional groupware categories presented in subsection 2.1.1 to one or more of the four (tele)cooperation phases in which they are most likely to be used — assuming a reasonable work division across the four phases. The table also provides examples indicating how the functional groupware system types may be applied during the respective collaboration phases.

Cooperation phase	**Functional groupware category**	**Example**
Asynchronous phase	Message systems	exchange of notifications
	Group knowledge bases	common storage of documents
	Workflow management systems	forwarding and processing of reports
	Project planning systems	maintenance of project status, milestones, etc.
	Co-authoring systems	joint preparation of documentation
Preparation phase	Message systems	coordinate preparation (who, where, what about?)
	Group knowledge bases	make necessary documents available
	Project planning systems	extract important topics to be discussed
	Meeting scheduling systems	agree on date and time
	(Virtual office environments)	ad-hoc initiation of group meetings
Synchronous phase	*Synchronous* co-authoring systems	jointly review a report
	Group (decision) support systems	support for brainstorming, voting
	Cooperative editors	sketch ideas, take minutes
	Application sharing systems	jointly solve problems, review documents
	Audio and video communication	provide audio(visual) communication
	Teleconferencing systems	*integration of the above functionality*
Postprocessing phase	Message systems	coordinate actions
	Group knowledge bases	distribute meeting report
	Project planning systems	update the project plan, milestones, status

Table 2.1: Functional types of groupware employed during the four cooperation phases

2.1.4. Types of Teleconferencing Systems

The previous subsection has provided a high-level overview of the various types of groupware systems — which included teleconferencing systems as one functional class. In contrast to most other classes, teleconferencing systems potentially integrate the functionality of several groupware system types within a single system. This subsection introduces the various types of teleconferencing systems in more detail and briefly indicates which groupware functionality is included in each teleconferencing system type.

The history of teleconferencing may be dated back the 1920s when the idea of adding video communication facilities to simple telephony in order to improve interpersonal telecommunication was first implemented. In 1929, the German *Reichspostzentralamt* demonstrated a *Fernseh-Sprechanlage* (literally translated: ''television intercom'') for the first time [Lautz 95]. Also since the 1920s, thoughts about a *Picturephone* have been around in AT&T Bell Laboratories and a first prototype of an apparatus for *two-way television* was demonstrated in 1930 [Ives 30]. In 1945, BUSH envisioned the MEMEX system for telecooperation, consisting of a scanner, printer, a mass storage medium, two screens, and a keyboard [Bush 45] — that was never actually built due to missing technical prerequisites (refer also to [Wenger 95]). The idea of video telephony as regular means for interpersonal communication was pursued since the early 1960s when experiments were conducted at AT&T (reported in [Egido 88] [Tang / Isaacs 93]). The *Picturephone* was publicly introduced at the 1964 World Fair [Egido 88] (refer also to [Molnar 69], [Dorros 69], and the other articles published in the AT&T Bell Laboratories Record special issue on the Picturephone in mid 1969). For further information on history of video telephony and videoconferencing refer to [Egido 88], [Schulte 93], [Wenger 95], and [Lautz 95].

Table 2.2 gives an overview of the entire spectrum of teleconferencing systems that were developed since then and shows possible ways of their implementation. One distinction is made according to the number of participants the respective system type is intended for: the ''two persons'' category refers to point-to-point communication relationships with one person acting at either side. ''More than two persons'' can be accommodated by systems that allow for several persons at one or both side(s) of a point-to-point connection as well as by interconnecting more than two systems of either type (e. g. by means of some central unit). The other dimension of this matrix refers to the media types transmitted within the teleconference: ''data-only communication'' systems make use of data channels transmitting textual information typed in by the users as the single means of interaction.[12] Audio-based teleconferencing systems transmit audio information as the primary means of interaction and may include — depending on the system type — video and/or data communication as well.

In the following, a brief description of the various system types shown in table 2.2 is given (for more information on the respective types refer e. g. to [Gerfen 86], [Schulte 93], and [Wenger 95]).

[12] Groupware applications such as shared editors are not considered to be teleconferencing systems: although such editors might be (mis-)used for text-only communication, they are designed as a complement to audiovisual communication rather than as a substitute.

Number of Participants	Audio-based communication	Data-only communication
two persons	• Telephony – via stand-alone telephone – with computer support • Video telephony – stand-alone system – computer-integrated	• Text communication – computer-based
more than two persons	• Audio conference – via stand-alone telephones – with special audio equipment – with computer support • Video conferencing – conference room – rollabout system – using video phones – computer-integrated (desktop) • Audiographic teleconferencing – conference room – computer-integrated (desktop) • Multimedia teleconferencing – computer-integrated (desktop)	• Computer conference – computer-based

Table 2.2: Overview of teleconferencing systems

- *Text communication / computer conference*

Tools of this type accept textual input from the participants via the keyboard and display all the input to all participants. Examples are so-called chat programs such as the UNIX programs *talk(1)* or *phone(1)*. As the information exchange is comparably slow and has little in common with human-to-human communication in face-to-face conferences, these systems are not considered any further in the remainder of this thesis.[13]

- *Telephony / audio conferencing*

Stand-alone telephones do not require any explication, neither need phone calls. Computer support for telephony refers to (in Germany usually ISDN-based) telephones that can be controlled by the computer to provide enhanced telephony services.[14] Examples for such services are provision of address books with automatic dialing, answering machines, callback lists, etc. Finally, dedicated conferencing devices may also be used to achieve a higher audio quality, e. g. table mounted devices that have multiple microphones, a loudspeaker, and acoustic echo cancellation functionality.

[13] Note, however, that it is well recognized that text-only or video-only (for sign language) communication systems are important to accommodate deaf people.

[14] Usually such phones are either connected to the phone line via some computer or modem hardware or can be controlled from the computer via the telephone PBX.

Audio conferences are established by means of a central conferencing unit — either within a telephone PBX local to one of the participants or offered externally by a conference call service provider. In any case, the central system either *mixes* the audio input from all or a subset of the conferees and returns the resulting signal to all of the participants. Or the central system broadcasts a single incoming audio stream to the entire conference (*switching*).

- *Video telephony / videophone-based conferencing*

Video phones allow audiovisual communication between the communicating partners via (ISDN) phone lines. Stand-alone video phones are devices consisting of a receiver, a keyboard, a small (color) monitor and a video camera. Computer-integrated video phones are implemented by add-on hard- and software for audio and video capturing, processing, and replay, as well as for connecting the workstation to the network.

Both types of video phones may be used to hold (multipoint) videoconferences by connecting — similar to the establishment of an audioconference — to a specific system, a Multipoint Control Unit (MCU). One MCU or a set of MCUs together act as a central system interconnecting virtually an arbitrary number of videophones. MCUs are responsible for mixing or switching audio as well as video information streams.

- *Video conferencing*

The traditional video conferencing makes use of specific videoconferencing rooms or rollabout systems. Depending on the room size, up to some ten conferees may typically participate at one site. Both types of equipment may include other telematic equipment (e. g. fax machines for document exchange) or document cameras as well.

These types of videoconferencing are usually employed by point-to-point interconnection of exactly two videoconferencing systems. However, multipoint configurations either using MCUs as mentioned above or interconnecting all sites in a full mesh are technically possible, too.

- *Audio/audiographic teleconferencing*[15]

Audiographic teleconferencing systems provide high-quality audio transmission with no support for full motion video. These systems are implemented either as specially equipped conference rooms, as rollabout systems or are built as computer-integrated systems. Besides the transmission of speech, audiographic systems include telematic equipment (e. g. for facsimile transmission) and/or are capable of capturing still images: of participants, of documents, or of other objects of discussion. Telepointers or annotation tools (for use in conjunction with captured images) may be supplied as well.

Like video conferencing, room-based or rollabout systems can accommodate several participants at each site. For multipoint conferences, either an MCU (which in this case has to implement data forwarding as well) or a fully meshed networking topology is needed.

[15] The audiographics conference is one particular type of *augmented audio conferencing*, that provide functionality somewhere in-between audioconferencing and audiovisual conferencing. Other types of augmented audio conferencing are *slow-scan-television* and *freeze-frame television* [Schulte 93]. In the following, only the most widely known term, ‘‘audiographics conference’’ will be used.

- *Desktop multimedia teleconferencing*

Following the outline given in the introduction, desktop multimedia conferencing systems provide audiovisual communication services, augmented by simple data communications, cooperative (groupware) applications (such as joint editors), and include elaborate mechanisms for conference control.

Like telephones and videophones, DMC systems are intended to serve exactly one user so that a system by itself allows for point-to-point communication only. Like in the previous scenarios, multipoint conferences are typically established by means of an MCU. However, if communication takes place via the Internet or in intranetworks, no dedicated central system may be needed at all [Casner / Deering 92] [ITU-T H.323]. The following subsection provides a more complete characterization of DMC systems with respect to the classification schemes presented so far.

2.1.5. Conclusion: Characteristics of a DMC System

The previous subsections have introduced various classification schemes for groupware systems in general and have also provided an overview of the various types of teleconferencing systems. Based on these classifications, this subsection provides a first characterization of DMC systems as a particular form of teleconferencing system. While the temporal and spatial relations between the involved persons are common to all types of teleconferencing systems, the groupware functionality included is specific to DMC systems. It should be noted, however, that the following description of included functionality is still not exhaustive and will be completed in the following sections.[16]

- *Temporal relationships of the involved parties / phases of collaboration*

  DMC systems are designed for synchronous collaboration and are not suited for support of asynchronous interactions. They are intended — besides physical meetings (and possibly phone calls) — as the central means for interaction during the synchronous collaboration phase.

  DMC systems have to support the preparation and the postprocessing phases as well, and they have to allow incorporation of information of the asynchronous phase in order to provide for seamless transitions towards and from the synchronous phase. These aspects, however, are beyond the scope of this thesis which deals only with aspects of synchronous communication.

- *Spatial relationship of the involved parties*

  For the application of DMC systems, it is virtually irrelevant whether or not the collaborating partners are co-located. The primary goal of DMC systems, however, is to overcome any geographic distance.

- *Functional types of groupware systems*

  DMC systems are hybrids that integrate several groupware systems into a single system. As the focus of DMC systems is on synchronous collaboration, those functional groupware

[16] For example, one important ingredient of DMC systems is security functionality. However, as this is neither a functional groupware category nor another classification scheme, security functionality has not been addressed so far. This applies to other types of functionality as well.

classes designed for synchronous use are the primary candidates for incorporation into a DMC system:

- audiovisual communication facilities as core functionality (of type j),
- remote control and application sharing systems (type i),
- cooperative editors (type h) possibly including synchronous parts of co-authoring tools (type g), and
- group (decision) support systems (type f).

As stated before, for the integration with other phases of the collaboration process, asynchronous groupware systems like meeting scheduling systems (type e) or co-authoring systems (type g) are of relevance, too, but their further consideration is beyond the scope of this thesis.

To summarize the above, for the purpose of this thesis, a desktop multimedia teleconferencing system can be viewed as a hybrid of a variety of synchronous groupware systems that are integrated into a workstation. The entire functionality of the DMC system does not only include the functionality of its constituents but also integrates all functions and presents it to the user as a seamless whole.

2.2. Modeling a Teleconference

While the previous section has outlined the technical characteristics of a DMC system, this section is to provide the foundation for more precisely deducing the functionality to be included in a DMC system. As repeatedly mentioned, the aim of DMC systems — as well as teleconferencing systems in general — is to provide a means allowing to conduct meetings without requiring physical co-location of the participants. Obviously, a necessary step towards identifying the functionality to be covered by a DMC system is to understand physical conferences. As a basis for developing a model for teleconferences, the precise meaning of the term *conference* within this thesis is defined, followed by a description of a typical conferencing scenario which is to be supported by DMC systems. From this scenario, the key characteristics of a conference are extracted and ways for their reproduction in teleconferences are described. A discussion of how conferences can be classified and which of these conferences are to be supported by DMC technology completes the modeling of (tele)conferences. These considerations form the basis for the following section that details the functionality to be provided by a desktop multimedia conferencing system in order to accommodate the target conference settings.

As a starting point for modeling a teleconference, the following definitions of the term *conference* are taken from the Oxford English Dictionary and the Merriam Webster:[18]

> *"(...) 4.a) The action of conferring or taking counsel, now always on important or serious subject or affair; 'the act of convening on serious subjects, formal discourse', but formerly more in the general sense of: conversation, discourse, talk; (...) d) In modern legal practice, a meeting for professional advice at which only one counsel is present: distinguished from consultation (...) 6. A formal meeting for consultation or discussion, e. g. between the*

[18] Other dictionaries give similar but less extensive descriptions [Chamber 55] [Americana 62] [Collier 94].

representatives of two sovereign states, the two Houses of Parliament or Congress, the representatives of societies, parties, etc. (...) 8. The action of conferring, bestowal" [Oxford 89]

"(...) 2: the act of consulting together, usually formal interchange of views; 3: a meeting for consultation, discussion, or interchange of opinions whether of individuals or groups: a: a meeting of representatives of different nations to discuss international problems or determine general policy; b: a meeting of members of two branches of a legislative (...); c: a meeting of those members of a legislative body who belong to the same party in order to plan party policy but without binding its members to a certain course of action; (...); 6a: an informal meeting for purposes of intense instruction between a teacher and a student or a small group of students; 6b: a brief and intense course often held during a vacation emphasizing practical experience and demonstration and usually attended by adults; (...)" [Webster 86]

These definitions already indicate that the term conference is used to cover a variety of different settings — for some reason not explicitly including scientific conventions which are commonly also referred to as conferences. The fact that the term ''conference'' may refer to a diversity of meetings is also supported by the intuitive understanding of a conference and the daily use of this term. The former use of ''conference'' has an even broader scope including many sorts of communications between two or more persons regardless of the subject. In any case, conferences are convened to ''discuss matters of common interest'' [Chamber 55]. As many conferences are not restricted to discussion — often a lot of work gets done during conferences —, the more appropriate wording is that conferences are intended ''to deal with, or work on, matters of common interest''.

Altogether, this indicates that conferences are not necessarily restricted to pure discussions nor need they take place as formal events. Basically any kind of meeting of two or more persons may be regarded as a conference. Examples are lectures at a university, discussions in the parliament, discussions between a banker and a client, the annual review within a company, workshops, project or workgroup meetings, brainstorming sessions, etc. Each of these physical conferences is a potential candidate for substitution by a teleconference. This yields a variety of potential application fields for teleconferencing technology.

Roughly the same basic type of technology is likely to be used for most group activities involving personal interaction of some sort: technology enabling audiovisual communication over distance and potentially allowing for further interactions.[19] However, the various types of conferences mentioned before are of quite different nature and consequently place different demands on a teleconferencing system. In other words, the conference semantics[20] may differ heavily in different settings and hence make different types of system support desirable. As a consequence, designing an entirely generic system to support any kind of conference is not a suitable approach, if

[19] The provision of virtual (office) environments is one example for using teleconferencing technology for a different purpose. Broadcasting seminars and workshops (in particular, IETF work group sessions) is to date the widest and most famous deployment of teleconferencing technology in the Internet [Casner / Deering 92] [Kirstein *et al.* 93] [MERCI 95].

[20] Further variations include the means of interaction, the user front end (e. g. workstation vs. TV set) and other equipment, the desired quality of voice and video, and so forth.

possible at all. Instead, the wide range of application areas — i. e. conference types — needs to be narrowed into a manageable portion in order to define a set of target conference types that shall be supported by DMC systems.

SCHINDLER identifies three different types of application areas for teleconferencing technology [Schindler 92a]:

- *Persons-based (or "plain") videoconferences*

  Plain videoconferences refer to teleconferences where only the high-quality transmission of video and voice of the communicating partners is of importance. Examples are TV talkshows or interviews with remote participants, corporate television broadcasts, and traditional video conversions between participants in two videoconferencing rooms.

- *Work-based (or "professional") videoconferences*

  In contrast to plain videoconferences professional ones focus on the (computer-based) work to be accomplished by using the conferencing system. Audio communication remains crucial as primary means for conversation and video remains an important additional communication channel. But both media may be reduced in quality to give room for transmitting (changes to) documents currently being discussed, editing operations, and so forth.

- *Entertainment / leisure-based (or "social") videoconferences*

  Finally, (probably more advanced) teleconferencing technology can be used as a basis for entertainment: examples include new types of games and distributed virtual reality applications. Also, creating and maintaining social contacts and providing human care especially for the elderly and for people with disabilities is an important application for this kind of teleconferencing.

With respect to these three categories, DMC systems are designed to support professional videoconferences that are subsequently also referred to as *business conferences*. However, the term ''business conference'' so far only defines that primarily serious matters are discussed during such conferences and that additional means for interactions beyond audio and video transmission are requested. As will be seen in the following, business conferences also may take many different shapes.

This section precisely defines the target set of conferences desktop multimedia conferencing technology — as understood in this thesis — is intended to support. Subsection 2.2.1 describes a typical business conference scenario and shows how its various elements are represented in a teleconference. Thereby, this subsection defines terminology and identifies required functionality. What is left open in this description, however, is how such a conference is run, how many persons may take part, how the conferees interact, etc. All these issues are dimensions of describing a business conference. Subsection 2.2.2 introduces an encompassing set of variables for describing (business) conferences from a technical point of view. Based on these variables, the target range of conferences to be supported by DMC systems is identified. Finally, subsection 2.2.3 addresses the applicability of DMC systems to meeting scenarios from the user rather than the technical perspective thereby completing the modeling aspects of teleconferences. Based on the outcome of these subsections, the next section derives the functionality a DMC system has to provide in order to be capable of adequately supporting business conferences between work group members.

2.2.1. Description of a Business Conference

This subsection outlines what is considered a business conferencing scenario and lists all the — with respect to this thesis — important features that are to be reproduced in a teleconferencing system. An encompassing description of a business conference scenario[21] is given below which deliberately includes repetitions of various facts that have already been stated earlier (or seem intuitively clear). All the relevant characteristics are highlighted by use of an italic font.

> Business conferences take place in a *meeting room* and involve *two or more persons* in a *synchronous cooperation*. Such conferences are *scheduled* for a certain *date and time* and take place at a certain *venue*. The participants are in general *notified in advance* about the conference and receive *additional conference-related information* such as the agenda and documents needed for preparation.
>
> The *conference host* provides all the necessary facilities for the conference (e. g. rooms, equipment) and is also the one to control *access to the conference*, i. e. admits only authorized persons to the conference room. Within the conference, a dedicated conferee may be appointed to be the *conference chairperson*; other participants may be assigned specific *roles* as well (e. g. secretary, presenter, observer); tasks and privileges are associated with these roles. The *conference policy* describes how the conference shall be run, who is admitted, which conferees have which roles, etc. The collection of all the information above describing a conference is called the *conference profile*.
>
> The conferees gather in the meeting room; once admitted, they may *come and leave at will*. As soon as all (required) conferees are present, the conference is formally *convened*. The formal conferencing phase lasts until the conference is explicitly *adjourned*. Outside (as well as during) the formal conference phase *side discussion* among small groups of participants may take place. If during the conference course further persons are needed (e. g. experts to give advice on a certain issue) these persons may be *invited* into the conference; of course, persons may also be invited for the entire duration of the conference. Participants may be *excluded* from the discussion if this is deemed necessary. Each conferee (re-)entering an on-going conference needs to be *updated on the current state of the discussion*.
>
> Conferences may be split into small working groups (‘‘*subconferences*’’) that are later rejoined again.[22] Also, several conferences may be going on in *parallel* (e. g. in the same or an adjacent building). Each conference is separated from the outside and from other conferences, meaning that *privacy* within a conference room in ensured.
>
> The intention of the conference members is to accomplish a (set of) task(s) in a cooperative manner. They *exchange papers* (and other items) and *interact* synchronously in a variety of ways: they do presentations, discussions, drafting, voting, decision-making, etc. To facilitate these interactions, they make use of various *meeting aids*, such as whiteboards, blackboards, overhead / video / slide projectors, audio replay systems, etc. If the conference is run by a chairperson, this person decides who may talk and who may access the meeting aids (*floor control*) to guarantee an orderly conference course.

The important characteristics of a business conference as highlighted in the above description are discussed in the following in the order of their appearance. For each property, its meaning for physical conferences is identified and the requirements for providing the respective functionality in teleconferences are derived.

---

[21] Note that this description does not in any way restrict the number of conferees, the styles of interaction, etc. This is subject to the following subsection.

[22] However, as a subconference is likely to be held in a different room as a conference of its own, subconferences are not considered separately.

- *Meeting room*

As stated before, in physical conferences the participants come together in a room in order to collaborate. The meeting room forms the medium in which all interactions take place: the conferees see and talk to one another, make use of meeting aids, physically distribute conferencing materials, etc. For teleconferencing, this medium is replaced by suitable telecommunication networks and protocols (together with appropriate equipment for the users) to convey (most of) the relevant information between the communicating persons.

- *Two or more persons*

As mentioned above, conferences generally involve more than two participants, and their number may change during the course of the conference. This implies that teleconferencing systems have to support multipoint as well as conventional point-to-point communication and allow for seamless transition between these two modes of operation.

- *Synchronous cooperation*

As outlined before, conferences are a highly interactive cooperation process with immediate feedback to all participants. Feedback is given via visual and audible communication channels that are reproduced in teleconferences. The potentially high degree of interactivity requires that all information exchanged in a teleconference is delayed as little as possible during transmission. Transmission requirements differ for the various information types (video, audio, application data), requiring different types of transmission services.

- *Scheduling*

Advance scheduling of physical conferences serves basically two purposes: a) to ensure that all participants gather at the same place at the same time and are able to prepare themselves for the conference; and b) to enable planning for a meeting room of sufficient size, the appropriate equipment, copying service, catering, etc. and reserving these ‘‘resources’’. For teleconferences, item a) basically remains unchanged, only the type of personal and location information differ. Reservations (item b) are optional in teleconferences and deal primarily with computing and telecommunication resources such as a central conference server to which all the participants set up a connection for the conference, the required communication lines, and bandwidth rather than rooms or equipment.[23] Teleconferences for which (advance) reservation is not required or not even possible include phone call-like ad-hoc teleconferences and low end teleconferences without guaranteed quality.

- *Date and time / additional conference-related information*

Each conference usually has defined starting and expected ending date(s) and time(s). Other configuration parameters of a conference include the meeting venue (see below), how many (and which) persons are to participate and which meeting aids are needed. These parameters are of importance for scheduling both physical conferences as well as a teleconferences.[24]

---

[23] Note that teleconferences may make use of teleconferencing rooms or other dedicated equipment at one or more sites which may need to be reserved as well.

[24] Note, however, that in practice most teleconferences will be shorter compared to face-to-face meetings (in the order of hours compared to one or more days), and that the starting and ending time may be handled more flexible, since travel arrangements are not an issue in teleconferences.

- *Venue*

The venue is the location where a physical conference takes place, e. g. country, city, street, building, room. Similarly, a ‘‘place’’ needs to be defined for a teleconference as well. However, this place is no geographical location but merely a unique reference to the conference including e. g. conference name, password, and electronic addressing information.

- *Notification in advance*

Conferees receive invitations to or notifications about conferences they may or are requested to attend. Such invitations or notifications contain all the information needed for participation, the most important ones being date and venue. The participants are informed by phone, fax, letter, etc. and are often expected to confirm their participation. In teleconferencing environments, notifications may be distributed by means of electronic mail, by announcing the conference using some well-known communications path, or by storing information about it on a conference server (e. g. a reservation system or an MCU). Also, participants themselves may actively search for interesting conferences, e. g. by querying conference ‘‘servers’’ about scheduled or ongoing teleconferences.

- *Conference host*

The conferencing facilities at a venue are provided by a host who is responsible for the meeting room(s) and the necessary equipment as well as for managing the entire conference course: reception of participants, guiding them to the room, catering, etc. In physical conferences, often one of the participating parties hosts the conference. This type of host is no longer needed in teleconferences. Nevertheless, coordination functions (reservation, access control, billing) and provision of connectivity (through MCUs) among all participants are the tasks of ‘‘teleconference hosts’’ — which may be PTTs, private companies, or conference parties. Point-to-point and teleconferences doing without centralized equipment may not need a dedicated host at all.

- *Access to the conference*

The conference host and the conference chairperson usually form the authorities deciding who may participate in a conference if the conference is not public. Persons that are denied entry to the conference room are thereby prevented from gaining access to any information exchanged during the conference course. This very property is of crucial importance for the deployment of teleconferences as well. Participants must be authenticated (at entry, re-entry, and repeatedly during the conference) to ensure that their access to the conference (and all its potentially confidential information) is legitimate.

- *Roles*

In physical conferences, each conferee plays a certain role that may be defined in two ways. Firstly, a participant has a certain status which may imply (lack of) certain privileges, e. g. being the conductor or an observer. Secondly, each conferee is expected to perform a set of tasks within the conference, e. g. to make a presentation, to vote, to listen, etc. What a role means for a certain conference — i. e. what privileges and tasks are linked to it — depends on the conference policy. A teleconferencing system has to support the concept of roles but should leave their meaning up to the conference policy. The same applies to privileges associated with certain roles. While not easily possible in physical conferences, teleconferencing systems may enforce adhering to restrictions imposed by the roles (e. g. not being allowed to speak) as well as taking actions by virtue of

a role (e. g. excluding somebody from a conference). Tasks conferees have to perform, however, are at the semantic level of the conference and whether or not a participant does what he is expected to remains invisible to the teleconferencing system.

- *Conductor / chairperson*

The conference conductor (also termed chairperson) is a dedicated conferee who is responsible for the orderly course of a conference. He has administrative privileges that enable him to perform this function, i. e. the conductor is a specific role in a conference. Most of the conferences have some kind of conductor — if no person is explicitly appointed, an informal conductor is usually established within the group. Conductorship may change during the conference course. Administrative responsibilities of the chairperson may include convening and adjourning the meeting, admitting participants, arbitrating the conference floor, inviting and excluding persons, etc. The same applies to teleconferences. However, while in physical meetings the conductor gains his ''power'' mainly through personal presence and in general through his privileged location (e. g. seated at the head of the table) teleconferencing systems have to provide technical means for controlling the conference course. The conductor may decide to use these means if the conference cannot be run through verbal human-to-human communication only.

- *Conference policy*

A conference brings together a group of people to discuss a certain topic in a certain setting. The conference course and the behavior of the conferees depend at least on the relationships among the participants, on the subjects being discussed, and on the formal requirements on the conference. These are the characteristics that distinguish an informal working group meeting from a highly structured plenary discussion, from a negotiation, etc. The rules for behavior and interaction within a conference are termed conference policy. In physical conferences, adherence to the respective conference policy (which in most cases is not made explicit) is demanded from all conferees following the rules of our civilization. While this holds true as well for teleconferences, the conference policy may also have implications on the mode of operation of a teleconferencing system. Besides relying on human-to-human interaction for running the conference[25] the conference policy may be made explicit to the teleconferencing system. Variables of the conference policy include whether or not the conference is conducted, how floor control is managed, which means of control may be used by the conductor, etc.

- *Conference profile*

The conference profile encompasses the definition of all the conference parameters including who may participate, what is to be discussed, where and when does the conference take place, etc. In physical conferences part of the profile is typically included in an invitation. In teleconferences, the conference profile becomes the central information record defining how a conference has to be initiated and run. The conference profile also forms the basis for advance reservations.

- *Convene / adjourn*

A physical conference starts when all or most of the important participants are present and — in case of a scheduled conference — the envisaged start time has been reached. The conference

[25] This is more difficult in teleconferences anyway since — due to technical constraints — compared to face-to-face meetings the participants are less aware of one another and each other's actions.

lasts until the predefined closing time (if any), until all issues are discussed, or until it becomes clear that no further conclusions will be reached. The decision to convene and adjourn the conference is made either by the conductor or by consent of the group. This is no different in teleconferences. Once the communication channels have been established, it is up to human-to-human interaction to decide when the actual meeting starts and when to close it. In contrast to face-to-face meetings, however, teleconferencing technology allows missing participants to be called via the system; even the entire teleconference may be initiated automatically by the system — calling all specified persons — at the scheduled time. Also, when the conference time is up, the teleconferencing system may terminate the conference automatically; this is of importance e. g. when a conference A utilizes shared resources (such as a central conference server) and the same resources are allocated to another conference B starting immediately after conference A is expected to be finished.

- *Side discussion*

In physical conferences, the involved persons have frequent opportunities to talk to one another — by twos or in small groups — outside the (strict) course of the conference: in coffee breaks, during meals, before start and after the end of the meeting, as well as during the meeting, of course. Teleconferences can poorly accommodate these ‘‘by chance’’ meetings: those who are early in a teleconference can talk to one another (in a single group) before the actual start of the conference. If small group discussions with certain persons shall take place, they have to be explicitly set up as a separate (parallel) conference. On one hand, this is less disturbing to the rest of the group compared to a physical conference. On the other hand, holding a side discussion (which is similar to a subconference) means that the involved participants are de-facto decoupled from the main conference and lose track of what is going on, when important topics are coming up, etc. while they remain in touch in a physical conference just by being in the same room.

- *Come and leave at will*

Once a person has been accepted to a conference (and has not been excluded afterwards) she may leave and re-enter the conference room at will. The control mechanisms for teleconferences have to account for dynamic conference membership as well. In addition, teleconferencing allows for simultaneous presence in several conferences, e. g. with one conference being the focus of a participant’s attention at a point in time while the other conferences are running in the background. A participant may arbitrarily switch between these teleconferences.

- *Invitation of conferees*

During the conference course, the participants of an ongoing conference may require guidance from other persons (e. g. experts on a certain matter) who are not yet participating and hence shall be invited to join the conference. If the invitees agree, they are to be brought into the conference. In physical conferences, the required persons can be invited to join the conference at any point in time — provided that they are at the same location. If not, involvement can at most be achieved using the telephone. Teleconferencing systems need mechanisms to invite and authenticate such ‘‘experts’’ but have to prohibit misuse of this facility. In contrast to physical conferences, the invitees’ locations are irrelevant.

- *Exclusion of conferees*

In conferences, there is the occasional need to (temporarily) exclude certain participants from the conference. For example, guests or experts may not be allowed to be present when confidential

matters are discussed. Also, although this is unusual for business conferences, conferees may be (deliberately) disturbing the course of the conference. In any of these cases, the conferees in question are expelled from the conference room by the conductor. A similar facility is to be provided in teleconferences. While in physical conferences the person to be excluded is verbally requested to leave the room, in teleconferences leaving the conference is enforced by shutting down the telecommunication ‘‘connection’’ to the respective person’s system.

- *Updated on the current state of the discussion*

As a consequence of dynamic leaving and joining a conference (as well as because of late arrival) temporarily absent participants have to be synchronized with the discussion (e. g. by action of the chairperson or by getting information from a neighbor). For teleconferences, this means, that besides informing the human being about the state of the discussion, in particular the teleconferencing application software and the underlying protocols need to support (re-)synchronization as well.

- *Parallel conferences / subconferences*

Many physical conferences may be held in parallel between which conferees may change provided that the conference rooms are co-located. Such parallel conferences can be independent; or they can be the result of splitting up a meeting into work groups (subconferences) that optionally rejoin later into a plenary session. Special cases of parallel ‘‘conferences’’ are phone calls or consultations between the members of a party. Support for parallel participation in multiple conferences gains importance in teleconferences since the need for physical co-location of the conference rooms disappears and the process of changing between teleconferences is simplified and seamless to the user and becomes less disruptive to the conference itself. However, switching between conferences bears the risk that the user looses track of the discussions and has to be resynchronized each time she (re)enters a conference.

- *Privacy*

In an ideal situation, every two meeting rooms are isolated from one another and from the outside. This means that information may not propagate between the two rooms or get to the outside unless conferees themselves forward the information (e. g. by changing from one room to another): under this precondition, privacy in a meeting room is ensured. For teleconferences, encryption is required to prevent eavesdropping in the network and on intermediate systems.

- *Exchange papers*

Many activities in a conference require the distribution of working documents and information updates to the conferees. Consequently, in teleconferences, electronic exchange of arbitrary documents prior to as well as during a conference is required.

- *Interactions ... meeting aids*

During a physical conference, the participants talk to and look at one another. They have presentations and discussions, make decisions, do engineering, vote, etc. Meeting aids such as overhead or slide projectors, flip charts, whiteboards, etc. are often utilized to support (certain types of) interactions between the conferees. In teleconferences, video images of the conferees (including part of their environment) and the conferees’ utterances are captured by cameras and microphones, respectively, and transmitted to the other participants. The physical meeting aids are substituted by teleconferencing applications that provide the respective functionality.

- *Floor control*

The course of a conference and the participants' behavior follows some commonly accepted rules. Among other things, the conferees have to agree on who is allowed to talk at a given point in time, who may draw on the whiteboard, etc. This is referred to as floor control. The conference floor may be controlled explicitly with (usually) the conductor granting and withdrawing the permission to speak or draw. Or floor control may be handled implicitly based upon a sense for cooperation and mutual respect among the conferees (e. g. when somebody starts talking the others listen and do not interrupt). Explicitly granting and withdrawing the floor has to be provided in teleconferences as well. Unlike in most face-to-face business meetings, however, adherence to the conductor's decision to grant or withdraw the floor may be enforced by the teleconferencing system. Handling floor control based on politeness is made more difficult in teleconferences: due to the technology involvement and the physical distance between the participants, the time span between one participant starting to talk and all others recognizing this increases significantly; therefore, it is more likely that several persons start talking virtually at once thus rendering ''free floor'' policy more difficult to use. As a consequence, means for explicit floor control may be perceived necessary in teleconferences even though they would not be employed in the same setting in a physical conference.

Finally, note that potentially multiple floors may be managed simultaneously: for example, one conferee may speak while a second one updates a flip chart accordingly. Access to some or all floors may be coupled or the floors may be treated independently. As these are common scenarios, DMC systems have to support the aforementioned ways of floor control for multiple (independent as well as coupled) floors.

2.2.2. Types of Teleconferences (Teleconferencing Styles)

The previous subsection has outlined a scenario for business conferences and identified those features that have to be reproduced by teleconferencing systems in order to support such conferences. Thereby, the previous subsection has defined the setting and the tools for a business conference. What has been left open, however, are a variety of parameters (also termed *attributes*) that further describe the type of a business conference. Examples are the number of participants and the detailed regulations of the conference policy such as how conductorship and floor control are handled. In this subsection, the technically relevant parameters — i. e. those that do impact the design and/or the functionality of the infrastructure to be developed — of a (business) teleconference are identified. Based on these parameters, a categorization scheme for teleconferences is presented according to which (not only) business conferences may be classified. This scheme is finally used to restrict the focus of this thesis to a well-defined subset of teleconferences.[26]

The category of *business teleconferences* may include a variety of conferences such as technical presentations, workgroup meetings, negotiations, R&D discussions, brainstorming sessions, and many more scenarios depending on how ''business'' is actually defined. Figure 2.3 shows on the left hand side a set of meeting types which are well-known and do not need further explanations.

[26] On the single scale of figure 2.3 below, two extreme positions can be identified with very different requirements that (today) call for different approaches to a system design. This emphasizes that building a *one size fits all* type of teleconferencing system is unlikely to succeed (at this point in time) which turns it indispensable that a target set of teleconferences be specified upon which the further design can be based.

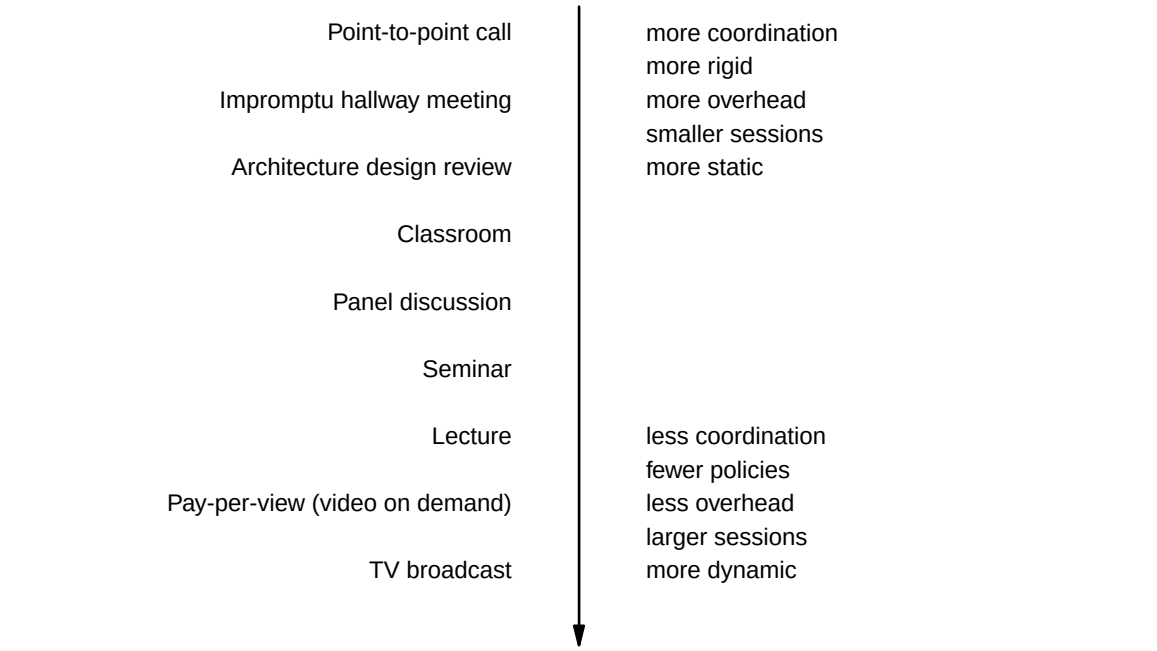


Figure 2.3: Meeting types [Schooler 93, p. 429]

Following the definition of the previous subsection, the author considers all meeting types from point-to-point calls to panel discussions and possibly seminars to be business conferences of some sort. The aforementioned examples obviously do fit into this range of meeting types.

Figure 2.3 also shows a simplified rough correlation between meeting types and a set of five meeting parameters. However, in the figure, the various parameters on the right hand side of the image are linked to one another and are entirely implied by the meeting type.[27] While this is indeed true for some of the conference parameters, the precise definition of a target range of conference styles requires a more thorough (and independent) consideration of the various conference attributes. In the following, an extensive (but not necessarily complete) set of technical parameters is presented which are treated as independent of one another. These parameters may be used to describe the technical characteristics of all the meeting types introduced before; hence, the *meeting type* is not included as a separate attribute.[28]

- *Interactivity*

Interactive conferences are characterized by a multidirectional exchange of views among many or all of the participants (design review, phone call). In contrast, non-interactive meetings are primarily concerned with conveying knowledge and information from a single or a small group of person(s) to an audience without (much) feedback (lectures, television). In-between, semi-interactive conference types such as workshops and seminars can be found. Business conferences are

[27] Note that this diagram was perfectly suited for its purpose: it was presented at the 27th IETF in Amsterdam in 1993 within the MMUSIC working group to provide an overview of possible future directions for protocol development in this group.

[28] The intended meeting type may already imply that some attributes (in general) take certain values or roughly limit the range in which some conference attributes may vary.

considered to include interactive and semi-interactive settings; consequently, this range has to be supported by a DMC system.[29]

- *Group size*

The possible group size ranges from two (for a simple phone call scenario) to more than a million (for a TV broadcast). Restricting business conferences to the meeting types indicated above, the number of participants can be expected to be always less than one hundred, and is probably less than twenty in most cases. This yields a target maximum group size of one hundred participants for DMC systems.

- *Conference duration*

Teleconferences may last for as short as a one minute (e. g. when teleconferencing is used in place of a phone call) but may also continue for several hours. A conference spanning several days would be modeled as a set of conferences — one each day for some hours. The goal here is to support teleconferences of arbitrary duration.

- *Geographic distribution*

Teleconferencing technology obviously focuses on support for geographically dispersed groups where the conferees may be located on the same floor, in the same building, or be many miles apart. Nevertheless, this technology can also be employed for co-located participants (e. g. in a conference room). Also, hybrid scenarios combining co-located and remote participants into one teleconference have to be considered.

- *Spontaneity*

Conferences may be established ad-hoc (like a telephone call), or they may be planned in advance. Pre-planning may reach from the very short term (e. g. the conference is to start within minutes), over schedules for the same day, to the long-term (e. g. weeks or months in advance). DMC systems have to support spontaneous as well as pre-planned conferences.

- *Conference establishment*

A conference is not actually convened unless most or all of the expected participants have joined. Participants may enter a conference in different ways: either the conferees actively enter a conference (*call-in*, *meet-me* conference), the conferencing system (e. g. at the conference host) contacts the conferees at the start time (*call-out*), or one or more conferees call-in and make the conferencing system contact the missing persons (*call-through*). Finally one conferee may contact another directly (implicitly establishing a conference) for an (initial) point-to-point conference (*direct call*). All four types of conference establishment need to be supported.

- *Admission control*

Admission control regulates which participants may join a conference. Anybody may enter an *open* conference whereas admission to a *closed* conference is limited to a predefined set of persons. Conferences (in particular open ones) may be publicly announced. Alternatively, information about their existence can be secretly distributed to only those who are intended to participate

[29] Note that DMC systems *may be used* for simple information dissemination, but they are not primarily *designed* for this purpose.

— as a first step towards admission control (‘‘security by obscurity’’). In addition, access to a closed conference may be regulated either by some password mechanism — i. e. the set of authorized persons is defined as those knowing the password (sharing a secret). Or admission control is based on personal identity by authenticating the persons upon entry and admitting only those listed in the conference profile. A DMC system should support the entire spectrum of admission control.

- *Conference policy*

The conference policy describes how strictly a conference is managed and is therefore related to the issues of conductorship and floor control explained below. On one end of the scale, conferences are rigidly managed with central floor assignment through the conductor, participants always talking only to the conductor who forwards statements, etc. Examples are highly formal international meetings (such as ITU-T plenary sessions at Study Groups meetings or meetings of other UN bodies). On the other end, conferences are informal with less policies governing the interactions among participants. Examples are phone calls or small group design meetings among persons of equal status. A DMC system has to accommodate both extremes as far as these policies can be actively supported by technology.

- *Conductorship management*

A conference may be chaired (*syn* conducted) by a participant, the so-called chairperson (*syn* conductor), or not. If a conference is conducted, the conductor may be a (statically) pre-assigned person or the conductor role may be passed around among (a set of privileged) conferees — all this is part of the conference policy. A conference may also switch between conducted and non-conducted mode depending on the task being performed. All these variations of conductorship control have to be supported by a DMC system.

- *Floor control*

As in part discussed before (refer to the item *floor control* in subsection 2.2.1), access to each of the one or more conference floors may be uncontrolled (free floor), i. e. anybody may talk, draw, etc. in parallel at any point in time. Or access to the floor may be administered: by the conference conductor who grants and revokes access to the floor explicitly, by some control entity — such as the conference host — that automatically handles floor requests according to the conference policy (e. g. maintains a speaker list), or in a joint fashion by all conferees who honor floor requests and pass the floor on to the next speaker or actor. All these policies have to be supported for multiple floors.

- *Coupling mode*

The coupling mode refers to the consistency of state information about a teleconference at the various participating sites. On one end of the scale (*tightly coupled*), all participants have exactly the same perception of the conference (e. g. who is present, who does what, which meeting aids are in use and what they are used for) at any point in time — like in a small meeting room. On the other end, consistency is not explicitly provided for (*loosely coupled*). Still, a common perception of the conference subject exists and people are roughly aware of the actions in the conference. But the exact membership need not be known, the members are more anonymous, and the knowledge may vary from site to site.[30] An example for this latter case is a talk with an audience

[30] Referring to figure 2.3, to achieve a tight coupling more overhead is required compared to a loosely

of four hundred listeners. Business conferences obviously belong to the tightly coupled conferences which are therefore the primary concern for the design of DMC systems.

- *Conference dynamics*

Conferences also vary in their dynamics, i. e. the frequency with which changes to the conference state occur. This attribute mainly aims at changes in membership, but changes in role assignments, in the applications being in use, what they are used for, etc. are included, too. One extreme are static sessions with predefined membership and applications that do not change throughout the entire conference course; examples are a ''phone'' call and a negotiation about a contract with a fixed set of participants. The other extreme are highly dynamic sessions with membership changes every few seconds; examples are multicast IETF workshops and television broadcast. As this thesis aims at business conferences, the focus with respect to this attribute is on static to moderately dynamic sessions.

Summary

Figure 2.4 summarizes the descriptions given above and illustrates the conference types that form the scope for DMC systems and therefore have to be supported by the teleconferencing infrastructure being developed in this thesis. Note again that, even though DMC systems are designed for a particular range of teleconferences, they may nevertheless be employed to hold conferences of other types as well; in such cases, however, support will not always be optimal.

2.2.3. Applicability and Limitations of Teleconferences

The previous two subsections have identified the functionality as well as the technical parameters that are relevant for reproducing face-to-face meetings by means of (desktop multimedia) teleconferencing technology. These discussions have shown that, objectively, teleconferencing and particularly DMC systems are capable of replicating virtually all kinds of meeting-related interactions. The use of elaborate groupware applications (meeting aids) allows for interactions between participants in teleconferences that go even beyond what can be achieved within typical meeting rooms. Nevertheless, it is intuitively clear that — in spite of all efforts — teleconferencing technology will not be able to replicate all details of a physical meeting: sitting alone at one's desk differs in many respects from being in a meeting room with others, and the groupware applications by means of which interactions take place are differently to operate compared to those tools that would be used in a conference room.

As a consequence, teleconferencing technology should not be judged as an equivalent substitute for face-to-face meetings, but rather as a complementary means for collaboration with different strengths and weaknesses.[32] Otherwise, misconceptions about the potential of teleconferencing

coupled conference. This and the previous policy-related items also subsume the issue of ''coordination'' mentioned in figure 2.3 from a technical and an organisatorial point of view, respectively.

[32] Some researchers fundamentally doubt that the attempt to nothing but reproduce physical meetings by means of telecommunication technology has chances to succeed because in this case the achievements of the technology are always measured relative to the physical meeting. ''Many current efforts to accomplish this attempt to create a sense of 'being there,' chiefly by establishing audio and video channels between distant locations. Any system which attempts to bring those that are physically distant into a physically proximate community by imitating physical proximity will always keep the former at a disadvantage. This is not because of the quality of the system, but because of what they attempt to achieve.'' [Hollan / Stornetta 92, p. 848]

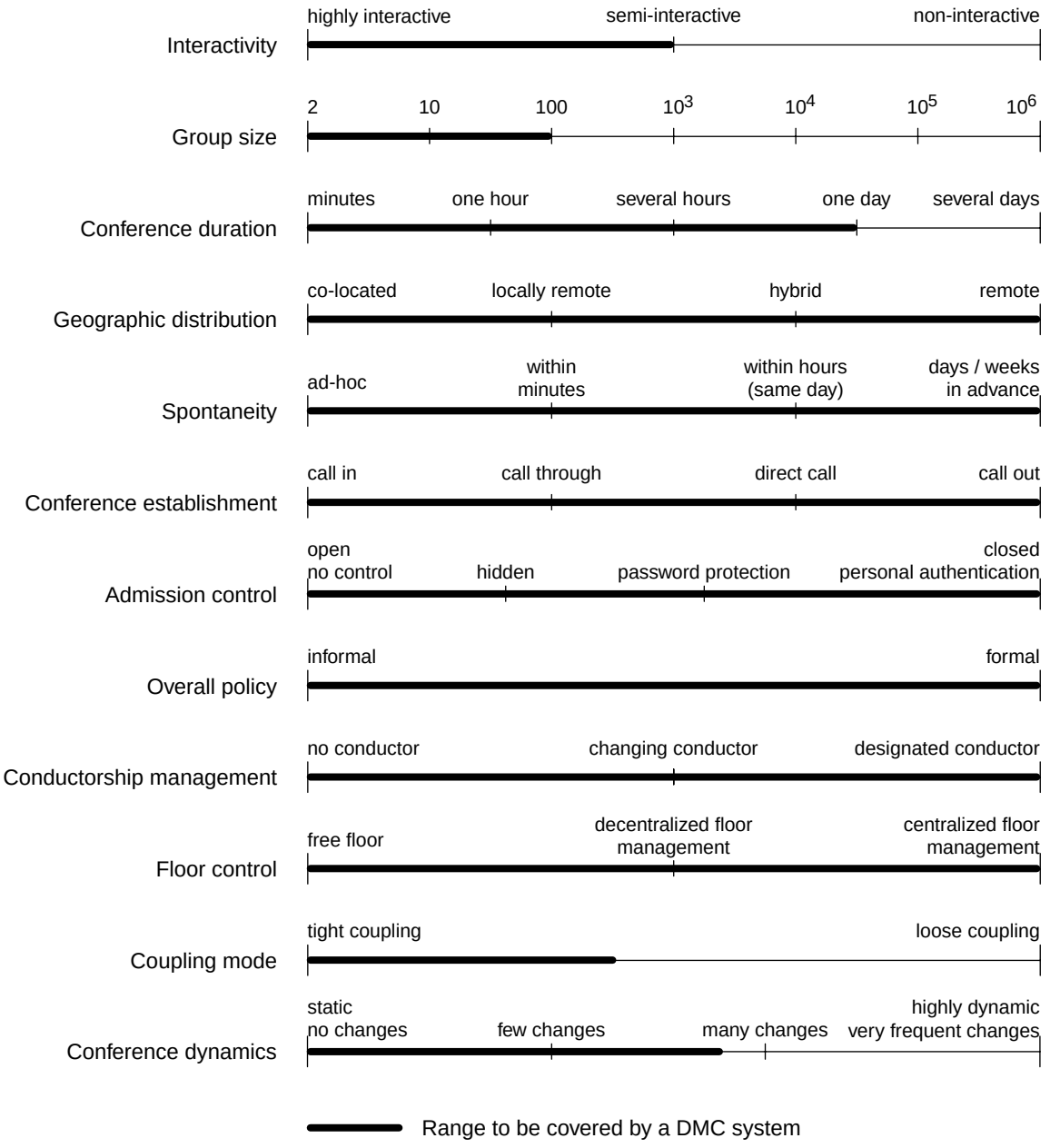


Figure 2.4: Target range of teleconferencing styles

systems may occur which may easily lead to disappointment of the users if their expectations are not met: ‘‘The adoption of video conferencing rooms and other multimedia technology has suffered from marketing myths that promote them as replacement for face-to-face interaction’’ [Egido 90].

The entire history of videoconferencing technology has shown that, in spite of many promising forecasts for virtually all kinds of teleconferencing systems, the actual deployment over the years has always fallen short of the expectations (refer also to e. g. [Darby 90], [Clarke 90], [Ott 95a]).[33]

[33] Not only videoconferencing systems failed to meet the expectations of their designers and users. This

A lot of research has been undertaken to determine the reasons for the past failures of teleconferencing systems; these reasons include the aforementioned style of marketing, immaturity of the technology itself, and shortcomings in functionality, among many others.[34]

This subsection completes the modeling discussion of teleconferences by addressing limitations and applicability issues of teleconferencing technology conceived most important by the author. Thereby, this subsection outlines where DMC systems may be used and which additional (non-technical) aspects have to be considered in their design and application: First, the most important limitations inherent to teleconferencing technology are discussed and, based on this, the characteristics of meetings being candidates for use of this technology are outlined. Finally, non-technical as well as technical acceptance criteria of teleconferencing systems are discussed.

Limitations inherent to Teleconferencing Technology

A lot of research has been undertaken to identify differences between face-to-face meetings and teleconferences that cannot be overcome by technical means (refer e. g. to [Kubicek / Rolf 86], [Egido 88], [Egido 90], [Clarke 90], [Straßburger 90], [Schulte 93], and [Ott 95a]). The most important of the manifold findings can be summarized as follows:

- First of all, the fact that the conferees are not in the same room leads to reduced ‘‘social presence’’ [Short *et al.* 76] of the individuals. The conferees are capable of interacting with one another, but the conference-specific atmosphere and the mutually perceived physical proximity are missing as are other sensual impressions (e. g. smells). Despite audiovisual communication channels, important cues such as spatial recognition of speech and immediate association with a person as well as the possibility to establish eye contact are missing. Overall, this results in a less personal character of teleconferences compared to face-to-face meetings — with potentially less interactions and reduced quality of discussions.

- Furthermore, due to the lack of a common physical environment, the opportunities for social interactions (e. g. during breaks, outside the meeting room) are missing. These are not only relevant to allow the participants to get to know one another (and thus to establish mutual trust among themselves). Moreover, it often happens that in the informal context of e. g. coffee breaks or business lunches, important decisions are prepared, disputes settled, and so on.

- Finally, as already mentioned before, holding a teleconference feels differently compared to being in a face-to-face meeting. In particular, interactions in face-to-face meetings do not require additional skills in contrast to operating a teleconferencing system (e. g. writing and drawing with a pen on a whiteboard or overhead transparency is performed intuitively, while using a groupware application that implements a whiteboard typically requires learning). Furthermore, participation from one’s office means that the potentially distracting office work is continuing which may lead to participants being less focused on the teleconference.[35]

---

applies to other types of groupware systems as well. Failures have been reported for scheduling systems, early e-mail and bulletin boards systems, project management tools to name just a few prominent examples (refer e. g. to the overview in [Grudin 88], to [Lynne Markus / Connolly 90], [Grudin 94a] [Grudin 94b]).

[34] Refer e. g. to [Grudin 88], [Egido 88], [Tonnemacher 88], [Mühlbach *et al.* 89], [Grudin 90], [Kolrep *et al.* 90], [Lynne Markus / Connolly 90], [Greenberg 91], [Gaver 92], [Schulte 93], [Grudin 94b], and [Ott 95a] among many others.

[35] The author has made this observation during innumerable audio conference calls as well as video and multimedia conferences. It should also be noted that most persons — e. g. colleagues and visitors — are not

Taking into account particularly the first two of these factors reduces the types of meetings that are candidates for substitution by teleconferences. In earlier investigations, the author has characterized meetings that may be held via DMC systems as follows [Ott 95a]:[36]

- The persons communicating via teleconferences are likely to know each other in advance rather than using the technology for the first contact.
- The subject of discussion will in general be of more impersonal (pertinent) nature. Interactive discussions are possible as are more unidirectional updates.
- Routine meetings may be held using DMC systems as may be conferences to handle special cases. However, this technology is unlikely to be used for negotiations, discussions of personal affairs, or similar highly delicate matters.
- In general, the top management is unlikely to use desktop teleconferencing for daily work at this point in time as are secretaries (for different reasons, of course). Clerks, engineers, lower, and middle management are expected to make more extensive use of this technology.
- R&D, design, marketing, and sales are likely to become the first divisions for multimedia teleconferencing as there the potential benefit seems to be the largest. Also, management of (international and/or inter-corporate) projects is likely to be supported by teleconferencing technology.
- Deploying DMC systems is basically independent of a company's size but, as intra-corporate usage is likely to dominate at the beginning, internationally operating enterprises are expected to spearhead the large-scale deployment. With respect to the industrial sector, manufacturers of this technology, other high-tech companies, and service providers (e. g. marketing agencies, airlines) are expected to take the lead in deploying DMC systems.
- As already identified during the discussion of the target range of teleconferencing styles to be supported by DMC systems, teleconferences may be planned in advance or held spontaneously, and they may have a formal or an informal character. However, spontaneously (or on a short-term) convened, informal meetings are more likely since they better exploit the DMC systems' advantages over face-to-face meetings.
- Furthermore, teleconferences may last from a few minutes to two or three hours and usually involve not more than some ten participants. Longer and (much) larger teleconferences are considered rather exceptional.

Acceptance of Teleconferencing Technology

Obviously, the number of candidate meetings matching the aforementioned characteristics is large (and the various studies quoted in chapter one confirm this perception). But even if there is a large number of meetings that potentially could be held by means of teleconferencing technology, the actual use of a specific teleconferencing system (and thus the eventually achieved benefits) depends on its acceptance by the potential users. Particularly with the advent of DMC systems and the resulting migration of teleconferencing technology from dedicated rooms for occasional usage (within closed groups) to everyone's desk for everyday use, acceptance of this technology

---

yet accustomed to workstation-based conferencing: while they do not disturb a person talking on the phone, they do not (yet) attribute a comparable significance to a desktop multimedia conference — even if the conferee is using a head-set.

[36] In addition, each phone call is a candidate for substitution by a desktop multimedia teleconference ''call''; this would result in enriched rather than reduced collaboration quality.

by the employees — primarily as a new working tool, but also as new means for some social interactions — becomes increasingly crucial to its success: an employee who rejects DMC systems as collaboration tool will barely use this technology voluntarily to increase the efficiency of business processes [Ott 95a].

Acceptance of DMC technology in turn does not only depend on the availability of this technology alone but in particular also on its feasibility for group work and for providing solutions to real-world problems that are perceived as such by the potential users.

For the design of DMC systems, technical feasibility for collaborations primarily means appropriate consideration of factors specific to group work (most of which are irrelevant to the design of single user applications).

Issues that need to be addressed in the technical design of a teleconferencing system include the following:

- seamless transition between individual and group work modes on one hand and between (physical) desktop and computer-based tools on the other [Ishii / Miyake 91];
- sufficient consideration of social and organizational roles of individuals and their skills as well as differences between groups concerning working style, structure, etc. in different situations [Greenberg 91] [Grudin 90];
- not requiring users to learn new (software) tools in order to be able to perform group work (''collaborative load'') [Marmolin / Sundblad 91], nor asking them to carry out complex procedures to set up collaboration sessions (''cognitive overhead'' [Patel / Kalter 93a]); and
- the adequate design of the complex user interfaces required for synchronous group collaboration [Ellis *et al.* 91].

Inappropriate consideration of these aspects — as is typical for most of today's teleconferencing systems — leads to users perceiving shortcomings in the system functionality and thus may constitutes a obstacle to broad acceptance of the system.

Besides these issues, the perceived quality of interactions carried out through the DMC system — which is typically measured by the users in terms of audio and video quality as well as in system response time — are of importance.

But even if all aforementioned technical conditions were satisfied, other (non-technical) reasons for rejection of DMC technology by individual employees remain the two most important ones of which are the following [Ott 95a]:

- Organisatorial mistakes of the management when introducing teleconferencing systems into (parts of) the enterprise — typical mistakes include lack of commitment to the new technology, exclusion of those who are to use the systems from the decision-making process, and non-adoption of business processes and organizational structures to make best use of the technology.
- Personal reasons of individuals — employees may be afraid of losing their status in a group, may not feel sufficiently qualified for the new technology, may fear consequences for the social relationships to colleagues, or may just be worried to lose the privilege to travel on business.

These issues are not specific to teleconferencing but are rather common to most technological innovations and are dealt with extensively in economics and management sciences (refer e. g. to the overview in [Bullinger 90]).

Finally, the perceived value of teleconferencing technology for each individual user as well as for an enterprise as a whole depends on the number of communication partners that effectively may be contacted using DMC systems (this applies equally well to all other types of telecommunication technology): a *critical mass* of users — within the same as well as in other enterprises — has to be reached.[37] That is, the number of partners a person can communicate with using (most or all features of) a DMC system must be sufficiently large so that the technology can be used on a regular basis rather than in exceptional situations only. As long as such a threshold is not reached, the systems in question are potentially rarely used and thus provide little or no benefit. In turn, systems that are not deemed useful are likely to be abandoned, particularly if their usage (or even their non-usage) incurs some cost, be it personal effort or money (refer e. g. to [Grudin 88], [Greenberg 91], [Cool *et al.* 92], [Schindler *et al.* 93], [Grudin 94b]). From a technical perspective, the most important prerequisite for achieving the required critical mass is that teleconferencing systems made by different vendors are interoperable so that a person is able to call any other by means of a DMC system as it is possible today when using the telephone.

Summary

Assuming a conference scenario to which teleconferencing is applicable at all, then of the acceptance issues for desktop multimedia conferencing technology discussed in this subsection are either related to the research area of human-computer interaction or to organizational and management sciences and hence are beyond the scope of the design of a communication infrastructure. However, the communication infrastructure developed in thesis should contribute towards achieving broad acceptance in those two areas that are within its scope: it should be sufficiently flexible to allow accommodating the entire range of teleconferencing types identified in subsection 2.2.2, and its implementation should be based on widely accepted international standards so that interoperability with other standards-compliant DMC system can be ensured.

2.3. Outline of a DMC System

Section 2.1 has introduced several classification schemes for groupware systems and characterized DMC systems according to these schemes, thereby providing a first outline of the functionality to be covered by DMC systems. The previous section has introduced a typical business conference scenario and identified the key features of such a conference as well as a set of technically relevant attributes of (business) conferences. Based on these results, the target range of conferences to be supported by a DMC system has been defined, and, finally, important applicability issues of teleconferences also impacting the DMC system design have been addressed.

This section combines the results of the previous ones and summarizes the functionality to be integrated by a desktop multimedia conferencing system, in terms of the incorporated groupware systems as well as the additional conference-specific functionality (section 2.3.1).

[37] ‘‘There is widespread awareness that group communication systems have increasing returns to adoption, meaning that the value of the technology to one user increases as the number of users increases.’’ [Resnick 93, p. 402].

The discussion of the various conference types in subsection 2.2.2 has shown that DMC systems need to be applicable to different conference scenarios in different groups held for different purposes, etc. Consequently, all functions have to be tailorable to the specific needs of a particular conferencing situation; this is discussed separately in subsection 2.3.2. Furthermore, for the design of DMC systems, not only the integration of a variety of collaboration functions is of importance but also that the DMC system itself is integrated into the office environment so that the user can efficiently make use of its functionality. This design aspect is addressed in subsection 2.3.3. Finally, subsection 2.3.4 presents the most important technical prerequisites for the actual implementation and the broad deployment of desktop multimedia conferencing systems.

2.3.1. Functionality of a DMC System

So far, this chapter has characterized desktop multimedia conferencing systems, described the features of business (tele)conferences, and outlined for which types of (business) conferences DMC systems as considered in this thesis are designed. In this subsection, the functionality to be provided by a DMC system — which in turn has to be supported by the underlying communication infrastructure — are summarized in a concise overview.

The total functionality described so far can be grouped into three categories that are discussed separately. *Conference control* provides the human user with the capability to participate in teleconferences — e. g. to move into and out of a ‘‘meeting room’’, to adhere to the conference policy, etc. — and thus may be interpreted to represent the presence of a human being in a conference. *Audiovisual communication* as well as *meeting aids* complement conference control by providing means to interact with other participants and thus represent the senses of a conferee as well as her capabilities to express herself. The major difference between audiovisual communication and meeting aids is that audio(visual) communication provides the means for basic interactions to enable collaboration in teleconferences at all, while meeting aids support collaboratively carrying out dedicated tasks *in addition* to the basic interactions.

Conference Control

The conference control forms the basis for the operation of a DMC system. It provides functions to establish, run, and terminate conferences as well as to allow the participants to move into and out of conferences. Conference control includes the following functional groups:

- *Conference configuration*

Conference configuration is concerned with defining the conference policy and guarding adherence to it. This includes managing privileges, assigning roles, checking permissions, etc. as described before. This includes in particular conductorship management as one particular role.

Further aspects of conference configuration are reservations, user management, accounting, billing, and other organisatorial or managerial functions. These functions, however, are (if provided at all) local to the conference service provider and are only of marginal significance to the interactions in the synchronous collaboration phase. Hence they are not considered any further in this thesis.

- *Participation management*

This comprises functions for establishing conferences in all three aforementioned conference styles (call-in, call-out, and call-through) and for terminating conferences manually as well as automatically. In point-to-point scenarios, similar functions are used to call another person, to indicate acceptance or rejection of an incoming call, and to terminate a call. These functions imply creation and termination of a conference as appropriate. Of course, extension of a point-to-point call to a multiparty conference is included as well (using the join and invite functions described below).

Furthermore, participation management contains functions to allow participants to control their individual participation in teleconferences. Specifically, joining and leaving a conference as well as changing from one conference to another are provided. Additional functions allow influencing other persons' participation in teleconferences: invitations may be used to include new participants into a conference (if they agree); participants may be expelled from a conference (without their consent). Permission to invoke the latter two functions are likely to be tied to certain roles.

- *Floor control*

A conference floor — which means basically the permission to speak, draw, present, write, etc. — may be explicitly assigned to k out of n (with $0 \leq k \leq n$) participants at a given point in time during the conference. As discussed extensively before, mechanisms are provided to allow participants to request the floor, release it, ask for the floor, pass it on, and even to preempt it. These mechanisms allow to implement more elaborate floor control functions on top.

- *Security*

Security functionality deals mainly with restricting access to a conference and to the information exchanged within this conference. Access to the conference is controlled by either participant authentication through public key systems or by password protection.

The single means for achieving confidential information exchange within a conference is to encrypt information prior to transmission. This requires distributing a key for en-/deciphering to all participants in a way that no potential intruder can get hold of the key. Frequent key changes are needed to prevent confidentiality from being compromised by eavesdropping and cryptanalysis of the information stream.

Audiovisual Communication

Audiovisual communication provides the primary means for human-to-human communication in teleconferences. At least an audio communication channel is expected to be always present while video channels are optional. The following control functions for audio and video need to be supported by a DMC system:

- *Audio control*

Besides local control functions such as volume selection and muting, audio control consists of selecting a mode for the distribution of audio information within a conference. Mode selection may be done centralized or decentralized (i. e. individually), depending on the conference policy in force.

When using centralized selection, either the audio information of a single participant is exclusively forwarded to all others, or, several or all audio sources are mixed, thereby allowing

multiple parallel speakers.[40] Unless all conferees may speak simultaneously, the (audio) floor control mechanisms are used to determine who is allowed to speak at any point in time.

- *Video control*

Video control comprises primarily the selection which participants' video images to display to each user. Ideally, a participant may choose freely whom to see, and may display an arbitrary number of video images (up to all the involved conferees including himself) thus allowing for *continuous presence*. In practice, however, the number of simultaneously displayed video images is restricted to a small number (e. g. one, two, or four) — due to limited space on the screen, bounded transmission capacity, the nature of the real-time transmission protocols in use, etc. Hence, strategies for selection of the displayed images have to be provided. These include manual control such as individual choice or centralized choice for all conferees (e. g. by the conductor) as well as automatic control, e. g. by coupling video and audio selection to show the current and the most recent speaker(s) or by coupling video with the conference floor.

Depending on the capabilities of the underlying transport protocols for video information and the conference topology, dedicated (central) systems[41] may be required to perform selective forwarding of information (''switching'') or (spatial) composition of several video streams into a single one (''mixing'').

In addition, (remote) camera control, selection of the camera or another device to source the video information for a given video stream, taking snapshots of a video stream, and the like belong to the area of video control.

Meeting Aids

The means for conventional audiovisual communication described above allow basic interactions among conference participants based upon sight (of one another) and sound. The conference description given in section 2.2.1 has indicated that additional functionality is required to reproduce more interactions possible in a physical meeting. This functionality is covered by various teleconferencing applications that are integrated (on demand) into a DMC system.

Further elaborating on the outline presented in subsection 2.1.5, the following functional categories are considered part of a DMC system:

- *Remote control / application sharing systems (type i.)* allow cooperative usage of single user applications in teleconferences. As for most application areas only single user applications do exist or are widespread in use, this type of groupware application is essential for effective collaboration in teleconferences.

- *Cooperative editors for text, graphics, etc. (type h.)* as well as synchronous parts of *co-authoring tools (type g.)* have — in contrast to single user word processors, drawing tools, etc. — knowledge about being used in a distributed fashion and can therefore provide specific

[40] Using some central unit for (selecting and) forwarding a single out of *n* incoming information streams is called audio switching. Providing a composite output stream of several incoming ones is termed audio mixing.

[41] As briefly introduced in subsection 2.1.4, these systems are termed *Multipoint Control Units (MCUs)* in ITU-T recommendations.

services relevant to work groups. Thereby, these groupware applications complement single user applications used via application sharing systems; they are currently less powerful in terms of editing functionality compared to single user applications though.

- *Group support systems (type f.)* comprise groupware applications that support certain types of group activities: voting, brainstorming sessions, organizing and structuring ideas, etc. Obviously, they support interactions very typical in meetings and hence are essential for a variety of (tele)conferencing scenarios.

In addition, the teleconferencing research community has developed a set of (new) groupware application types that are specifically designed for use within teleconferences with the aim of reproducing further interactions of face-to-face meetings in teleconferences as well:

- *Joint viewing tools* allow displaying still images (e. g. of objects), documents (e. g. transparencies), and other text or graphics-based information simultaneously to all participants.

- *Telepointer* extend joint viewing tools or application sharing systems by ‘‘group pointers’’ that behave similar to mouse pointers in today’s window-based user interfaces but are visible to all participants. These pointers are used for pointing on windows viewed by all participants to direct the other persons’ attention to specific items, areas, etc. on the screen, e. g. during a presentation.

- *Shared whiteboard and annotation applications* are used for sketching, describing, and commenting on ideas with the results being immediately displayed to all conferees — as is done in face-to-face conferences on whiteboards or using overhead projectors. Such teleconferencing applications also allow to import (arbitrary) documents and overlay textual remarks or graphical annotations.

In general, these three components are integrated in a single groupware application, as their functionality overlaps and all features are highly desirable in teleconferences.[42]

Furthermore, some traditional *telematic services applications*, in part also contained in *office information systems*, are incorporated in DMC systems — after appropriate (protocol) modification to make them suitable for the new (synchronous, multipoint) environment. Of particular interest are file transfer (e. g. based on the ITU-T multipoint file transfer protocol for teleconferences, T.127 [ITU-T T.127]) and multipoint facsimile applications (e. g. as envisaged for the audiographic teleconference defined by the ETSI, [ETS 300101]). Inclusion of such services allows easy and immediate dissemination of e. g. working documents or the achieved results within a teleconference.

[42] Examples for such integrated groupware systems are applications making use of the ITU-T still image conferencing and annotation service as defined in [ITU-T T.126] as well as the whiteboard developed at LBL for the MBone [Jacobson / McCanne 93].

2.3.2. Tailorability of DMC System Functionality

The previous subsection has summarized the functionality to be provided by a DMC system. In order to make DMC systems applicable to a range of meeting types, it is required that this functionality — conference control functions and the teleconferencing applications implementing audiovisual communication as well as meeting aids — can be adapted to a variety of conference settings, with different persons, conference styles, topics being discussed, etc. Tailorability does not address the basic functionality of a DMC system and its components — which has been defined above — but rather how the interactions using the various functions are regulated in the context of a teleconference. That is, tailorability refers to how far the behavior of DMC system components can be influenced by the *conference policy*.

Tailorability of system components comprises at least two dimensions: a) to the behavior of a system (and its users) in a certain conference setting as well as b) to the presentation of the system to its users. (b) aims at accommodating experienced as well as novice users in terms of offered functionality, visible complexity of operations, etc. and belongs to the area of human computer interaction (in the context of groupware systems); therefore, this is not considered here any further (refer e. g. to [Grudin 90] and [Greenberg 91]). (a) refers to the adaptation of the system functions to match the needs of different groups and conference styles through appropriate policies and is discussed in the remainder of this subsection. Both kinds of tailorability are crucial to broad acceptance of the system.

A conference policy is primarily derived from the type of the conference to be held and the topics to be discussed. Certain roles (such as conductor, regular participant, member of the audience) can typically be identified in conferences; and in most cases privileges are associated with these roles rather than with individuals.[43] In essence, a conference policy describes how the previously defined mechanisms are applied: which participants are permissible in a conference, which authentication scheme (if any) is used, which roles are defined, which privileges are associated with which roles, which participants may perform which roles, etc. Thereby, the conference policy defines the meeting type. Two extreme conference policies have already been described before: the non-conducted meeting of equally privileged participants and the strictly ruled meeting with the chairperson assigning the floor only to a subset of predetermined speakers. These and further predefined policies may be inherently provided by the DMC system, but the system should still allow (dynamic) definition and use of other policies as well.

However, all control mechanisms in DMC systems should leave room for policies on a human-to-human basis and limit technology-based regulations to a few really necessary functions. This is because an etiquette is likely to be developed around the deployment of teleconferencing — just as there is a conference culture for the various types of face-to-face meetings [Pankoke-Babatz / Prinz 89] [Ellis *et al.* 91]. These interpersonal agreements about behavior are often sufficient to exercise control and thus can substitute many technical enforcement

[43] Policies may even take into account the organizational and social structure of a group whose members "come together" for a certain teleconference. That is, privileges of conferees in a teleconference may reflect their position in the enterprise relative to the other participants. How far these differences are made explicit in a teleconference, however, depends on the characters of the persons themselves, on the relations between the group members, and how belonging to different levels in the organization hierarchy, to different divisions, etc. affects dealing with one another in everyday work practice.

mechanisms. Also, ‘‘soft’’ control by humans can more smoothly be adapted to changing situations in contrast to a choice between discrete settings of a system. Therefore, putting each and every action under system control is at least inconvenient to use, and an appropriate service is probably impossible to implement.[44] As a consequence, whatever policies are implemented in a DMC system, adaptability also implies the ability to turn them off selectively as well as entirely and hand control over to the human users (refer e. g. to [Ellis *et al.* 91] [Greenberg 91] [Rodden / Blair 91] [Dourish 93]).

2.3.3. Integration Aspects

Most of the teleconferencing functionality described in the previous subsection need not necessarily be implemented in a DMC system; specifically equipped conference rooms (as well as rollabout systems) are in principle able to provide the same functions (by incorporation of workstations if needed), typically at a much higher quality compared to DMC systems (with respect to cost and quality, rollabouts are in-between room-based and DMC systems).[45]

Regardless of the conferencing quality achieved, however, dedicated conference rooms, rollabout systems, and even stand-alone videophones (as well as conventional telephones) are separate pieces of (office) equipment and require personal overhead for their use (e. g. conference rooms and rollabout systems require advance booking, participants have either to move to the conference room or to prepare their office for the rollabout system, etc.; refer again to [Ott 95a] among others). Co-location of teleconferencing systems with the workplace and exclusive availability of a system to an employee are necessary but still not sufficient conditions to achieve a seamless integration of this technology with the daily work process. Even a stand-alone desktop videophone still remains an isolated appliance with various implications for everyday use of the system:

> For example, if the teleconferencing system is different to operate compared to the known office equipment, this creates learning barriers for the human users (e. g. [Schindler *et al.* 93]). Also, the information used and produced in teleconferences is strictly separated from the information available in the remaining office environment, particularly from that stored on the workstation, and it requires additional — manual — effort for the user to transfer information between these two information ‘‘spaces’’ — if this is possible by technical means at all [Ishii / Miyake 91] [Schindler *et al.* 93].

The key to seamless transitions between teleconferences and daily work is — in addition to co-location with the workplace and permanent availability — the sweeping integration of every day work tools (mainly the workstation) and the teleconferencing technology.[46] To address the

[44] As an example on how complex encompassing policy descriptions even for simple coordination tasks may get refer to [Pankoke-Babatz / Prinz 89].

[45] Conference rooms are designed to solve exactly the problem of getting people ‘‘together’’ with much effort spent in order to achieve optimal system function (e. g. perfect acoustics, high quality video images, etc.). It should be explicitly stated the high quality is achieved by the use of equipment at much higher cost (about one to two orders of magnitude) in conference rooms compared to DMC systems (refer to the examples summarized in [Ott 95a]).

[46] This is done in part by conference room or rollabout systems that allow to include text and graphics input either by means of scanners or document cameras or by the inclusion of a workstation. The latter alternative also may support transferring conference results from the conference back to the office environment. These approaches, however, do not eliminate the extra personal overhead and the need for learning new tools for teleconferences. Also, the integration of information remains awkward.

aforementioned needs, the target for the design of desktop multimedia conferencing systems is the full integration of teleconferencing and computer-based office technology. Physically integrating the teleconferencing system into the workstation is the first step towards this. In addition, however, full integration of the DMC system and the workstation environment needs to be achieved in the following areas (refer to [Schindler *et al.* 93]):

- *In the system environment.*

  The existing workstation functions must not be affected [Schindler 91]. This means that the teleconferencing equipment needed to transform a workstation into a DMC system is a pure *add-on* to the computer and does not interfere with the function of components already in place. Hardware, operating system, communication infrastructure, and particularly all application software continue to be used without changes. Thus, to the user, the DMC system appears simply as yet another application system available on the workstation.

- *In the area of available application software.*

  The continued utilizability of the application software as mentioned above is even extended to the duration of a teleconference. In addition to running independently beside a teleconference as a local application, each existing application program shall be usable jointly within a teleconference. This requires application sharing functionality to be provided as part of the DMC system (as identified before).

- *In the design of the user interface.*

  The presentation of the DMC system functionality through its user interface is coherent with those of other application programs on the respective workstation, operating, and window system type. I. e. the user interface complies with same style guides as other application software.

Note, however, that although the integration concept of DMC systems allows to embed teleconferencing equipment seamlessly within the daily working procedures and thus to obtain most benefit for routine work, this does not imply that using a DMC system is always the preferred way of holding a teleconference with business partners. That is, DMC systems are not expected to rule out the use of room-based systems — just as teleconferences will not substitute face-to-face meetings altogether (refer e. g. to section 2.2.3 and to [Ott 95a]).

2.3.4. Enabling Technologies and Environmental Factors

The previous two subsections have outlined the functionality of DMC systems as well as a set of design requirements for the seamless integration of telecooperation and other work processes. While the requirements discussions are necessary to delineate DMC systems from other types of teleconferencing and groupware systems, for actually implementing a DMC system, the base technology needs to be in place and other preconditions need to be satisfied. To complete the outline of a DMC system, this subsection briefly addresses those technological and environmental factors that together provide the foundations for the implementation of desktop multimedia conferencing systems.

What is of importance for the evolution of a new type of complex technology is — besides the potential to solve problems that are commonly perceived as such — the availability of mature

base technology to build on. In addition, the cost for a potential user to obtain and use the new technology has to be outweighed by the expected benefits obtained from its usage. GRUDIN has motivated the emergence of groupware systems in general as follows [Grudin 94a]:

> *In the mid-1980s, the terms groupware and CSCW were coined and conference series and literature appeared. Conditions that emerged in workplaces to encourage this included (a) computation inexpensive enough to be available to all members of some groups; (b) a technological infrastructure supporting communication and coordination, notably networks and associated software; (c) a widening familiarity with computers, yielding groups willing to try software; (d) maturing single user application domains that pushed developers to seek new ways to enhance and differentiate products.*

With respect to the specific area of DMC systems in business environments item (c) is not applicable: most employees are usually reluctant to change their working procedures or tools (refer to section 2.2.3 as well as to [Ott 95a] and the references given there). Item (d) is only of limited relevance since the motivation for DMC systems primarily stems from the idea of integrating cooperation facilities and audiovisual communication — especially by including single user applications that are established in the market place and cannot simply be replaced. The availability of application sharing systems and the lack of market pressure are probably reasons why many manufacturers of successful single user software products still do not seek this way of differentiation.[47]

Items (a) and (b) fully apply to DMC systems and are elaborated on in the following. What has not been mentioned by GRUDIN is the importance of international consensus — new item (e) — when concerned with communication systems as a prerequisite for interoperability and thus for acceptance by the user community.

(a) First of all, the development in the area of PCs and workstations has led to the availability of powerful computers (at reasonable prices) thereby providing the foundation for DMC systems which usually contain computation-intense components. In addition, advances in speech and video compression techniques as well as the rapid development of what is commonly referred to as ''multimedia technology''[48] (e. g. peripheral audio and video devices for PCs) — though mainly for the entertainment industry — have provided the foundations for ''multimedia''-capable workstations.

(b) The second important precondition for teleconferencing systems in general being satisfied today is the widespread availability of networks suitable for multimedia communications: Local Area Networks (LANs) such as Ethernet and FDDI as well as Wide Area Networks (WANs) such as ISDN (in Europe) and PSTN together with powerful modem technology and appropriate compression algorithms. Also, at the time of writing, the global Internet is slowly moving towards becoming a suitable platform for teleconferencing; the technology for obtaining quality of service guarantees from the global Internet is available (but its wide

[47] As soon as international standards defining protocols for e. g. distributed word processors or spreadsheets emerge that allow interoperation of products from different vendors, inclusion of this feature into single user software becomes commercially interesting (or even a requirement).

[48] In the market-place, ''multimedia'' is commonly associated with computers being equipped with high-volume storage media (usually CD ROMs) and hardware for replaying and maybe capturing continuous media, namely audio and video.

deployment of this technology has not really started yet). In particular, ubiquitous connectivity through WANs and the Internet allows reaching virtually all potential collaboration partners through the medium teleconference. Broadband networks (e. g. using ATM) are of interest for future DMC systems, but are of limited importance in the marketplace today.

(e) Finally, international standardization bodies as well as various industry consortia have provided (and continue working on) standards for teleconferencing equipment, and have established consensus about the important aspects of multimedia telecommunication and teleconferencing:

- Telecommunication protocols for information exchange in teleconferences, for setting up and managing teleconferences, and for various types of teleconferencing applications, provide the basis for interoperability between teleconferencing systems of different vendors.
- Standardized algorithms and formats for encoding and representing audio and video information on one hand enable interoperability between different systems. On the other hand, the establishment of a consensus promotes — because of the reduced business risk — investments of chip manufacturers into the development of chip sets for the respective coding algorithms. This in turn forms the basis for the development of hardware for capturing, compressing, and replaying audio and video information (e. g. [Fischer / Gupta 90], [Fox 91], [Wenger 95]).
- Agreements between vendors about application programming interfaces (APIs) in teleconferencing systems allow for easy integration of system components from different vendors.[49] Furthermore, usage of standardized APIs increases portability across different computer platforms. Finally, such APIs simplify interoperability testing by providing common points of reference for test tools [Schindler *et al.* 95] [TELES 96].

2.3.5. Conclusions for the Design of DMC systems

This section has addressed functionality to be covered by DMC systems and outlined further design requirements with respect to the integration into the daily working process. It also has outlined the technological and environmental foundations upon which DMC systems are built. In summary, a DMC system can be characterized as follows (refer to [Schindler 94b]):[50]

- integration of distributed working tools with audiovisual communication;
- provision of multipoint communication to allow for participation of more than two sites, and usage of the necessary technology (e. g. MCUs) in a way that hides the underlying complexity from the user;
- security of all multimedia communications by means of encryption and authentication of all participants;

[49] An important example for a ‘‘standard’’ developed by the industry and accepted by the market-place in Germany is the *Common ISDN Programming Interface (CAPI)* [CAPI 2.0] that provides ISDN applications with a manufacturer-independent interface to ISDN boards.

[50] As important non-technical (economic) factors SCHINDLER points out that the investment cost for a DMC system has to be in accordance with its add-on character and therefore should not exceed the cost for the workstation itself. Also, the operating cost (mainly in terms of telecommunication cost) should remain in the same order of magnitude of a telephone call.

- implementation of the DMC system as add-on functionality to the workstation already in place;
- inclusion of application sharing mechanisms to allow using single user applications within teleconferences;
- consistency of the user interface of the teleconferencing system with style guides for the respective workstation platform;
- conformance to international standards; and
- utilizability on top of different underlying networks transparent to the user.

2.4. Summary

This chapter has introduced the subject of teleconferencing and has outlined the characteristics of and requirements on a particular type of technology intended to offer teleconferencing functionality: desktop multimedia conferencing (DMC) systems — which constitute the background for this thesis.

The first section has presented the research area of groupware systems along with the most important classification schemes for groupware systems: a functional classification with a detailed overview of teleconferencing systems, the time space matrix, and the four phases of telecooperation. DMC systems have been characterized with respect to these classification schemes: DMC systems encompass audiovisual communication as well as (single user and multi-user) application-based collaboration functionality. They are primarily designed for synchronous collaboration of geographically dispersed users, but can accommodate co-located users and hybrid scenarios as well.

Following this general introduction to groupware and teleconferences, an encompassing definition of teleconferences and teleconferencing terminology has been given in the section 2.2. Starting from the definition of a ‘‘conference’’, possible deployment scenarios for teleconferences have been presented and so-called professional or business conferences have been identified as the target for support by DMC systems. Features of such conferences have been identified and ways of reproducing these in teleconferences have been outlined. Furthermore, from the variety of conferencing styles applicable (not only) to business conferences, the target range relevant for DMC systems has been chosen: highly interactive, tightly coupled conferences of arbitrary duration with up to some one hundred participants that may but need not be geographically dispersed; spontaneous conferences fall into this range as well as scheduled ones, many different ways of conference establishment are possible; and, finally, a variety of conference policies have to be considered, e. g. for access control, floor control, conductorship management, etc. In this context, also the limitations inherent to teleconferences when compared to physical meetings have been addressed as have non-technical and technical applicability issues of teleconferencing systems.

Based on the results of the first two sections, section 2.3 has defined the functionality to be provided by a DMC system. Furthermore, the important technical factors for the seamless integration of a DMC system into the workplace have been identified. Also, the technological foundations for the development and broad deployment of DMC technology have been addressed: in particular, the need to base the design of DMC system on commonly accepted international standards has been pointed out.

Overall, this chapter has defined the functionality for DMC systems that can be summarized as follows:

Provision of

- *conference control services,*
- *audiovisual communication and control services,*
- *meeting aids services, and*
- *security services*

with this functionality being

- *tailorable to accommodate a variety of conference settings,*
- *capable of working in arbitrary (heterogeneous) networking environments, and*
- *based on existing and emerging international standards.*

This thesis addresses the design and implementation of an infrastructure for DMC systems that provides communication services for non-real-time information (i. e. data) to groupware applications. The infrastructure has to accommodate teleconference in the range identified in subsection 2.2.2 and has to provide all data and control services to the teleconferencing applications required to implement the aforementioned DMC system functionality. This includes the provision of multipoint communication facilities, conference control functionality, and security services all of which form the basis for the implementation of meeting aids. The provision of (generic) services and protocols to support the design of meeting aids is to be addressed as well.

The subject of audiovisual communication in DMC systems is deliberately excluded from the scope of this thesis because audiovisual communication aspects form a research area of its own and are already covered in related research and engineering activities [Wenger 95]. However, as part of the conference control functionality to be developed in this thesis, means have to be provided for the proper integration of groupware applications providing audiovisual communications and other real-time services.

3

The MCL Infrastructure for Teleconferencing Systems

This chapter introduces an architecture for multipoint communication in teleconferences: the *Multipoint Communication Layer (MCL)*. The MCL defines a concept for an infrastructure that provides conferencing-specific communication and coordination services. The functional range of the MCL is intended to cover those data communication and coordination services required by synchronous groupware applications that are integrated parts of DMC systems. Therefore, the functional range of the MCL is partially derived from the results of chapter two: functionality for direct peer-to-peer communication of application entities is included in the MCL infrastructure as is conference control functionality for setting up and running conferences and for managing groupware applications as components of a conference. Altogether, the MCL constitutes a framework for multipoint teleconferencing as well as for the integration of groupware applications and conference control within DMC systems and defines an infrastructure that provides all the necessary services.

The motivations for developing a widely applicable teleconferencing infrastructure are manifold and follow the same considerations that have motivated the introduction of communication infrastructures in general (the most prominent description of an infrastructure being the ISO Reference Model for Open Systems Interconnection [ISO 7498] [ITU-T X.200]):

- Concentrating basic communication and conferencing functionality and providing these services by means of an infrastructure or a toolkit to all teleconferencing applications supports the concept of separation of concerns in DMC system design. Groupware application designers are relieved from dealing with the non-trivial aspects of group communication and management and may concentrate on the actual application functionality. This is of particular importance since group / multipoint communication constitutes a research area of its own right that is by no means as well understood as is point-to-point communication. Design and implementation of such protocols is very difficult rendering repeated protocol design (by each application developer) not only expensive in terms of time and effort, but also very prone to errors. Therefore, a generic infrastructure helps avoiding duplication of effort for those functions that otherwise would need to be implemented for each groupware application independently.

- Provision of a widely applicable multipoint communication platform enables the use of well-defined and well-documented interfaces — local protocols or application programming interfaces (APIs) — at the level where applications access the infrastructure services. Defining interfaces in a platform independent manner improves portability of applications to different hardware and operating systems to a great extent.[1] Such interface definitions have to be agreed upon in a far-reaching (international) consensus and accepted by the market so that simple portability of groupware applications between DMC systems of different vendors is achievable. This then allows exchanging groupware modules providing equivalent functionality as well as adding new ones to a DMC system regardless of the respective modules' origins, thereby providing the foundation for customizable DMC systems created from a set of building blocks.

- Finally, the definition of a common infrastructure contributes to achieving interoperability among different DMC systems: Firstly, because a commonly used infrastructure covering all the required communication and control functionality avoids the risk of different and incompatible sets of services being used by two DMC systems. Secondly, because the protocols implementing the services (from low level data transmission up to the groupware application protocols) need to be globally agreed upon. Obviously, wide interoperability can only be achieved if the design of an infrastructure is closely tied to international standardization of the overall architecture and its components as well as of the services provided and the protocols implementing these services.

The subject of this chapter is the conceptual outline of the Multipoint Communication Layer as a functionally encompassing — i. e. in that respect ''ideal'' — multipoint communication and conferencing infrastructure. Ideal means that the infrastructure covers all the conferencing-related functionality identified so far as being required by groupware applications: for communication with their peer entities as well as for their integration into a DMC system. A concrete implementation of the infrastructure has to be based on international standards and offer its services through standardized interfaces as far as possible. However, while the MCL services outlined in this chapter account for all the identified requirements, at the time of writing, not all of these services are covered by *existing* international standards:

- From the conceptual viewpoint, this means that gaps in the standardized teleconferencing services (and protocols) for non-real-time groupware applications are identified. Based on the MCL architecture, contributions are made to the standardization bodies with the aim that these gaps be filled in *emerging* and *future* (revisions of the existing) standards.

- For the implementations performed for this thesis, a dual approach is followed: those services that are covered by international standards are implemented in compliance with the respective standards. The other services are rated with respect to their relevance to the respective standardization bodies: those perceived important in the short term are prototyped based on non-standard protocols, provided that the implementation can be performed in a way that is backward compatible to purely standards-based systems. Functionality that is not important to the

[1] For example, the BSD UNIX concept of *sockets* has been a common abstraction and programming interface to communication services provided by the operating system. Despite incompatibilities in various details across the many implementations of *sockets*, their invention has significantly simplified porting applications that require inter-process communication from one system platform to another [Leffler *et al.* 89].

near-term standardization and/or cannot be provided without harming interoperability is not considered further in the engineering part of this thesis.

The first section of this chapter briefly reviews various architectures for teleconferencing systems with respect to the functionality provided and the grouping of this functionality. Based on this review and the results of chapter two, section 3.2 gives an outline of the MCL concepts, its functionality, and its internal sublayer structure. This section also defines the interfaces of the MCL to lower layer communication protocols and its demarcation from MCL-based groupware applications. The functionality of the three generic sublayers of the MCL is discussed in the subsequent two sections. Section 3.3 deals with the basic communication services not specific to teleconferencing while section 3.4 addresses conference management functionality in detail.

3.1. Communication Architectures for Teleconferencing Systems

Chapter two has given an overview of the functionality to be offered by a DMC system to a human user. As the Multipoint Communication Layer (MCL) developed by the author is intended to provide the basis for group communications in DMC systems, it has to support groupware applications in implementing the previously outlined functionality. Part of the functional range to be covered by the MCL can be derived from the findings of chapter two: this includes the required conference control services and the groupware application protocols to be supported. Also, MCL services should be designed to be tailorable to meet the needs of groupware applications in different teleconferencing scenarios. In addition, the MCL needs to provide a variety of services that are invisible to the user, but simplify the design and implementation of groupware application systems and allow them to be integrated into a DMC system. In this section, the particular services required from the (DMC) system designer's perspective are addressed in addition to user visible services, thereby completing the outline of the functional range of the Multipoint Communication Layer. Input on the necessary additional functionality is obtained from past implementation experience as well as from reviewing other group communication architectures for synchronous groupware applications and specifically teleconferencing systems.

The design of the MCL was an evolutionary process including many interactions with the environment over time: On one hand, the Multipoint Communication Layer experienced repeated (minor) refinements that consisted either of adding functionality or of small adjustments to the MCL architecture to better fit the commonly perceived model of data communication in teleconferencing systems (that was only slowly emerging). On the other hand, the author has contributed to shaping this model in the two most important international groups, the ITU-T and the IETF (see below). The results of the research projects described in this section served primarily as input to the design of the Multipoint Communication Layer. The two major standardized architectures for teleconferencing systems presented in this section also provided input to the MCL, but their development has as well been influenced by the author; so that some current (draft) standard specifications reflect various concepts developed during the research carried out for this thesis.

The development on the Multipoint Communication Layer started in early 1992[2] in the context of the EURO.VISION[3] project.

- Initially, the MCL architecture design was primarily based on implementation experience gained from work on the DIDAMES system (see below).

This was complemented by reviewing the results of various other research projects that have developed groupware systems for synchronous collaboration:[4] collaborative editors, drawing tools, multimedia conferencing systems, etc. Of particular importance to the development of the MCL architecture were those research projects that had developed communication architectures as basis for internal use by their synchronous groupware systems as well as those that provided development environments (such as groupware development toolkits or teleconferencing platforms) in order to simplify the implementation of synchronous distributed groupware applications. Out of those, the more prominent research projects with influence on the MCL include

- the Group Communication Architecture of the MERMAID conferencing system,
- the GroupKit toolkit,
- the Touring Machine System, and
- the IBM Lakes architecture.

The initial work on the MCL was only marginally influenced by international standardization because, at this point in time, not much of standardized infrastructures for data communications in teleconferences was in existence.[5] This changed from 1993 onwards, when group communication architectures for data communications and conference control became an intensely pursued goal in major standardization organizations such as the ISO[6], the ITU-T[7], and the IETF[8] as well as the ETSI and other regional standardization bodies. These emerging standardization activities were of particular importance to the design of the Multipoint Communication Layer because an explicit design goal for the MCL design was to base its implementation on international standards as far as possible. Therefore, the author has actively participated in the relevant working groups of the IETF and the ITU-T (both of which have been at the leading edge in standardization of teleconferencing) and has learned from but also contributed to the architectures for teleconferencing of the respective bodies:

---

[2] Refer to [Ott 92a], [Ott 92b], [Ott 92c], [Ott 92d], [Ott 92e], [Ott 92f], and [Ott 93] for further information on this early work.

[3] In the beginning, research and development of the MCL as part of the EURO.VISION project were closely coupled with two projects of the RACE Programme of the European Union: EuroBridge (R2008) and MIMIS (R2025).

[4] The reader is referred to the extensive collections of articles in [Greif 88], [CSCW 88], [CSCW 90], [ECSCW 91], [CSCW 92], [ECSCW 93], and [Baecker 93] as well as the encompassing overview of groupware systems given in [Malm 94].

[5] The drafts of standards that have been available to the author (which also were taken into consideration) include two early drafts from the T.120 series (T.122 and T.123) and a document on audiographic teleconferencing from the ETSI (European Telecommunications Standards Institute) [ETS 300101]. Refer to the sections 3.1.7 and 3.1.8 below.

[6] International Organization for Standardization.

[7] International Telecommunication Union, Telecommunication Standardization Sector; formerly known as Comité Consultatif International Télégraphique et Téléphonique (CCITT).

[8] Internet Engineering Task Force.

- the IETF Architecture for Internet Multimedia Conferencing and
- the ITU-T Architecture for Multimedia Conferencing.

The remainder of this section reviews the aforementioned seven communication architectures (for teleconferencing systems). The intention of this review is not to provide a detailed description of the various architectures, but to highlight the key features considered to be relevant to the design of the Multipoint Communication Layer. Therefore, these architectures are assessed with respect to the following items:

a) the types of information[9] that are handled (transmitted and/or controlled) by the infrastructure — to be able to qualify the infrastructure functionality provided with respect to the information types since different types require different services;

b) the range of functionality provided by the architecture — to verify and augment the services to be offered by a teleconferencing infrastructure from the system perspective; and

c) the conceptual subdivision of this functionality into identifiable (groups of) services — to gain knowledge about the design concepts for teleconferencing architectures.

Subsequently, a very brief presentation is given on a number of further representative architectures that do not address group communication functionality in an as encompassing manner, but the review of which also contributed valuable additional insights to the MCL design.

3.1.1. The DIDAMES Integrated Videoconferencing System

The DIDAMES integrated videoconferencing system, a DMC system, has been developed within the DIDAMES[10] project from 1989 to 1991.[11] The DIDAMES system provides multipoint teleconferencing through a central (MCU-like) system, the *Conference Service Provider System (CSPS)*, among up to fifteen participants each one using a *Conference Service User System (CSUS)*. Communication is supported across various networks including broadband ISDN, narrowband ISDN, and Ethernet. To the user, the DIDAMES conferencing system provides audiovisual communication with up to four simultaneously displayed participants and includes two types of application sharing as well as file transfer functionality.[12]

Internally, a CSUS of the DIDAMES videoconferencing system is constructed of many largely independent service entities that are combined at the application layer by a sophisticated user interface to provide an integrated conferencing service to the user. It is this service infrastructure — also providing the multipoint communication functionality — that is of interest for this review.

---

[9] The information types are categorized into real-time information (e. g. audio and video) and non-real-time information. Non-real-time information are subdivided into (conference) control information, and information of other groupware applications (such as shared editors). For a precise definition refer to section 3.3.1.

[10] DIDAMES (Distributed Industrial Design And Manufacturing of Electronic Subassemblies) was a research project (R1060) within the RACE program of the EEC.

[11] The DIDAMES system is an early precursor of the EURO.VISION system — the standard compliant DMC system for which the architecture described in this thesis has been developed.

[12] For a conceptual outline and overview of the DIDAMES videoconferencing system refer to [Schindler / Heidebrecht 90]. Technical details are described in [Beyer 91], [Dzwillo 91], [Ott 91a], [Ott 91b], [Bastian 92], [Berresheim 92], and [Beyer 92] among others.

Assessment of functionality

a) The architecture of the DIDAMES conferencing system deals with both real-time and non-real-time information. The management layer (see below) processes and forwards control and most other non-real-time information using TCP/IP. Real-time information and screen output of the centralized application sharing are handled at the device layer (see below) for efficiency reasons.

b) The following functions are offered by the DIDAMES system:
   – conference control functionality (including conference configuration, participation control, and floor control),
   – multipoint connectivity through a central system,
   – internetworking by using the Internet protocols (IP, TCP, and UDP) and coupling of different networks through the central system,
   – audio and video control,
   – centralized and replicated application sharing,
   – floor control with various policies for both application sharing and speech, independent as well as coupled.

c) Conceptually, the DIDAMES conference service user system can be subdivided into three layers: the application layer, the management layer, and the device layer.
   • The application layer contains the teleconferencing user interface that integrates the services of the lower layers (and thus is de-facto a groupware application). This layer can also include arbitrary other groupware applications that make use of the services offered by the management layer.
   • At the management layer, six services can be distinguished from which the integrated teleconferencing functionality is constructed:
     – conference control service — for configuring, creating, terminating, joining and leaving conferences, for storing and retrieving persistent information about conferences and participants, and for maintaining transient conference state,
     – video control — for positioning and zooming video images and for selecting the source to be displayed in a particular image;
     – audio control — for audio input selection, muting, and audio floor control for speech;
     – distributed use — for replicated application sharing including synchronization mechanisms and specific floor control;
     – remote use — for centralized application sharing including specific floor control; and
     – workstation integration — for providing access to the local PC and DOS/MS Windows environment in a separate window.

     Conference control, distributed use, and remote use each include independent service-specific multipoint communication facilities based upon point-to-point connections.
   • The device system layer forms the lowest conceptual level of a DIDAMES system. It contains device drivers that provide the management layer with a convenient interface to the hardware and perform operations that need to be located close to the hardware for efficiency reasons. Two communication services are included:

- efficient transfer of audio and video information as well as of redirected screen output;
- adaptation to different networks and provision of a socket interface to the Internet protocol suite.

3.1.2. The Group Communication Architecture of the MERMAID System

The Group Communication Architecture (GCA) [Maeno *et al.* 91] forms the communication platform of the MERMAID[13] teleconferencing system [Watabe *et al.* 90] that has been developed at the C&C research laboratories of NEC in Japan. The intention of the MERMAID system, a DMC system, is to efficiently support cooperation in (geographically dispersed) work groups. With GCA, a new architecture has been developed because ''conventional communication architectures, based on point-to-point interactions, such as the OSI seven layer model, cannot deal fully with group communication and cooperative work.'' [Maeno *et al.* 91, p. 2765]

The Group Communication Architecture is based on point-to-point communication and creates a multiple-user architecture offering services for setting up and controlling interactions among groups of applications. Thus, GCA provides all required group communication functionality to cooperative applications that reside on top.

Assessment of functionality

a) The GCA deals with data and control information that is distributed using Internet protocols. In addition, the GCA controls the distribution of real-time information. Actual forwarding of real-time information is performed by a dedicated (MCU-like) device using a separate transmission medium (ISDN lines).

b) The functionality of the GCA ranges from low-level communication services to application-specific protocols including
   - convening of groups (and invitation of members);
   - maintenance of information about present and past conferences;
   - conference control including participation control, floor control, audio and video control;
   - information forwarding within a single domain (a LAN environment) and across domains;
   - interfacing to different networks; and
   - a set of group applications such as whiteboard, telepointer, and database access.

c) The GCA essentially consists of five functional components: (1) the human interface, (2) the group communication control function, (3) the office support application function, (4) the information management base, and (5) the communication interface, some of which are further subdivided. Out of those five components only (2), (4), and (5) are relevant to the design of a communication infrastructure.

[13] *MERMAID* is the abbreviation of *Multimedia Environment for Remote Multiple Attendee Interactive Decision-making*.

(2) The group communication and control function is subdivided into the following parts:
   - group application layer — applications that use the lower layer functions;
   - group presentation — effective control of media: e. g. voice mixing, video control;
   - group session — control of the conversation, e. g. floor control;
   - group connection — provision of group communication (conference) control such as starting, ending, joining, and leaving a conference; and
   - group relation — maintenance of information about groups and their members: registration and deletion of groups and members, convening, setting up, dissolving meetings.

(4) The information management base stores and provides access to information about users and conferences, e. g. names, addressing information, etc.

(5) The communication interface keeps the higher level communication functionality independent of the networks by providing a unified interface to the underlying network.

3.1.3. The GroupKit Toolkit

GROUPKIT is a groupware toolkit for creating synchronous interactive conferencing applications that has been developed at the University of Calgary [Roseman / Greenberg 92]. The functionality of GROUPKIT is derived from user as well as from groupware application programmer requirements.

GROUPKIT provides an object-oriented run-time architecture for setting up and controlling group communication between distributed processes. One of the key characteristics is a strict decoupling of objects that provide mechanisms from those that define the policies governing the mechanisms' use. GROUPKIT also includes application layer functionality with the focal point being support for WYSIWIS-style shared working surfaces.[14]

Assessment of functionality

a) GROUPKIT deals with non-real-time and control information and uses mechanisms of the INTERVIEWS [Linton *et al.* 91] dispatch library (presumably) on top of TCP/IP for information distribution.

b) The functionality envisioned for the GroupKit toolkit includes:
   - conference control (pre-planned as well as ad-hoc conferences);
   - support for persistent sessions by maintaining session state between meetings on permanent storage media;
   - a robust communications infrastructure supporting unicasting and multicasting in process groups including RPC-style communication facilities for message passing between objects;

[14] The requirements discussion for the GROUPKIT in [Roseman / Greenberg 92] addresses more concepts than are actually implemented. Nevertheless, the following assessment also includes concepts that have not been reported to be implemented because the goal is to acquire knowledge about the overall architecture rather than a specific implementation.

- synchronization and concurrency control mechanisms for objects, such as locking, transactions, etc.;
- provision of flexible and extensible policies for all control mechanisms; and
- general support for distributed applications: mechanisms to enhance shared workspaces by a translucent overlay with telepointing and annotation functionality.

c) This functionality may be grouped into three categories:
- application-specific functions — including control objects that implement policies, telepointing and annotation support;
- conference control functions — for creating, terminating, joining, leaving, and running a conference, controlling the conference policies, and saving the conference state information between sessions; and
- communication functions — for inter-application communication and object message passing.

3.1.4. The Touring Machine System

The Touring Machine System has been developed at the Bellcore Information Networking Research Laboratory [Arango *et al.* 93].[15] The goal of the Touring Machine project is the development of a common system infrastructure for distributed multimedia applications to reduce the burden on the (groupware) application developer.

The Touring Machine system defines a set of logical abstractions: the session, the transport, and the application level. Thereby, the Touring Machine system allows setting up group communication relationships (sessions) for distributed applications by specifying an abstract transport topology and the necessary resources that are then mapped onto the underlying networks and their characteristics. On top of the session control level of the Touring Machine infrastructure, an API is defined that hides the complexity of the underlying layers from the application developer. Applications that make use of the Touring Machine system are called *clients* and access the system services through this API.[16]

Assessment of functionality

a) The Touring Machine system deals with interconnection configuration and resource handling for real-time as well as non-real-time information. Actual information forwarding performed by the infrastructure is restricted to control and other non-real-time information. For distribution of (analog) audio and video information, dedicated hardware bridges are used.

b) The Touring Machine system provides functionality for setting up and modifying communication relationships among groups of distributed applications. The services offered by the infrastructure include:

[15] This subsection addresses the second version of the Touring Machine system. The first version [Bates / Segal 90] that was completed in 1990 provides a subset of the second version's functionality.

[16] A set of applications have been developed in the context of the Touring Machine project, e. g. the Cruiser teleconferencing application [Cool *et al.* 92] and the Rendezvous architecture for creating shared workspaces [Patterson *et al.* 90].

- client registration and authentication — to participate in group communication, application clients have to make themselves known to the infrastructure and provide information about themselves, such as addressing information and policy preferences;
- session management — conference control functionality including negotiation of the session policy;
- logical to physical topology translation — for actual session setup, the Touring Machine translates the abstract specification given by a client into a physical interconnection topology;
- resource allocation — reserving and sharing network (trunks, bridges, etc.) as well as workstation (camera, microphone) resources;
- network configuration — control of (physical) interconnection between devices (information sources and sinks) including configuration of audio/video bridges and switches;
- storage and retrieval of information — about users, registered clients, ongoing sessions, stations and their addresses, etc. including access control; and
- inter-client message passing — point-to-point exchange of datagrams between any two Touring Machine clients without the need to establish a separate session.

c) At the application level, a station manager coordinates the access of multiple clients per workstation to the Touring Machine infrastructure. The infrastructure itself is divided into a session control level, a transport level, and a name server component.
   - The session control level deals with logical control of communication sessions: application clients are grouped into sessions according to the session definitions. The entire session management (creating, deleting, joining, leaving, etc.) is done at this level, including negotiations about session policies, maintaining internal session state, etc. The session policies themselves, however, are defined by the clients.
   - The transport control level translates logical connection topologies of the session level into physical ones, handles resource allocation and network configuration. Thus, this level provides network transparency to the session layer.
   - The name server component is orthogonal to the two control levels. It stores system state information concerning both levels and realizes access control to this information.

3.1.5. The Lakes Architecture

The Lakes Architecture has been developed at the IBM research laboratories as an ‘‘architecture for collaborative networking’’ [IBM 94]. The goal of the Lakes architecture is to provide multimedia group communication services to (groupware) applications independent of underlying networks and specific personal computer platforms.

The Lakes architecture defines a platform that offers a uniform API towards collaborative applications. The focus is not specifically on teleconferencing but more general on setup and handling of information stream(s) between groups of application entities and/or (hardware) devices. Teleconferences may be modeled on top of these facilities at a higher layer.

Assessment of functionality

a) The Lakes infrastructure supports transmission of any type of information, real-time as well as non-real-time, and allows flexible specification of the transmission characteristics for each information stream. For each information type, the appropriate Lakes protocols are employed for transmission.

b) Due to its exclusive focus on establishment and control of information streams, the Lakes architecture offers rich functionality in this respect, but does not provide any conference control functionality at all. The functionality for (real-time) stream management includes:
   - call control between groups of application entities, i. e. setup and teardown of communication relationships;
   - provision of unidirectional transmission facilities (channels) from a single source to one or more destinations with application-defined transmission requirements, such as QoS, signal type (analog or digital), data class (voice vs. video vs. data);
   - optionally linking various channels in so-called channel sets to achieve global ordering of information delivery, inter-media synchronization, and merging of information streams;
   - synchronization and resource sharing by means of (global as well as per-application) tokens;
   - support of generic features for application sharing (e. g. window mirroring, redirection of input and output, etc.) thereby allowing the inclusion of non-Lakes applications into Lakes sessions; and
   - maintenance of node information in a Lakes network in an *address book*.

c) A system based on the Lakes architecture consists of collaborative applications (one of which acts as the call manager), (logical) devices, and the Lakes platform itself.

   The services offered by the Lakes platform may be categorized into
   - application services — namely the building blocks for application sharing;
   - synchronization services — via the token mechanism; and
   - transport layer services — including group communication establishment, QoS-based information transmission, and provision of cross-channel relations;

   The services offered by the call manager include
   - name resolution (address lookup),
   - placing outgoing as well as accepting incoming calls, and
   - implementation of policies for the above functions.

3.1.6. The IETF Architecture for Teleconferencing

Within the IETF, various working groups have contributed to the development of a teleconferencing architecture. These efforts started in the late 1980s with the work on multicasting for the Internet Protocol in order to support group communications [RFC 1075] [Deering 91] [RFC 1112]. Afterwards, improvements to the multicast routing infrastructure — the Multicast Backbone, *MBone* — [RFC 1584] [Deering *et al.* 96] [Ballardie *et al.* 96] have been achieved and

resource reservation mechanisms to achieve a reasonable service quality have been worked out [Zhang *et al.* 93] [Braden *et al.* 96] [RFC 1633].

The first conferencing-specific building block of the IETF conferencing architecture has been a protocol for transmitting real-time information over packet networks: the Real-time Transport Protocol (RTP) [RFC 1889] [RFC 1890]. Public announcement services for teleconferences on the MBone (based on the Session Description Protocol (SDP) [Handley / Jacobson 96], Session Announcement Protocol (SAP) [Handley 96], and their predecessors) have — together with RTP (and publically available software) — promoted the widespread use of loosely coupled teleconferences in the Internet. While also small group meetings are held many of the MBone teleconferences are broadcast-style seminars with only limited interaction. At the time of writing, these services are being augmented by means for calling and inviting participants into conferences [Handley *et al.* 96b] and for tight control of teleconferences [Bormann *et al.* 96a]. Although a lot of research work specific to teleconferencing has been done in the context of the IETF [Schooler 91] [Handley / Wakeman 94] and despite a system model has been accepted early [Schooler 93] [Weinrib 94] [Frivold / Lang 94], a complete picture of the Internet teleconferencing architecture has not been published until 1996 [Handley *et al.* 96a].

Finally, the recently matured security architecture for the Internet (to achieve authentication and confidentiality of information exchange) is particularly relevant to teleconferencing since the Internet multicast infrastructure inherently does not provide any notion of controlling access to transmitted information [RFC 1825] [RFC 1826] [RFC 1827].[17]

As can be seen from the above description, the IETF conferencing infrastructure is not the outcome of a focused effort towards support of teleconferencing (and many parts of the infrastructure are used for other purposes as well) nor has it been developed by a single group. Nevertheless, the following assessment does consider the existing infrastructure as a whole regardless of the origins of its various components.

Assessment of functionality

a) The IETF infrastructure basically covers transmission of all types of information. With respect to supporting groupware applications, however, the focus is, besides provision of conference control, on the exchange of real-time information.

b) The following functions may be considered belonging to the Internet teleconferencing architecture:
   - network layer multicast as basic support for group communication;
   - (network) resource reservation to allow communication across the Internet at a defined quality of service;
   - real-time transport (RTP) primarily intended (but not restricted) to enable transmission and playback of audio and video information (RTP also provides a few control mechanisms that allow running loosely coupled conferences);

[17] Note that some security functions are also integrated into several of the protocols mentioned above.

- scalable reliable multicast (SRM) as a generic framework for applications that require to some (application-specific) degree reliable dissemination of information;[18]
- conference announcements to make information about a scheduled or ongoing conference known to the public or a target group of persons;
- call setup mechanisms to establish a point-to-point conference and to invite persons into ongoing conferences;
- a conference profile description that identifies a conference, provides time and address information, and defines which applications are to be used during the conference;
- conference control for tightly coupled conferences providing functions like conference creation and termination, joining and leaving a conference, etc.; and
- support of security functions such as transmission of encrypted information and authentication of message originators.

c) The IETF conferencing architecture consists of several ''layers'':
   - the Internet Protocol layer with its multicast facilities;
   - the transport layer that includes UDP and TCP as well as RTP, SAP, and other protocols for transmission of structured information and also optionally embeds security services;[19]
   - application level services for conference announcement, call setup, and conference control; and,
   - orthogonal to the other parts, an infrastructure for (network) resource reservation and provision of quality of service guarantees over the various physical networks.

   Furthermore, a variety of related application services including access to directory services as well as user and service location protocols are provided.

3.1.7. The ITU-T Architecture for Multimedia Conferencing

The ITU-T multimedia conference architecture essentially consists of two parts: (low-level) network-specific protocols for establishing and multiplexing physical connections; and a (high-level) network-independent conferencing infrastructure for multipoint communication, conference control, and support of teleconferencing applications.

The low-level protocols have been primarily designed for point-to-point audiovisual communication, in particular video telephony. They define the communication procedures for placing an audiovisual or a multimedia call between two systems, *multimedia terminals*, and offer functionality for combining several independent (physical) network connections as well as multiplexing

[18] Protocols based on the SRM scheme [Floyd *et al.* 95] are deployed in two distributed conferencing applications used in the Internet: *wb* [Jacobson / McCanne 93] and *nt* [Handley 96]. A lot of research has been done in the area of reliable multicast for the Internet (e. g. MTP [RFC 1301], MTP-2 [Bormann *et al.* 94b], RMP [Whetten *et al.* 94], RMTP [Lin / Paul 95], RAMP [Koifman / Zabele 96], and MFTP [Miller *et al.* 97], among others). However, at the time of writing, in the IETF reliable multicast is considered primarily a research rather than an engineering issue (stated on the 36th IETF in summer 1996 in the Open Transport Area meeting). Hence, the Internet Research Task Force (IRTF) is to start working on this subject (stated on the 37th IETF in December 1996 in the Open Transport Area meeting).

[19] As reliable multicast following the SRM scheme, if required for an application, is implemented in the application itself, SRM is not included here.

the resulting hybrid connection to allow simultaneous transmission of virtually any number of information streams (audio, video, control, and telematic application data). Thus, these protocols provide the low-level infrastructure for multimedia communications.

The low-level protocols are developed by Study Group 15 of the ITU-T[20] and are defined in several series of recommendations:

- the H.310 series of recommendation for ATM networks,
- the H.320 series of recommendations for ISDN,
- the H.321 recommendation for use of H.320 on top of ATM,
- the H.322 recommendation for use of H.320 on top of isoEthernet[21],
- the H.323 series of recommendations for corporate (inter)networks, and
- the H.324 series of recommendations for analog telephone networks and mobile systems.

Along with the definitions of the system components, system operation, and communication procedures, a set of audio and video encoding recommendations has been defined.[22] Refer to [Wenger 95] for discussion of the various encoding schemes and for more information especially on the H.320 series of recommendations.

The high-level conferencing infrastructure is described in the T.120 series of recommendations. Its development started around 1990 with the first recommendation of this series being approved by the ITU-T in 1993. At the time of writing, eight recommendations have been approved that define multipoint communication and conference control functionality as well as application protocols. Work on several further recommendations is in progress but has not been completed: these are intended to provide means for controlling real-time information streams, application sharing, and reservation of conferences.

Assessment of functionality

a) All types of information required for multimedia conferencing are covered by the ITU-T architecture. The H.3xx series of recommendations define the multiplexing of audio, video, T.120, and low-level control information. The T.120 series of recommendations addresses multipoint communication for non-real-time as well as high-level conference control information.

b) The T.120 series of recommendations offers the following functions, some of which are directly mapped onto the low-level protocols:
   – a network-independent multipoint communication environment based on a set of point-to-point connections, providing an arbitrary number of applications with unicast, multicast, and broadcast services;

[20] The work has started in the late 1980s for video telephony over the ISDN.

[21] *isoEthernet* is short for *isochronous Ethernet* [IEEE802.9] that defines a variant of the conventional Ethernet [IEEE802.3]. isoEthernet reserves a fraction of the bandwidth of an Ethernet to offer 96 (isochronous) B channels of 64 kbit/s each, access to which is arbitrated through a D channel similar to the procedures (and using the same service interface) of ISDN.

[22] Audio encodings include G.711, G.722, G.723.1, G.728, and G.729; video encodings comprise H.261, H.262, and H.263.

- a token mechanism and optional global ordering of transmitted data for synchronization;
- conference control functionality including means for enforcing policies;
- reservation services for conferences and conference resources;
- support of conductorship and (for conducted conferences) of floor control;
- administrative functions for managing applications within the conference including capability negotiation, resource arbitration, and naming service;
- control of audiovisual information exchange;
- application protocols for telepointer, annotation, whiteboard, file transfer, application sharing, and remote device control;
- support of security functions such as authentication at conference establishment and encryption of information streams during the course of a conference; and
- separation of policy and mechanisms for conference control as well as application protocol functionality.

c) The ITU-T multimedia teleconferencing architecture defines the following functional layers:
   - network-specific multiplexing and control for establishing and controlling the flow of audio, video, control, and non-real-time information (this is achieved by the low-level protocols);
   - network-dependent transport stacks (OSI layers two to four) to provide a unified interface to the different networks for the T.120 infrastructure;
   - a multipoint communication service for multipoint communication and synchronization;
   - a conference control service for managing the conference course (create, terminate, join, leave, etc.; floor control; conductorship) as well as for internally administering the use of applications (T.124);
   - application protocols as building blocks for teleconferencing applications;
   - a higher layer entity (referred to as the *node controller*) that is responsible for implementing the conference policies; and,
   - partially orthogonal to the other layers, functionality for controlling audio and video communications (within the T.120 as well as the H.3xx recommendations).

3.1.8. Miscellaneous Architectures

This subsection briefly describes several further architectures for teleconferencing, from research and from standardization. These are narrower in scope compared to the above and are therefore not described in that much detail. Nevertheless, the projects listed in the following provide valuable further information relevant to the design of an infrastructure.[23] Each of these projects addresses functions that have not been covered above: with respect to application-specific support or integration of applications.

[23] Many other infrastructures or infrastructure components described in the literature are well recognized but are not described here because they address topics too specific for this discussion of general conferencing architectures. Examples are [Ellis / Gibbs 89], Rendezvous [Patterson *et al.* 90], [Rodden *et al.* 92], [Narayanaswamy / Goldman 92], CoEX [Patel / Kalter 93b], DistEdit [Knister / Prakash 93], COLA [Trevor *et al.* 93], and the window sharing work of [Stefik *et al.* 87a]).

- The *MultimETH* system [Lubich 89] is a multimedia teleconferencing system that incorporates distributed editing of structured multimedia documents. The underlying architecture consists of a set of (hierarchical) functional components including a layer for multipoint communication based on point-to-point connections to a central server and a conference management layer for basic conference control operations that also supports a notion of roles and permissions. As application services, a chat and a voting protocol as well as a protocol for distributed editing are supported.

- *MMConf* [Crowley *et al.* 90] is an architecture that supports shared synchronous groupware applications and provides a separate conference manager that implements conference control functionality. MMConf is oriented towards window sharing applications and provides a set of generic functions — e. g. telepointing and drawing, intercepting input, floor control, and file transfer — to support this particular category of groupware applications.

- *MMCC* [Schooler 91] is a conference management entity that performs conference control functions and configures local applications. For overall coordination between different end systems, MMCC entities interact through a conference control protocol. The MMCC architecture assumes independent *media agents* implementing audio, video, and groupware applications that individually handle the communications with their peer entities. These media agents are coordinated locally by MMCC through separate (in part media agent-specific) local configuration protocols to convey the impression of multiple groupware applications belonging to the same conference.

- *CoDraft* [Kirsche *et al.* 93] is a shared graphics editor based upon a multiparty communication platform. This communication platform covers basic conference control functionality and provides various modes for multicasting. At the application level, CoDraft includes support for locking and voting as well as multiparty file transfer for distribution of larger amounts of information.

- The ETSI draft standard *ETS 300101* [ETS 300101] defines protocols, procedures, and equipment characteristics for a multipoint audiographic teleconference based upon ISDN. It provides multiplexing facilities for carrying audio and control information as well as data of telematic applications in a multipoint environment. The draft ETSI standard defines a sophisticated conference control protocol that includes the basic functions for conference and participation management as well as conductorship and floor control. In addition, several application protocols such as ‘‘telewriting’’, still image transfer, and multipoint facsimile are defined.

3.1.9. Conclusions for the MCL Architecture

In the introduction to this chapter, one major motivation of a teleconferencing infrastructure has been described as the provision of conference-specific services to groupware applications in order to avoid repeated implementation of the same (complex) functionality. In this section, several teleconferencing and/or communication architectures aiming at this objective have been analyzed with respect to three criteria: a) the information types for which the architecture is designed, b) the communication and conferencing functionality provided, and c) the concepts for subdividing this functionality.

The outcome is an overview of which types of services need to be provided by an infrastructure to address the needs of teleconferencing applications. Overall, the required functionality outlined in section 2.3.1 is largely matched by the sum of all architectures examined in this section. From these investigations, additional knowledge has been gained about internal management functions, potential user applications, and about concepts for structuring the services:

a) Types of Information handled by the Architectures

The investigated infrastructures distinguish between real-time information (usually audiovisual information) and non-real-time information. Three variations are found with respect to which information types are dealt with and which are not:

> *All infrastructures cover transmission of non-real-time information, i. e. application data and control information.*
>
> *Real-time information is addressed only by a subset of the infrastructures:*
>
> – *Some include services and protocols for control as well as transmission of real-time information.*
> – *Others deal with control of but not with the actual distribution of real-time information.*

One of the concepts that has become apparent from examining various communication and conferencing architectures is a clear separation of real-time and non-real-time information with respect to control and transmission. Even if an architecture targets transmission of both information types (as e. g. in the cases of the Lakes, the ITU-T, and the IETF architectures), the transport services and the protocols employed differ due to the largely contrary requirements of both information types. As a consequence, handling of real-time and non-real-time information is realized independently with integration — if any — being done at the interface to the application.

> *As the Multipoint Communication Layer is designed for non-real-time information only, the following summary does not consider any functionality specific to handling of real-time information, neither for transmission nor for control.*

b) Functionality of the Architectures

The review of infrastructures in this section reveals a range of functionality common to several (if not all) of the aforementioned architectures:

> *Most (if not all) of the considered architectures support*
>
> – *multipoint communication;*
> – *basic conference control (i. e. conference startup and termination as well as membership control); and*
> – *an information base for storing and retrieving persistent conference-related information (e. g. for conference reservation).*

Some of the infrastructures provide

- *means for synchronization between groupware applications, e. g. to coordinate access to shared resources by means of tokens or locks;*
- *conductorship and floor control;*
- *administrative functions for managing the use of groupware applications as part of a conference (e. g. naming, resource allocation); and*
- *definition of generic protocols for certain classes of applications, such as telepointer, whiteboard, joint editing, application sharing, and file transfer.*

Despite its importance, only a few of the above architectures support security functionality including authentication of participants, access control to a conference, key distribution, and encryption of information streams.

This list provides an overview of the functionality to be covered by the MCL architecture with one exception: storage and retrieval of persistent information is beyond the scope of the MCL since this does not address synchronous interactions between users through groupware applications.[24]

c) Conceptual Subdivision of the Functionality

The overall design concept of all infrastructures is to isolate generic communication and conferencing functionality from the groupware applications. The following concepts for organizing this functionality have been found in the investigated infrastructures:

The following concepts are common to most or all of the architectures:

- *independent handling of real-time and non-real-time information;*
- *keeping the infrastructure independent of the underlying network;*
- *distinction between communication (transport) services and administrative services (including conference control);*
- *distinction between generic services — such as transport and conference administration — and services specific to certain (classes of) groupware applications; and*
- *restricting the infrastructure to provide mechanisms but leaving definition and implementation of policies to the respective groupware applications.*

These concepts form the basis for the design of the MCL infrastructure for teleconferencing that is presented in the remainder of this chapter.

[24] However, as section 3.4 points out, this type of functionality can easily be integrated into a DMC system based on the synchronous MCL services.

3.2. The MCL Architecture

The architecture of the Multipoint Communication Layer is based on the design concepts derived from the investigations in the first section of this chapter. The services provided by the MCL cover all the functionality identified in chapter two and the previous section to be essential for a multipoint communication platform which aims at supporting groupware applications and integrate them into a DMC system. This section gives an outline of the MCL architecture and its concepts while the subsequent two sections address the services provided by the MCL in more detail.

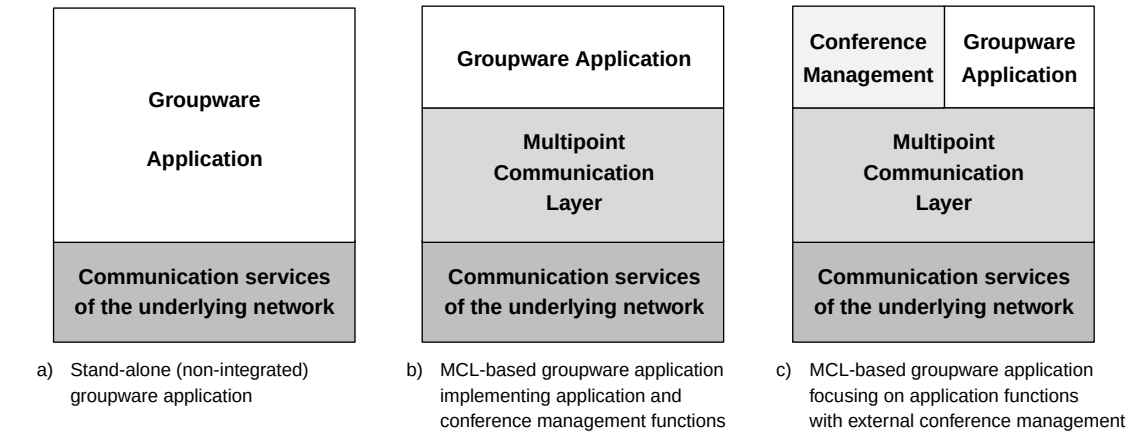


Figure 3.1: Introduction of the Multipoint Communication Layer

Figure 3.1 depicts the location of the Multipoint Communication Layer between the communication services offered by the underlying networks and a groupware application (refer also to [Ott 92b], [Ott 92c], and [Ott 93]). This allows the amount of communication and conferencing-related functionality otherwise covered by a stand-alone groupware application (a) to be reduced for MCL-based groupware applications (b). A dedicated conference management application is considered to be responsible for all the functions and policies related to the conference as a whole, such as joining and leaving (c).

The Multipoint Communication Layer creates a network independent platform for groupware applications based on the communication services offered by underlying protocol layers, e. g. implemented within the operating system (subsection 3.2.1). The multipoint communication and conferencing functionality provided by the MCL is subdivided into four conceptual sublayers that reflect the aforementioned design concepts (subsection 3.2.2): the lower two sublayers, MCL-m/t and MCL-syn cover generic transport and synchronization mechanisms; MCL-adm deals with all functions related to conference management; and MCL-app addresses services specific to certain groupware applications. Security functionality may be integrated as needed across all communication layers of a DMC system (subsection 3.2.3). The MCL services are restricted to provide mechanisms for communication and conference control; policies have to be implemented in the groupware applications (subsection 3.2.4).

3.2.1. Service Interface to Underlying Communication Services

One of the design concepts for the MCL is to offer an infrastructure independent of underlying networks and the communication protocols. This is required to make the MCL applicable in a variety of network environments, including but not limited to ISDN, PSTN, and any networks based upon the Internet Protocol suite. Different networks have different characteristics and provide a different service quality to the user. In order to achieve network independence in spite of these differences, a common service interface has to be defined upon which the MCL functionality is based. This interface then forms the borderline to the lower layers that are usually provided as part of the operating system. For each network that is to be used underneath the MCL, a protocol hierarchy has to be defined that offers this lower layer interface.

Besides being network independent, the service interface must be widely accepted and implementation of the interface should be available on different operating system platforms for later implementation of the MCL.

The single layer of the OSI reference model for which these requirements are met, is the transport layer [Ott 92c] [Ott 92d]:[26]

- The lower OSI layers one to three define the characteristics of a network (transmission medium, representation of bits, link layer protocols, network addressing, routing, etc.) with different networks offering different transmission services. Consequently, a network-independent interface cannot be offered at or below the OSI network layer.
- An ‘‘internetworking layer’’ such as the Internet Protocol may provide network-independence between layer three and four. However, the services offered by the Internet Protocol itself are very basic so that one of the Internet transport protocols, TCP or UDP, is generally used on top of IP. Use of plain IP would require to duplicate this piece of readily available functionality within the MCL. The same applies to other (less widespread) internetworking protocols (such as IPX), too.
- In contrast to the lower four layers, OSI layers five to seven already deal with application semantics. Also, they are widely perceived as too complex and heavyweight for efficient communication.[27] Today, the upper three layers are rarely used and included in virtually no operating system.

The transport layer allows its users to abstract from the services offered by an underlying network layer. Transport protocols enhance the quality of service offered by the network layer to match the transport service user’s needs, thereby making the transport service largely independent of the network in use.

> The transport layer is able to conceal most network specifics from the transport service user for a given transport connection. For example, error control can be adjusted, flow control added, etc. so that a high quality level can be demanded at the transport service interface. Other quality of service parameters are largely beyond the control of the transport layer. For example, the transmission

[26] Note, that in the OSI community the transport layer is considered the lowest layer allowing for a common service interface and quality across different networks as well (refer e. g. to the OSI service overview in [ITU-T X.220].

[27] The complexity of the three upper OSI layers has also been recognized by the standardization bodies. An initiative to allow a more light-weight specification of some of these layers has been started in 1993.

latency and the throughput of a connection have to be accepted as delivered by the network unless other or additional network connections are used and/or network resources can explicitly be reserved in order to increase the available bandwidth and reduce the latency. These latter parameters are largely orthogonal to the design of the MCL (see below). The only assumption is that any transport protocol hierarchy being part of a DMC system's multimedia communication architecture is capable of providing acceptable throughput and latency characteristics to the MCL (refer to chapter four).

Like the entire OSI reference model, the transport services assume point-to-point communication and are consequently defined for such environments. Furthermore, most of today's established wide area networks — such as PSTN and ISDN — are based upon line-switching and thus provide point-to-point communication only. It is one of the MCL's tasks to define a multipoint communication environment on top of point-to-point connections to enable teleconferencing with existing network infrastructure. Besides traditional point-to-point telecommunication, provision of multipoint communication facilities as an (inter)networking service is gaining importance — which is motivated by application areas such as teleconferencing, distributed systems, etc. Networking hardware for various Local Area Networks and, in particular, the Internet protocols support multicasting, i. e. one-to-many and many-to-many information distribution, and offer this service at the (inter)network layer. This functionality should be exploited where available in order to achieve the most efficient information distribution, especially in larger teleconferences.

For example, using a fan-out of four point-to-point transport connections (e. g. TCP) instead of a single multicast transport connection, requires the same piece of information to be transmitted four times instead of once by the sender which incurs ''cost'' in terms of local resources and computation time. Furthermore, the PDUs containing the information potentially traverse the same network(s) more than once thereby reducing the effective network bandwidth. Finally, the larger number of PDUs potentially increases forwarding latency and packet loss in routers.

Due to the inherently different properties of point-to-point and multicast (multipoint) communication, a separate transport service interface has to be defined for multicast communication and entirely different transport protocols have to be employed to provide a comparable quality of service.

Point-to-Point Communication

For point-to-point communication, transport services with similar service interfaces are defined for all network and internetwork protocols, and implementations of the transport interfaces come with virtually all of today's operating system platforms. A suitable and well-defined service interface for the transport layer is the OSI transport service definition [ITU-T X.214] that also maps easily to non-OSI transport protocols such as the *Transmission Control Protocol (TCP)* [RFC 0793] with minor additions [RFC 1006], SPX, and AppleTalk.[28]

The OSI transport service defines a variety of parameters to describe the desired quality of service. Taking into account that the MCL is intended to distribute information reliably and that ideally no readily available functionality shall be duplicated within a the MCL, the quality of service expected from the transport layer is error-free, flow-controlled, and sequence-preserving transmission of information units of virtually arbitrary size. Achieving this requires a connection-oriented

[28] For descriptions of the latter two refer e. g. to [Martin *et al.* 94].

communication at the transport layer, so that the following three transport services of X.214 are used:

- `T-CONNECT` — to establish a point-to-point connections to a peer entity,
- `T-DATA` — to transmit and receive information on this connection according to the aforementioned quality of service, and
- `T-DISCONNECT` — to tear down the connection.

Multipoint Communication

For multipoint communication, neither a commonly agreed on transport service interface nor a (set of) standardized multicast transport protocol(s) have been defined so far. From the ITU-T and the ISO, proposals do exist for multicast transport service definitions: X.6 [ITU-T X.6] and the revisions to X.214 [Moulton 94] [Mauthe *et al.* 95] the latter of which is a joint effort between the ISO and the ITU-T working groups. However, at the time of writing, neither the ISO nor the ITU-T have defined a standardized multicast transport protocol to provide (a subset of) these services. In the area of multicast transport protocols that offer reliable transmission services comparable to the OSI transport layer for point-to-point communication, many research projects have produced a variety of protocol specifications for which prototype implementations exist (refer to section 6.2.1). However, at the time of writing, none of these are being considered a candidate for standardization.

The single widely accepted multicast service interface with the appropriate protocols in place has been defined in the IETF at the network layer: IP multicast. IP multicast defines a group-based addressing scheme (class D Internet addresses) along with three ''services'':[29]

- `JOIN` — to join a multicast group and subsequently receive all the datagrams addressed to this very group with best-effort quality of service;
- `LEAVE` — cease information reception of datagrams destined to a given multicast group; and
- `DATA` — to originate datagrams to and receive them from a multicast group (this service is identical to unicast transmission of datagrams apart from the address type).

Like in unicast IP, datagrams are delivered *best-effort*, i. e. datagrams may be lost, duplicated, reordered, and their payloads may contain bit errors. Due to the lack of a standardized multicast transport protocol the existing *User Datagram Protocol (UDP)* which provides for application addressing and error detection (but not correction) is employed in conjunction with IP multicast.

Following current practice, the approach taken for the MCL design in order to exploit multicasting facilities is to make use of the IP multicast service interface with UDP as the transport protocol.[30] Reliability has then to be added specifically for the purposes of the MCL as a separate layer on top of UDP (refer to chapter six). Once a standardized reliable multicast transport becomes available, the aforementioned reliability layer is intended to be substituted by this protocol. To prepare this later migration, the MCL makes reasonable assumptions about the services provided by the reliability layer that are — at current knowledge — likely to be met by a future reliable multicast transport service.

---

[29] IP multicast has been pioneered by DEERING [Deering 91]. For maintaining group membership refer to [RFC 1112] and [Fenner 97]. For multicast routing of IP datagrams through complex internetworks refer to [RFC 1075] [RFC 1584] [Pusateri 96] [Ballardie *et al.* 96], and [Deering *et al.* 96].

[30] This restricts the use of multicasting to IP-based network environments. Note, however, that the

These complementary service interfaces for point-to-point and multicast networks delineate the Multipoint Communication Layer towards the lower protocol layers as depicted in figure 3.2.

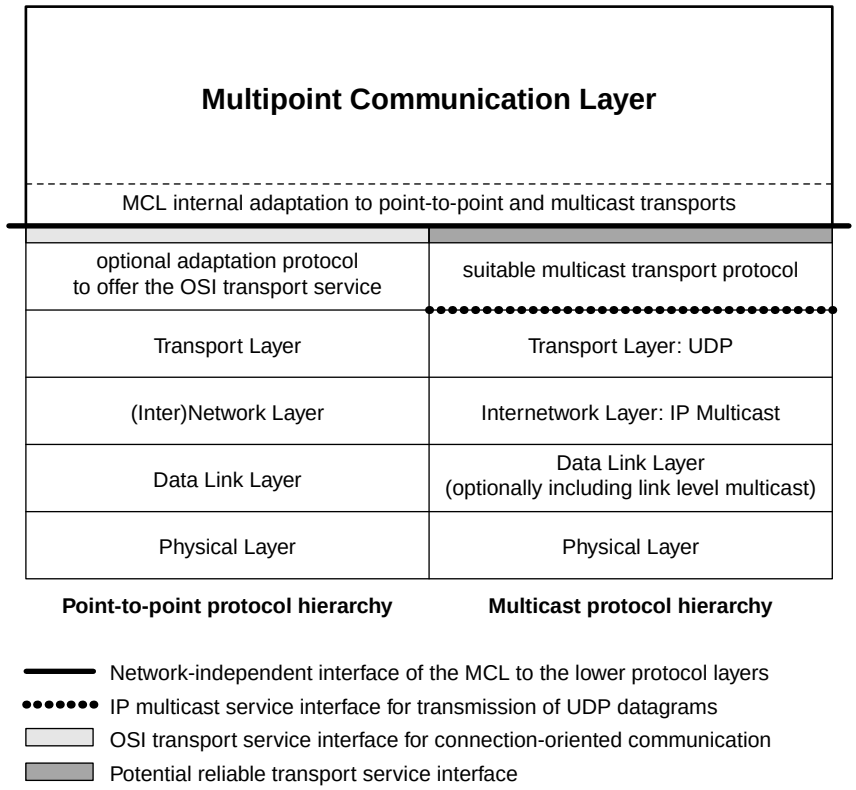


Figure 3.2: Interfacing the MCL to point-to-point and multicast transport

3.2.2. Structure of the MCL

As stated above, the Multipoint Communication Layer covers the functionality identified in chapter two and in the previous section. The MCL's internal structure to provide this functionality reflects the concepts described in the previous section. These concepts include the separation of transport services from conference management services, and that of generic infrastructure services from application-specific protocols. Furthermore, the implementation of policies (for conference management as well as for applications) is left to the groupware applications themselves.

The functionality of the MCL is subdivided into four largely self-contained conceptual[32] sublayers each of which aggregates related services out of those summarized in section 3.1.9:

- MCL-m/t (multipoint transport) offers multipoint communication services on top of arbitrary networks;

Internet is at the time of writing the single protocol architecture that precisely defines many-to-many information transmission at all. Furthermore, the moderate quality of service requirement for multicast transport allows easy adaptation to other environments as soon as they evolve.

[32] The term *conceptual* is used to emphasize that a specific implementation need not reflect this structure nor are specifications necessarily organized following this subdivision.

- MCL-syn (synchronization) provides various means for synchronizing groupware applications within a teleconference;
- MCL-adm (administration) deals with all the conference-related functionality including the integration of groupware applications; and
- MCL-app (application) supplies a selection of services specific to certain classes of groupware applications which may be used to construct complex teleconferencing applications on top.

Figure 3.3 depicts the structure of the Multipoint Communication Layer and indicates which sublayer provides which of the teleconferencing services identified before. A rough outline of the respective sublayer's functionality is given below. Note that security functionality is relevant to all four sublayers as well as to the transport used underneath and the groupware applications on top. Therefore, security functionality is dealt with separately in the following subsection.

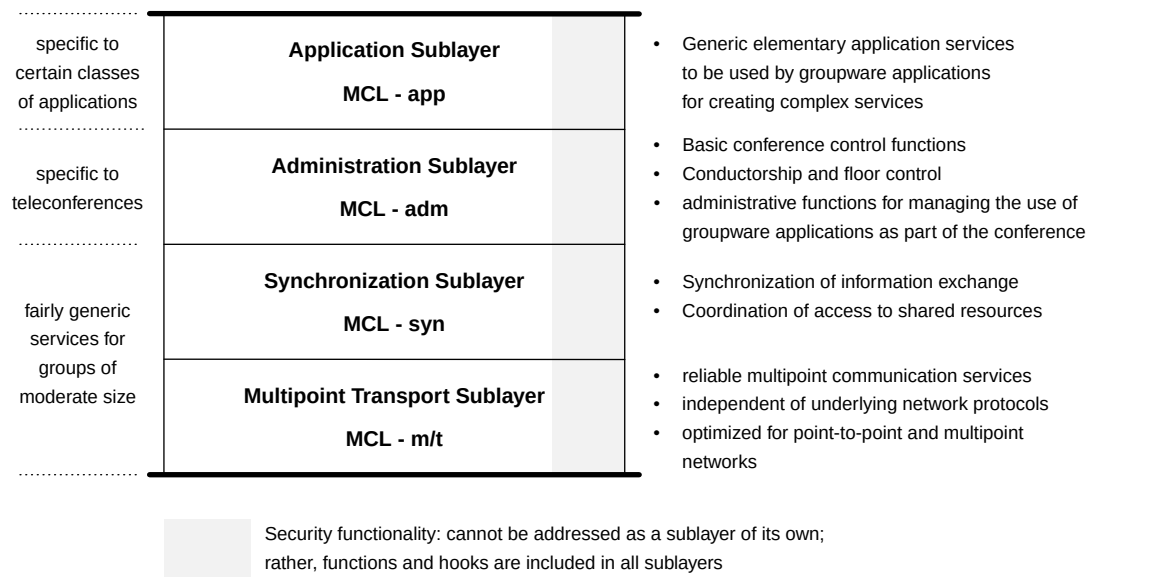


Figure 3.3: Conceptual layers of the MCL

MCL-m/t: Multipoint Transport Sublayer

The main task of the Multipoint Transport sublayer of the MCL is to provide uniform multipoint communication facilities to higher MCL sublayers and groupware applications across all kinds of networks. As independence of physical networks and layer one to three protocols is already achieved by the underlying transport protocols, the MCL-m/t only has to bridge between point-to-point and multicast networks. On one hand the MCL-m/t constructs multipoint communication services on top of point-to-point networks. On the other hand, the MCL-m/t makes use of multicast capabilities of networks where available to optimize resource utilization. The service interface offered to the higher layers remains identical, regardless of whether or not an underlying transport supports multicasting. Also, arbitrary networks (with and without support for multicasting) may be combined to form a single uniformly looking multipoint communication environment.

From the service point of view, the MCL-m/t is capable of joining any number of DMC systems located in arbitrary networks into a single group communication relationship. All members of this group are considered part of the same conference and communication is possible between and restricted to all DMC systems belonging to the group. That is, the group defines the borders for information exchange in a teleconference.

Within the group, multiplexing and subaddressing are provided so that each groupware application entity on any DMC system in the group can be reached individually (unicast). Also, sets of teleconferencing application entities may form subgroups so that information may be destined to all of them simultaneously (multicast/broadcast). All communication within the group is reliable, i. e. information sent to one or more entities reach all of them. Finally, ordering of information units from a single as well as from multiple senders is included here. A more detailed description of the transport services offered by the MCL-m/t is given in section 3.3.1.

MCL-syn: Synchronization Sublayer

The Synchronization sublayer of the MCL makes use of the MCL-m/t services for information exchange and provides services that simplify the design of (distributed) groupware applications with respect to achieving consistency. These services essentially include three groups: synchronization between various information streams, semaphore-like mechanisms to coordinate access to shared resources, and various checkpointing facilities. Refer to section 3.3.2 for details about which services are offered by the MCL-syn.

Note that the delineation between transport and synchronization services is less clear than the borders to and between the upper two MCL sublayers. To some degree, it is a design decision whether or not to separate transport and synchronization services and where to draw the borderline. In fact, many infrastructures evaluated before do not support this separation. From a conceptual point of view, however, synchronization is oriented at application semantics while transport is concerned with moving information units among applications — which are two entirely different aspects of communications. Therefore, the MCL infrastructure concept supports a strict separation of transport and synchronization functionality (although the standards on which the implementation described later in this thesis is based on do not).

MCL-adm: Conference Administration Sublayer

First of all, the Conference Administration sublayer of the MCL offers conference control services to the (human) user. These *user-visible services* essentially comprise the functionality outlined in section 2.3.1, i. e. conference configuration (including assignment of roles and privileges), participation management, floor control, and security services (e. g. authentication). In the context of participation management, MCL-adm triggers setup and teardown of transport connections of the MCL-m/t in order to establish, modify, and destroy the communication group for the conference. The MCL-adm also provide mechanisms to implement a variety of policies.

In addition, the MCL-adm provides the functionality to integrate a set of groupware applications into a DMC system. These are called *internal management functions* because they are hidden from the user. The internal management functions allow to keep track of local groupware applications, to distribute information about these applications to other systems, and to maintain a coherent view of the current conference state across all local applications.

Section 3.4 gives an extensive description of the services provided by the Conference Administration sublayer.

MCL-app: Application Sublayer

The Application sublayer of the MCL covers services that simplify the design of groupware applications at the level of the application functionality rather than the communication services. The model for the MCL-app is derived from to the structure of the OSI application layer, i. e. two categories of services exist:

- On one hand, abstractions of the underlying multicast communication are offered by providing certain programming paradigms. As these services are independent of the groupware application in use, following the OSI reference model, they are termed *Common Application Service Elements (CASE)*.[33] In this thesis, no detailed specification of common services to be included in the MCL-app is given. Nevertheless, examples of potential MCL-app services are listed in table 3.1.

- On the other hand, a set of services each specific to certain classes of applications is provided from which complex groupware applications may be constructed by combining the various services as needed. In the context of the OSI reference model application layer, such services are termed *Specific Application Service Elements (SASE)*.[34] In the discussions in chapter two, a set of functional groupware categories has been identified to be relevant to teleconferencing systems. Services and the respective protocols that implement functions of these groupware categories are candidates for the application-specific part of the MCL-app (table 3.2[35]). Again, details about the services to be provided are beyond the scope of this thesis.

Unlike in the OSI reference model, in MCL-app the common and the specific services are not necessarily considered to have a hierarchical relationship, i. e. specific services are not based on common services. The sole distinction in the MCL-app is the level of specialization to serve a certain type of groupware application. Apart from that, common and specific services constitute building blocks of the same toolkit from which groupware applications may be constructed.

Interactions of the MCL sublayers

In the OSI Reference Model, providing a layered architecture means that the (N)-layer makes use of (N-1)-services in order provide the (N)-services to the (N+1)-layer. For each layer (N), only the interactions with its neighboring layers (N-1) and (N+1) are defined. In the OSI Reference

[33] In the OSI reference model, three basic CASEs are defined: the Association Control Service Element (ACSE) [ITU-T X.217] [ITU-T X.227] for setting up and tearing down connections, the Reliable Transfer Service Element (RTSE) [ITU-T X.218] [ITU-T X.228] that provides reliable transfer of information units of arbitrary size, and the Remote Operations Service Element (ROSE) [ITU-T X.219] [ITU-T X.229] (revised and extended in [ITU-T X.880] [ITU-T X.881] [ITU-T X.882]) defining remote procedure calls. Two further service have been defined for application level synchronization: Commitment, Concurrency, and Recovery [ITU-T X.851] [ITU-T X.852] and Transaction Processing [ITU-T X.860] [ITU-T X.861] [ITU-T X.862]).

[34] Examples in the OSI reference model for services that may be used to build sophisticated application based upon standardized protocols are File Transfer, Access, and Management [ISO 8571], Cooperative Document Handling [ITU-T T.190], and Document Transfer and Manipulation [ITU-T T.431].

[35] The specific services listed in the table have already been described in chapter two. Therefore, no repeated explanation is given here.

Model, the term ‘‘sublayer’’ is used to indicate the subdivision of a layer and denotes a grouping of functions of an (N)-layer. One or more sublayers of an (N)-layer may be bypassed.

Common Services	Description	Examples
Application Integration	executes the potentially complex procedures to properly integrate a groupware application into a DMC system; this service combines the lower three MCL sublayer services appropriately and offers a simple service interface to the application; an entity providing such a service is referred to as *Application Resource Manager* in the ITU T.120 series of recommendations	Generic Application Template [ITU-T_T.121]
Voting	provides a mechanism for a group of entities to decide about actions, state changes, etc. based upon agreement according to certain voting rules (e.g. simple or 70 per cent majority, weighed votes, etc.)	Agreement Protocol [Shenker *et al.* 95], ISO proposal on voting and polling [ISO/IEC 92]
Transparent Data Channel	simulates a point-to-point connection-oriented or connectionless channel for transparent transmission of information between any two entities in a teleconference; this allows to run traditional point-to-point communication protocols in a multipoint teleconferencing environment	this issue was repeatedly discussed in the T.120 working group of the ITU-T
Remote Procedure Call (RPC)	offers the point-to-point procedure call paradigm in a multipoint environment for client-server style interactions; a simple point-to-point RPC could for example be run in a transparent data channel	Sun RPC [RFC 1057] [RFC 1831]
Multipoint RPC	provides RPC semantics adapted for multipoint environments: examples for adaptations are gathering (and optionally aggregating) information from multiple entities upon request; seeking information from any suitable rather than a particular entity (anycasting); and triggering certain actions at the receiving entities without expecting a response	M-RPC [Schumann 96] [Tanaka 96], Group RPC [Wang *et al.* 91]

Table 3.1: Potential common services of the MCL-app

Specific Services	Examples
Joint viewing	T.126 [ITU-T_T.126], wb [Jacobson/McCanne93], CoDraft [Kirsche *et al.* 93]
Telepointer	[Nakajima 93], T.126 [ITU-T_T.126], wb [Jacobson/McCanne93], CoDraft [Kirsche *et al.* 93]
Shared whiteboard	T.126 [ITU-T_T.126], wb [Jacobson/McCanne93], CoDraft [Kirsche *et al.* 93]
Shared editor	DistEdit [Knister / Prakash 93], ShareKit [Edlich 95]
(Multipoint) file transfer	T.127 [ITU-T_T.127], [RFC 1235]
Application sharing	SharedX [Altenhofen 90] [Abdel-Wahab/Jeffay 92], RemoteUse [Ott 91a], DistributedUse [Ott 91b], Xy [Bormann *et al.* 94a], T.SHARE [Romano 97]

Table 3.2: Potential specific services of the MCL-app

The architecture of the Multipoint Communication Layer follows these OSI conventions. The MCL defines a a set of services to be used by groupware applications (the ‘‘higher layers’’) and interfaces through the OSI transport services to the underlying networks. The overall services offered by the MCL are defined by the union of the services provided by the individual sublayers. The services of the lower three sublayers are functionally complementary to one another rather than forming levels of abstractions of the same functionality. The MCL-app sublayer provides various levels of abstractions hiding details of lower sublayers but still allows access to lower layer services. It is at the discretion of the MCL service user to base its functionality exclusively on MCL-app services or to use lower sublayer services directly (e. g. if the MCL-app does not satisfy the specific application’s needs).

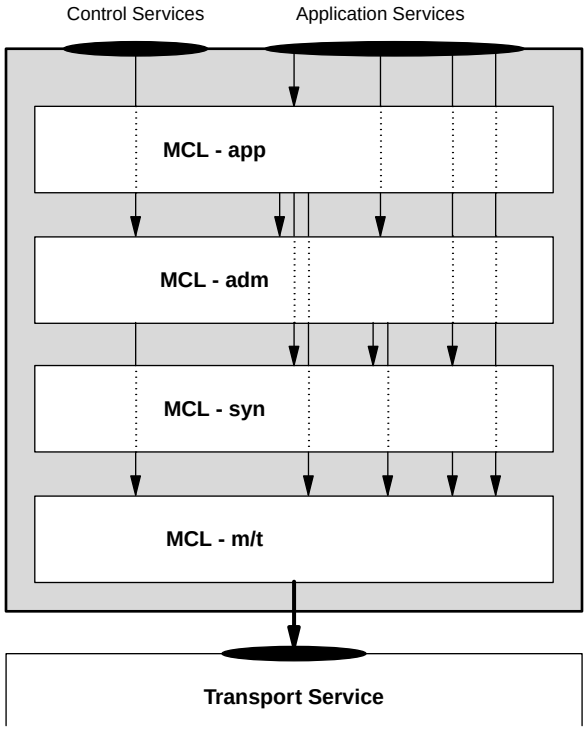


Figure 3.4: Usage relations within the MCL

Within the Multipoint Communication Layer, the MCL-m/t provides the foundation for all inter-system communication and consequently all other sublayers make use of MCL-m/t services. MCL-syn provides add-on services that facilitate synchronization which are used by higher layers (including the groupware applications) to the extend needed. MCL-adm makes use of both MCL-m/t and MCL-syn services to provide the conference administration functionality.[36] Together, these three layers provide the infrastructure functionality of the MCL. The various parts of the

[36] Some of the MCL-adm services require local interaction between application entities and a dedicated conference management entity (refer to section 3.4). These local interactions are not carried out through the MCL-m/t service. Rather their implementation is considered local matter. For the specific implementation of the MCL done in the context of this thesis, however, the means for local interactions are specified in chapter four.

MCL-app make use of any combination of the infrastructure services required to perform their respective functions.

Figure 3.4 depicts the internal usage relationships between the MCL sublayers and the services offered to the MCL service user. In addition to what has been described above, on the left hand side a control service access point is shown. While all the other MCL services are applied within an established teleconference, the control services are used for its creation and destruction. At the MCL-adm sublayer this means verifying access to the conference and updating state information accordingly. At the MCL-m/t sublayer, point-to-point or multipoint transport connections carrying the information streams of the teleconference are set up or torn down. To prevent race conditions on a DMC system, access to the MCL control services is restricted to a single MCL service user entity, e. g. the conference management entity introduced in beginning of this section (figure 3.1).

3.2.3. Security Aspects

The MCL and its four sublayers as discussed so far do not address security functionality. In particular, no separate sublayer, e. g. an *MCL-sec*, has been defined to provide the necessary services. The rationale behind this is that security functionality cannot easily be centralized at a single (sub)layer within a DMC system. Rather, security functions are potentially required at all levels of a teleconferencing system depending on the application scenarios, the security requirements of the users, the environment the DMC system is used in, the groupware applications, etc. and hence have to be tailored to specific needs. The respectively required security functions may have to be applied within the Multipoint Communication Layer at different sublayers, underneath the MCL in the transport protocol hierarchy, and/or above the MCL within a groupware application itself.

In general, security requirements for information exchange are expressed by means of the following four commonly accepted security attributes:[37]

- *Confidentiality* mechanisms ensure that an intruder is unable to interpret a piece of information in the form it is transmitted ‘‘over the wire’’ (prevention of eavesdropping). Confidentiality is achieved by enciphering the information before transmission.

- *Authentication* of the originator of a piece of information prevents intruders from introducing false information without this being recognized by the recipient. Authentication is realized by means of digital signatures.

- *Integrity* of information guarantees that a piece of information received by a recipient is identical to what the originator has sent, i. e. integrity services detect modifications to the information while it is in transit. Integrity may also be achieved by digital signatures.

- *Non-repudiation*, finally, does not address attacks of intruders but rather malicious behavior of either of the communicating parties themselves. Providing non-repudiation ensures that the sender of a piece of information cannot deny to have sent it afterwards and that the receiver cannot deny reception of information later on. Mechanisms required to implement non-repudiation include digital signatures and trusted third parties.

[37] Refer e. g. to [Tanenbaum 89], [Schindler 90], [Bormann 91], and [Schneier 96].

The following list identifies at which layers in a DMC system which of the above security mechanisms may be placed and also points to standards that address security functions at the respective levels.

- *At the physical network interface*

  Encryption techniques may be applied to all data being sent or received across a physical network interface (e. g. an Ethernet or ISDN interface). However, encryption is usually restricted to the payload information because control information (non-data bits in the Ethernet frame, the D channel protocol in ISDN) is required for addressing and other control functions. At this level, usually no means for authentication and key exchange are provided so that these functions need to be performed out of band.

- *Within the (inter)networking architecture*

  For many networking architectures, security concepts to be applied at and below the transport layer. The security concept for the Internet Protocol architecture [RFC 1825] [RFC 1826] [RFC 1827] provides authentication of peers upon connection setup, authentication of individual data packets (for connection-oriented as well as a connectionless transmission), and the encryption of the payload information being transmitted (including or excluding transport protocol headers) — per connection or per datagram sent. Such concepts may be extended to include mechanisms for exchanging encryption keys at the beginning of a conversation.

- *Within a multiplexed connection*

  The ITU-T H.32x series of recommendations provide for video telephony and videoconferencing aggregation of multiple line-switched connections and multiplex the resulting aggregated transmission capacity to provide subchannels for different media (e. g. audio, video, data, control). Comparable to encryption per connection for IP networks, these subchannels may selectively be encrypted for transmission using the same or different encryption keys. Key exchange is supported and keys may be changed dynamically during transmission. The communicating peers may be authenticated initially upon setup of the aggregated connection and then repeatedly throughout their lifetime [ITU-T H.233] [ITU-T H.234] [ITU-T H.245] [Toga 97].

- *At the transport layer*

  Independent of whether or not a security infrastructure is in place for the respective transmission medium, the transport service user may decide to encipher all information being sent across a particular transport connection. As with encryption at the physical network interface, no infrastructure is in place for authentication and key exchange so that these functions have to be performed out of band or by higher layer protocols.

- *At the MCL-m/t sublayer*

  The MCL-m/t sublayer can efficiently support encryption of the information it conveys. Encryption may be done per multipoint communication group, i. e. all information belonging to the same teleconference is encrypted with the same key. Alternatively, encryption keys and modes may be selected on a per communication channel basis, i. e. per destination address (or even per source).

- *At the MCL-adm sublayer*

  While the MCL-m/t sublayer can deal with encryption of all information transmitted through the Multipoint Communication Layer, the MCL-adm is the place to carry out the corresponding control functions such as user (and application) authentication as well as key exchange. Control protocols for tightly-coupled conferences such as T.124 [ITU-T T.124] and SCCP [Bormann *et al.* 96a] perform some the these control functions and leave encryption to the underlying infrastructure or to the groupware applications. However, the MCL-adm may encipher the control information streams of the conference — which may be done at the MCL-adm itself or using MCL-m/t functions.

- *At the MCL-app sublayer*

  The MCL-app sublayer has knowledge about (sets of) communication channels being used within the same application session. Therefore, at this sublayer security functions (authentication, key exchange, and encryption) can be coordinated to that respect. Also, MCL-app services are conceivable that provide groupware applications with security functionality, e. g. offering a selection of authentication protocols including means for negotiating the one to use.

- *Within the groupware application*

  Finally, the groupware applications may perform all the security functions by themselves. This allows them to tailor the use of various security functions to their specific needs (e. g. decide on a per-message basis whether and how to encrypt and/or authenticate a message).

This overview gives an impression of the various possibilities to integrate security functionality into the communication protocol hierarchy of a DMC system. It also shows that addressing security functions solely within the Multipoint Communication Layer is insufficient to fully secure a DMC system: as audio and video information are not sent via the MCL, confidential transmission of these information types cannot be achieved by means of the MCL. However, the MCL can provide mechanisms for user and groupware application authentication, key distribution, and encryption of control as well as non-real-time data transmitted by means of MCL services. Additional security mechanisms may be applied underneath the MCL or within the groupware applications to further improve the DMC system security. MCL services may be used for key distribution and changing the keys in use. Furthermore, the MCL-app sublayer can provide groupware applications with a collection of security protocols and mechanisms to improve security within the groupware applications. Security services that may be provided at the MCL-m/t and the MCL-adm sublayers are addressed in sections 3.3 and 3.4, respectively.

Note, however, that the provision of security services and their application in daily use of DMC systems is rendered more difficult by further technical and non-technical factors (e. g. [Tanenbaum 89]):

> The major technical obstacle is the lack of a world-wide public key infrastructure that would allow personal authentication of any participant based on public key algorithms without prior knowledge of the participant.[38] For the short term, this problem needs to be circumvented by having ‘‘closed’’ groups that allow mutual authentication. A closed group, for example, could be all the customers making use of conferencing services offered by a particular MCU operator.

[38] One approach to a suitable public-key infrastructure has been the OSI Directory Service [ITU-T X.500] the first vision of which has been completed in 1988. Although further revisions added and clarified

On the non-technical side, laws in many countries prohibit or restrict use and/or export of certain types of cryptographic technology. Usage restrictions may prohibit the actual use of effective cryptographic algorithms so that confidentiality may not be achievable for teleconferences that involve or are carried out within certain countries. The most prominent example for such a country is France. Export restrictions (e. g. of the USA) on security technology may prevent availability of powerful algorithms to users in many (non-industrialized) countries and therefore again limit the applicability of security technology if sites in such countries are involved in a teleconference.

3.2.4. Outline of MCL-based Groupware Applications

The Multipoint Communication Layer provides its services to groupware applications that are part of a DMC system so that the application need not care about protocol issues of group communication, conference integration, and even certain application protocols. Rather, groupware applications to be integrated into an MCL-based DMC system are required to make use of the services of at least some of the MCL sublayers: use of the MCL-m/t is mandatory because it provides the basic multipoint communication facilities; use of MCL-adm is also required since otherwise integration into the teleconferencing system and interoperability are not achieved. Direct utilization of MCL-syn and MCL-app services is optional and depends on the groupware application and its needs.

In order to use the MCL services correctly, a groupware application may be based entirely on a (set of) specific MCL-app services that already incorporate proper use of the lower sublayers of the MCL. Alternatively, the groupware application can make use of the Application Resource Management service of the common MCL-app services for correct ‘‘integration’’ into the teleconference but implement its protocols itself by directly accessing MCL-m/t, MCL-syn, and possibly common MCL-app services. Finally, a groupware application may directly access all the lower sublayer services and combine them on its own to integrate itself into a conference as well as to implement its application protocol.

While the Multipoint Communication Layer provides the communication facilities in the context of a teleconference, the groupware application has to implement a variety of further functions to serve its specific purposes. These include but are not limited to the following:

- application-specific policies according to which the services of the MCL infrastructure are used;
- user interface to present the application functionality to the human user;
- the actual application functionality in terms of which MCL-app services to use and how to combine them; and
- additional application-specific services and protocols if the services provided MCL-app are insufficient for certain purposes.

It has repeatedly been mentioned that, besides groupware applications that implement meeting aids and other means for human-human interaction in a teleconference, a dedicated conference management application exists that is responsible for carrying out user actions related to the

---

functionality, there is no broad support, neither by the industry nor by the telecommunication service providers. At the time of writing, efforts in the IETF are underway to create a public key infrastructure that is based on some of the concepts of X.500 but is much more light-weight. This specification of this infrastructure shall become an Internet ‘‘proposed standard’’ in 1997.

conference as a whole (such as joining or leaving a conference).[39] The conference management application is entirely based on the MCL-adm services. In particular, it uses the control services to set up, join, leave, and terminate a conference as well as to control the conference course — which are unavailable to other (''regular'') groupware applications. It communicates locally with the other groupware applications by means of the MCL-adm services in order to control their integration into the conference.

Apart from its specific tasks, the conference management application is comparable to the other groupware applications. The conference management application is responsible for implementing the desired conference policies based upon the MCL mechanisms. It may provide an interface to the human user as well as interfaces to other local system components that may complement or substitute the user. With respect to security, for example, if user authentication is to be performed, it could be the task of the conference management application to provide and verify public key certificates, i. e. to interface to a public key infrastructure. Also, a conference management application on an MCU may provide an interface to a conference booking system.

3.2.5. Summary

This section has outlined the overall structure and functionality of the Multipoint Communication Layer, its interface to lower communication layers, and the functions to be implemented in MCL-based groupware applications. Figure 3.5 summarizes the above discussions. The Multipoint Communication Layer is divided into four sublayers. The functionality covered by the MCL as well as its internal structure reflect the design concepts highlighted in section 3.1.9.

The lowest sublayer, MCL-m/t, implements an infrastructure for multipoint communication on top of the transport layer service interface. It is capable to interconnect any number of DMC system across different networks making use of point-to-point as well as multipoint transport ''connections''. The next high sublayer, MCL-syn, provides generic services to simplify synchronization of groupware applications in a teleconference. Both MCL-m/t and MCL-syn are generic services (compared to the upper two MCL sublayers); their respective functionality is defined more precisely in sections 3.3.1 and 3.3.2 with security services of these sublayers being addressed in section 3.3.3. MCL-adm provides functionality specific to teleconferencing and realizes conference control and integration of groupware applications into a DMC system. The services provided by MCL-adm are described in detail in section 3.4. Finally, the MCL-app sublayer offers services specific to certain classes of groupware applications. An overview of candidate MCL-app services — common and generic ones — has been given in this section. Like groupware applications, it may heavily depend on the particular usage scenario and environment of the DMC system which types of MCL-app services are required. Due to the manifold conceivable application services, their respective complexity, and the resulting potentially huge amount of MCL-app services, a detailed specification of even a single MCL-app service is considered beyond the scope of this thesis.

[39] Depending on the particular implementation, this functionality may of course be distributed across several applications that coordinate their actions at the control service interface in order not to produce contradicting or inconsistent service requests. From a conceptual point of view, however, these are treated as a single application.

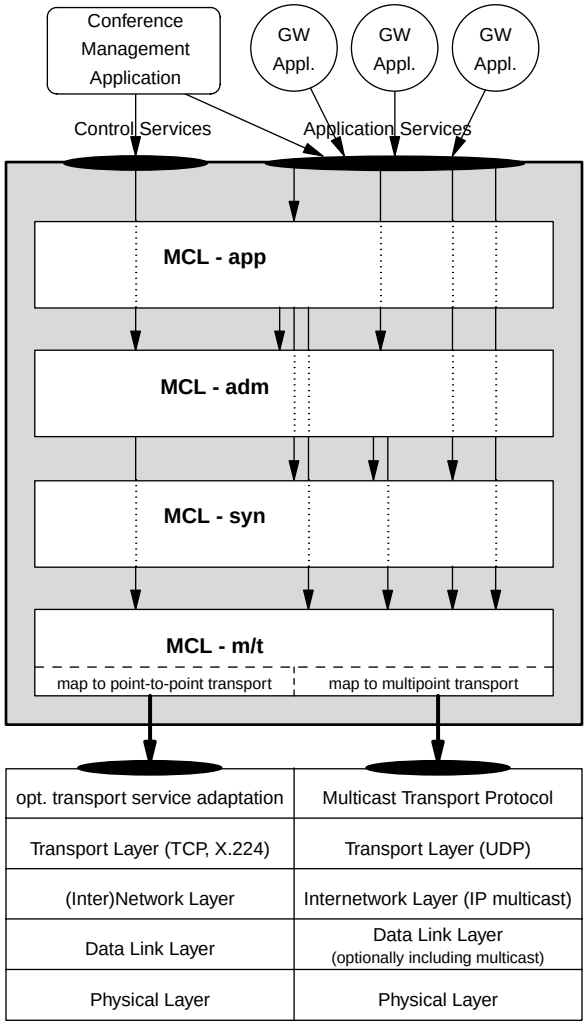


Figure 3.5: Summary of the MCL

3.3. Conferencing-Independent Communication Services

The Multipoint Communication Layer intends to provide a framework for a teleconferencing infrastructure. In teleconferences, various styles of interactions between DMC system components take place and a variety of information types are dealt with. Therefore, at its lower two sublayers, MCL-m/t and MCL-syn, the Multipoint Communication Layer offers generic communication services for information exchange in groups of applications, i. e. services that are independent of teleconferencing.

In the following, important services offered for transmission of information as well as for achieving synchronization in communication relationships are identified and their characteristics are described. It is defined which of these services are to be provided by the MCL infrastructure. The discussion of these issues is split into three parts:

1) the transport services — provided by the MCL-m/t sublayer;
2) the synchronization mechanisms — offered by the MCL-syn sublayer; and
3) security considerations — identifying security attributes to be considered at the lower two sublayers of the infrastructure.

In each of the following subsections, at first, an overview of the most important attributes used to characterize the MCL-m/t and MCL-syn services is given. For each attribute, a broad spectrum of *possible* service characteristics is described based upon which, in the end of each subsection, the *target* service characteristics to be supported by the respective MCL sublayer are defined.

3.3.1. Transport Services

This subsection addresses the characteristics of the transport service provided by MCL-m/t. The first item discusses the number of communicating entities and their interrelationships. Following this, the remaining items provide a broad overview of the properties according to which information transmission characteristics (*quality of service, QoS*)[40] at and above the transport layer between two or more communicating entities are defined.

- For the *number and relations of communicating entities*, point-to-point (two entities) and multipoint (more than two entities) communication can be distinguished.

  Within a group of communicating entities, any *number of senders* may be distributing information to (parts of) the group. Each piece of information may be destined at any *number of recipients*. Basically, three modes are defined: unicast (a single recipient), multicast (a well-defined subset of all possible recipients), and broadcast (all possible recipients).

  Communication between these entities may be unidirectional or bidirectional, i. e. an entity either acts exclusively as a sender or exclusively as a receiver (one-way communication); or an entity performs sending and receiving functions. In the latter case, sending and receiving may be alternating (two-way alternate communication) or in parallel (two-way simultaneous communication).

  The above classification scheme covers all types of interactions in groups of communicating entities: one-to-one, one-to-many, one-to-all, many-to-many, and many-to-one.

In a teleconference, conference control and a variety of groupware application entities are likely to be active at the same time. Obviously, entities implementing different application protocols do not communicate with one another, but need to interact with different sets of peer entities (each set of which forms a subset of the total number of groupware application entities in the conference). To accommodate this setting, the infrastructure has to provide multiplexing functionality and has to offer addressing schemes that allow to communicate with individual entities as well as with subgroups or all entities within the scope of a teleconference. Consequently, all interaction styles discussed above — one-to-one, one-to-many, one-to-all, many-to-many, many-to-one — are required.

[40] These groups and their attributes are to a large degree derived from commonly accepted taxonomies (sometimes with minor differences in terminology) that have been developed in the research areas of computer communications as well as distributed systems; some of them are defined in international standards. Refer for example to [ITU-T X.214], [ITU-T X.224], [ITU-T G.701], [Moulton 94], and [Mauthe *et al.* 95].

The following items address the most important transport service characteristics related to the issue of QoS:

- *Packetization of information* concerns two levels: the service interface that defines how information is received from and delivered to the service user and the protocol that defines the units in which the information is transported between the peer entities.

  At both levels, information may either be packetized, i. e. information units of a defined length (termed packets, protocol or service data units, cells, frames, etc.) are transmitted and are identifiable as such at the recipient. Or information is transmitted as a continuous stream of bytes or bits with no other boundaries preserved at the protocol or service interface. Arbitrary combinations of both methods at the protocol and the service level are possible.

- *Error control* consists of two parts: error detection and error correction.

  Error detection has to deal with two types of errors: false and missing (or duplicate) pieces of information. To detect false information, redundancy is transmitted along with a piece of information and checked at the receiving side. Missing pieces of information or duplicates are detected by numbering all the transmitted units.

  Once an error is detected (at the receiving side), corrective action — if desired — may be taken in two ways: either by retransmitting the false or missing piece of information until it has been received correctly or by supplying sufficiently redundant information that enables the receiver to reconstruct the correct piece of information (forward error correction).

  Depending on the application requirements and the semantics of the information transmitted, three types of error control are conceivable:

  – no error control at all — if false or missing information does not harm the receiving application —,
  – error detection but no correction — if information processed by the receiving entity have to be correct but pieces of information may be missed without affecting the application —, and
  – error detection and correction — if the receiving application needs to rely on receiving all the information as well as the information being correct.

  The last type that prevents bit errors as well as information losses (and duplication) occurring during transmission from becoming visible to the recipient is also termed *reliable* communication.[41]

- *Flow Control* provides protection against a receiver being overrun by the sender(s) transmitting more information in a period of time (transmission rate) than the receiver can handle.

  Flow control may be done entirely at the discretion of the source (*source-controlled*) or may use or rely on control feedback from the receiver(s) (*receiver-controlled*).

  Source-controlled flow control may use rate limiting, i. e. shape the information traffic emitted in a way that in the average a certain threshold is not exceeded. This approach is referred to as *rate control*.

[41] Note, however, that absolute reliability cannot be guaranteed: in error detection, bit errors may not be detected by the algorithms in use and when employing forward error correction, the redundancy provided may not be sufficient to correct certain transmission errors.

Receiver-controlled flow control uses backpressure schemes in which the receiver either explicitly asks the source to stop transmitting or implicitly does so by not acknowledging the incoming information.

- For information exchange in groups different schemes for *ordering* have been defined. Information distributed in a group by one or more senders may be delivered to each recipient in a non-determined order (*unordered*), or some common ordering properties for all recipients may be provided. Ordering of transmitted information may be done with respect to a single source as well as multiple sources.

  The simplest case of ordered delivery is *(single) source ordering*: for all information units from a single source the sequence in which the source originated them is preserved at all receivers (e. g. by means of a sequence number).

  In contrast, provision of *global* or *total ordering* [Lamport 78] [Birman / Joseph 87] means that all the information transmitted by any one of multiple senders in a group is delivered to all the receivers in the same sequence.

  Causal ordering [Birman *et al.* 91] is used in group communications in distributed systems. This scheme guarantees that messages are delivered to the recipients (which may be a subgroup) after messages that causally precede them (e. g. caused them to be sent).

  Ordering based on absolute time or logical clocks [Lamport 78] provides essentially the same service like global ordering except that the order of delivery is determined by the time at which a message has been sent rather than being chosen ‘‘at random’’ by some ordering entity.

- *Time-related attributes* describe the relationships between transmission and reception time of information as well as the relative timing between information units.

  Real-time information has a defined upper bound for the delivery to the recipient (a maximum transmission latency); if this bound is not met, the information becomes useless to the recipient and the application relying on in-time delivery is affected.

  In contrast to real-time information, *time-critical* information does not have a strict upper bound for delivery to the recipient but shall nevertheless be delivered quickly to the recipient. If information is delayed during transmission, the functionality of an application is not seriously affected, but the (perceived) service quality of the application is degraded.

  Timing-independent information does not have such requirements (is non-real-time and non-time-critical) and is useful for the receiver regardless of when the information is received.

  Information streams have either an *internal time base* or are made up of with respect to timing unrelated *discrete* units. Time-based information streams are used to represent continuous media (e. g. audio or video) that due to their nature do require that the intra-timing between subsequent information units is preserved across the transmission for correct reproduction at the recipient. In contrast, information streams consisting of discrete units (e. g. editing commands) do not have an internal time base and therefore no intra-timing is required.

The MCL infrastructure solely aims at groupware applications that do require reliable exchange of information and, by definition, is not to include functionality to support any time-related attributes. Hence, error control and flow control need to be provided while timing-related attributes need not. In addition, all types of ordering are of potential value for groupware

applications, so that unordered message transmission is supported as are per-source ordering, causal ordering, and global ordering. Information exchange is done packet-oriented (transmission of *messages* of virtually arbitrary size is offered at the service interface) because this simplifies multiplexing and allows easy subaddressing within a teleconference without the need for a prior connection setup. Table 3.3 summarizes the transmission properties supported by the MCL.

Transport Service Property	Provided by the MCL
number and relations of communicating entities	one-to-one, one-to-may, one-to-all, many-to-many, many-to-one
packetization of information	packet-based information exchange or virtually arbitrary size
error control	reliable communication (error detection plus retransmission)
flow control	combination of rate and backpressure flow control
ordering	no, per-source, causal, and global ordering
timing	not provided

Table 3.3: Information transmission properties supported by the MCL infrastructure

3.3.2. Synchronization Services

Synchronization mechanisms have their origin in the area of operating systems where concurrent access to common system resources has to be coordinated. In distributed systems (e. g. distributed operating or database systems) synchronization mechanisms are again applied to control access to shared resources but also to consistently distribute shared state information and maintain this consistency when modifications occur. In the area of telecommunications, synchronization also refers to structuring the information exchange during a communication relationship with respect to time and/or (application) semantics. The following synchronization services are considered for inclusion in the Multipoint Communication Layer:

- *Inter-stream synchronization* relates pieces of information being sent independently (usually by the same source) and belonging to the different information streams to one another. In particular, processing of information from different streams may be aligned. In the context of real-time streams, inter-stream synchronization is used to preserve timing-relationships between two streams (e. g. between audio and video from a single sender to achieve lip synchronization).

- *Tokens* [ITU-T X.215] [ITU-T T.122] serve similar purposes as semaphores and locks in that they provide a mechanism to synchronize the actions of different communicating entities with respect to common state, e. g. to arbitrate access to shared resources. Tokens can be used to provide mutual-exclusion for access to resources (e. g. to construct write locks) as well as for concurrent resource utilization (read locks) [Herrtwich / Hommel 89].

- *Checkpointing* [ITU-T X.215] [ITU-T X.851] means explicitly establishing (and saving) common distinct state — the checkpoint — after transmission and/or reception of a particular piece of information at some or all entities in a group. The aim of checkpointing is to confirm (at the application level) the actions that have been taken prior to setting the checkpoint and to

be able to reset the shared state to one of the previously established checkpoints (fallback), e. g. if failures occur later in the communication relationship.

- *Transactions* [ITU-T X.215] [ITU-T X.851] [ITU-T X.860] are another means for synchronizing state information at some or all entities in a group communication relationship. Transaction concepts combine token and checkpointing to provide synchronization mechanisms for complex operations in distributed applications. For such operations, they coordinate concurrent access to shared resources, such as shared state information, and perform internal checkpointing so that operations that could not be completed successfully by all involved entities may be ‘‘undone’’ (rollback). Transaction concepts allow more complex synchronization schemes to be implemented but they require knowledge of the application semantics (e. g. how to detect and resolve conflicts in concurrent operations) [Jalote 94].

Inter-stream synchronization mechanisms for real-time information streams do not need to be provided by the MCL since real-time information itself is not supported. For data communication between groupware applications, such a mechanism is provided to allow different information streams exchanged between the same set of groupware application entities to be synchronized when needed (e. g. high priority control and low priority data messages).

Tokens are to be included in the MCL functionality in order to support groupware applications as well as MCL-internal management. Examples for the usage of tokens are representation of the conference floor(s) and conference conductorship as well as coordination of updates to shared databases, drawing surfaces, etc.

From a conceptual point of view, checkpointing and transactions both are highly intertwined with the respective service users. As in the session layer of the OSI model, means for signaling checkpoint-related actions to the peer(s) are provided by the infrastructure (including consistency checks for services invoked). The actual checkpoint processing (e. g. saving or restoring state information) has to be carried out by the service users (eventually the groupware applications) as only they have the required knowledge about the semantics of each operation. Transaction concepts are not supported at all, because transaction processing depends even more on the application semantics. However, with token and checkpointing mechanisms, the building blocks for implementing transactions at the application level are provided. The services to be offered by the MCL are summarized in table 3.4.

Synchronization mechanisms	**Provided by the MCL**
inter-stream synchronization	provided for non real-time information
tokens	exclusive and shared
checkpointing	means for signaling checkpoints and rollback provided
transaction concepts	not provided

Table 3.4: Synchronization mechanisms supported by the MCL infrastructure

3.3.3. Security Services

Section 3.2.3 has already introduced the subject of security including the four commonly accepted security attributes, and has roughly outlined where within a teleconferencing system security mechanisms may be applied. This section specifies which security mechanisms are included in the MCL-m/t sublayer in order to support which of the aforementioned security attributes:

- *confidentiality*,
- *authentication*,
- *integrity*, and
- *non-repudiation*.

Confidential information exchange is crucial to the acceptance of teleconferencing. Therefore, encryption is an essential function of a DMC system and needs to be provided by the communication infrastructure. For dynamically selecting and changing the encryption key(s), key exchange mechanisms have to be provided as well. As the infrastructure supports multiplexing, independent encryption keys and algorithms need to be supported for different information streams and different application sessions.

Authentication may be done at three levels: 1) initially upon entry of a new participant into a conference (possibly including repeated subsequent verifications), 2) for each groupware application entity (initially when joining an application session and repeatedly to re-confirm its identity), and 3) for each application message being transmitted. Items 1) and 2) are an issue of conference control (and are potentially relevant to key distribution) and are therefore addressed in the next section. Per-message authentication (item 3) as well as per-message integrity may be performed at the MCL-m/t level, but groupware applications may perform these functions more efficiently because they are able to decide which messages need to be authenticated or checked for integrity and which need not. Therefore, and because of processing overhead and additionally introduced latency per-message authentication and integrity validation is not provided by the MCL.[42]

Non-repudiation is not relevant to synchronous transmission services offered by a communication infrastructure but rather of importance to application layer services such as document or file distribution (e. g. representing meeting minutes or business letters). Therefore, non-repudiation mechanisms are not included in the MCL infrastructure.

Altogether, MCL-m/t provides hooks to achieve confidentiality by integrating hooks for encryption and providing means to signal key changes synchronized with the flow of (encrypted) information. Authentication and key distribution mechanisms are not addressed within MCL-m/t (but are left for MCL-adm). Table 3.5 summarizes the above discussion.

[42] Initial authentication of users and applications together with encryption provide a reasonable amount of message integrity as well: an intruder cannot simply introduce or modify messages because these messages would afterwards not be deciphered correctly by the recipient(s) provided that the encryption key has not been compromised. However, this is insufficient to detect modifications caused by conference participants who possess the respective encryption key.

Security attributes	Provided by the MCL-m/t
confidentiality	hooks provided for encryption synchronized signaling of key changes provided
authentication	not provided (refer to MCL-adm)
integrity	not provided
non-repudiation	not applicable to synchronous communication

Table 3.5: Security mechanisms included in the MCL-m/t sublayer

3.3.4. MCL-m/t and MCL-syn Service Summary

The MCL infrastructure is designed for multipoint teleconferencing systems in which arbitrary numbers of interactive groupware applications (operated by humans) participate. Therefore, the MCL has to offer flexible many-to-many communications within a teleconference for an arbitrary number of concurrently running applications. The MCL does not support transmission of real-time, time-critical or time-based (continuous) information, i. e. audio-visual information cannot be conveyed through the infrastructure. Rather, the focus is on information that requires reliable and flow-controlled transmission. In addition, MCL provides two types of ordering, includes basic means for supporting synchronization, and offers hooks for integrating security functionality to ensure confidentiality of the exchanged information.

3.4. Conference Administration Services

This section deals with the functional range of the conference administration sublayer — also referred to as conference management. The conference administration sublayer makes use of the transport and synchronization services of the lower two MCL sublayers to communicate with its peers. The conference management functionality is subdivided into two complementary groups:

- *Conference control services* — services that are used to affect the course of the conference and that are intended to be invoked by the human user. Therefore, they are also referred to as ‘‘user-visible’’ conference management services. This group essentially includes all the conference control functionality described in chapter two.
- *Internal management services* — services that are used to integrate groupware applications into the teleconference and to establish application sessions of interoperable peer application entities within a teleconference. They are termed ‘‘internal management’’ services because the human user is unaware of them.

Figure 3.6 depicts a system model for realizing conference management functionality:[43] The conference management service is provided by a conference management entity. The conference management entity and the application entities both make use of the transport services for inter-system communication with their respective peer entities. In addition, local communication channels exist through which the application entities access the administration services of the conference management entity. In the figure, the conference control services are accessed by a user interface entity that presents them adequately to the human user.

[43] This model is used as a basis in all the discussion throughout this section.

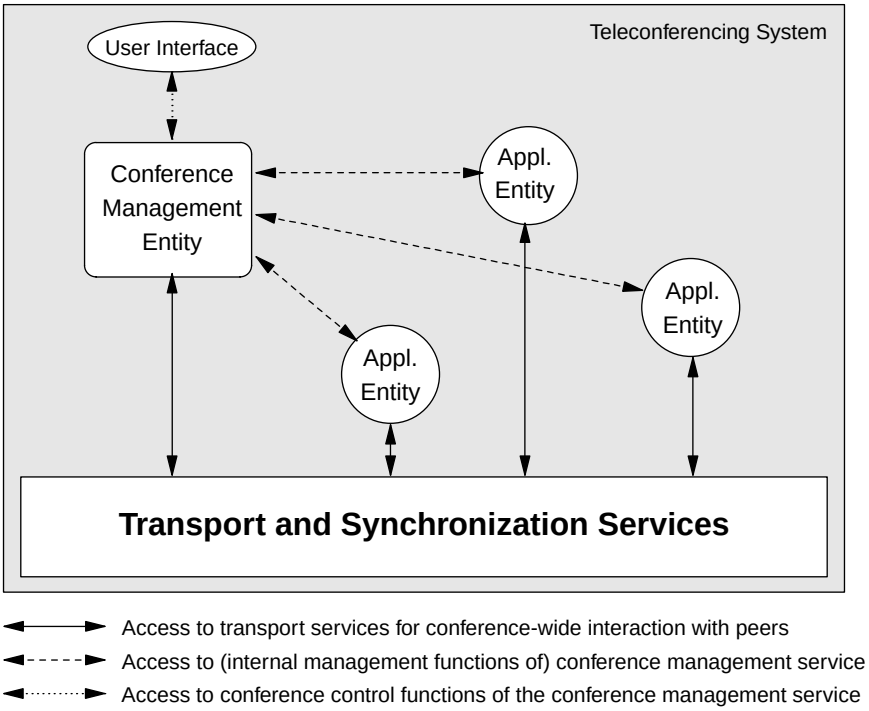


Figure 3.6: System model for conference management services

To complete the conference management model, figure 3.7 extends the system model to show the interactions between two teleconferencing systems in a point-to-point conference and the various protocols in use:[44]

- Inter-system communication occurs between peer entities running on different system: the conference management entities communicate with one another using a conference management protocol. Also, a set of application entities exchange information within application sessions. These application sessions are isolated from one another with (different) application-specific protocols being employed in each session.
- Intra-system communication is used to coordinate the otherwise unrelated local groupware applications on each teleconferencing system and integrate them with the conference management entity as well as to provide access to the conference control services.

Derived from the appearance in this figure, inter-system communication protocols are termed *horizontal protocols*[45] and protocols for local interactions are called *vertical protocols* [Schooler 91] [Ott *et al.* 94].[46]

[44] A scenario of only two systems is used for the sake of clarity of the presentation in the figure. The number of sites is conceptually irrelevant to the model presented here. For the same reason, the transport services used for inter-system communication is not shown in the figure.

[45] Basically all protocols specifying interactions and encodings ‘‘on the wire’’ — virtually all protocols defined by international standardization bodies — belong to this category.

[46] Vertical protocols typically make use of local IPC mechanisms and may be hidden from groupware applications underneath well-defined APIs.

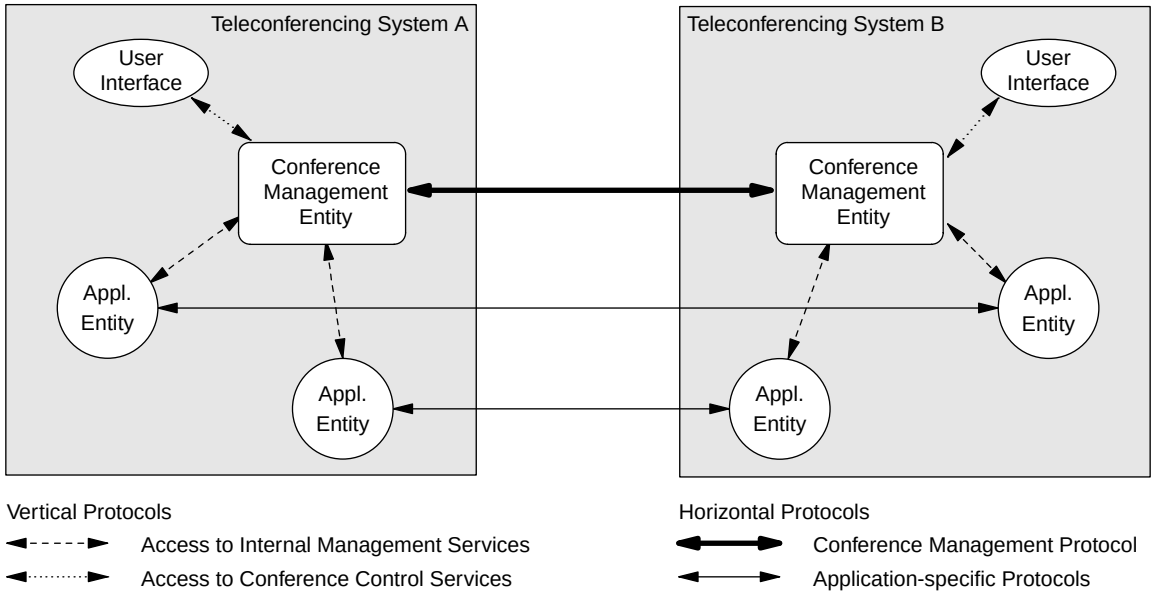


Figure 3.7: Protocols for conference management services

In summary, conceptually the conference management entity offers two different service access points: one for establishing and running the conference and one for attaching groupware applications to make them become part of a teleconference. The respective services are locally accessed via vertical protocols. A horizontal conference management protocol conveys all the necessary information between conference management entities to provide these services. These two categories of conference management services are described in the following two sections.

3.4.1. User Visible (Conference Control) Services

The user-visible conference control functions have already been discussed extensively in chapter two. Therefore, in essence, this subsection recaps the requirements described in section 2.2 and defines the subset of the functionality presented there that is to be supported by the MCL.

In section 2.3.1, conference control services have been subdivided into four functional groups: conference configuration, participation management, floor control, and security. For each of these groups the respective services are briefly listed in the following. Table 3.6 summarizes all the (user visible) conference control services provided by the conference administration sublayer of the MCL architecture.

- *Conference configuration*

  Conference configuration essentially refers to defining a conference profile. Means for specifying and enforcing different conference policies are provided. Conferees may be assigned roles that are associated with permissions to carry out certain actions in a conference. The number of roles and permissions as well as their interpretation is beyond the scope of the conference control functionality being part of the MCL infrastructure.

  It has already been pointed out repeatedly, that functionality that is of not purely synchronous nature is not included in the MCL. With respect to conference configuration, organisatorial

functions such as reservation, accounting, and related ones are therefore not specified here. However, at the end of this subsection, an outline is given how to implement such external functions based upon the MCL services.

- *Participation management*

  First of all, participation management comprises services for setting up conferences, point-to-point ‘‘calls’’ as well as group sessions as a prerequisite for any participation. A service for terminating a conference is provided as well.

  Furthermore, functions for changing the membership of a conference are included. Participants may join and leave a conference on their own. They may be invited to or excluded from a conference by another participant. Changing from one conference to another is modeled by leaving one and then joining the other conference.

- *Floor control*

  Floor control offers mechanisms for requesting the floor, for accepting or denying the request, for passing it on to a designated successor as well as for releasing the floor. The floor may be assigned exclusively to a single participant or shared by several ones.

  Any number of (application-specific) floors are supported with one being dedicated to be the ‘‘global conference floor’’. Neither the meaning of a floor nor the assignment policies applied to a floor are restricted by the conference control.

- *Security functions*

  Security functions at the conference control level are restricted to participant authentication and (the initiation of) changing the encryption keys of the conference.

  Authentication is performed when a participant enters the conference and may be repeated arbitrarily during the conference course. Authentication may be done based on a shared secret (i. e. a password) or based upon personal authentication (i. e. confirming the identity of a participant).

  Key changes require identifying which key(s) to change and distributing the new key(s) to the intended participants. Furthermore, at each participant’s site, the key(s) have to be forwarded locally to the affected application(s).[47] Finally, switching to use the new key(s) has to be initiated and then synchronized among all application entities on a per application session basis.

As will become clear from the next subsection, all of these services interact with internal management services, too: in order to become an integrated part of a teleconference, a groupware application entity needs to be informed about (changes to) the conference state potentially affecting the application and may even have to be able to trigger changes to the conference state.

Interaction with Conference Reservation and Announcement Services

Reservation and announcement services have explicitly been excluded from the functionality provided by the conference control services of the MCL. The following considerations show that the MCL conference control services are orthogonal to the aspects of conference reservation or

[47] This is, in fact, an internal management function, but it is mentioned here for completeness.

Functional Group	Conference Control Service Semantics	
Conference configuration	PROFILE DEFINITION	define permissible participants available floors and their characteristics available roles and associated permissions required permissions to invoke services assign roles to participants ...
	ROLE ASSIGNMENT	assign role initially assign role by virtue of another role request role pass on / share role deny role request give up role
Participation management	SETUP	initiate a conference place a point-to-point call accept a point-to-point call
	TERMINATION	terminate a conference terminate a point-to-point call reject a point-to-point call
	JOIN	join a conference rejoin conference after temporary absence join conference as part of changing a conference
	LEAVE	leave conference leave conference as part of changing a conference
	INVITE	invite conferee as part of conference setup invite conferee into an ongoing conference
	EXCLUDE	exclude conferee as part of conference termination exclude conferee from an ongoing conference
Floor control	FLOOR ASSIGNMENT	assign floor initially assign floor by virtue of a certain role request a floor pass on / share a floor deny floor request give up a floor
Security	AUTHENTICATION ENCRYPTION CONTROL	authentication upon joining repeated authentication during a conference distribution of a session key switching to new session key

Table 3.6: Conference control services

announcements and that the MCL functionality is sufficiently complete to enable implementation of those services on top.

Figure 3.8 depicts a model of a conferencing system that interfaces to a conference reservation system and may act as a conference server (e. g. an MCU). The user interface directly accessing the conference control services is substituted by a reservation control entity. The control entity obtains information about scheduled conferences and their profiles from a data base. The data

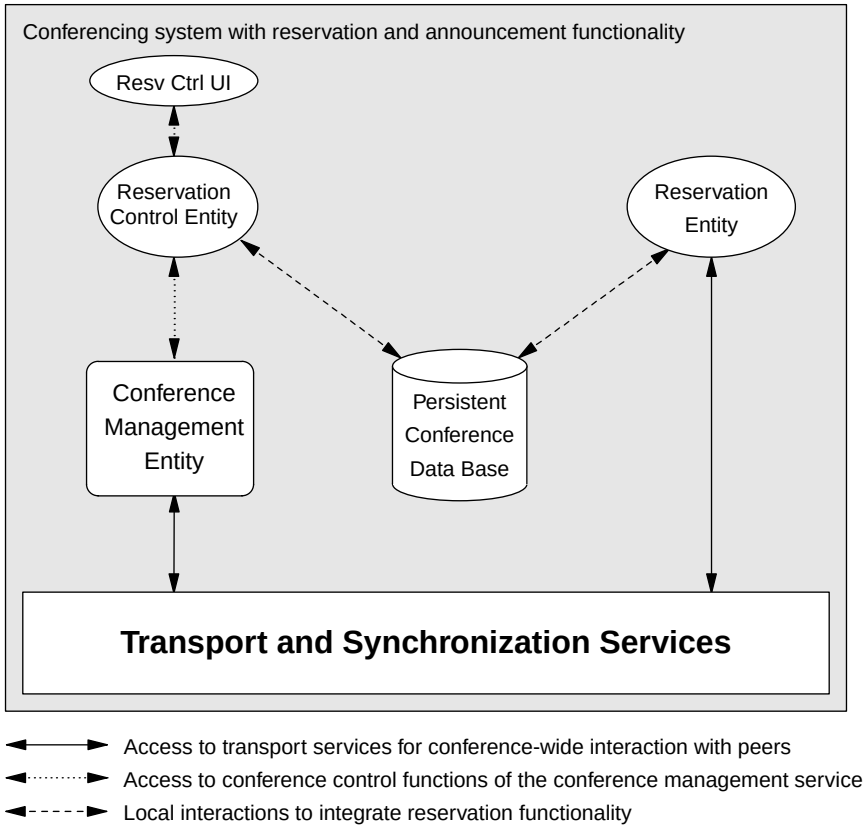


Figure 3.8: Interaction of conference control with a reservation and announcement system

base itself is maintained by a reservation system that may be accessed and modified e. g. via a specific reservation protocol, via HTTP, e-mail, or may be updated by means of an announcement protocol. The control entity may also be accessed by an operator through a dedicated reservation control user interface.

The information retrieved from the data base enables the control entity to set up and terminate conferences at the scheduled times (if desired), admit permissible participants, and assign roles and privileges according to the respective profile. However, all these actions may be performed using the aforementioned conference control services (table 3.6); there is no need for additional reservation services in the infrastructure itself. In a similar fashion, the control entity may export e. g. accounting information and store it in the conference data base. This may be complemented by integrating network management functionality into various components of the infrastructure in order to obtain more detailed knowledge about the current state of the infrastructure at any point in time.

3.4.2. Internal Management Services

The user-visible conference management services described in the previous subsection provide functionality to control the course of a teleconference and to reflect the participants' behavior. For actual interaction among the conferees, however, groupware applications are included as part of a teleconferencing system. These groupware applications are considered independent entities

rather than merged with the conference control entity. They make use of the same transport services but are otherwise unrelated to the conference management and consequently do not have any knowledge about ongoing conferences and their respective state information.

> Currently deployed Internet conferencing applications — audio and video communication tools as well as a whiteboard and a shared text editor — do include rudimentary conference control functionality for loosely coupled conferences (and individually keep inexact state information about a conference). On one hand, this enables these applications to operate *stand-alone*, i. e. without a conference management entity. On the other hand, there is almost no cross application interaction and consequently no consistent behavior of the applications in response to actions taken by the human user. In fact, the individual applications are part of independent conferences that happen to involve the same users.

It is the task of the internal management services to integrate the conference management entity and the groupware application entities in order to make them appear as a coherent teleconferencing system [Schooler 91] [Handley / Wakeman 94] [Ott *et al.* 94]. In order to accomplish this integration, mechanisms to achieve the following two properties have to be provided:

- *Interoperability* between the groupware application entities of different teleconferencing systems regardless of the supplier of a particular application (provided that the same classes of groupware applications are available on the involved systems at all).
- *Consistency* in the behavior across all groupware applications involved in a teleconference with respect to the actions taken by a user for that particular teleconference through the conference management entity (or even through any of the applications).

Interoperability

Interoperability means that, if two or more teleconferencing systems are interconnected, the participants have to be able to collaborate using all the media and tools available to them for information exchange. Achieving this without requiring explicit user intervention is one of the key aspects to teleconferencing technology being perceived as usable by the human users.

> A well-known example for perfect interoperability is the telephone system. Virtually any phone in the world can be called from any other phone, and the persons at both phones will be able to communicate. The quality may vary and different phones and network providers will offer different additional features, but this does not affect the basic conversation service.

A DMC system offers functionality for audio communication, video communication, and various meeting aids. These are referred to as *user services* for the remainder of this section. Each DMC system implements a set of user services by means of groupware applications. Two DMC systems potentially use different groupware applications from different vendors to implement the same user service. Each user service may be ‘‘instantiated’’ more than once in a conference in conjunction with different tasks (called *application contexts*). The following example illustrates this scenario.

> Two users at two sites converse via audio communications and in parallel edit two different text files. This scenario is depicted in figure 3.9 with different groupware application classes (audio vs. text editor) being indicated by geometric shape, an application session is depicted as a pair of ‘‘white’’ groupware application entities that are interconnected through solid lines.

User A uses the *vat* audioconferencing tool with G.722 audio encoding used with RTP/RTCP, UDP, and IP as transport mechanism, user B uses a different audioconferencing application called *rat*. They both make use of the *nt* text editor using the *nt*-specific protocol on top of UDP and IP. A and B have other groupware applications available, a video tool (*vic*) and a whiteboard (*wb*), respectively. While only a single audio application session does exist, one editor session is used to make notes about the conversation and the other is used to draft a common statement on the usability of teleconferencing.

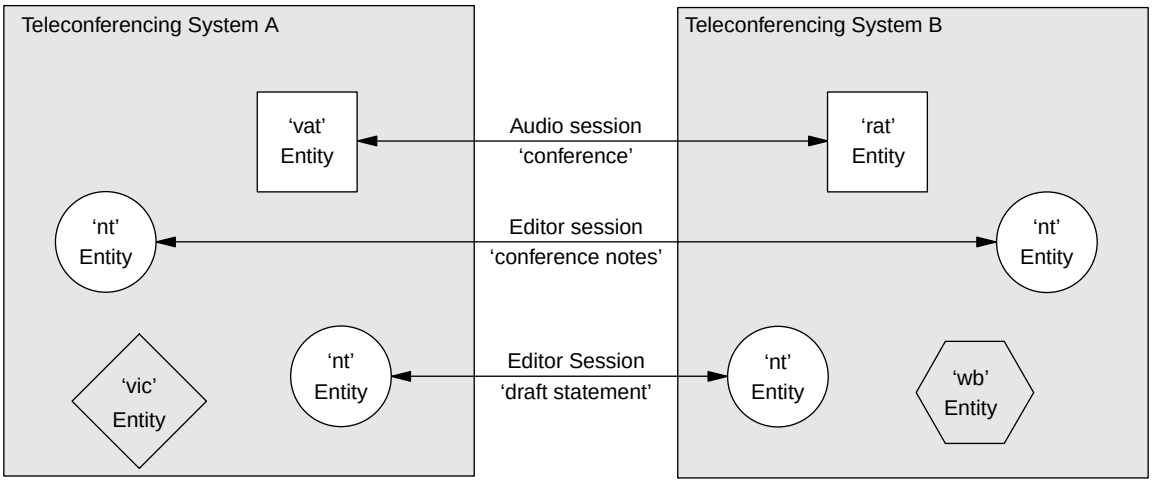


Figure 3.9: Sample interoperability scenario

In order to achieve a minimum of interoperability between each two DMC systems, two prerequisites must be fulfilled. First, at least one (meaningful) application session has to be established. This requires that both systems have at least one available groupware application class in common.[48] Second, interoperation within each application session has to be achieved. This is only possible if the peer entities share at least one protocol for information exchange. This is only achievable if the protocols for the various application categories are standardized.[49] A minimum set of required functionality — a so-called *baseline* — has to be defined (again through a standard) to guarantee that a common denominator can always be found, with respect to the available groupware application classes as well as with respect to the protocols used per application class.

Assuming that some basic level of interoperability is guaranteed, several issues have to be addressed to eventually establish the communication relationship of the above scenario (refer to [Ott *et al.* 94] for an extensive discussion of these interoperability issues).[50]

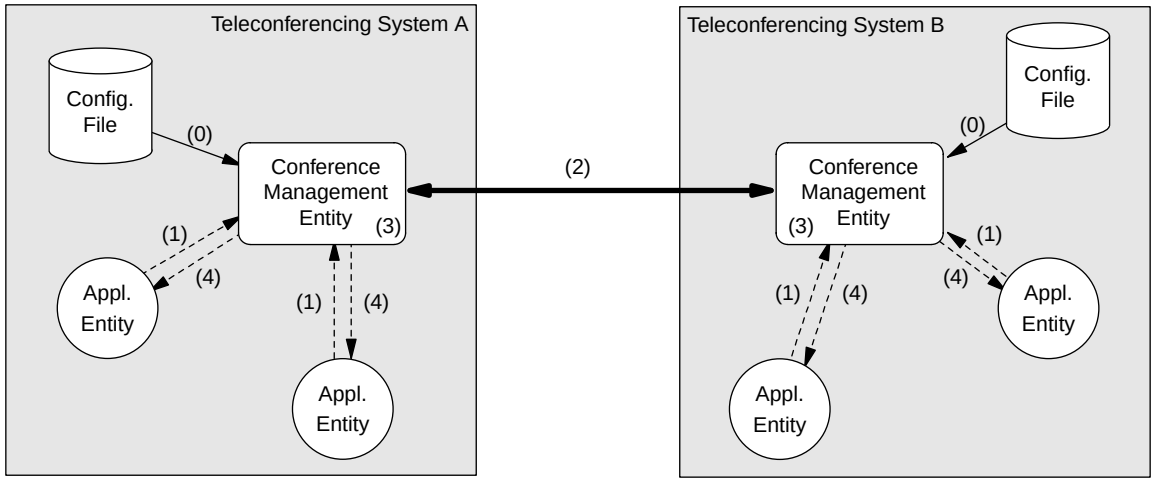
[48] From a pragmatic point of view, this ''least common groupware application category'' should provide audio communication facilities.

[49] Standardization may be done by international standardization bodies, by industry consortia, or by market forces setting standards.

[50] Alternatives are to automatically start up the necessary groupware application entities initially following the conference profile and to invoke the application entities manually when they are needed. Automatic initiation of application entities at conference startup requires a priori knowledge about what is needed in the conference and does not allow for subsequent changes (except if done manually). Manual invocation of applications is, however, in neither case user-friendly. In both cases, it is assumed that the

- Each involved DMC system needs to determine for which application classes groupware applications are available at the other site(s).
- For each application class available at some or all of the involved systems, the protocols (including transport protocol stacks) supported by the respective groupware applications have to be determined. Along with each protocol, knowledge about protocol-specific parameters may be required, too.

The set of information describing the characteristics of a DMC system that are relevant to interoperability is termed *capability set* or simply *capabilities*. One of the tasks of each conference management entity is to obtain information about its local DMC system's capabilities. This is done either by applications registering dynamically with the conference management entity (through a vertical protocol) or by static configuration (e. g. by means of an initialization file).



0. Initialization with DMC system and groupware application capabilities through a configuration file
1. Registration of groupware application entities and their capabilities
2. Exchange / update of registered groupware applications and their capabilities
3. Computation of common capabilities (internally within the conference management entity)
4. Dissemination of capability information (common as well as individual) to all application entities

Figure 3.10: Capability exchange procedure

From the capabilities of all DMC systems involved in a teleconference is derived which application sessions can be set up and which systems may be involved in a specific application session that makes use of certain protocols and protocol parameters. The procedure of mutually announcing the capabilities in a teleconference is called *capability exchange*. The capability exchange is carried out by the conference management entities using their horizontal protocol. The results of the exchange process can be obtained by each application entity from its local conference management entity through the vertical protocol. The entire process is depicted in figure 3.10.

After the conference management entities on the involved DMC systems know about all the systems' capabilities, application session among interoperable groupware application entities may be

appropriate groupware applications are available at the other site(s).

set up. An application entity actually creates an application session by registering the session with the local conference management entity. The latter one ensures that such a session does not yet exist in the conference. Furthermore, the application claims and registers all the resources required to run the application session.

Once the session registration is completed, the application entity may either actively invite one or more peer entities (this is roughly comparable to a traditional connection setup) or it may passively wait for other application entities to join the application session. If, at the time of invitation, no entity of the addressed groupware application is running at a site that is being invited into the session, such an entity can be instantiated dynamically (*application invocation*).[51]

Regardless of the particular way used to set up and populate an application session, the following services need to be provided:

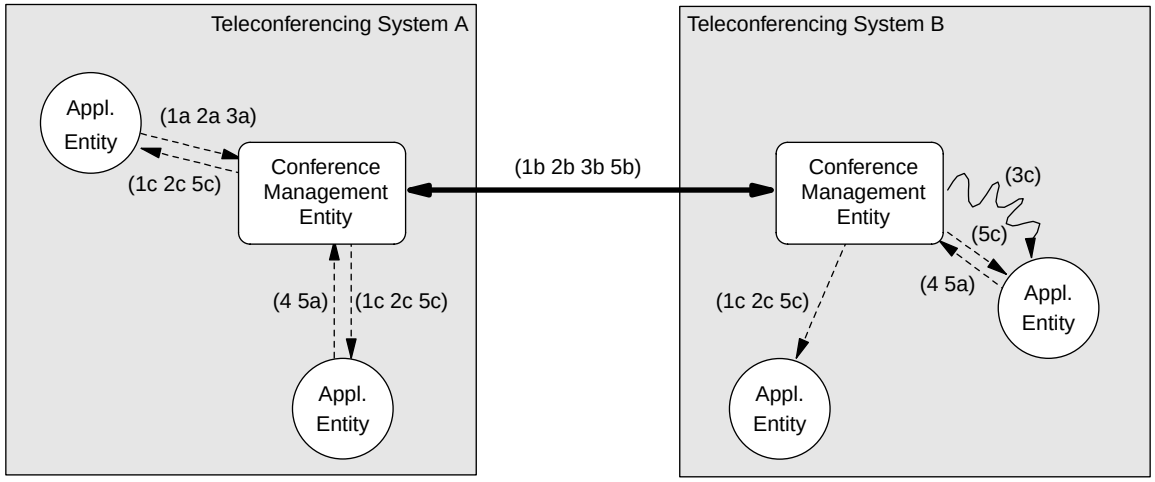
- An application session needs to be identifiable and addressable by a groupware application entity, i. e. an application entity must be able to select the correct application session to join.
  - When creating a session, the protocol(s) and the protocol parameters actually in use for that session — the (*active capabilities*) — are initially selected (and subsequently modified, if necessary) considering the capabilities of those applications that are intended to participate in the session. When a particular session is already active, a newcoming application entity is able to decide whether it is able to interoperate with the other entities in that session based on the session's active capabilities. Also, the entities already involved in a session can compute whether or not a potential newcomer would degrade the functionality of their session — which may influence the decision to accept or reject the newcomer for the session.
  - A set of attributes is required to specify what the session is actually used for (e. g. to distinguish the statement editing session from the conference notes editing session in figure 3.9). These attributes are complemented by a session identifier that uniquely distinguishes an application session in a teleconference and that may be passed around as a handle (e. g. in an invitation to an application session). With the unique identifier and/or the session description, (the user of) a newcoming application entity is able to choose the right session if multiple ones using the same protocol are going on in parallel. The entire description of a session is referred to as the *application session context*.
- Applications and distinct application entities need to be identifiable, too, and they need to be addressable: during setup of a session, a groupware application may need to be invoked on one or more DMC system(s), which requires the invoking entity to identify the target application to be instantiated. During the course of an application session, the application entities need to identify their peers and obtain information about them: examples are their respective transport addresses to unicast information to each of them, the capabilities of a particular entity (in contrast to the capabilities of the application class on that DMC system), vendor information, etc.
- Furthermore, an identification mechanism is needed for all kinds of resources belonging to an application session. These resources include, for example, transport addresses used for

[51] Note that this discussion intentionally disregards any issues of conference policies allowing or disallowing certain actions.

session-wide information exchange, identifiers of tokens for synchronization and checkpoints for recovery, etc.

All three of these functions require a registration service that allows to associate type and value of a piece of information with a (structured) identifier. A lookup mechanism has to be included to map identifier (and type) to the value stored in the registry. Finally, means for monitoring the contents of some or all registry entries are needed to inform a groupware application entity about changes to present as well as the creation of new entries that are relevant to the application session(s) this particular entity participates in.

The registry functionality allows application entities that are to collaborate to find one another, to establish a common application session, and to make use of the same resources. Also, using a registry prevents application entities intended to be independent of one another from accidentally using the same resources for different purposes and thereby creating conflicts. Naming conventions are needed for the registry to ensure that groupware applications from different vendors use the same systematics to store and lookup information. Otherwise conflicts may arise or the application entities may simple not be able to locate the information of interest about one another. Besides recording information about application sessions, creation and termination of as well as participation in sessions have to be dealt with. Functions to create and terminate, join and leave sessions as well as to invite and exclude other application entities are needed. Also, an invocation service is required in order to (remotely) instantiate groupware applications that are supposed to participate in a particular application session.



1. Session creation (a: initiation, b: update, c: notification)
2. Registration of session parameters, context, resources, etc. (a: initiation, b: update, c: notification)
3. Invocation of remote groupware application (a: initiate, b: forward, c: execute instantiation)
4. Lookup of session details (especially addresses and other resources)
5. Join session (a: initiation, b: update of session information, c: notification)

Figure 3.11: Sample session establishment

Figure 3.11 shows an application entity on DMC system A that creates an application session and remotely invokes the corresponding groupware application on teleconferencing system B. This leads to an application entity being instantiated (e. g. by *fork()* and *execve()*) that first looks up

details about the application session and then joins it. A second application entity of the same class is passively present on systems A and B. On system A, this entity recognizes the registration of the session (e. g. by monitoring the registry), also looks up the information necessary for joining the session, and then joins it. The entity on system B also receives all these notifications but does not take any actions.

Local Consistency

In this thesis, a teleconferencing system is modeled as being composed of many groupware application entities plus a conference control entity. Potentially, each of the application entities has its own user interface and shares common state with its peers about the application session(s) it is involved in. On each DMC system the collection of all these entities represents a human user in a teleconference.

To the user, a coherent view of a DMC system has to be presented regardless of the number of application entities that make up the DMC system. Therefore, all potentially user-visible components of a system — i. e. all groupware application entities and the conference control entity — have to behave in a consistent fashion in response to actions taken by the user. In order to achieve this goal, an important demand is that all these entities may be controlled from a single point with respect to (user) actions concerning his participation in the conference. If a user requests the conference floor, changes or leaves the conference, etc., he has to be able to do so by means of the conference control user interface so that only a single action is required. The user shall not have to perform the same action repeatedly for each active application entity.[52] Rather, the application entities are to be notified about the user action internally. This simplifies the usage of the system and helps to avoid user errors.

Besides coherently reacting on user input, all application entities on a DMC system have to consistently reflect the state of the conference to the user. In order to share the same view of the conference state, they receive their knowledge about the conference as a whole from a common source. If the groupware application entities are not controlled by some dedicated entity that also informs them about the conference and the user's actions, all the entities have to collect and maintain conference-related state information by themselves. This increases the probability of inconsistencies among the state information held by the several application entities which, finally, may introduce the ''Who-is-right?'' problem if several entities with different understandings of the conference status try to synchronize their respective state.[53]

Finally, local coordination of application entities provides a hook for integrating mechanisms into a teleconferencing system that are used by and orthogonal to the individual groupware

[52] A variation of this concept is that the user may initiate any action from any of the application entities provided that they offer a suitable user interface. If the user is e. g. focused on reviewing a text and wants to make a comment (audibly as well as in the shared text editor), it is convenient to be able to invoke the ''conference floor request'' from the text editor's user interface and make the conference management entity carry out the request (rather than having to pick the conference control user interface to do so). In the conference model described in this section, this is easily achievable by allowing (certain) groupware applications to access the conference control services via the respective vertical protocol as well.

[53] Having each groupware application entity maintain its own view of the conference state implies that the necessary functionality for doing so has to be replicated in all groupware applications, thereby making the application functionality and hence the protocols more complex.

applications. That is, present and future functions to be covered by the infrastructure may be implemented by a single entity per system that offers these services through vertical protocols. For example, this technique can be applied to integrate common security mechanisms into a teleconferencing system: a dedicated security entity is conceivable that performs user authentication and key distribution between DMC systems and forwards appropriate keys to the application entities as well as notifications about when those keys are to become effective.[54]

The obvious design choice is that the conference management entity performs the role of the local coordinator and a) forwards user actions to the affected application entities as well as b) keeps the application entities informed about the current conference state. Additional functionality may either be provided by the conference management entity as well if this is feasible — as in the case of authentication and key exchange. Alternatively, a specialized service provider entity may offer services not covered by the conference management.[55]

Application entities may react differently when receiving notifications of any kind — also termed *events* — from their local conference management entity. The reaction depends on the event type and contents, on the conference and application session policies currently in force, and, of course, on the groupware application (entity) itself. An application entity has at least the choice to

- actively take the action requested in the event, if any;
- update information visible to the user as a result of the event;
- recognize the event and update internal state information accordingly for subsequent usage;
- recognize the event but ignore it because the current application session (or local) policy specifies that the particular event be ignored;
- ignore the event because it is unknown to the application entity.

This means that in principle events signaled through vertical protocols are understood as ‘‘hints’’ to the receiving groupware application entity — that are generally followed by the application entity unless the current policies define exceptions or the respective events are not supported by the application entity. However, some events do require the recipient to take a certain action. These *mandatory* events include requests to register and deregister the application capabilities in a conference, to instantiate an application entity in order to participate in an application session, and to terminate an application entity. All other events are *optional* to understand and an application entity’s reaction may be influenced by the policy.

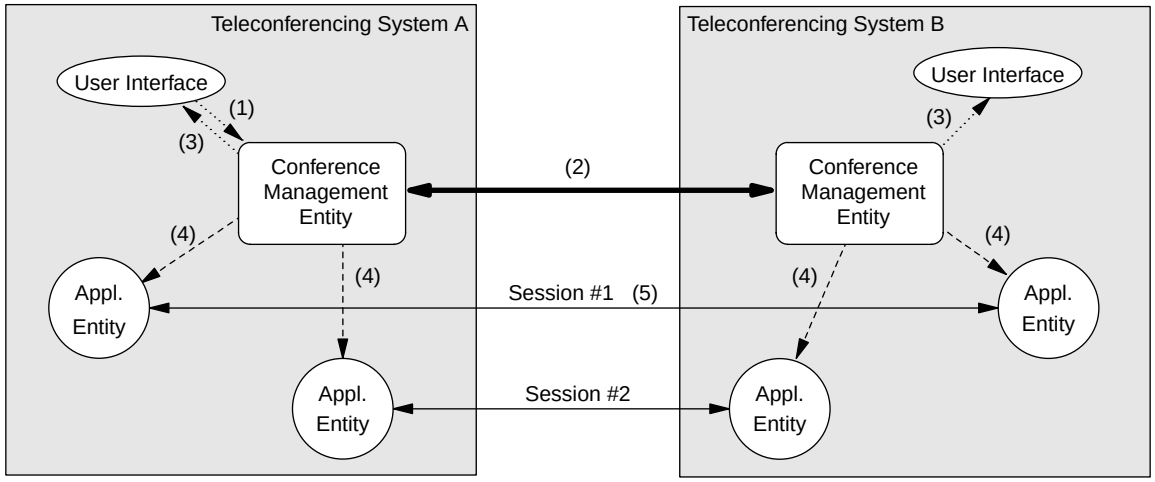
To have optional messages and to explicitly allow for unknown ones to be ignored without this being an error, provides the foundation for the extensibility of the vertical protocols. Extensibility is important for at least two purposes: a) to add new service as well as new events for present services; b) to implement application-specific local (and remote) message passing via the vertical protocols for coordination between entities of certain groupware applications.[56] In both cases,

[54] Assuming that communication for vertical protocols is within a single DMC system and therefore sufficiently secure against intruders of any kind.

[55] In the context of this thesis, it is always the conference management entity that is assumed to be the local coordinator. However, design and implementation of the MCL allow for additional coordination entities to be introduced later on.

[56] An example for application-specific message interchange via a vertical protocol is the local synchronization of `vat` and `vic` which deal with audio and video information in an Internet teleconference, respectively. Entities of these applications exchange information about the current speaker and provide lip synchronization via message exchange on the so-called *message bus* [Handley / Wakeman 94].

application entities that do understand the events can benefit from the information received without others being disturbed in their operation.



1. Floor request initiated by the user
2. Determination that the floor request is granted
3. Indication of the floor request result to the users
4. Information of application entities about the modified floor status (in parallel to 3.)
5. Reaction of application entities to the request (via their horizontal protocol)

Figure 3.12: Consistent reaction on a floor request

Figure 3.12 illustrates how central control and information consistency are both achieved via the conference management entities. The user of DMC system A initiates — via the conference control user interface — a request for the conference floor. The request is granted (as determined via the horizontal conference management protocol). This result is fed back to user on system A and is indicated to the user of system B as well. In addition, all the application entities are notified that now the user of teleconferencing system A holds the floor. The application entities of application session #1 (the shared editor with a jointly edited document) react on this change which may have been triggered specifically for this session. From this point in time on, user A may input text while user B is restricted to observing. The entities involved in application session #2 (the shared editor used for the conference minutes) do not care about the ‘‘floor change’’ event because the session policy defines that this session’s floor is independent of the conference global floor (e. g. user B is the designated scribe for the minutes).

Service Summary

Interoperability services enable a set of DMC systems to automatically determine all possible ways of interoperation by means of a procedure called capability exchange. Furthermore, these services allow application entities to locate, optionally initiate, and contact peer entities as well as to set up and participate in application sessions. As the interoperability services are carried out using the conference management protocol, the horizontal application protocols may remain unchanged. That is, the application protocols may focus exclusively on their respective functionality. All the additional communication is done locally through interaction with the conference

management entity via a vertical protocol. Table 3.7 summarizes the services to be provided in order to achieve interoperability.

Functional group	Interoperability services
Capability exchange	register application capabilities deregister application capabilities modify application capabilities update affected capability information if a change occurs
Active session capabilities	register session capabilities deregister session capabilities modify session capabilities update affected capability information if a change occurs
Application session control	create / register session terminate / deregister session modify session information lookup session information
Application entity control	join session leave session invite application entity into session invoke application entity exclude application from session lookup application information
Registry	allocate resource register resource / item deregister resource / item lookup resource / item modify resource / item monitor registry inform monitoring entities about registry modifications

Table 3.7: Interoperability services of the conference management

Coordination services help maintaining consistency across all groupware application entities on a DMC system. Actions of the local user as well as activities of other participants are indicated to each application entity. Also, the entities are informed about changes to the conference state relevant to them. This enables a coherent behavior of all system components so that a uniform DMC system is presented to the user. The provision of coordination services by the conference management and its local access through a vertical protocol relieves horizontal application protocols from dealing with these aspects. The design of the coordination services to flexibly react on the events of the conference management entity provides the foundation to integrate application-specific policies. Finally, the coordination services are easily extensible to include future enhancements to conference management services and to allow application-specific information exchange. Table 3.8 gives an summary of the coordination services to be provided by the conference management.

Functional Group	Coordination Services
Application Control	conference started / joined indication conference ended / left indication conference (focus) changed indication instantiate application entity for a session terminate application entity of a session
Status Updates	select state changes to be reported update reports – conference membership – application session membership – floor status change – conductor change – session / conference key change – and others
Local message passing	unicast generic message multicast generic message

Table 3.8: Coordination services of the conference management

Finally, note, that the conference control services and the internal management services are in practice not as strictly separated as indicated at the beginning of this section (refer to figure 3.7). Rather, on one hand, the conference control user interface entity is likely to obtain information updates via the coordination services as well as information about system capabilities and application sessions via the interoperability services. On the other hand, several application entities may — depending on the DMC system implementation and the local policies — be able in invoke conference control services on behalf of the user. Consequently, the conference control user interface entity (or any other application entity performing this controlling function), has access to and makes use of both conference control and internal management services; groupware application entities may or may not have direct access to conference control services.

3.4.3. MCL-adm Service Summary

This section has described the conference administration services covered by the MCL-adm sublayer. It has been identified that the conference administration services may be subdivided into two service groups: user-visible conference control services and internal management services.

User-visible conference control services include the functionality described in chapter two to configure and run a conference; to manage participation; and to authenticate participants. Furthermore, mechanisms to deal with floor control, conductorship, and other policy issues are provided.

Internal management services are largely invisible to the user but provide the foundation for system interoperability and for local consistency across the components of a DMC system. Following these two main tasks, these internal management services are further subdivided into interoperability and coordination services. Interoperability services provide means to express the capabilities of a DMC system and its groupware applications as well as mechanisms to establish, maintain, identify, and destroy application sessions. Coordination services provide local groupware applications with current state information about the teleconference(s) in which the user and

the respective application entities are involved. In particular, each application entity is notified about state changes that potentially impact it.

3.5. Conclusion

This chapter has introduced the concept of the Multipoint Communication Layer as a communication platform for building (non-real-time) groupware applications and integrating them into DMC systems. In the beginning, an overview of existing communication architectures for teleconferencing systems has been given. From the results of this overview, design concepts for the MCL have been derived and the range of functionality to be covered by the Multipoint Communication Layer has been defined. These concepts and the functionality are reflected in the design of the Multipoint Communication Layer. The resulting MCL architecture describes a network-independent infrastructure that conceptually distinguishes between generic transport and synchronization services, conference-related functionality as well as means for application-specific support. With respect to all its services, the MCL is restricted to the provision of mechanisms and leaves the definition of policies for the utilization of these mechanisms up to the groupware applications.

The Multipoint Communication Layer is composed of four sublayers that have been introduced in this chapter. The services provided by the lower three sublayers have been defined. The lowest sublayer, MCL-m/t, interconnects any number of DMC systems in a teleconference and provides the multipoint communication facilities for information exchange in the conference. The next higher sublayer, MCL-syn, offers generic synchronization services. Finally, MCL-adm introduces the notion of conferences; it supplies conference control functionality and offers the mechanisms to integrate groupware applications into teleconferences. For the uppermost sublayer, the MCL-app, a rough outline of potential services has been given: these include common services (such as remote procedure call and voting mechanisms) as well as specific services (such as the basic building blocks for whiteboard and file transfer applications). Security functionality is not addressed in a sublayer of its own but is spread across several sublayers of the MCL as appropriate: encryption mechanisms are located within MCL-m/t, authentication and key distribution in MCL-adm. It has been pointed out that further security mechanisms may be implemented underneath the MCL within the network and above the MCL in the groupware applications.

The remainder of this thesis deals with a specific implementation of a subset of the MCL that is entirely based on international standards. The next chapter shows how this set of standards maps onto the MCL structure and outlines this implementation as well as its integration into a specific DMC system architecture.

At the time of writing, however, not all the MCL services are covered in a single set of international standards. As compliance with such standards is a prerequisite for an interoperable implementation, only those parts of the Multipoint Communication Layer have been implemented for which international standards do exist, or extensions to the exiting standards could be easily implemented in a backwards compatible manner. The author is contributing to the international standardization process in the ITU-T and in the IETF with the aim of addressing those services not covered so far as well as improving standardized services and protocols that are incomplete or inefficient in their current definition.

4

Implementation Outline and Component Interfaces

The previous chapter has introduced the Multipoint Communication Layer outlining an architecture for data communication in teleconferences and describing the services to be provided by such a communication platform for teleconferencing systems. Starting from this chapter, the remainder of this thesis deals with the implementation of the MCL services.

It is obvious that in the area of teleconferencing any meaningful implementation has to be based on existing and emerging international standards in order to find acceptance in the teleconferencing community. This is due to the fact that the bottom line in teleconferencing (and telecommunications in general) is interworking with other implementations. For the implementation performed within this thesis, the ITU-T T.120 series of Recommendations is taken as the standardized basis to provide the services of the Multipoint Communication Layer. The T.120 series has been chosen for several reasons:

- at the time of writing, the T.120 series is the single sufficiently complete standardized infrastructure that covers the MCL services to a large degree;[1]
- the implementation of a DMC system also requires standards for audiovisual communication to be in place (such as the ITU-T H.3xx series) that are integrated in a standardized fashion with the data and control infrastructure; this is only well-defined for T.120 and H.3xx (in the T.130 Recommendations); and
- the T.120 series is broadly accepted by the teleconferencing industry.

This chapter provides the linking element between the MCL concepts introduced in the previous chapter and the particular implementation of the MCL based on the T.120 series. In the first section, the use of the ITU-T T.120 series of Recommendations as the standardized basis to implement the MCL functionality is described. The services of the relevant Recommendations are described and it is shown which parts of the MCL services they do cover.

[1] This is elaborated on further in section 4.1.

As already stated in section 1.3, the design and implementation efforts for the T.120-based data communication infrastructure described in this thesis are embedded in the context of the development of the EURO.VISION desktop multimedia conferencing system and the associated software development kit (DMC SDK). Therefore, the remaining sections of this chapter after section 4.1 are devoted to the integration of the T.120-based infrastructure into the local system environment. The two issues of major importance to achieve the integration are the overall DMC SDK architecture as well as the interfaces between the T.120 components and the surrounding parts. Section 4.2 provides an overview of the software development kit for desktop multimedia conferencing (DMC SDK) and indicates how the MCL implementation fits into the architecture. Following this, sections 4.3 and 4.4 address the concepts and implementations of the interfaces towards the underlying networks (including the implementation of the transport protocol hierarchies themselves) and towards the groupware applications using T.120 services, respectively. A brief summary concludes this chapter and leads over to the implementation descriptions of the core components of the T.120 (and thus the MCL) infrastructure.

4.1. The ITU-T T.120 Series of Recommendations

The ITU-T T.120 series of recommendations defines an infrastructure of ‘‘Data Protocols for Multimedia Conferencing’’ [ITU-T T.120]. As already outlined in subsection 3.1.7, this infrastructure includes establishment of a multipoint communication environment for a teleconference based on point-to-point connections and the provision of appropriate communication mechanisms for groupware applications within this environment. Furthermore, conference control facilities to run a conference and internally manage its constituents are provided as are services and application-specific protocols to simplify the design of groupware applications.

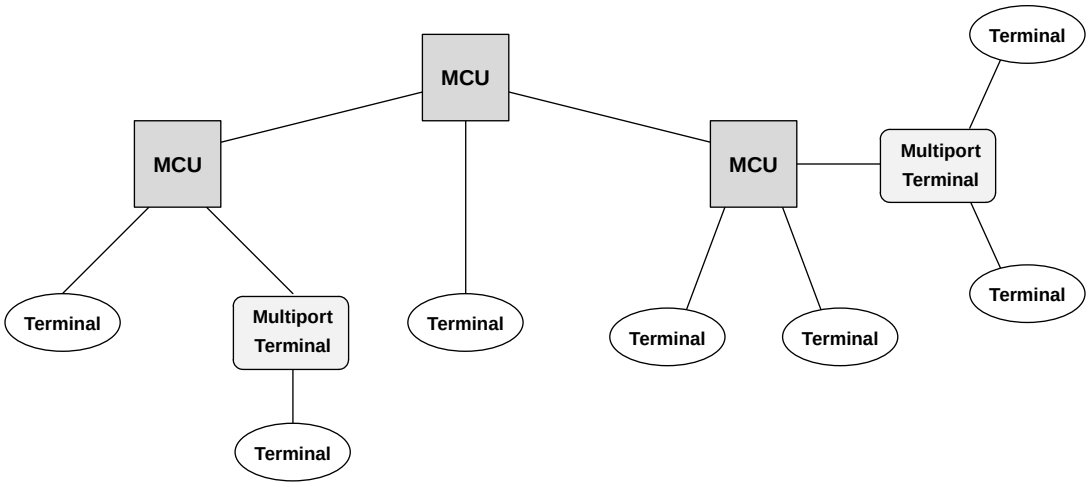


Figure 4.1: Interconnection of nodes that form a T.120 conference

Figure 4.1 depicts a set of T.120 systems interconnected to form a T.120 conference. In the figure, three different types of components are distinguished — collectively referred to as *nodes* — that are defined as follows in the T.120 series [ITU-T T.124]:[2]

[2] Refer also to the ITU-T F.700 series of Recommendations [ITU-T F.701] [ITU-T F.710] [ITU-T F.720]

- *Terminals* are ‘‘end-point audiographic or audiovisual equipment. (...) A terminal is limited within a conference to a single MCS connection[3]’’, however, a terminal may well support participation in several conferences in parallel. Terminals are intended to be primarily used by conference participants. DMC systems are primarily designed to be used as terminals (but may provide additional functionality as well).
- A *Multiport Terminal* refers to ‘‘end-point audiographic or audiovisual equipment that also includes the ability to bridge T.120 information.’’ Multiport terminals provide the same functionality to the human user as terminals. In contrast to terminals, multiport terminals support more than one MCS connection and thus may interconnect several terminals and MCUs in a teleconference. DMC systems may but need not be multiport terminals.
- A *Multipoint Control Unit* is ‘‘commonly referred to as an MCU or bridge, (it is) a multi-port device that serves to connect terminals and other MCUs in a multipoint fashion. (...) An MCU is not primarily intended for as an end-point for user communication.’’ Multipoint control units are required in (line-switched) point-to-point networks to interconnect more than two terminals. Typical DMC systems do not have the capability to act as an MCU.

Interconnection of T.120 nodes is based on point-to-point links and is done in a hierarchical fashion with an MCU typically acting as the root of the connection hierarchy — except for a simple point-to-point call where no MCU is needed at all. All connected nodes together constitute a T.120 conference. The T.120 series of Recommendations defines a set of services along with the required protocols to establish such a conference topology, exchange information within the boundaries of the conference, and control the course of the conference. Figure 4.2 gives an overview of the structure of the T.120 Recommendations. In this figure, the following (groups of) components are identified:

- *ITU-T T.120 Infrastructure Recommendations* provide the communication and conference control mechanisms of the T.120 Recommendations. T.123 defines the transport protocol hierarchies to be employed for the various supported underlying networks in order to offer a uniform transport service interface. The Multipoint Communication Service (MCS) offers the required multipoint information exchange facilities for groupware applications within a teleconference and also provides synchronization services. Finally, the Generic Conference Control (GCC) offers conference management services.

- *ITU-T T.120 Application Protocol Recommendations* define application protocols for specific purposes — e. g. file transfer or still image transmission — that are intended to be used by any application that needs this type of service. The application protocol recommendations define so-called *application protocol entities (APEs)* in terms of the services they offer, the protocols employed, their parameters (capabilities), and the interactions with MCS and GCC.

- *Non-Standard Application Protocol Entities* are included in the figure to indicate that standard-conforming T.120 teleconferencing systems may well include non-standard protocols, e. g. to implement proprietary services or to provide MCL services not yet covered by ITU-T recommendations. The T.120 infrastructure provides the necessary mechanisms allowing a

[ITU-T F.730] [ITU-T F.740].

[3] An MCS connection is a connection between two T.120-based systems for a particular conference (refer to subsection 4.1.2).

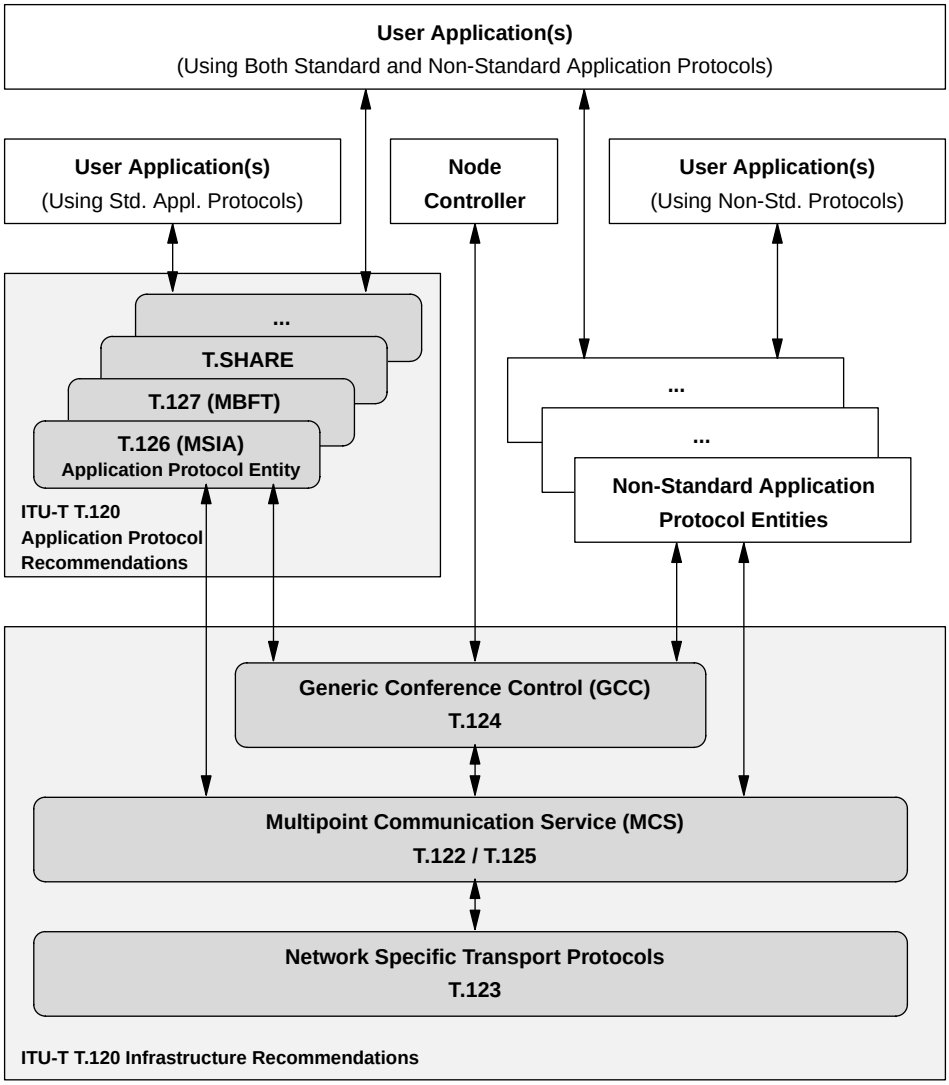


Figure 4.2: Overview of the T.120 series of Recommendations

DMC system to make information about the existence of such protocols (and their respective parameters) known to other systems and to determine whether common non-standard protocols are supported by some or all of the conference participants. In this thesis, however, no further consideration is given to non-standard protocol entities.[4]

- *The Node Controller* is the entity in a node that controls the operation of the entire system with respect to a conference. That is, the node controller sets up and tears down connections, initiates conference control operations, implements the conference policy, etc. It is likely that either the human user interacts with the node controller through some (graphical) user

[4] In the context of this thesis, services determined during the MCL design to be useful to groupware applications are deliberately not implemented as non-standard application protocol entities, as this would provide the services in a proprietary fashion. Rather, such extensions are fed into the standardization process to integrate them into revisions of the standards as far and as soon as possible.

interface or (e. g. in case of MCUs) the node controller automatically operates the system following some predefined rules. In the EURO.VISION system, a user interface application includes the node controller and thus performs all necessary local system control functions as well as the integration of T.120 and H.3xx services [TELES 96]. Hence, the node controller — which is in particular not an infrastructure component — is not described in this thesis.

- *User Applications* — regardless whether based on standardized and/or non-standardized protocols — are the groupware applications that finally provide the functionality of meeting aids and other tools to the human user. Design and implementation of user applications is beyond the scope of this thesis.

- One ITU-T T.120 recommendation is not explicitly mentioned in the figure, the *Generic Application Template (GAT)*, which is defined in T.121. In contrast to the other recommendations, GAT does not explicitly define a new service and protocol. Rather, the generic application template prescribes how standard and non-standard application protocols have to use the infrastructure recommendations so that co-existence and collaboration of any number of application protocol entities is ensured.

Design and implementation of the three T.120 infrastructure recommendations are the focus of the engineering work in the context of this thesis. The emphasis is on MCS and GCC as these two realize the three generic sublayers of the MCL. In addition, the generic application template and the application protocol recommendations are of interest as they belong to the MCL-app sublayer. The following subsections address the respective ITU-T recommendations in detail and set them into relation to the respective MCL services. This section concludes with a summary of how T.120 maps into the MCL concept and which of the MCL services are covered by T.120.

4.1.1. Transport Protocol Hierarchies

The ITU-T Recommendation T.123 [ITU-T T.123] defines transport protocol hierarchies for a variety of networks. These hierarchies have to be used in conjunction with the respective networks in order to provide the uniform transport service interface for T.120.

The MCS requires this transport service interface to follow the ISO Transport Service definition [ITU-T X.214] and to provide the following three services: `T-CONNECT`, `T-DATA`, and `T-DISCONNECT`. An established transport connection is expected to convey (arbitrary sized) SDUs flow-controlled, error-free, and in sequence between the peers. In T.123, a so-called basic profile defining the transport protocol stack to be used underneath MCS is described for each supported network type: ISDN, CSDN[5], PSDN[6], PSTN, broadband ISDN, and LANs (including intranetworks as well as IP-based internetworks), with one or two alternate profiles being provided for most networks.[7]

---

[5] CSDN stands for Circuit-Switched Data Network; an example is Datex-L in Germany.

[6] PSDN stands for Packet-Switched Data Network; an example is Datex-P in Germany.

[7] Note that if several choices to implement the same service on the same network are available, a standardized way is required to negotiate at connection setup time which transport protocol hierarchy shall be used for a particular connection. For achieving good interoperability, this requires a common base line that all the DMC systems have to implement in order to ensure that communication is always possible.

For the development of the T.120 infrastructure in the context of the DMC SDK, however, only those transport protocol stacks are of interest that are capable of integrating the transmission of T.120 data with real-time information. As no standards for audiovisual communication across PSDN and CSDN are defined in the ITU-T H.3xx series, these two network types are not considered any further. Those transport protocol hierarchies that are implemented in the EURO.VISION DMC SDK and therefore have to be supported by the T.120 implementation are shown in figure 4.3:[8]

- the basic ISDN profile of T.123 that enables use of T.120 together with the H.320 audiovisual communication services;
- the basic LAN profile of T.123 that is compliant to the H.323 Recommendation on audiovisual communication services over LANs; and
- an alternative PSTN profile (also described in T.123) that allows use of T.120 in conjunction with audiovisual communication services.

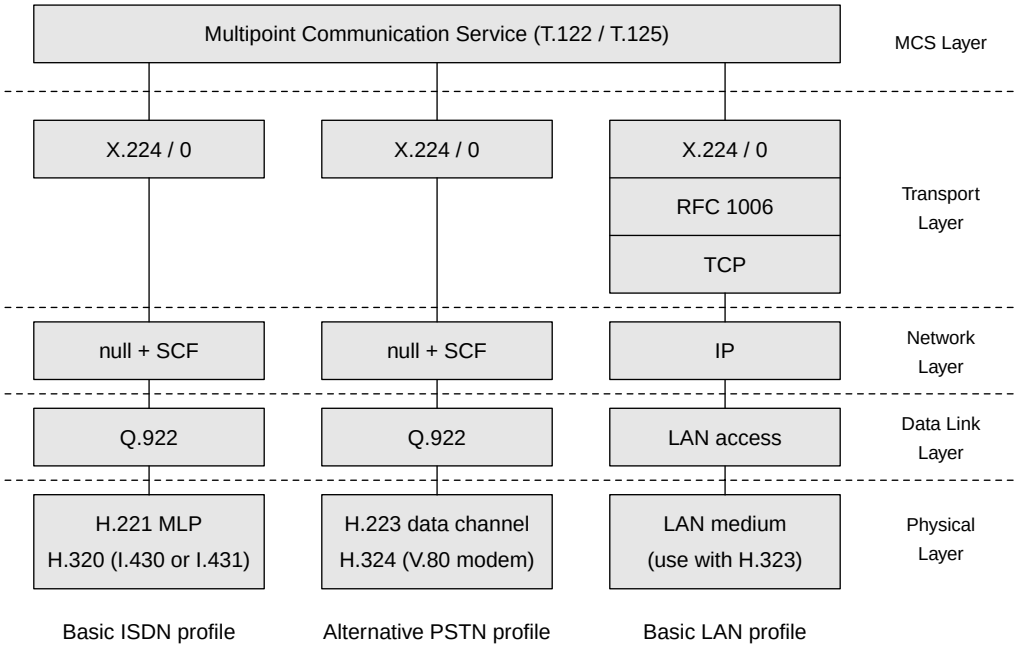


Figure 4.3: Implemented transport protocol profiles of T.123

As becomes apparent from figure 4.3, the ISO transport protocol class 0 (X.224/0) [ITU-T X.224] is used to provided the requested transport service regardless of the underlying network. The ISO transport protocol class 0 assumes that error control and flow control are already available from the network layer service[9] and only provides the mechanisms for segmentation/reassembly and addressing of a particular transport service user. However, these network service requirements of

[8] The DMC SDK does not yet cover ATM networks so that the respective protocol stack is not included. The same applies to the SPX/IPX protocol hierarchies for communication in LANs.

[9] The protocol is designed for use on top of so-called *class A networks* (refer to [ITU-T X.224]) that are characterized as networks that do error detection and error correction sufficiently well with respect to the demands of the transport service user.

X.224/0 are not fulfilled by any of the three depicted network multiplexers: neither the H.221 MLP channel, nor the H.223 data channel, nor the IP datagram service offer reliable communication services. Hence, additional data link and transport protocols are used to enhance the poor network service and meet the aforementioned requirements.

Furthermore, for multimedia communications — i. e. simultaneous transmission of several media streams (audio, video, T.120 data) between two nodes — the network layer has to provide multiplexing to allow independent transmission of the distinct media streams. While the connectionless datagram service provided by IP inherently provides multiplexing facilities, a separate multiplexing layer is required for the line-switched, connection-oriented network services of ISDN and PSTN (which is defined in the H.3xx series of ITU-T recommendations).

The protocol suites that are defined in T.123 to provide the required services to the transport layer on the various networks are discussed in the remainder of this subsection.

- *ISDN / PSTN*

  For ISDN and PSTN, the lower three layers perform similar functions, layers two and three are identical.

  Layer one defines transmission of bit streams over one or a set of aggregated ISDN B channels or a single telephone line using V.80 modems [ITU-T V.80]. The next layer is the network-specific multiplexer allowing for simultaneous transmission of distinct media streams. For ISDN-based communication, the multiplexer is defined in H.221 [ITU-T H.221]. The H.221 *Multi-layer Protocol (MLP)* channel is used to carry T.120 information (this channel is utilized according to the rules defined in [ITU-T H.242]). For PSTN, multiplexing is defined in H.223 [ITU-T H.223] and a control protocol allows to create dedicated T.120 data channels [ITU-T H.245].

  Layer two provides reliable, flow-controlled, in-sequence delivery of information units through the LAPF (link access procedures to frame mode bearer services, an HDLC variant) protocol defined in Q.922 [ITU-T Q.922]. Q.922 allows multiplexing the single data channel provided by layer one so that a Q.922 control channel and virtually any number of independent reliable data link channels may be established in parallel.

  At layer three, the *synchrony and convergence function (SCF)* of Q.933 (refer to [ITU-T Q.933] and [ITU-T T.123]) is used to set up and tear down individual layer two channels by using the Q.922 control channel. This feature of Q.922 and Q.933 is used to establish multiple transport connections across the same link. Once a channel is established, information units to be transmitted on this channel are transparently passed through to layer two.

- *LAN / Internet*

  When concerned with IP-based inter- or intranetworks, the specifics of the physical networks at layers one to three are invisible to T.120 so that the following considerations address only the IP and TCP layers:[10]

  The IP layer provides a connectionless datagram service and thus already offers (datagram-based) multiplexing of the underlying medium in order to convey control, real-time, and one

[10] All relevant issues of transmitting IP packets over any type of physical network are covered in the respective RFCs published by the IETF.

or more T.120 data channels. As the IP service is connectionless, error and flow control are not provided so that a second (transport) layer is required to offer the reliable services upon which X.224/0 can be based. The Transmission Control Protocol (TCP) is used to provide the reliable, flow-controlled, and in-sequence delivery of a byte stream on top of the IP service. As X.224/0 expects the network to transmit NSDUs as identifiable packets, a packetization mechanism is provided on top of TCP to demarcate packets contained in the byte stream [RFC 1006]. This allows X.224/0 to be used as a second transport protocol on top of TCP.

- *Use of Multicasting*

  The definition of the services required by MCS is oriented at the model of circuit-switched networks and so are the protocol hierarchies to provide these services including the stack for IP-based networks. The use of TCP — *the* ubiquitously available reliable protocol in the Internet — as the basis for reliable communication is the logical conclusion from the approach of creating conferences from a hierarchy of point-to-point connections. While all the other networks inherently provide point-to-point services only, this restriction is not imposed by IP networks. In general, they do offer multicasting services, at least within subnets.

  At the time of writing, however, T.123 does not cover profiles for using multicast capabilities of networks underneath the MCS. Neither a service definition for the transport service interface nor a (set of) protocol suite(s) have been standardized. This is mainly due to the facts that a) no standardized reliable multicast transport protocol does exist and that b) the MCS protocol itself is primarily designed for point-to-point connections. Chapter six addresses an extension to the transport profiles as well as to the MCS protocol that allows leveraging multicast capable network environments without disturbing interoperability with non-extended DMC systems.[11]

4.1.2. The Multipoint Communication Service Provider

The *MCS provider* is the entity on a DMC system that implements the Multipoint Communication Service (MCS). The MCS provides the low level communication infrastructure enabling information exchange between arbitrary groupware application entities in a multipoint teleconference. The multipoint communication environment for a conference is constructed by interconnecting the nodes participating in the conference through a set of point-to-point connections, the *MCS connections*, each of which is based on a T.123 compliant protocol stack.

In order to implement four independently flow-controlled priorities, an MCS connection typically consists of four transport connections each of which is assigned to convey data of a certain priority.[12] It is exclusively used for a single conference, i. e. if two nodes are to communicate with one another in two conferences, then two MCS connections are required.

---

[11] The approach taken in this thesis has been proposed to the relevant working group in the ITU-T, and since summer 1996 extensions to T.123 and MCS following the concepts developed by the author are being worked on in this group [Ott 96b] [Ott 96c] [Galvin 97].

[12] Optionally, in an MCS domain less than four transport connections per MCS connection may be used which, however, then requires data of several priorities to be mapped onto a single transport connection so that the independent flow control for the affected priorities is lost.

The interconnection topology is always hierarchical (see figure 4.1) — even if only two nodes are involved — with the root node being termed the *Top MCS Provider*. A set of nodes interconnected by MCS is termed an *(MCS) domain*, and is identified by an *(MCS) domain selector* to be able to distinguish between multiple domains at the same MCS provider. Domains are strictly separated from one another even if they share one or more MCS providers; i. e. no cross-domain communication or coordination is possible by means of MCS services. A set of *Domain Parameters* are negotiated upon setup of the first MCS connection when creating to a domain.

The MCS offers its services through two different types of service access points, a *Control MCSAP* and an *MCSAP*. The Control MCSAP is exclusively used to set up and tear down MCS connections thereby constructing and destroying domains. The Control MCSAP is not tied to a particular conference at a T.120 node, and per node only a single entity (termed *controlling entity*[13]) may attach to the Control MCSAP; this restriction is imposed to avoid conflicts for setting up and tearing down MCS connections.

Once a domain is created by interconnecting a set of MCS providers, groupware applications (referred to as *MCS Users*) have to attach to the domain in order to make use of MCS services, e. g. to exchange information with their peers. Groupware applications become MCS users for a particular domain by attaching through an MCSAP to this MCS domain. A groupware application may be attached to any number of domains in parallel. There is also no limitation on the number of MCSAP instances per T.120 node and per domain and therefore no limit on the number of groupware application entities.[14] When attaching to an MCS domain, each groupware application is assigned an *MCS User Id* that is unique with respect to the domain and identifies each MCS user in all subsequent actions. This identifier also serves as an address for unicasting information to a single MCS user.

Figure 4.4 depicts a sample configuration of four nodes with MCS providers hosting two MCS domains and several MCS users being attached to either or both these domains. Note that the controlling entities (i. e. the GCC providers) are also attached to User SAPs of the domain(s) hosted by the respective MCS providers because they need to exchange information with their peers in the respective domain and they require a User SAP attachment to do so (refer also to the next subsection).

Within a domain, the concept of communication *channels* is used to provide flexible unicast, multicast, and broadcast services and thus allow multiplexing of the underlying MCS connections. A channel is identified by a unique number and collectively refers to zero, one, or more MCS users that are *members* of the respective channel. Information units transmitted in a domain are addressed to a channel. They are then distributed by the Multipoint Communication Service to all members of that channel. Three different channel types are distinguished:

- *Multicast channels* are used to distribute information to a subset or all MCS users in a domain. Any MCS user may become member of any multicast channel and give up membership at any time. Multicast channels are used to implement multicasting and broadcasting

[13] The GCC provider (see subsection 4.1.3) acts as the controlling entity for the MCS and is itself controlled by the node controller.

[14] The MCS specification restrict the number of MCS users per domain to 64,535 but this limit is far beyond what is expected to be needed for tightly coupled conferences with some hundred participants.

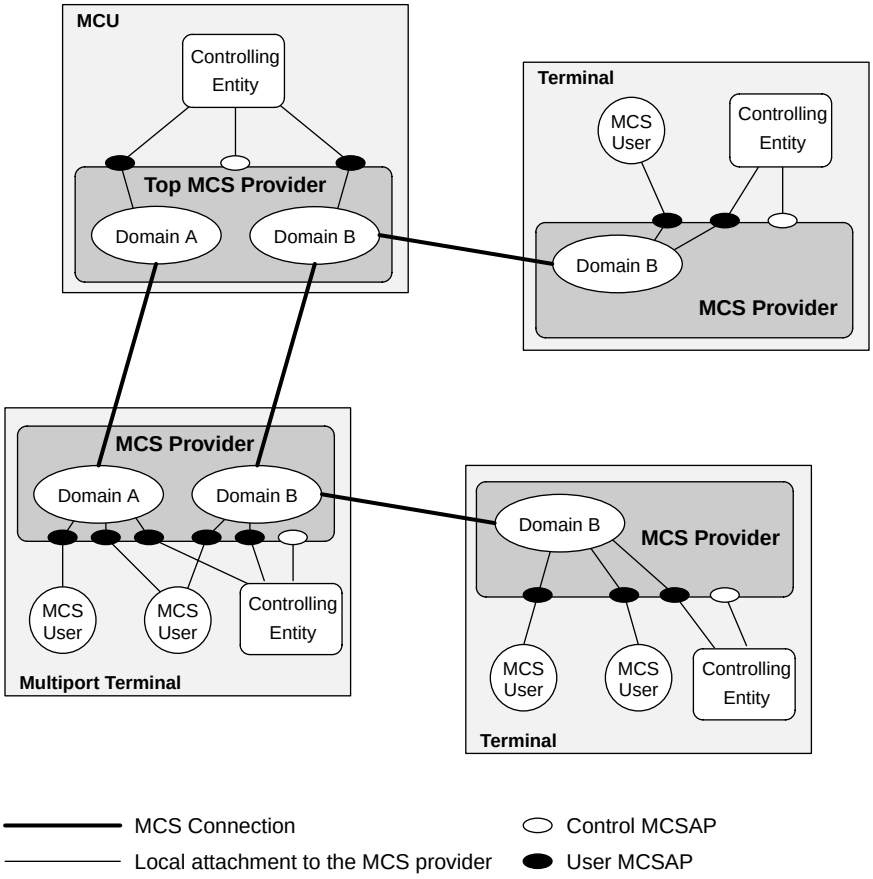


Figure 4.4: Sample configuration of four MCS providers in two domains

facilities. A set of 1000 identifiers is pre-assigned for ‘‘static’’ multicast channels. Further identifiers for dynamic multicast channels are assigned by the MCS provider upon request. Static multicast channels are considered to exist permanently (even if they do not have members) while dynamic ones are created when the first MCS user joins the channel and are destroyed when the last member leaves.

- *Single member channels* refer to exactly one MCS user and are used to provide a unicasting service. For each MCS user id assigned upon attachment of a groupware application, a corresponding single member channel is created with the channel id being equal to the MCS user id. Only this MCS user is allowed to become member of the single member channel. Through this mechanism, each MCS user automatically obtains a single unicast address per domain.
- *Private channels* are used to control membership in a channel and thus the information distribution through this channel.[15] Private channels are dynamically created as needed by MCS users. It is up to the creating entity to decide which other MCS users may join this particular

[15] Note that the concept of private channels does not provide for real *privacy* because MCUs and multiport terminals that forward data passed along for such a channel could easily listen in.

channel, whether to exclude certain members from the channel, and when to destroy the channel. The channel ids for private channels are assigned dynamically.

With the channel concept, the MCS allows its users to abstract entirely from the underlying interconnection topology of the MCS domain and from the location of their peers. Instead, a flat addressing space is presented to the MCS users in which each entity or subgroup is directly addressable. In addition to multipoint information transmission, the MCS defines the concept of tokens to provide mechanisms for synchronization between MCS users within a domain. For further details refer to section 5.1.

The MCS provider following the T.122 and T.125 Recommendations covers a subset of the functionality defined for MCL-m/t and MCL-syn as indicated in table 4.1. Compared to the MCL service overview discussed in section 3.3, MCS provides virtually all MCL-m/t service requirements. However, only one of three MCL-syn services is covered by MCS and security issues are not dealt with at all. Section 5.1 gives a more detailed description of the of the MCS services and the underlying MCS protocol. For further details refer to the MCS service definition [ITU-T T.122] and the MCS protocol specification [ITU-T T.125].

Property	To be provided by MCL-m/t and MCL-syn	Covered by MCS
Number of entities	one-to-one, one-to-many, one-to-all, etc.	all variations provided
Unit of transmission	packet-based or stream-based	packet-based
Error control	reliable communication	provided
Flow control	combination of rate and backpressure flow control	back pressure flow control
Ordering	per-source, causal, and global ordering	per source / global ordering
Inter-stream synchronization	for non-real-time information	not provided
Tokens	exclusive and shared	provided
Checkpointing	means for signaling checkpoints and rollback	not provided
Confidentiality	hooks for encryption synchronized signaling of key changes	not provided not provided

Table 4.1: MCL-m/t and MCL-syn services provided by the MCS

4.1.3. The Generic Conference Control Service Provider

The *GCC provider* is the entity on a DMC system that implements the Generic Conference Control service (GCC) [ITU-T T.124]. The GCC provides the mechanisms to set up and control T.120 conferences and to manage groupware applications as part of such a conference. There is a one-to-one mapping between GCC conferences and MCS domains: a GCC conference encompasses all nodes involved in an MCS domain with all applications attached to the domain.

The GCC provider binds to the Control MCSAP of the MCS provider and thus acts as the controlling entity for the MCS. Furthermore, the GCC attaches through an instance of an MCSAP to each domain active at its MCS provider in order to be able to exchange information with other

GCC providers in the respective domain (i. e. the GCC conference). Like the MCS provider, the GCC provides two types of service access points, a *Control GCCSAP* and a *GCCSAP*.

- Through the Control GCCSAP the operation of the T.120 node as a whole with respect to T.120 conferences is controlled. As in MCS, only a single entity may attach to the Control GCCSAP. This control function is performed by the T.120 node controller. The Control GCCSAP offers the (user visible) conference control services roughly comparable to those described in section 3.4.1 which include all decisions about setup and teardown of MCS connections.
- A regular GCCSAP is used by a groupware application entity to register with the GCC service provider for a particular conference. The GCCSAP provides services that are functionally similar to the internal management services described in subsection 3.4.2. There is virtually no limit on the number of GCCSAP instances at a GCC provider and consequently no limit on the number of groupware application entities registered in parallel to the same conference. A groupware application entity that wants to register with several conferences or several times with the same conference requires a separate GCCSAP instance per registration and per conference.

Groupware applications that are to participate in a T.120 conference thus attach to an MCSAP to be able to use MCS services and they attach to a GCCSAP to register themselves in the conference. The node controller attaches only to the Control GCCSAP but is capable of creating and destroying as well as manipulating all the conferences. Figure 4.5 illustrates the interactions of MCS, GCC, two T.120-based application protocol entities, and the node controller on a T.120 node with two active GCC conferences, the two corresponding MCS domains, and one T.120 application participating in each conference.

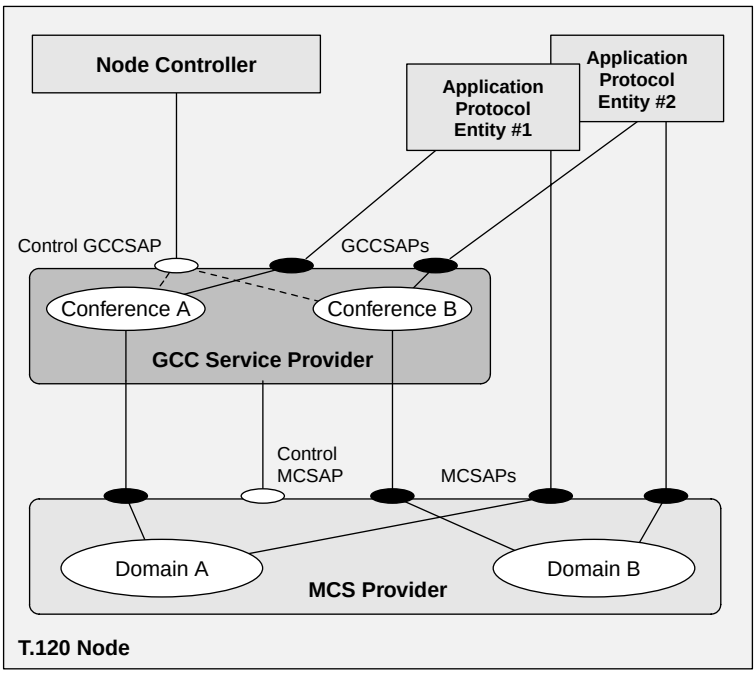


Figure 4.5: Model of a T.120 node with two conferences and two application entities

Although the underlying Multipoint Communication Service is capable of hiding the details of the hierarchical structure from the Generic Conference Control, the latter exploits knowledge about the conference topology for its control protocol. Like MCS, GCC uses a centralized management concept for the conference: the Top GCC Provider that is co-located with the Top MCS Provider performs the role of the central coordinator. The Top GCC Provider is in charge of maintaining all state information about a GCC conference and distributing it to the other GCC providers. Furthermore, the Top GCC Provider arbitrates resources, assigns identifiers, and is responsible for validating and processing most of the GCC service requests.

The Generic Conference Control service provides a variety of services for conference configuration, establishment, and termination (which are offered at the Control GCCSAP). These services include means for creating a conference and its conference profile, querying information about ongoing conferences, modifying (parts of) the profile, and terminating a conference. Furthermore, joining and leaving a conference, inviting a user to, excluding her from a conference, as well as moving a user to another conference. For entry to a conference, GCC provides hooks for arbitrary authentication mechanisms; however, only simple password protection is fully specified so far in T.124.

In a conference, GCC supports two predefined roles, the *convener* and the *conductor*. The convener is the GCC provider that has initially created the conference and defined the conference profile. A GCC conference always has a convener (although the convener need not be present during the entire conference) and this role does not change throughout the lifetime of a conference. For conductorship, the conference profile defines whether the conference may be conducted (i. e. have a designated conductor) or not. If the conference is conductible, a conductor may but need not be present; conductorship may be passed between GCC providers.[16] The convener and conductor roles may have privileges associated with them. These privileges are defined in the conference profile. However, it is beyond the scope of the profile to define which GCC providers may act as the conductor.

The Top GCC Provider is responsible for maintaining all state information of the entire conference, distributing it to newcoming T.120 nodes, and providing updates to all GCC providers as necessary. The conference state of GCC consists of three parts:

- The *conference roster* is a list of all GCC providers participating in a conference. For each GCC provider, information is stored about its T.120 node and the human user(s) the respective GCC provider represents in the conference.
- The *application roster* contains information about the application protocol entities of the conference. For each application protocol entity that is registered with any of the GCC providers in the conference, a record in the application roster contains the protocol entity's capabilities and other information required for interactions in the conference. From the application roster, also active application protocol sessions and the participating application protocol entities can be derived.

[16] Note that the conductor in the GCC sense is responsible for coordination of groupware applications at the GCC layer. This function is likely to but need not be correlated to the role of a conference chairperson visible to the user.

- The *application registry* is a database of all resources that are used within a GCC conference and its application sessions. These resources include MCS channels, MCS tokens, per-conference unique GCC handles, and generic attributes without any GCC-defined meaning. As part of the registry, services are offered to obtain unique identifiers along with the registration of resources. The application registry supports the concept of ownership and access permissions and provides mechanisms to notify application protocol entities when database entries are changed.

Overall, the GCC covers most services provided by the MCL-adm sublayer and consequently includes most of the conference control information outlined for the MCL-adm. Table 4.2 gives an overview of which services of the MCL-adm sublayer (as discussed in section 3.4) are supported by the Generic Conference Control service. For a detailed description of the services offered by GCC and an outline of the underlying T.124 protocol refer to section 7.1.

MCL-adm service group	**Covered by Generic Conference Control services**
Conference Control Services	
Profile definition	permissions for certain roles to invoke privileged operations and some further control flags (see also subsection 7.1.1)
Role assignment	convener and conductor
Setup	provided
Termination	provided
Join	provided
Leave	provided
Invite	provided
Exclude	provided
Floor assignment Authentication	not provided simple password protection plus hooks for n-way authentication schemes
Encryption control	not provided
Interoperability Services	
Capability exchange	provided
Active session capabilities	provided
Application session control	provided
Application entity control	join, leave, and invoke
Registry	provided
Coordination Services	
Application control	indication of conference start / end and application entity invocation
State updates	application session membership and conductorship
Local message Passing	not provided

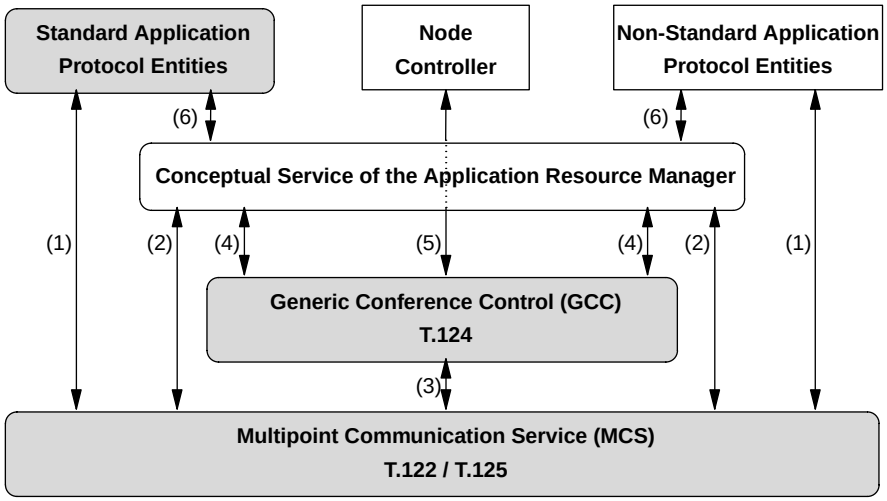
Table 4.2: Comparison of GCC and MCL-adm services

4.1.4. Generic and Specific Application Services

At the time of writing, the T.120 series contains four specifications that define services that conceptually belong to the MCL-app sublayer. One of these, the *Generic Application Template (GAT)* defined in T.121 [ITU-T T.121], specifies generic services while the other three, T.126, T.127, and T.SHARE (see below), define services and protocols for applications with particular needs. This subsection gives only a brief overview of the services defined in those four (draft) recommendations, as these do not belong to the core infrastructure of the T.120 Recommendations.

Generic Application Template (T.121)

The Generic Application Template defines a set of rules prescribing how standard as well as non-standard application protocols have to make use of the infrastructure services of MCS and GCC when they are to be used within a T.120 conference. GAT defines the concept of an *Application Resource Manager (ARM)* that invokes MCS and GCC services on behalf of the application protocol entity. In T.121, the ARM is described to be part of the application protocol entity. Conceptually, however, an ARM entity combines various infrastructure services and offers a (simpler) ARM service to the application protocol entity thereby ensuring compliance to the T.121 rules. Figure 4.6 depicts the location of the conceptual ARM service interface hiding many MCS and all GCC services from the application protocol entity. The guidelines laid down in the GAT specification are used in all of the standard application protocol recommendations discussed in the following.



(1) MCS services for data transmission and usage of resources
(2) MCS services for allocation of resources
(3) MCS services of the Control MCSAP
(4) GCC services offered at the regular GCCSAP
(5) GCC services offered at the Control GCCSAP
(6) Application Resource Management services mapped to (2) and (4)

Figure 4.6: Conceptual service interface of the Generic Application Template

Multipoint Still Image and Annotation Protocol (T.126)

The ITU-T Recommendation T.126 defines a protocol and a service interface for *Multipoint Still Image and Annotation* functionality (abbreviated as *MSIA* or *SI*). T.126 is intended to be used as a standardized means for information interchange by all groupware applications that require graphics information to be exchanged in a non-real-time fashion in the context of a teleconference.[17] T.126 does not define an application itself but rather provides a set of services to be used by all groupware applications that need (a subset of) the class of functionality specified in SI.

The basic elements of a T.126 application session are *workspaces*. In principle, a workspace is a user-visible rectangular area of pixels to which all the drawing operations defined in T.126 may be applied. A workspace is constructed from any number of ''stacked'' *planes*; the contents of higher planes conceals contents of lower ones. If the optional telepointer service is supported for the a workspace, the uppermost plane is always the *virtual pointer plane* containing one or more telepointers. Workspaces are created and destroyed as needed by the application; no upper limit is defined by SI for the number of workspaces concurrently in use in an application session. Upon creation of a workspace, its characteristics (including the number of planes, the dimensions in pixels, the number of supported colors, etc.) are defined. Workspaces are made visible to participants of a conference through so-called *views* on the workspace. A view is the rectangular region of a workspace intended to actually be displayed to a user. Each user may have multiple views into the same workspace, but a single view — the *focus view* — is commonly presented to all users. Figure 4.7 depicts a workspace consisting of two planes plus the virtual pointer plane and shows a view into that workspace.

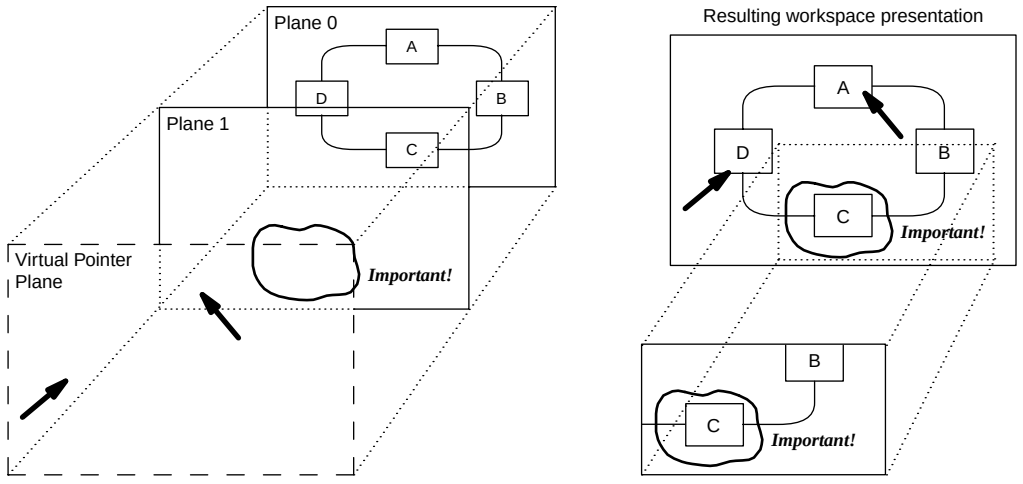


Figure 4.7: Sample T.126 workspace configuration

[17] This excludes, for example, continuous video sequences, graphics exchanged asynchronously prior to the conference, and on-line access to graphics databases from the scope of T.126.

Most of the services offered by T.126 are related to workspaces. These services can be categorized into workspace management and archiving, bitmap handling, annotation operations on workspaces, telepointer services, and remote events:

- Workspace management includes functions for workspace creation and deletion, synchronization, modifications to workspace parameters, and handling of views into workspaces. Furthermore, services are provided to archive workspaces. New workspace archives may be created to save workspaces; existing ones may be opened to retrieve and modify workspaces. Archived workspaces may be acted upon using the same services (described below) as dynamically created workspaces; in particular, (bitmap) copy operations may be used to store information in and retrieve information from archives. Examples for the usage of archives are to preserve workspaces created during an SI application session for future sessions, or to distribute workspaces (containing e. g. the transparencies of a presentation) prior to a T.120 conference, etc.
- Within a T.126 application session, bitmaps may be exchanged to be displayed on a particular plane of a workspace. Later on, bitmaps may be edited and deleted. For transmission of bitmaps, various compression formats are defined, including group 3 facsimile encoding and JPEG. These services allow slide shows and overhead projectors for presentations to be implemented.
- A set of basic drawing operations (e. g. polylines and ellipses) are supported by SI to allow annotation of workspaces, i. e. to create, modify, and delete drawings in a workspace. The operations are parameterized with a set of attributes (such as color, line thickness, etc.) Each drawing operation is applied to a particular plane of the workspace. Text and complex graphics operations (such as splines) are not directly supported by the 1995 revision of T.126 but may be transmitted by means of bitmaps. The annotation services provide means to e. g. annotate transparencies during a presentation that are represented as a (set of) bitmap(s), to implement whiteboard functionality, etc.
- Furthermore, T.126 supports an arbitrary number of telepointers. The characteristics of telepointers have already been described in subsection 2.3.1. In T.126, a telepointer is represented as a special type of bitmap that is created, manipulated, and deleted via the aforementioned bitmap services so that arbitrary shapes can be used. Telepointers are displayed on top of all other planes in the virtual pointer plane. The telepointer services complete the set of services necessary for interactive presentations, design discussions, brainstormings, etc.
- Finally, a set of services is provided for remote control of an SI-based groupware application entity. The remote control event types include key strokes, pointing device movements, and printing commands. Furthermore, services for controlling who may and who may not exercise remote control of an entity are provided. In essence, remote events combined with the bitmap and drawing services are able to provide rudimentary means for application sharing; however, this type of application is now covered by the Recommendation T.SHARE (see below).

In addition to the above workspace-based services, SI defines *hard copy functionality* that allows — independently of workspaces — to transmit bitmaps intended for processing by a particular hard copy device.

Given the wide variety of functionality defined in T.126, there are many ways to make use of SI services. For most application scenarios, however, a complete implementation of all the SI

services is not necessary (and too expensive). To accommodate use of SI for different scenarios and still preserve interoperability, T.126 defines several profiles that specify the mandatory SI services for various application scenarios. Such profiles are defined to accommodate hard copy applications (e. g. fax transmission), presentations, and whiteboard applications.

Multipoint Binary File Transfer Protocol (T.127)

The Multipoint Binary File Transfer (MBFT) protocol defined in ITU-T Recommendation T.127 [ITU-T T.127] is a standard application protocol to be used by groupware applications that require file transfer functionality in the context of a T.120 teleconference. Like SI, MBFT does not define a specific file transfer application but offers a set of services applications can make use of. In contrast to other file transfer protocols such as *ftp* [RFC 0959] and the EUROFILE transfer profile [ETS 300383], MBFT does not follow the strict client-server paradigm: the roles of the MBFT application protocol entities may change dynamically during an MBFT session. Also, MBFT provides facilities for multipoint file distribution, i. e. the simultaneous transmission of a file to several recipients.[18]

The core service of MBFT is the transmission of a file to multiple recipients. The recipients are all or a subset of the participants in a T.120 conference capable of running an MBFT entity. The sender of the file decides whether to broadcast the file to all MBFT capable nodes or whether to restrict the circle of recipients by using the MCS concept of private channels.

Transmission of a file may be initiated by the sender or by one of the recipients. The sender may choose at any time to *offer* a file to a set of recipients. Alternatively, an MBFT entity may trigger an offer from another entity (the potential sender) through a *file request* service which is either implicitly confirmed through a subsequent offer or explicitly denied. In a file offer, the sender provides an extensive description of the file (following T.434 [ITU-T T.434]) so that the potential recipients can decide whether or not they do want to receive the file. The same description format is also used in the file request to indicate which file the sender shall transmit.

Following the offer and the (optional) response by the potential recipients the sender explicitly indicates start of the transmission. Subsequently, the ‘‘blocks’’ of the file are transmitted and finally the transmission is either concluded successfully or aborted. In order to save bandwidth during transmission, MBFT includes an option for file compression.

Finally, MBFT includes a service to retrieve directory listings from another MBFT entity (as far as access is granted) thereby enabling other entities to find out about files of interest. In contrast to typical client-server file transfer protocols, MBFT does not include services such as navigation in the directory structure, directory creation, and renaming or deletion of files.

Application Sharing (T.SHARE)

The Application Sharing (AS) defined in draft ITU-T Recommendation T.SHARE[19] defines mechanisms for platform-independent sharing of window-based applications. That is, an instance of a single-user application running on one T.120 node in a teleconference is made visible to

[18] Refer also to [Ott 96a] for a comparison of file transfer services.

[19] An official recommendation number has not been assigned by the ITU-T at the time of writing.

some or all other conferees whose T.SHARE application protocol entities are part of a T.SHARE application session. The other conferees may also provide input to this single-user application instance thus using it as if the application instance was running on their local node.

In a T.SHARE application session, *hosting* and *viewing* nodes are distinguished. Hosting nodes export one or more single-user applications to an application session thus making the contents of the respective applications' windows visible to the other nodes in the T.SHARE session. On a hosting node, the windows of exported applications are termed *hosted windows*. Viewing nodes import windows of applications hosted by other nodes and display them locally. On a viewing node, the windows of imported applications are termed *shadow windows*. Windows that are neither hosted nor shadow windows are referred to as *local windows*. T.SHARE-capable nodes in a T.120 conference may act as hosting nodes or as viewing nodes or as both, they also may choose not to participate in a T.SHARE application session at all. There is no limit on the number of hosted or shadow windows per node, neither are on the number of simultaneous T.SHARE application sessions.

T.SHARE follows the general concept of centralized window sharing systems.[20] A hosting node transmits the output of exported applications to the peer entities viewing the applications. The transmitted output consists of state information, color information, and *order* and/or *bitmap* data:

- Orders contain drawing operations that match the effects that calls of the exported applications to the local graphics library or window system have on the hosted windows.[21] The remote recipients apply the received orders to their viewed windows which results in identical contents of the hosted and the viewed windows.
- Bitmaps do contain rectangular pieces of the contents of a hosted window rather than the sequences of graphics operations that result in this contents. Therefore, bitmaps can be used to update parts of windows that have been created at the hosting node before the viewing node started to receive the output stream (this also helps to accommodate latecomers in a T.SHARE session). Furthermore, they may be used to transmit the results of local operations that cannot be expressed by orders (efficiently). Finally, of course, bitmaps are usable to represent bitmap operations in a hosted window.

To some degree, orders and bitmaps are two means to achieve the same result.[22] This provides a means for optimizations to the hosting node, e. g. to minimize the latency and/or to minimize the bandwidth required for the T.SHARE session. The hosting node's T.SHARE application protocol entity may decide whether to use orders or bitmaps for each drawing operation and how to interleave the two data types in the output stream. Other means for optimizations that may be applied at the hosting side include e. g. collapsing several updates of overlapping areas into a single update and not transmitting information that is recognized to have been obsoleted.

---

[20] Refer to [Stefik *et al.* 87a] and especially to the terminology definition of [Schindler / Ott 97].

[21] The standardized orders are quite generic and provide only a small subset of the drawing operations that are typically available as calls to a graphics library. The motivation for the generic approach taken by T.SHARE is that the protocol is kept independent of the underlying window system and its library interface — that differs heavily from one system to another. As a result, for many library calls, a translation from the specific call with its specific parameters to one or a sequence of several of the T.SHARE orders is required.

[22] All orders may be replaced by bitmaps but not all bitmaps may be replaced by orders.

To remotely control a hosted application, a viewing node has to obtain permission from the hosting node which may be withdrawn at any time, i. e. the hosting node is in control of all its hosted windows. This mechanism also ensures that at most one AS protocol entity at a time is able to provide input to the hosted application. The input stream supplied to the hosted application consists of keyboard and pointing device (such as mouse, track ball, etc.) events. These events are again transmitted in a window system-independent encoding and are fed into the hosted application by the hosting node's T.SHARE entity.

4.1.5. Extension for Control of Real-Time Information

The scope of the T.120 series is restricted to deal with handling of non-real-time information only (which is indicated by the term ''Data Protocols'' in the recommendations' titles). That is, transmission and control of audio, video, and other real-time information is explicitly excluded from the scope of T.120. These issues are addressed in other ITU-T recommendations, namely the H.3xx series. However, while the H.3xx series covers transmission and control of real-time information in a network-specific manner (for ISDN, PSTN, ATM, IP networks, etc.) and is largely restricted to point-to-point calls rather than multipoint conferences, T.120 is architected to be independent of underlying networks and thus allows bridging between teleconferencing systems located in different networks. To leverage this property for real-time control, an extension to T.120 has been defined — the T.13x recommendations — that allows conference-global (network-independent) control of real-time information. While the T.13x services are used to make these conference-wide decisions, actual control and multiplexing of the affected links continues to be carried out using the respective H.3xx recommendation. Figure 4.8 shows the extensions consisting of the following five recommendations:

- Draft T.130 provides the outline of the architecture for the extensions to the T.120 series of recommendations [ITU-T T.130].
- The future ITU-T T.131 is to specify on a per-network basis the mapping of T.13x services onto the call control, multiplex control, capability exchange, and other services defined for the respective networks.[23]
- T.132 will define the low level services for conference-wide capability negotiation and routing of real-time information streams from a source to one or more recipients.
- Draft T.133 specifies the high level services for controlling audiovisual communication in a T.120 conference. These services include the audio and video control services discussed in section 2.3.1 and provide mechanisms for floor control among other services.
- Draft T.RDC defines a protocol for remote control of arbitrary devices capable of sourcing or sinking real-time information (such as VCRs, cameras, microphones, etc.) [Woollett 97].

While these extensions are required as part of the overall DMC system functionality, they are only of secondary importance to the non-real-time infrastructure developed in this thesis (for further information, refer to section 8.3). Also, they are in the stage of being revised extensively in the ITU-T working groups and the real-time extensions to T.120 would constitute a research thesis of

[23] As T.131 performs the integration of call control and multiplex control for audiovisual and data communication on top of different networks — i. e. coordinates the use of several protocols —, this service cannot be integrated into a single protocol hierarchy but rather has to reside on top of all of them. Therefore, T.131 is located within the node controller of the T.120 architecture.

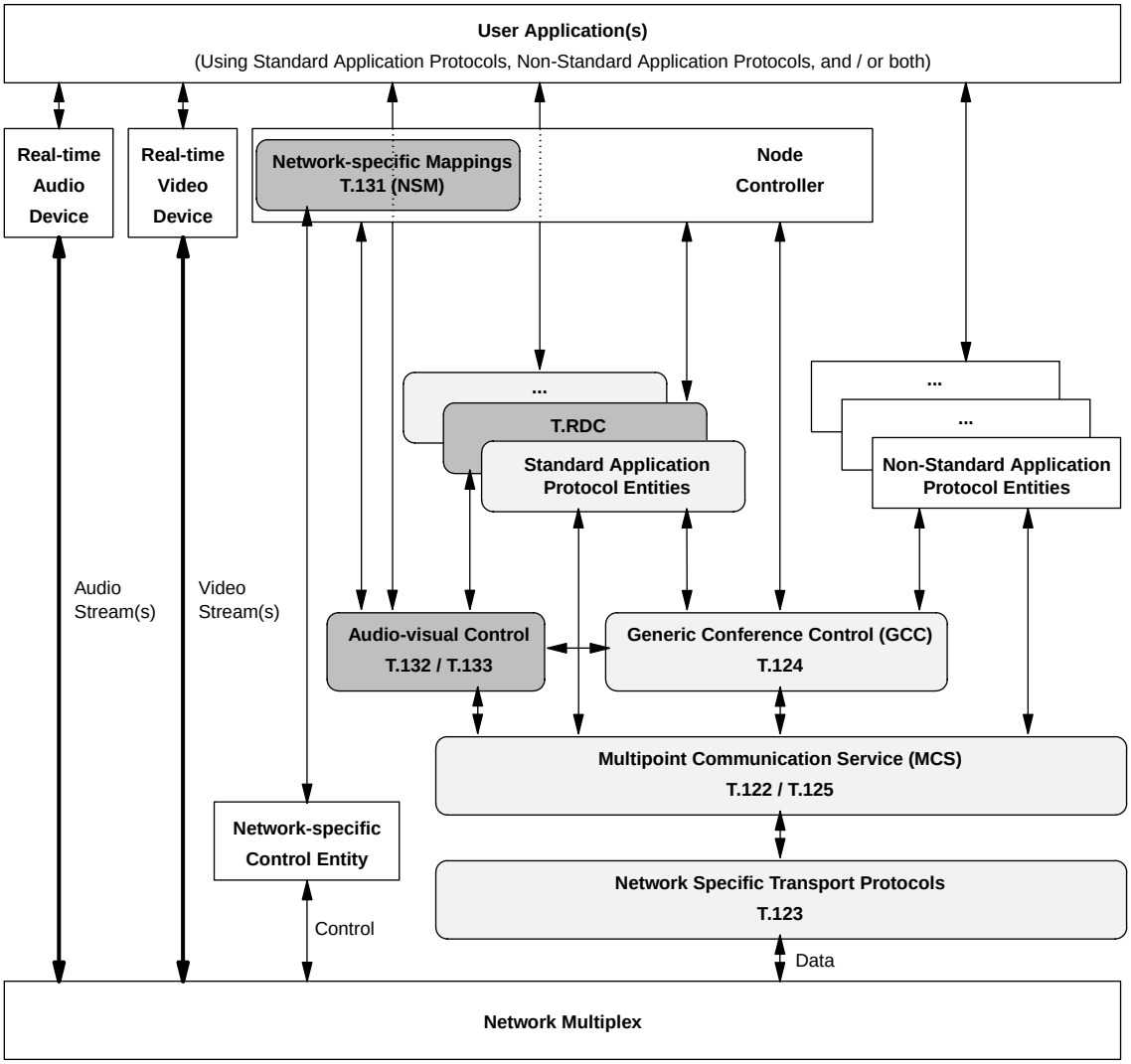


Figure 4.8: T.13x extensions to T.120 for real-time control services

its own due to their complexity and their close interactions with the various different network infrastructures. For these reasons, the T.13x recommendations are not described in further detail, nor are they covered in the implementation efforts carried out for this thesis.

4.1.6. Conclusion: Mapping between MCL and T.120 Services

This section has presented the ITU-T T.120 series of recommendations as the set of standardized services and protocols to be used for the implementation of the services identified for the Multipoint Communication Layer. In summary, the infrastructure recommendations of the T.120 series — T.122, T.123, T.124, and T.125 — cover most of the generic MCL services defined in chapter three. T.123 provides the uniform transport service interface required for network-independence of the MCL. T.122 and T.125 (the MCS) cover most service of MCL-m/t and some of MCL-syn.[24] T.124 (GCC) matches very closely the MCL-adm services. At the application

[24] A separate recommendation for dealing specifically with synchronization issues is not provided in T.120 so that some synchronization mechanisms not covered by MCS are currently included in the

protocol layer, the T.120 series includes three recommendations that cover many of the services required to build groupware applications providing the DMC system functionality identified in section 2.3.1. Thus the T.120 application protocol recommendations belong to the application-specific part of the MCL-app while the rules for using the T.120 infrastructure defined in T.121 map to the generic part of MCL-app. Figure 4.9 illustrates where the individual T.120 series recommendations do fit into the concepts of the Multipoint Communication Layer and indicates where services are not covered.

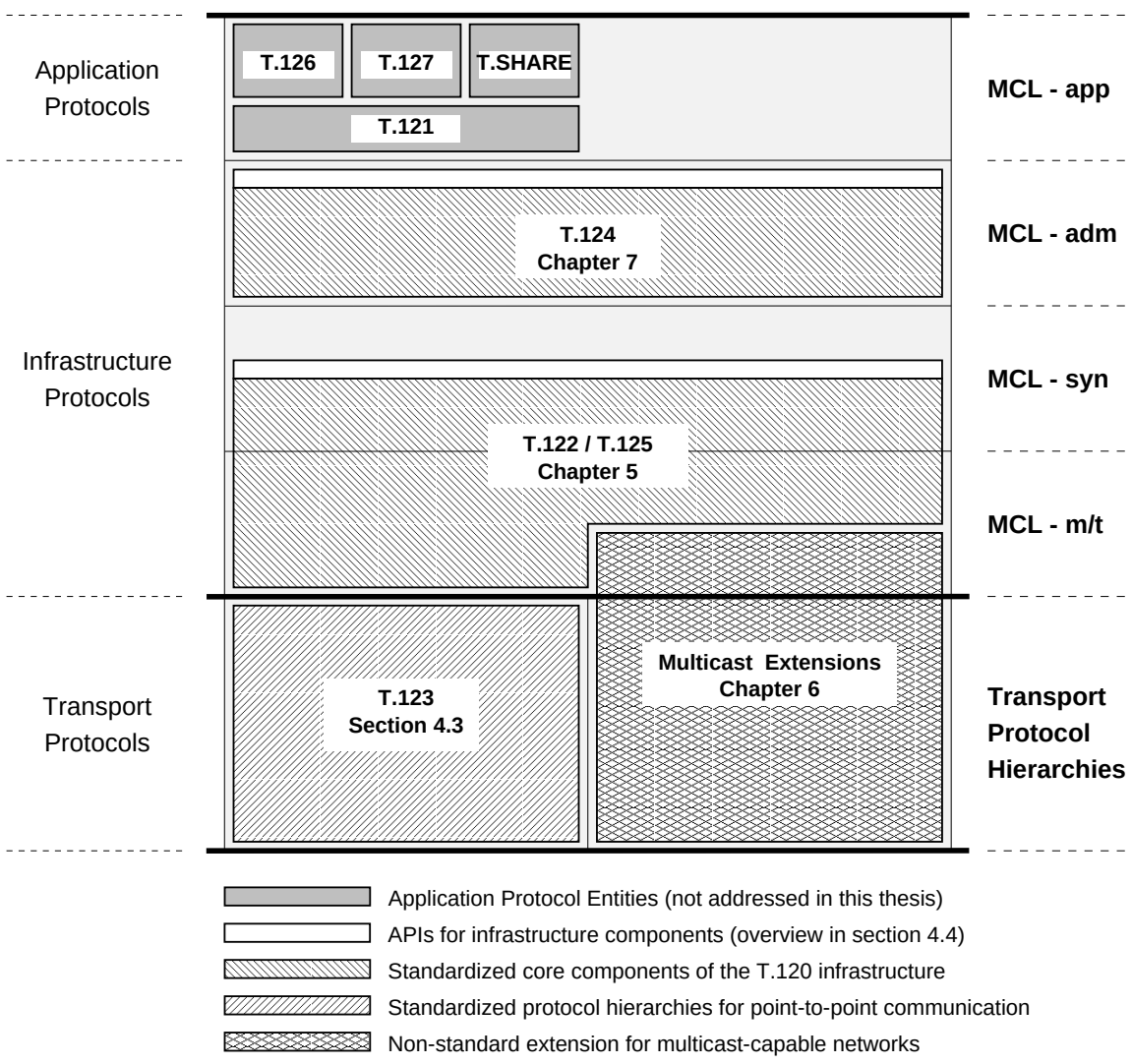


Figure 4.9: T.120 recommendations in relation to the Multipoint Communication Layer

Three types of services have been identified to be missing in T.120 when compared to the MCL concept: T.120 does not support the usage of multicast-capable networks in T.123 and MCS; certain synchronization services are not covered in MCS but have to be replicated in each application protocol requiring them; and, most important, the entire T.120 architecture provides virtually no

application protocols as needed.

security services apart from the hooks built into the T.124 conference setup procedures.[25] These shortcomings are dealt with differently in the context of the DMC SDK to which this thesis contributes:

- An extension to efficiently make use of multicast-capable networks for T.120 services has been originally developed in this thesis. Its design and implementation are briefly described in chapter six.

- The missing synchronization services have been directly incorporated into the application protocol recommendations within the ITU-T that do require the respective functionality. Therefore, no additional steps need to be taken at this point in time.

- The lack of (well-defined) security services in the T.120 recommendations is circumvented in three ways: first of all, for each of the underlying physical networks security services (authentication and encryption) have been or are being defined at the time writing [ITU-T H.233] [ITU-T H.234] [ITU-T H.245] [Toga 97]. These, however, apply to the lower layers (refer to the next section) and are therefore outside the scope of this thesis. Secondly, to enable stronger authentication based on GCC services, in the IMTC, a set of authentication schemes are being defined which are to be incorporated in the ITU-T work [Braun 96a]. These will be integrated into the GCC implementation as soon as their specification is sufficiently complete. Finally, T.120-based applications can perform encryption and key exchange independently of the T.120 services at the application layer which, again, is outside the scope of this thesis.

Figure 4.9 also shows where in this thesis the implementation of the various T.120 protocol modules is described. The focus of these descriptions is obviously on the central components of the infrastructure, MCS and GCC, that are described in chapters five and seven, respectively, as well as on the multicast extensions to transport protocol stacks which is addressed in chapter six. The implementation of the transport protocol stacks as defined in T.123 is briefly described in section 4.3, the implementation of the interfaces of the infrastructure recommendations to application protocol entities is outlined in section 4.4. Although prototype implementations of T.126 and T.127 have been carried out in the context of this thesis, they are not described here since — as already mentioned repeatedly — the focus (of design and implementation) is on the MCL infrastructure components. For information on the T.126 and T.127 implementations refer to [Gehrcke 95] and [Kerkhoff 97], respectively.

4.2. The DMC SDK Software Architecture

The preceding section has outlined the services of the T.120 series of ITU-T recommendations and how these fit into the model of the Multipoint Communication Layer developed in chapter three. Before the implementation details of the various components are described, this section briefly introduces the context in which the implementation has taken place.

As stated before, the T.120-based implementation of a multipoint communication platform for groupware applications has been undertaken in the context of the EURO.VISION system, and one design goal has been to integrate the implementation into the software development kit for

[25] And, of course, the T.120 architecture does not (and cannot) cover all conceivable MCL-app services.

desktop multimedia conferencing systems (DMC SDK) [Schindler *et al.* 94] [Schindler *et al.* 95] [Wenger 95].[27] This section gives an overview of the architecture of the DMC SDK underlying EURO.VISION and introduces the interfaces of importance for the integration of the T.120 infrastructure components.

4.2.1. Architecture Overview

An overview of the software architecture of the EURO.VISION system is illustrated in figure 4.10 that also shows which system components make use of which other system entities. The following types of entities are distinguished (white entities are based upon existing and emerging standards while hatched ones offer proprietary functionality):

- Functional modules implementing protocols and offering services conforming to international standards are depicted as white rectangles.
- APIs providing access to one or more functional modules (DMC APIs) are represented as bold dotted white ellipses.
- I/O devices (hardware and/or software) and their associated drivers are shown as hatched rectangles.
- The APIs for the devices are depicted as thin solid ellipses.
- DMC applications and the node controller which are using and coordinating the use of all these services (thus combining them to form a complete DMC system) are shown as a single bold, dotted, and hatched rectangle.

Between the various entities, different types of information are exchanged: real-time (i. e. audiovisual) information exchanges are depicted as thick white lines or as hatched thick lines if standardized and proprietary encodings and protocols, respectively, are underlying the interchange. Flows of non-real-time information types are indicated by bold black lines for standardized and by dot-dashed lines for proprietary information exchange. Note that proprietary information types are only used for purely local interactions. In all cases, the arrow of the line points to the used entity.

To human the user, the EURO.VISION *application* provides essentially all the services that have been identified in chapter two of this thesis to be of importance for a DMC system. In order to offer the DMC system functionality, the EURO.VISION application (which includes the T.120 node controller) accesses the entities implementing the local and communication services through well-defined APIs. These APIs and the services they provide access to are briefly described in the following subsection. For more detailed information about the overall software architecture, the design goals for the DMC SDK, and the various functional modules refer to [Schindler *et al.* 95] and [TELES 96].

[27] The motivation for designing a software development kit roughly follows the arguments for the development of a data communication infrastructure presented in the beginning of chapter three: providing multimedia communication facilities as services accessible through a set of well-defined interfaces allows other than pure teleconferencing applications to make use of and integrate this functionality without having to deal with the highly sophisticated technical details.

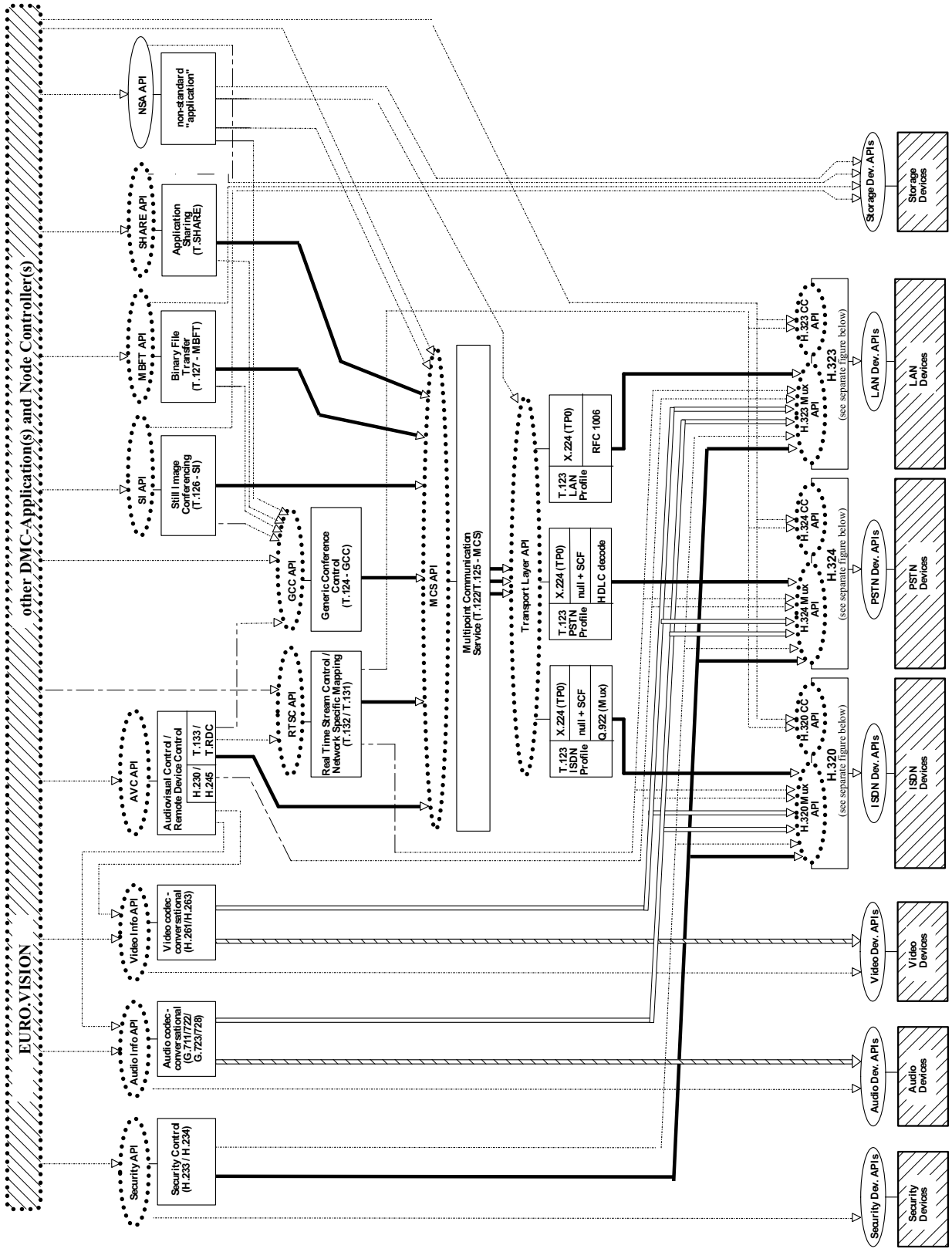


Figure 4.10: Software architecture of the EURO.VISION system

4.2.2. Application Programming Interfaces of the DMC SDK

In the DMC SDK, the functions offered by each of the individual entities are accessible through well-defined APIs. APIs for certain types of communication systems are increasingly subject to standardization (by industry fora such as the IMTC as well as by standardization bodies such as the ETSI and the ITU-T), and several of the DMC SDK APIs have been or are in the process of being standardized. Standardizing APIs in a complex DMC system architecture is desirable (if not necessary) for several reasons. First of all, they help different developers to structure and modularize their systems in a similar fashion. This provides the basis for combining modules from different vendors within the same DMC system and allowing substitution of (arbitrary) modules. Furthermore, platform independent APIs simplify porting DMC systems to different operating systems. Finally, well-defined APIs provide the basis for defining test scenarios and test suites for the various protocol modules and for (automatically) performing interoperability tests based on these test suites [Schindler *et al.* 95] [TELES 96].

In the DMC SDK underlying the EURO.VISION system, the following APIs are defined (refer again to figure 4.10):

- *Device APIs* provide typical device driver functionality such as data I/O and control functions for the hard- and software devices following the device API conventions of the operating system environment. In the DMC SDK, device APIs are defined for the security device (e. g. chip card reader), audio and video devices (for capturing, optionally en/decoding, and presenting audio and video information), network devices (e. g. ISDN boards, Ethernet boards, modems), and for permanent storage media. In most cases, these APIs are hidden from the user applications by the higher level APIs.
- *Network APIs* offer access to the various supported physical networks following the H.3xx series of recommendations. Network APIs are included in the DMC SDK for ISDN (using the H.320 recommendations), PSTN (following H.324), and for (local) IP networks (based on H.323).[28] For each network, a *call control API* and a *MUX API* are defined. The call control API provide functions for setting up and tearing down ‘‘physical’’ connections (for ISDN, the standardized CAPI is used here [CAPI 2.0], for PSTN and LAN, the APIs are derived from the CAPI). The MUX API exports functions for controlling the multiplexers for the multimedia information stream sourced and received by the DMC system. That is, the MUX API is used to specify in which ratios audio, video, control, T.120, and other information are mixed and interleaved to produce a single information ‘‘stream’’ that is sent across the respective network. Furthermore, the MUX API offers functions to actually transmit and receive the various types of information.
- The *Audio and Video Info APIs* provide information about the current state of the audio and video hardware and offer functions to control capturing and presentation of both types of real-time media streams.
- The *Security API* offers functions for mutual authentication of communicating peers, exchange of encryption keys, and individual (de)activation of encryption for particular information streams (audio, video, data) following the services defined in the H.233 and H.234 recommendations. A standardized security API is being worked on in the IMTC.

[28] Refer also to subsection 3.1.7.

- The *Transport Layer API* defines access to the OSI transport layer service based on the T.123 transport protocol hierarchies for the various networks. This API is primarily used by the Multipoint Communication Service (T.122/T.125), but other (non-standard) applications may make use of the transport layer API as well. The transport layer API follows the *Transport Layer Interface (TLI)* [TLI 88] [XTI 92].
- *Teleconferencing APIs* consist of the standardized T.120 infrastructure APIs and the application level service APIs. The former include
  - the MCS API providing access to the T.122 services in the MCS provider;
  - the GCC API offering the functionality of the GCC provider;[29]
  - the AVC API offers functions for high-level control of audio and video in a teleconference including remote device control; and
  - the RTSC (real-time stream control) API provides functions for both conference-global control of audio and video streams (the T.132 services) and for mapping network-independent multiplex and call control requests to the APIs of the respective specific networks.

  The T.120 application APIs offer access to the services of the respective T.120 protocols. APIs are supported for the functional modules implementing T.126 and T.127. In the future, also a T.SHARE API will be added.
- *Non-standard APIs* would be used to provide access to proprietary extensions (implemented in non-standard functional modules) that may be added to the EURO.VISION system and the DMC SDK architecture. An example for a non-standard application protocol and API is the WYSIWIS application sharing system of EURO.VISION that utilizes a proprietary protocol [Völz 93].[30]

4.2.3. Interfaces for the Integration of T.120

Out of the aforementioned variety of APIs, for the implementation of the T.120 multipoint communication infrastructure and its integration into the DMC SDK, four types of interfaces are of interest:

- the network-specific MUX APIs upon which the T.123 transport protocol stacks are based;
- the transport layer API that is provided by the T.123 implementation and used by the MCS provider;
- the MCS and GCC API as the interfaces to the T.120 infrastructure components that are implemented as separate processes; and
- the SI and MBFT APIs providing access to the T.120 application services that are implemented as libraries.

For the implementation of these APIs three different concepts are used that account for the different nature of the services and the implementation requirements on the respective service provider modules.

---

[29] The MCS and the GCC API have been standardized by the IMTC; refer to [Johnstone 96] and [Braun 96a], respectively.

[30] Note that with the availability of the first draft of T.SHARE since August 1996, efforts are underway to port the WYSIWIS application to the T.SHARE service platform.

The primary means to implement communication protocols in or close to operating systems is the STREAMS mechanism [STREAMS 90] as it provides for efficient protocol processing and allows arbitrary combinations of protocol layers — as are needed for the three different hierarchies of T.123 to be included in the DMC SDK. Therefore, all T.123 protocols are implemented as STREAMS modules with a TLI interface provided by the X.224 module to its service users. The lowest T.123 protocol module interfaces to the network specific multiplexers via a STREAMS interface, too. The implementation of T.123 is briefly described in section 4.3.

While the STREAMS mechanism is well-suited for the comparably simple protocol modules of the T.123 hierarchies, the nature of the MCS and GCC providers do not permit these entities to be implemented as STREAMS modules: the MCS and GCC providers potentially have to store large amounts of information about any number of conferences and the associated T.120 application protocol entities meaning that there is no foreseeable upper limit on their consumption of memory resources. This in turn requires them to be able to work with virtual memory that can be paged if the respective provider grows too large; however, paging of memory is not possible with STREAMS drivers and other OS kernel modules. Therefore, the MCS and GCC providers are implemented as separate processes that use message passing as IPC mechanism to communicate with one another as well as to offer their services to T.120 applications. The MCS provider also makes use of the TLI interface of T.123 to communicate with other MCS providers. The details of the message exchanges between the MCS and GCC providers and their respective users are hidden underneath the standardized IMTC API libraries that offer a function-call-oriented interface. The message passing mechanism employed by the MCS and GCC providers as well as the IMTC APIs are sketched in section 4.4.

Finally, the modules implementing the application protocols of the T.120 series are realized as libraries that are linked to the user application. These libraries also offer a function-call-oriented API. In contrast to the MCS and GCC providers, no specific message passing mechanism is required. Rather, the functions exported by the protocol modules constitute the respective APIs. This is not discussed further in this thesis (for detailed information on the implementation of the protocol modules for SI and MBFT and their APIs refer to [Gehrcke 95] and [Kerkhoff 97], respectively).[31]

4.3. Implementation of T.123 Transport Protocol Stacks

All the T.123 protocol hierarchies underneath MCS are implemented as STREAMS modules. The first part of this section gives a brief overview of the STREAMS mechanism to provide the basis for the description of the T.123 implementation in the second part.

4.3.1. The STREAMS Mechanism

The STREAMS mechanism is a powerful mechanism for flexible and efficient implementation of communication services [STREAMS 90]. A *Stream* consists of a *stream head*; zero, one, or more *stream modules*; and (typically) one *stream driver*.[32] Information exchange in a Stream is done by

[31] The T.121 service that combines certain MCS and GCC functions is also provided as a library which is used by the MBFT implementation. For the definition of T.121 API refer to [Birkicht 96b].

[32] In the following, the STREAMS mechanism is shortly referred to as ‘‘STREAMS’’, a particular instance of a stream is written capitalized as ‘‘Stream’’, and the lowercase ‘‘stream’’ is used as a general prefix.

means of messages. Data as well as control messages flow *downstream* from the stream head through the protocol modules to the stream driver and *upstream* from the stream driver through the stream modules to the stream head. Applications interact with a Stream through a well-defined interface at its stream head[33] in order to send and retrieve information as well as to control the behavior of the protocol modules and the device of the stream. The device driver of a Stream implements the interactions with the actual hardware or some virtual device.

The protocol modules (if present) may arbitrarily modify the information flowing up- and downstream in order to implement communication protocols. The upstream and downstream interfaces of protocol modules are well-defined and identical for all protocol modules. The individual protocol modules implement their respective functionality isolated from one another and exchange information with their upstream and downstream neighbors using these very protocol module interfaces. Consequently, the protocol modules in principle do not require any knowledge about their neighbor modules, neither do they have to know whether they are directly connected to the stream head or the stream driver. This transparency enables arbitrary stacking of protocol modules to implement protocol hierarchies, thus allowing flexible implementation of protocol hierarchies with each protocol layer being represented as one (or more) STREAMS protocol module(s) that are then dynamically combined to construct the protocol stack as needed.

The STREAMS mechanism does not only support one-to-one relationships between STREAMS modules; *1:n*, *m:1*, and *m:n* relationships between upstream and downstream elements may be implemented as well. This is achieved by the concept of multiplexers; these are STREAMS drivers that are capable of connecting to more than one upstream and/or downstream neighbors.

As stated before, the STREAMS mechanism defines interfaces for accessing the stream head as well as interfaces and procedures for passing message between stream head, stream modules, and stream drivers. In addition, the STREAMS concept includes memory management functions, prioritized messages, flow control within each stream, and scheduling services for STREAMS modules and drivers.

The STREAMS mechanism is available on most modern UNIX platforms. To accommodate MS Windows operating systems as well, a *STREAMS kernel* providing the entire STREAMS environment fully compatible to UNIX implementations has been developed expressly for these environments [Schindler *et al.* 95].

4.3.2. T.123 Implementation

The overall structure of the T.123 implementation for ISDN, PSTN, and LANs and its interactions with the MCS provider are depicted in figure 4.11. All T.123 protocols are implemented as multiplexers and therefore conceptually had to be realized as streams drivers. For each of the required protocols — X.224/0, SCF (Q.933), LAPF (Q.922), and RFC 1006 — an individual STREAMS driver has been implemented.

[33] This interface includes the typical UNIX functions for driver access — `open()`, `close()`, `read()`, `write()`, and `ioctl()` — and additional functions for message oriented communication with the Stream — `putmsg()`, `getmsg()`, `putpmsg()`, and `getpmsg()`.

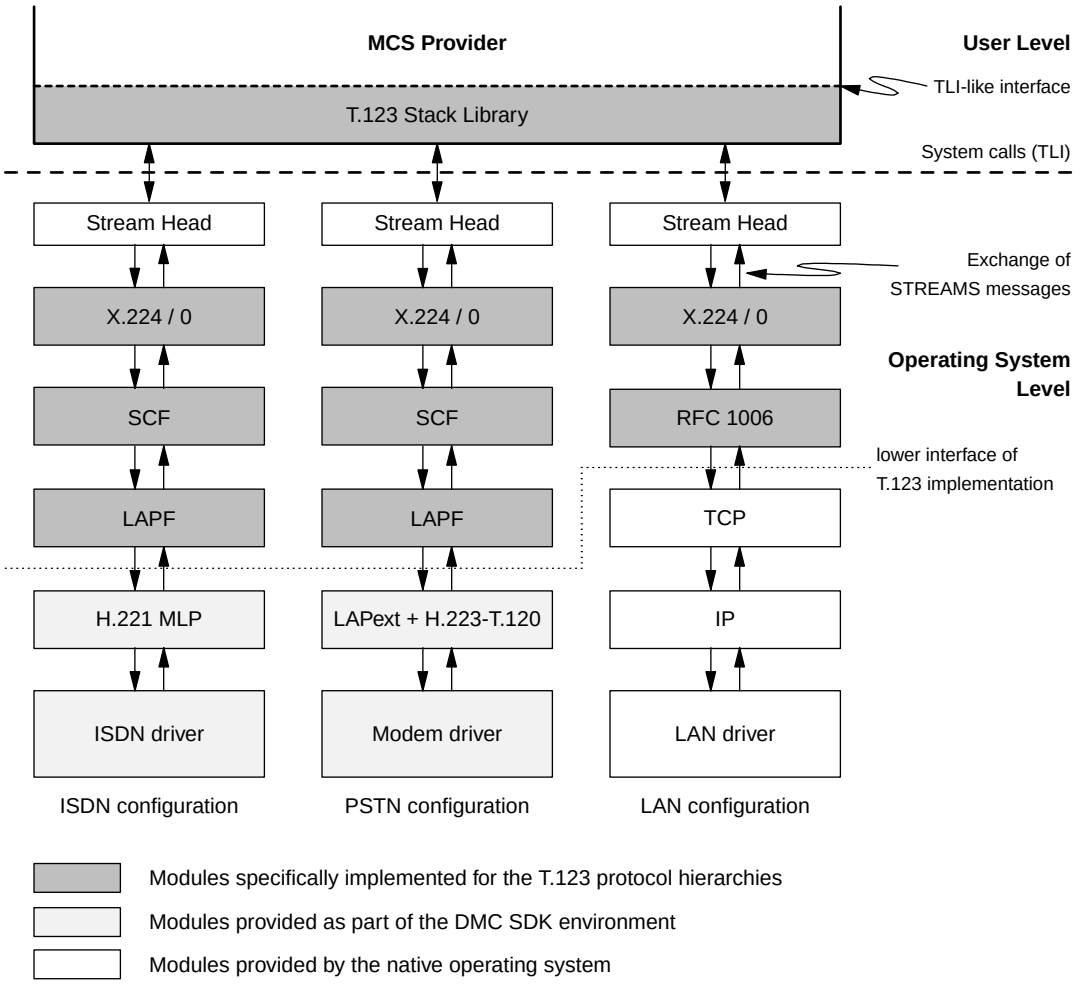


Figure 4.11: T.123 implementation overview

Furthermore, network-specific adaptation drivers are required for ISDN and PSTN — both of which have identical stacks with only slight differences at the Q.922 layer. In ISDN, the *h221mlp driver* provides access to the *Multi-layer protocol* channel of the H.221 multiplexer. For PSTN, a separate driver, the *lapext driver*, performs the Q.922 functions of *octet stuffing*, flags insertion and detection as well as CRC calculation and offers access to a data channel provided by the H.223 multiplexer. These multiplexers then access the ISDN and PSTN (modem) device drivers, respectively. These configurations are the same for UNIX as well as MS Windows platforms.

The configuration is less complicated with the LAN protocol stack. Besides X.224/0 which is common to all three stacks only the TCP framing based on RFC 1006 had to be implemented. TCP and IP modules are always provided by the operating system environment. However, while in UNIX, TCP and IP are implemented using STREAMS so that these modules can be used directly, in the MS Windows environment an adaptation driver (not depicted in figure 4.11) is inserted underneath the RFC 1006 driver that maps the streams interface to the Winsock API.[34]

[34] The Winsock API is the basis for networking in various MS Windows operating systems and is oriented at the BSD sockets interface. The Winsock API is documented in [Winsock 96].

As bottom part of the MCS, the *T.123 stack library* performs all the required interactions for the MCS to set up, configure, and tear down the protocol stacks. The T.123 stack library deals with all issues of address binding, initiation of outgoing and acceptance of incoming connections including configuring the STREAMS drivers as required. The latter includes opening the necessary STREAMS devices and linking them appropriately to build the protocol stacks which are then used by the MCS provider for information exchange. This functionality which is specific to each of the supported networks is performed invisibly to the core of the MCS provider. Overall, the stack library hides the complexities of the interactions with the underlying STREAMS drivers and provides a simple TLI-like interface to the MCS provider core which simplifies porting of the MCS provider — even to operating system environments that do not provide STREAMS. The interface functions offered by the T.123 stack library to the MCS provider are the following:

- The `m_listen_request()` is used to establish a passive connection endpoint for a particular protocol hierarchy and transport address that waits for incoming connections.
- Calling `m_connect_request()` initiates the establishment of a connection to the MCS provider identified by its transport address. The transport protocol hierarchy to be used for this particular connection is also passed as parameter.
- `m_accept_request()` allows a provider that has previously issued an `m_listen_request()` to accept an incoming connection setup request.
- With `m_disconnect_request()`, an established X.224/0 transport connection is closed.
- Finally, `m_data_request()` is used to transmit data across the transport connection; the data is retrieved on the receiving side by means of `m_data_indication()`.

For an encompassing and detailed description of the T.123 implementation (including the implementation of all the streams drivers and the stack library) refer to [Simm 95], for details on the various protocols and the setup, data exchange, and teardown phases refer to [ITU-T Q.933], [ITU-T Q.922], and [ITU-T T.123].

4.4. Implementation of the MCS and GCC Provider APIs

This section describes the implementation of the APIs through which the services of the T.120 infrastructure service providers are accessed by user applications.[37] The MCS and GCC service related functions offered by the APIs are in principle a one-to-one mapping of service primitives onto API calls and events. Furthermore, the APIs also provide means for locating and contacting the providers, setting capability and configuration parameters in the providers as well as retrieving this information, and for flow controlling the entire information exchange through the APIs.

Figure 4.12 gives an overview of the architecture of the APIs of the MCS and GCC provider. The MCS and GCC provider are both implemented as individual processes that provide their respective services through an API that is based on a message-oriented inter-process communication mechanism (and roughly comparable to the concept of *remote procedure calls*). Application processes that make use of MCS and GCC access the respective services through the application programming interfaces that have been standardized by the IMTC (uppermost thick dark line on the left hand side of the figure).

[37] This very interface is used by the GCC service provider to access the MCS services, too.

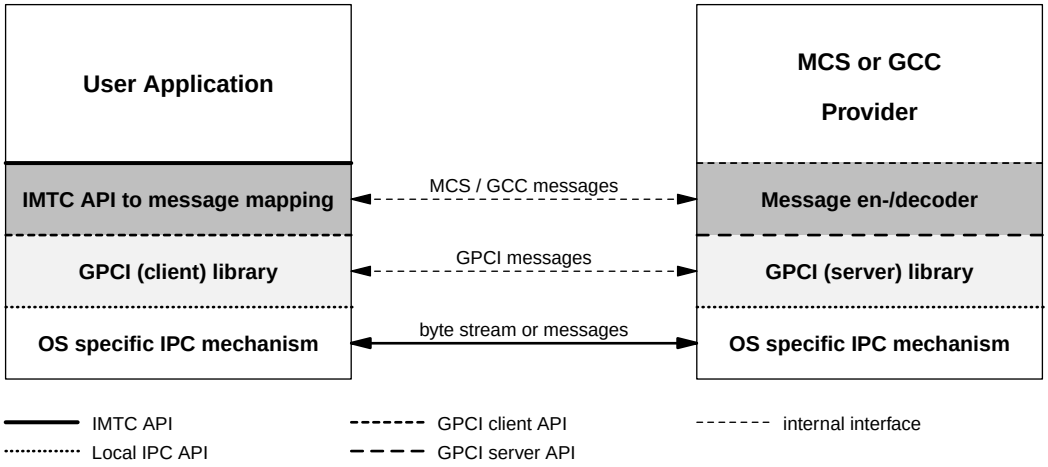


Figure 4.12: Overview of the API architecture

As the IMTC API specifications are function call-oriented but a message passing mechanism is required in the DMC architecture, the respective API implementations map the functions calls onto messages that are exchanged through an underlying message passing interface. The mapping is performed in the dark grey shaded box on the left hand side of the figure, corresponding functions for encoding and decoding the messages are also implemented in the respective service provider (also dark grey shaded).

In order to simplify portability of the APIs, an operating system-independent message passing mechanism is provided as part of the DMC SDK for local inter-process communication [Schindler *et al.* 95]: the *Generic Programming Communication Interface (GPCI)* that is standardized by the ETSI [ETS 300470] (light grey shaded in figure 4.12). The GPCI mechanism itself is implemented based upon the IPC functionality available from the respective operating system. The OS-specific IPC mechanisms are used at the lowest level to actually pass information from the user to the provider process and vice versa.

The following two subsections address the GPCI message passing interface and its implementation (including the mapping onto operating system-specific IPC mechanisms) and the implementation of the IMTC API, respectively.

4.4.1. The GPCI Message Passing Library

The GPCI message passing library is based on the concepts of the Generic Programming Communication Interface (GPCI) defined in [ETS 300470] which provides a model of a message-oriented API for the exchange of arbitrary information.

In the GPCI model, a user entity accesses services of a provider entity via the GPCI mechanism.[38] To do so, the user and the provider entities exchange messages that are termed

[38] The GPCI specification defines three entity types: local applications (LAs) using services provided by others, communication applications (CAs) that use and provide services via GPCI, and provider modules (PMs) that only provide services. This model is somewhat similar to the STREAMS concept, with LAs comparable to applications accessing the stream head, CAs to stream modules, and PMs to streams drivers.

Interface Data Units (IDUs). GPCI defines a total of seven functions to be used for information exchange between a provider and a user entity. All functions marked with a single asterisk (*) have only local effects (i. e. they do not lead to an exchange of IDUs with the service provider), those marked with two asterisks (**) interact with the operating system-specific implementation of the GPCI (but not necessarily with the corresponding provider) in order to provide the desired information:

- `GPCI_ListResources()**` is used by user entities to determine which service providers are available. This implicitly assumes that service providers have a mechanism to make themselves known to some (naming and registration) entity in the system or that other conventions for determining available services do exist.
- `GPCI_Link()` is used by a client to establish a GPCI link to a provider for subsequent information exchange. To contact the provider, this function makes use of the knowledge gained from `GPCI_ListResources()`.
- `GPCI_Unlink()` closes a link between a user and a provider.
- `GPCI_Put()` conveys information encapsulated in an IDU from the user to the provider.
- `GPCI_Get()` is used by the user to actively retrieve an IDU (if available) from the provider.
- `GPCI_Poll()**` provides a means to the user to determine whether or not an IDU is available at the provider for retrieval.
- `GPCI_SetSignal()*` may be used by the user to register a function that is called each time an IDU becomes available at the provider.

The aforementioned functions are essentially sufficient for the conceptual description of the user (or client) side. However, to embrace the provider (server) side in the concept as well, the following functions needed to added:

- `GPCI_Register()**` is used by a provider to register and deregister its services.
- `GPCI_LinkAccept()` and `GPCI_LinkResponse()` are used by the provider to accept (or reject) a link establishment request from the client and to negotiate the link parameters (including the maximum IDU size and the window size for flow control).

To finally provide a complete basis for a specific implementation of a GPCI library that is to be used by service providers and service users, further local management functions — that are operating system specific and hence not contained in the standardized GPCI specification — had to be added:

- `GPCI_Init()*` and `GPCI_Finit()*` are used to initialize and de-initialize the GPCI library, respectively. De-initialization includes closing all links and releasing all system resources.
- `GPCI_Lock()*` and `GPCI_Unlock()*` synchronize access to GPCI resources and thus enable the usage of the IMTC APIs by multithreaded applications.
- `GPCI_ProcessEvent()*` and `GPCI_ProcessEvents()*` are used to handle incoming messages, timeouts, or other (internal) events and call the appropriate functions (previously registered with `GPCI_SetSignal()`.
- `GPCI_GetFdList()` and `GPCI_Dispatch()` are functions specific to the UNIX operating systems for building applications that implement their own main loop and cannot rely on `GPCI_Poll()` but rather make use of the UNIX system calls `poll()` or `select()`.
- `GPCI_SetPostMessage()` is required in the MS Windows environment to select a window to which notifications about events shall be sent by the GPCI library.

A detailed specification of all these functions constituting the GPCI message passing library can be found in [Düwel 95] and [Melcher 96].

For an actual implementation of the GPCI library, the GPCI functions at some point have to make use of inter-process communication mechanisms provided by the operating systems. How the GPCI functions are mapped to various operating system environments is outlined in [Braun 95a] and [Braun 95b]. In the context of this thesis, the GPCI message passing library is implemented for the UNIX and the MS Windows operating systems. The following brief description gives an overview of the implementation for the UNIX operating system and points out the differences to the implementation for MS Windows.

In the UNIX operating system environment, the GPCI message exchange mechanism is mapped onto communication via UNIX domain sockets that are available on virtually all UNIX platforms (similar IPC concepts may be used as well). Register and deregister requests are mapped onto creation and deletion of a passive listening socket using the name of the service as the socket address in a dedicated directory of the UNIX file system. To list available resources, the names of the socket entries from that directory are retrieved, and afterwards additional information about each service may be queried from the respective providers. Thus, the providers use the service names as rendezvous points to which the user entities connect by using the `connect()` system call. As UNIX domain sockets typically provide a byte-stream oriented (rather than a packet-oriented) data transmission service but GPCI requires identifiable messages (the IDUs) to be conveyed, a simple IDU structure is defined for all message exchanges[39] along with message structures for resource information and link parameters. Transmission and reception of IDUs is performed with `read()` and `write()`, respectively. Notifications about events on links (such as incoming connections, availability of data, or closed connections) are recognized by using the standard UNIX system calls `poll()` or `select()`. With the function `GPCI_GetFdList()`, the GPCI library supplies the GPCI file descriptors (and the respective events) for which `poll()` shall be called. Depending on the program structure `GPCI_Dispatch()`, `GPCI_ProcessEvents()`, or `GPCI_ProcessEvent()` have to be called subsequently by the application using the GPCI library to initiate the actual processing of events. Upon each call to one of the latter three functions, internal GPCI data structures are updated as necessary and the previously registered callback functions are called as appropriate. If no longer needed, GPCI links are destroyed using the `close()` system call.

In the MS Windows implementation of the GPCI library, a dedicated message server is introduced to perform all the GPCI functionality. That is, GPCI user and provider communicate via the message server rather than directly. The GPCI message server provides the GPCI naming services, link parameter negotiation, and performs the actual message forwarding including queuing and flow control. An internal message format — roughly similar to the one defined for UNIX — is used to preserve message boundaries across the transmission. The final major difference to the UNIX implementation is that notifications about events are implemented by posting Windows messages to (invisible) windows of the user applications and the providers that have been registered previously with the GPCI message server.

[39] This structure essentially consists of a *message length* field for preserving message boundaries and a *message type* field identifying the message contained in the IDU.

Further details on the implementations for MS Windows as well as for UNIX can be found in [Schindler *et al.* 95], [Braun 95b] and [Ott 95b].

4.4.2. The Function Call Interfaces of the IMTC

The function call oriented APIs for the MCS and GCC providers are defined in [Johnstone 96] and [Braun 96a], respectively. These specifications mainly define functions that provide access to the standardized services of MCS and GCC. Request and response service primitives are generally mapped one-to-one onto function calls to the API while indications and confirmations are mapped one-to-one to event notifications. In these APIs, an event is a data structure consisting of an identifier for the event and an event-specific set of parameters. The parameters of function calls and events are largely equivalent to the parameters of the corresponding service primitives, except that the API requires a few additional parameters for local management, e. g. to distinguish several (instances of the same) service user(s) on the provider side. A set of additional functions is provided by each API to control the relationship between each service user and service provider: to establish a (logical) link between them and destroy it later, to gain information about the provider, to perform local flow control, etc.

The MCS and GCC APIs as defined by the IMTC are tailored to an environment in which service providers are implemented as shared libraries,[40] but they are not well suited for providers being implemented as separate processes that are accessed through message passing mechanisms. The remainder of this subsection describes, by which mechanisms the function-call-oriented IMTC APIs are mapped to message-oriented APIs that employ GPCI underneath. The implemented mapping mechanism does neither cause any syntactic or semantic incompatibilities to the IMTC specifications nor does it — despite the message exchanges — incur any noticeable degradation in performance:

- The information exchange between the API library and the provider requires inter-process communication which is performed using the GPCI message passing library. Both providers register services for their respective Control and User Service Access Points with the GPCI. Service users learn about the provider addresses by means of GPCI functions and then establish GPCI links to the providers as needed. When a T.120 application attaches to a provider — to a User MCSAP or GCCSAP as well as to the respective Control SAPs — a new GPCI link is created. Each GPCI link is multiplexed at the API layer above GPCI to provide several virtual channels: the MCS API requires independent transmission of up to four priorities and the GCC API has to distinguish between asynchronous and synchronous requests to avoid internal deadlocks. Therefore, the MCS and GCC APIs implement independent flow control mechanisms to their respective providers on top of GPCI using a simple windowing mechanism with separate message queues for the different priorities and request types.

- Initiating MCS and GCC services and receiving notifications from the providers requires that API functions as well as event types and their respective parameters are mapped onto provider-specific messages for transmission from the API to the provider and vice versa. For the MCS and GCC APIs, the required message formats are defined in [Ott 95c] and in [Birkicht 96a] using the ASN.1 syntax notation [ITU-T X.208]. The message encoding is

[40] This is e. g. reflected in their function-call-oriented nature.

specified in [Ott 95c]. In essence, messages consist of all the parameters found in the corresponding function calls and events of the IMTC API plus a few management parameters. The management parameters include the function or event type, a message sequence number for matching local user requests to local provider responses (and vice versa), the virtual channel identifier for this message, and a channel-specific acknowledgment number for flow control. The encoded MCS and GCC messages are transmitted as the data portion of GPCI IDUs using the GPCI transmission and reception functions.

- Some of the API calls require synchronous information retrieval from the provider which incurs two message transfers, from the API library to the provider and back. In order to save the overhead of a round trip of messages, the API library stores part of the state information it obtains from the provider so that the API library is able to provide the return values of certain API calls without having to query the provider. However, this approach is not followed for API calls for which the amount of state information to be stored may get arbitrary large and/or for the API calls that are expected to happen only infrequently. For such calls, the API library implements a mechanism that allows to send a request message and synchronously await the response.[41] While for most of the MCS API calls API-local state information is sufficient, the (much more complex) GCC API relies on the synchronous request mechanism for a number of calls.

Further details about the implementation of the MCS API library can be found in [Melcher 96], for additional information on the GCC API library refer to [Radig 96].

4.5. Summary

This chapter has introduced the ITU-T T.120 series of recommendations as the standardized basis for the implementation of the Multipoint Communication Layer. The conceptual considerations about a multipoint communication infrastructure described in chapter three have been linked to those standards based upon which the design and implementation efforts described in this thesis have been carried out.

Furthermore, the context of the implementation efforts carried out for this thesis has been introduced: the software development kit for desktop multimedia conferencing systems of the EURO.VISION system. The DMC SDK architecture with its functional modules and APIs has been presented. Those interfaces important to the implementation and integration of the T.120-based multipoint communication infrastructure have been identified and their implementation concepts have been outlined. Finally, some aspects of the implementations of two of these interfaces — that are relevant but not central to the focus of this thesis — have been briefly described: the transport layer API (including the implementation of the underlying T.123 protocol hierarchy) and the MCS and GCC service provider APIs.

---

[41] In principle, after having sent a synchronous request to the provider, the API library examines each incoming message from the provider and checks whether this is the response to the synchronous request. In this case, the synchronous call returns the received results. Otherwise, the API library queues the message internally and continues waiting for the response. The queued messages are indicated to the application as soon as the synchronous call has completed and the application has returned control to the API library.

Overall, this chapter has provided an overview of where the MCS and GCC providers are embedded in the DMC SDK and how they interact with its other components. The remainder of this thesis addresses the implementation of the MCS and GCC provider in detail.

5

Implementation of the MCS Provider

This chapter addresses the design and the implementation of the Multipoint Communication Service (MCS) provider. The first section reviews the ITU-T recommendations specifying the MCS (T.122 and T.125) in more detail in order to provide the basis for subsequent presentation of the implementation. The second section gives the overall outline of the MCS provider structure the two main components of which are described in the subsequent two sections. A brief summary concludes this chapter.

5.1. The ITU-T Recommendations T.122 and T.125

The overview of the Multipoint Communication Service specification given in this section is divided into two parts: firstly, the MCS services (T.122) are briefly described and afterwards the MCS protocol characteristics (T.125) to provide these services are presented. The latter are also of importance for chapter six that deals with the extensions to the MCS protocol for multicast-capable networks.

5.1.1. MCS Service Definition

As already indicated in the overview in section 4.1.2, the Multipoint Communication Service provides four groups of services: domain management, channel management, data transmission, and token management. The individual services comprising these groups are briefly described in the following.

The domain management services are used to create and destroy domains as well as to attach groupware applications to and detach them from a particular domain. Creation of a domain is done by setting up an MCS connection between MCS providers (`MCS-CONNECT-PROVIDER`) for this particular domain identified through a domain selector.[1] All MCS connections to an MCS

[1] Domains may also be created locally at an MCS provider without any interactions with other MCS providers [Johnstone 96]. However, as such actions do not involve any peer-to-peer communication, they are considered local matter and not defined in the ITU-T recommendations.

provider that are established with the same domain selector are considered to belong to the same domain, and these connections constitute (part of) the MCS hierarchy for this domain. An MCS provider drops out of a domain if either the provider itself or its (superior) peer disconnects the MCS connection belonging to that domain (MCS-DISCONNECT-PROVIDER). Once the domain is established at the MCS provider of a node, MCS users may attach to the domain using MCS-ATTACH-USER and detach again by means of MCS-DETACH-USER.

The channel management services of MCS allow to control the membership in channels. An MCS user becomes member of a particular channel by using the MCS-CHANNEL-JOIN service provided that the requesting MCS user application is allowed to do so.[2] This service is also used to allocate a new dynamic multicast channel which is created and joined atomically. MCS-CHANNEL-LEAVE is used by an MCS user to give up membership in a channel. A private channel is created by issuing an MCS-CHANNEL-CONVENE causing the issuer to become the ''owner'' of the newly allocated channel. No other MCS user is allowed to join this channel unless invited by the owner through MCS-CHANNEL-ADMIT. The owner of a private channel may exclude previously admitted members from the channel by means of MCS-CHANNEL-EXPEL, and it destroys the private channel with MCS-CHANNEL-DISBAND if the channel is no longer needed (all members will then automatically be expelled). The channel management services of the Multipoint Communication Service permit configuring MCS channels for information distribution to exactly one MCS user application (unicast), any subgroup of applications (multicast), or all MCS users in a domain (broadcast).

The data transfer services of the MCS perform the actual transmission of information. MCS provides reliable (i. e. error controlled) and flow controlled transmission of information as identifiable units from a sender to one or more receivers. Each information unit is addressed to a particular channel, thereby selecting the recipient(s), and is associated with a certain priority, thereby defining the T.123 transport connection to be utilized for its transmission. Information transmission is performed with either of two services, MCS-SEND-DATA and MCS-UNIFORM-SEND-DATA, that provide different ordering properties. MCS-SEND-DATA delivers all information units from the same source in the same order to all recipients (per-source ordering) while MCS-UNIFORM-SEND-DATA causes information units from all senders to be delivered in the same sequence to all receivers (global ordering). These ordering properties only apply to information units addressed to the same channel and sent at the same priority. Otherwise, no ordering relation is defined.

Finally, MCS provides a set of token management services to simplify synchronization between MCS users. A token is identified by a number that is unique within a domain. No predefined meaning is associated with a token. MCS tokens are either free (unused), owned exclusively by a single MCS user, or are shared by one or more MCS users. MCS users may request to obtain tokens exclusively (MCS-TOKEN-GRAB) or sharable with other MCS users (MCS-TOKEN-INHIBIT). The MCS-TOKEN-RELEASE service is used to release a token: if the exclusive owner or all shared owners release a token, it returns to the ''free'' state. With MCS-TOKEN-TEST an MCS user can determine the current state of a token. If an MCS user wants to grab or inhibit a token that is grabbed, it may use MCS-TOKEN-PLEASE to request it from its current owner who may respond with a MCS-TOKEN-GIVE to forward the token to the requester. If an MCS user

[2] An application is not allowed to join either a private channel to which it has not been invited or a single member channel that is not its own.

wants to grab a token that is inhibited, it also may use MCS-TOKEN-PLEASE to request it from its current owners. If all of these respond with an MCS-TOKEN-GIVE the token ownership is transferred to the requester, otherwise the token remains inhibited (however, those who have responded with a MCS-TOKEN-GIVE are considered to have released the token). Finally note that the standard does not require a recipient of an MCS-TOKEN-PLEASE request to respond at all. Those who do not answer are treated as if they had ''rejected'' the MCS-TOKEN-PLEASE request.

5.1.2. MCS Protocol Characteristics

The MCS protocol [ITU-T T.125] defines the interactions between MCS providers for setup, operation, and teardown of a domain. MCS providers exchange MCS protocol data units (MCSPDUs) using the underlying transport connections (the T-DATA service of the transport layer). MCSPDUs are defined and encoded following the Abstract Syntax Notation One (ASN.1).[3]

In each MCS domain, the Top MCS Provider is the central entity that maintains all relevant state information about a domain, assigns unique identifiers, and arbitrates access to resources. Each of the other MCS providers keeps state information about the subtree below it in order to perform data forwarding, routing of control MCSPDUs, and handling failures of nodes in the subtree.

The concept of the Top MCS Provider avoids protocol overhead to deal with race conditions that occur in distributed systems without a central coordinator. This allows the MCS protocol to be kept relatively simple: most service requests are mapped onto client-server-like interactions between the requesting MCS provider and typically the Top MCS Provider.

In essence, three different classes of interactions between MCS providers are found in the MCS protocol specification: establishment of an MCS connection to create or extend a domain, initiation and processing of an arbitrary management service (MCS user attachment, channel management, or token management), and data transmission. These three classes of interactions are described in the following to give an overview of the way the MCS protocol works.[4]

Establishment of an MCS Connection

Setting up an MCS connection is a point-to-point interaction between the initiator of the connection establishment (the *calling MCS provider* or *caller*) and the provider to which the setup request is directed (the *called MCS provider* or *callee*). At the calling MCS provider, the connection setup is initiated by the controlling entity issuing an MCS-CONNECT-PROVIDER.request at the Control MCSAP. The calling MCS provider then issues a T-CONNECT.request to the transport layer in order to establish a first transport connection to the called MCS provider.[5] This first connection is referred to as the *initial connection*. As soon as the initial transport connection is

[3] ASN.1 is defined in the ITU-T recommendations [ITU-T X.680], [ITU-T X.681], [ITU-T X.682], [ITU-T X.683], [ITU-T X.690], and [ITU-T X.691].

[4] Note that several details of the MCS protocol are deliberately not included in the following description in order to keep the presentation simple.

[5] The transport layer takes whatever actions are necessary to establish a transport connection for the respective network. Details of this procedure are not of interest to the MCS protocol. If multiple networks and the required T.123 protocol stacks are available, the MCS provider must be able to specify (e. g. by means of a network address type) over which network the transport connection shall be established.

established, the calling MCS provider sends a `Connect-Initial` MCSPDU to its peer. This MCSPDU contains the proposed domain parameters (minimum, maximum, and target values) and some other information. Out of the information conveyed in the initial MCSPDU, the following parameters are of particular importance for the connection setup:

- The *MCS protocol version* is part of the domain parameters and is used to determine which common protocol versions both MCS providers support and select one for this connection.
- The *number of MCS priorities* of the domain is also part of the domain parameters and indicates how many transport connections shall be used for the MCS connection.
- The *upward-flag* indicates which of the two MCS providers shall be the superior node in the connection hierarchy of the MCS domain.
- *User data* are included to allow higher protocol layers (i. e. the GCC) to convey information piggybacked onto the MCS connection setup request based upon which the called MCS provider's controlling entity can decide whether or not to accept the MCS connection (e. g. authentication information).

Other domain parameters include the maximum size of an MCSPDU that may be transmitted in this domain; the number of channels, tokens, and MCS user ids supported; and the minimum throughput required on each connection of the domain.

Each domain parameter is negotiated during setup of the initial transport connection: the calling MCS provider proposes a set of values for the domain parameters in the `Connect-Initial` MCSPDU; the called MCS provider may adjust these values within a range specified by the caller. If (the range for) one of these parameters is not acceptable to the callee, it rejects the MCS connection setup by sending an appropriate `Connect-Response` MCSPDU. Otherwise, the MCS provider indicates the connection setup at its Control MCSAP. The node controller at the callee then decides based upon the domain parameters, the caller's address, and the user data to accept the connection or not and responds with an appropriate `MCS-CONNECT-PROVIDER.response` that may also adjust the domain parameters (again within the range offered by the caller) and leads to a `Connect-Response` MCSPDU being sent back to the calling MCS provider. If the callee refuses the MCS connection, the transport connection is disconnected, otherwise the setup proceeds as follows.

Upon reception of the `Connect-Response` MCSPDU, the caller initiates the establishment of as many additional transport connections as are requested by the number of MCS priorities negotiated during the initial handshake. On each of these additional transport connections, the caller transmits a `Connect-Additional` MCSPDU that identifies to which initial connection (and thus which MCS connection) the respective additional transport connection belongs and which priority it shall be used for. The called MCS provider acknowledges receipt and acceptance of each of the `Connect-Additional` PDUs by responding with a `Connect-Result` on the respective additional transport connection. As soon as all `Connect-Result` MCSPDUs are received by the caller and if all connections are accepted by the callee, the MCS connection setup phase is completed and an `MCS-CONNECT-PROVIDER.confirm` is generated at the Control MCSAP of the calling MCS provider.

If the called MCS provider refuses the initial or any of the additional connections or any transport connection establishment fails, the entire MCS connection setup has failed and this is signaled

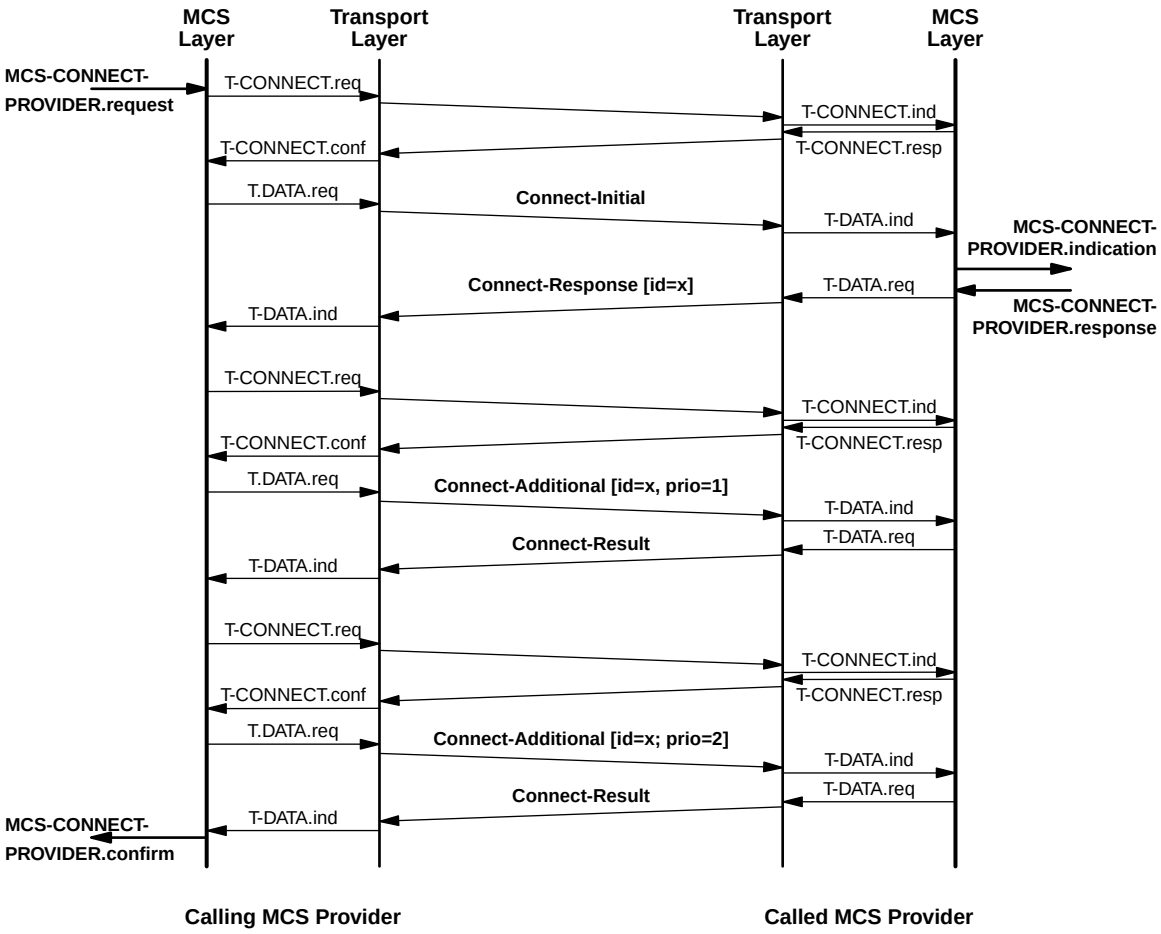


Figure 5.1: Establishment of an MCS connection

through an `MCS-CONNECT-PROVIDER.confirm` at the Control MCSAP of caller specifying the failure reason. Figure 5.1 illustrates the connection setup phase.

After the MCS connection setup phase is completed, further actions may be required to complete the setup of the MCS domain. Depending on the roles the two MCS providers have played prior to the connection setup with respect to the domain for which the MCS connection was established, three different cases may be distinguished:

- If the domain did not exist at the MCS provider that is connected as the subordinate node, no specific actions are needed.

- If the domain did not exist at the MCS provider that is connected as the superior node, it assumes the Top MCS Provider role and has to learn about the resources already in use in the previously existing part of this domain. To achieve this, an update phase is entered during which MCSPDUs are exchanged that inform the new Top MCS Provider of the assigned user ids, channels, and tokens as well as their respective status.

- If the domain existed at both MCS providers now interconnected, the MCS provider that joins as the subordinate loses its Top MCS Provider role. In both domains, user ids, channels, and

tokens have been allocated independently so that clashes in the identifiers may occur after merging these two domains. Independent of such conflicts, the Top MCS Provider of the joint domain has to gain information about the MCS resources in use. Again, an update phase is initiated during which the Top MCS Provider learns about all the resources in use. Afterwards, duplicate user, channel, and token identifiers are invalidated in the domain of which the subordinate node previously was Top MCS Provider.

In any case, the domain height (i. e. the depth of the tree of MCS providers) of the resulting domain is checked against the domain parameters and the domain is verified to be hierarchical and not to contain loops.

Exchange of Control MCSPDUs

The Multipoint Communication Service employs a centralized model for maintaining domain-global state information, arbitrating shared resources, and validating requests issued by MCS users. Therefore, the Top MCS Provider as the central entity of a domain is involved in the processing of virtually any exchange of MCSPDUs to satisfy a service request unless the outcome of a request can be decided at another MCS provider closer to the requester without harming integrity of the MCS domain state. With the respect to the involvement of the Top MCS Provider, two different types of MCS service requests can be distinguished both of which are described in the following:

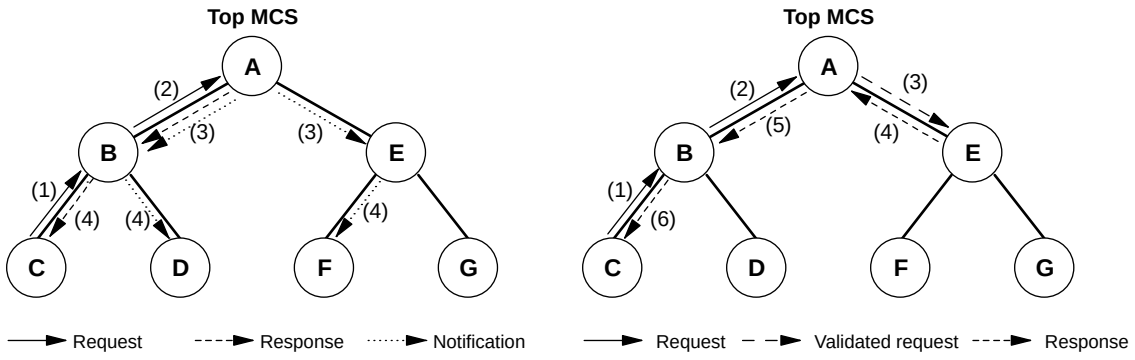
a) *Non-directed service requests* are used by MCS user applications to request a service from an MCS domain as a whole rather than from a particular peer entity. Typically, the Top MCS provider handles such a request but in some cases other nodes may possess sufficient information to process the request, too, so that forwarding the corresponding MCSPDU to the Top MCS provider is not necessary. Whenever central arbitration of resources is needed, however, the Top MCS provider processes the request. Non-directed requests may be further categorized according to two criteria:
   – whether or not the service requester expects a response (confirmed vs. non-confirmed service) and
   – whether or not the request leads to state changes that require notifications being sent to other MCS providers; notifications are only generated by the Top MCS Provider.

   For example, `MCS-ATTACH-USER` and `MCS-CHANNEL-JOIN`, both require a confirmation, but only the response for `MCS-ATTACH-USER` has to come from the Top MCS Provider. `MCS-CHANNEL-LEAVE` is an example for a request that does not require a confirmation at all. Notifications are generated for neither of these three services; examples for requests that induce notifications are `MCS-DETACH-USER` and `MCS-CHANNEL-DISBAND` (neither of which is confirmed). Figure 5.2a depicts the flow of MCSPDUs for `MCS-CHANNEL-DISBAND`.

b) *Directed service requests* are destined to a particular (set of) MCS user application(s). The corresponding MCSPDUs are always transmitted to the Top MCS Provider which validates the request with respect to the permissions of the issuer and then forwards the request MCSPDU to the destination(s) — or the Top MCS Provider refuses the request if the request is not permissible. Three types of directed service requests can be distinguished:

– requests for which the protocol requires a confirmation from the addressed MCS user (e. g. `MCS-TOKEN-GIVE` requires the recipient to either accept or refuse the offered token);
– services that may but need not induce a separate service being invoked by the recipient of the request (e. g. `MCS-CHANNEL-ADMIT` invites the recipient to issue an `MCS-CHANNEL-JOIN` but the recipient need not do so); and
– services that are indicated to the recipient for information only without any need or possibility to react (e. g. `MCS-CHANNEL-EXPEL`).

Figure 5.2b illustrates the flow of MCSPDUs for the `MCS-TOKEN-GIVE` service.



a) MCS-CHANNEL-DISBAND.request from node C processed by the Top MCS provider that generates a response and sends a notification to the nodes D, E, and F at which MCS users are attached to the channel.

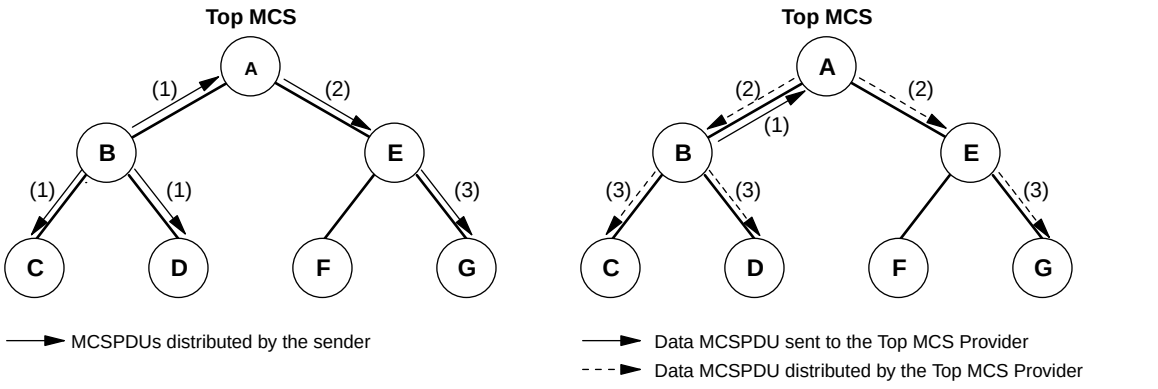
b) MCSPDU for a MCS-TOKEN-GIVE sent from node C to node E with a response sent back from E to C

Figure 5.2: Typical exchange of control MCSPDUs

Data Transmission

Data transmission is done per channel and per priority. The channel to which an information unit is addressed defines the recipients while the priority specifies which transport connection of each MCS connection is used to forward the data MCSPDU carrying the information unit. A data MCSPDU addressed to a certain channel is forwarded to all MCS providers that have at least one MCS user as a member of this channel.

Each MCS provider within a domain keeps track of which of its local MCS user applications is member in which channels. Furthermore, for each of their MCS connections to subordinate MCS providers, a non-leaf MCS provider maintains a list of channels that have members in the subtree rooted at the respective subordinate. From this information, the MCS provider knows for which channels it needs to receive information, to which locally attached MCS users it has to deliver data addressed to a particular channel, and to which subordinates to forward information of a given channel. This knowledge is established when processing and forwarding channel management MCSPDUs. Performing selective forwarding avoids that data MCSPDUs are unnecessarily duplicated and sent over MCS connections that do not lead to any recipients.



a) MCSPDUs for MCS-SEND-DATA initiated at node B and addressed to a channel of which MCS users at nodes C, D, E, and G are members; the MCSPDUs are distributed on the shortest path to all nodes.

b) MCSPDUs for MCS-UNIFORM-SEND-DATA initiated at node B and addressed to a channel of which MCS users at nodes C, D, E, and G are members; the MCSPDUs are first sent to the Top MCS Provider which then disseminates them to all recipients

Figure 5.3: Data transmission with MCS

The MCS provides two services for data transmission in a domain. `MCS-SEND-DATA` is used to distribute information units on the shortest path from the sender to the recipients. The corresponding data MCSPDU traverses each MCS connection at most once (figure 5.3a). All information units originated by the same MCS user on the same channel and priority are delivered to all members of the channel in the order they were transmitted. The `MCS-UNIFORM-SEND-DATA` is used to achieve globally ordered data transmission: the MCS provider of the sending MCS user transmits the corresponding MCSPDU upwards to the Top MCS Provider which then initiates dissemination of the data MCSPDU to all recipients. The Top MCS Provider ensures that the information received for the same channel and priority from multiple senders is transmitted to all receivers in the same order (figure 5.3b).

5.2. Structure of the MCS Provider Implementation

As already indicated in the previous chapter, the MCS provider is implemented as a separate process. The MCS provider process uses the TLI interface to access the T.123 protocol stack realized as STREAMS drivers, and provides its service to the GCC provider and the user applications via the GPCI message passing mechanism (which is concealed on the client side by the IMTC MCS API). The overall MCS structure is independent of the underlying operating system, and employing the two standardized communication mechanisms allows to keep the interactions with other components of the T.120 infrastructure OS-independent, too. The sole parts of the MCS provider differing significantly between OS platforms are its `main()` function doing the system-specific process initializations and the main loop in which incoming events — signaling and control of which is done in an OS-specific manner — are recognized and their processing is initiated.

Figure 5.4 gives an overview of the internal structure of the MCS provider: it consists of a communicator layer and a connection layer. The communicator layer implements the MCS protocol and is entirely independent of the operating system. The connection layer, which provides the abstraction from the specifics of the operating system and the communication libraries in use, contains the OS-specific `ConnectionManager` (that implements the main loop) as well as the

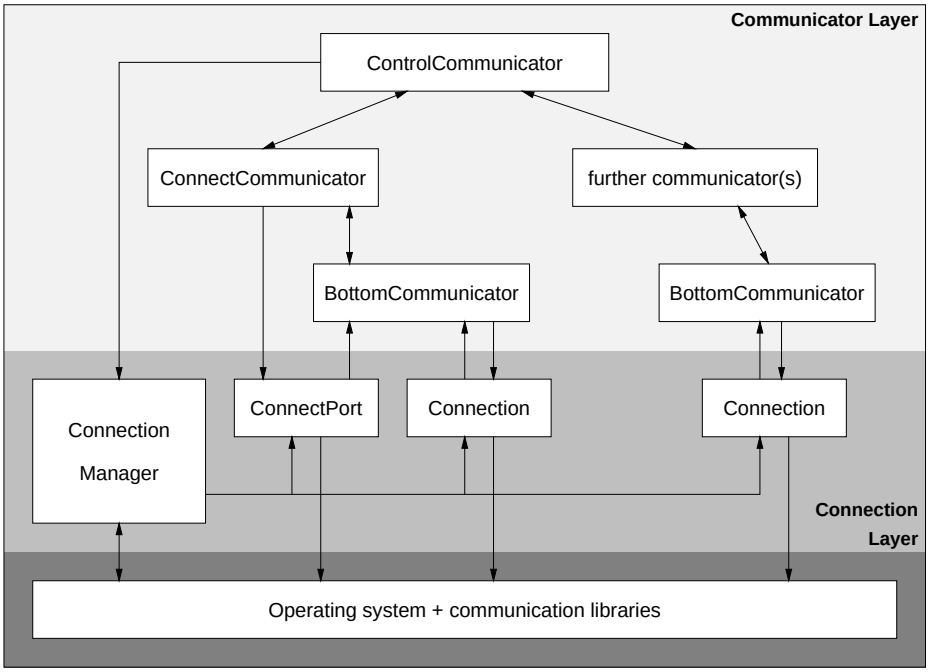


Figure 5.4: Structure of the MCS provider

T.123 stack library. All the MCS provider functionality is implemented by means of object classes for these two layers so that the MCS `main()` function does nothing but instantiate the respective objects and then call the central processing function repeatedly until the MCS provider is to terminate:

Figure 5.5 shows the essentials of the MCS main function which includes the main loop: after some initializations, the `ConnectionManager` and the `MCSProviderController` communicator (both of which are described in the subsequent sections) are created. Afterwards, the function `MCSGetDefaultLocalPorts()` is used to obtain the names to be used for registering the Control and the User MCSAP with the GPCI system (these service names are defined in an initialization file). Subsequently the provider controller is initialized which leads to the `ConnectPorts` for the local service access points being created.[6] Finally, the main loop of the MCS provider is entered which calls `waitForEvent()`. The parameter of `-1` as a timeout indicates that `waitForEvent()` shall not return because of a timeout. This means that the function waits for one or more events to occur, processes them, and then returns. If either the MCS provider shall be shut down or an unrecoverable error has occurred during processing, the variable `all_done` will be set causing the main loop to terminate. The remainder of the `main()` function cleans up the resources utilized by the MCS provider and then exits.

The following two sections describe the implementation structure of the two layers of the MCS provider and the C++ objects used to realize them.

[6] Note that the controlling entity of the MCS (i. e. the GCC) is responsible for creating `TransportConnectPorts` for passively awaiting T.123 connections so that no action in this respect is taken automatically by the MCS provider.

```
UnixConnManager*          mcs_manager;
MCSProviderController*    mcs_controller;
int                       all_done;

void main (int argc, char *argv [])
{
       int    n_ports;
       char   **ports;

       // do some local initializations
       all_done = FALSE;
       ...

       // create the initial internal provider structure
       mcs_manager         = new UnixConnManager ();
       mcs_controller      = new MCSProviderController (mcs_manager);

       if ((mcs_manager == NULL) || (mcs_controller == NULL))
              MCSExit (-1);

       // bind to local addresses for Control and User MCSAPs
       n_ports = MCSGetDefaultLocalPorts (&ports);
       if (mcs_controller->initialize (n_ports, ports) == NULL)
              MCSExit (-1);

       // the main loop of the MCS
       while (!all_done)
              mcs_manager->waitForEvent (-1);

       // orderly release all data structures
       delete mcs_controller;
       delete mcs_manager;
       exit (0);
}
```

Figure 5.5: Main function of the UNIX MCS provider

5.3. The Connection Layer

The connection layer of the MCS is designed to provide an abstraction from specific underlying communication mechanisms and their characteristics as well as from operating system specifics. Thus the task of the connection layer is to allow a portable implementation of the MCS protocol engine. The connection layer of the MCS contains three different types of objects:

- `Connection` objects or short `Connections`[7] relay information from the communicator layer to the (OS or library-specific) communication interfaces for active connections of the MCS provider (to user applications attached to User MCSAP, to the GCC provider attached to the Control MCSAP, or to other MCS providers connected via T.123 transport connections);

[7] In order to avoid confusion: when using the term ‘‘`connection`’’ typeset in Courier font, a connection ‘‘object’’ as found at the connection layer is meant; otherwise, the term refers to some transport or IPC connections.

- `ConnectPorts` passively listen for any incoming connections and notify the communicator layer about such events;
- The `ConnectionManager` implements the main loop of the MCS provider: it keeps track of all transport and local IPC connections and notifies the corresponding connection layer objects about events that have occurred as well as about elapsed timers.

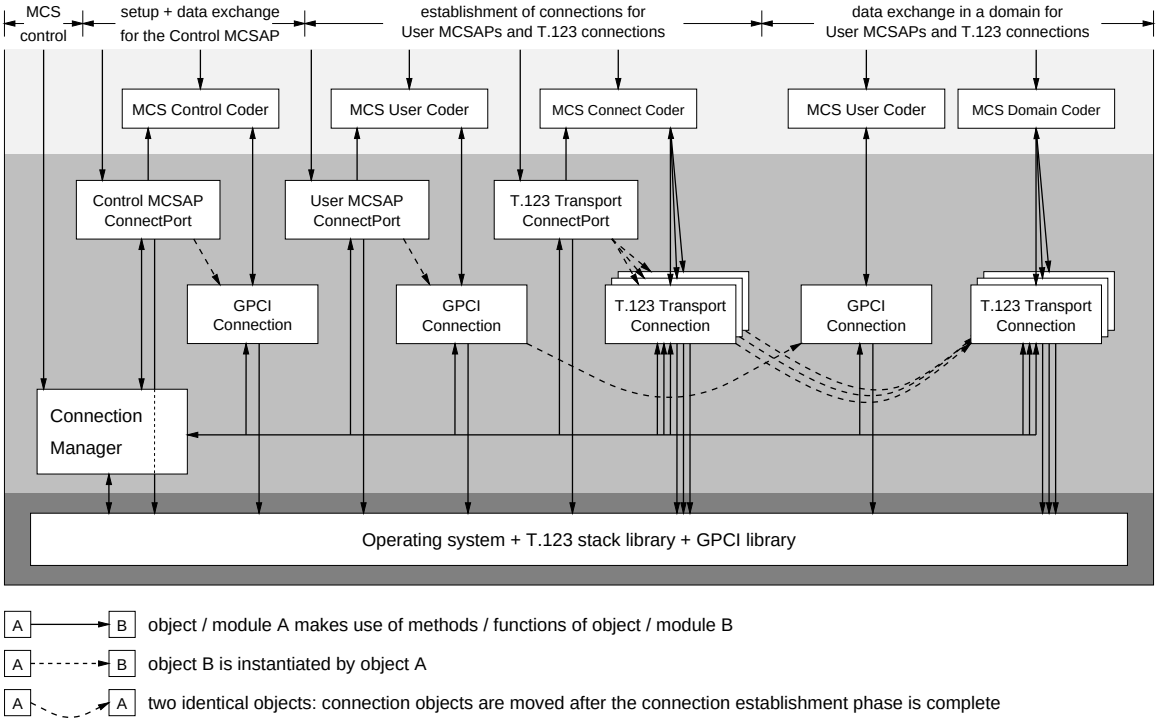


Figure 5.6: Overview of the connection layer

Figure 5.6 depicts all object types used in the MCS provider and also shows their usage relationships. The `ConnectionManager` acts as the central point of control that distributes event notifications to the other objects and thus initiates any kind of processing within the MCS provider. Three types of `ConnectPorts` are used to wait for processes attaching to Control and User MCSAP as well as for incoming T.123 transport connections from other providers. Finally, two types of `ConnectionObjects` handle active establishment and teardown of connections as well as data transfer. The `GPCIConnection` provides access to the GPCI library for local message exchange with entities that are attached to the Control and User MCSAPs while the T.123 `TransportConnection` forms the interface to the T.123 protocol stacks. The internals of these object types are described in the following subsections.

Connections

A `Connection` serves as abstract communication interface to the communicator layer. As illustrated in figure 5.7, a connection object consists of three parts: its abstract interface definition that is common to all `Connections`, a `ConnectionInterface` object to access arbitrary underlying communication mechanisms, and a `ConnectionHandler` object:

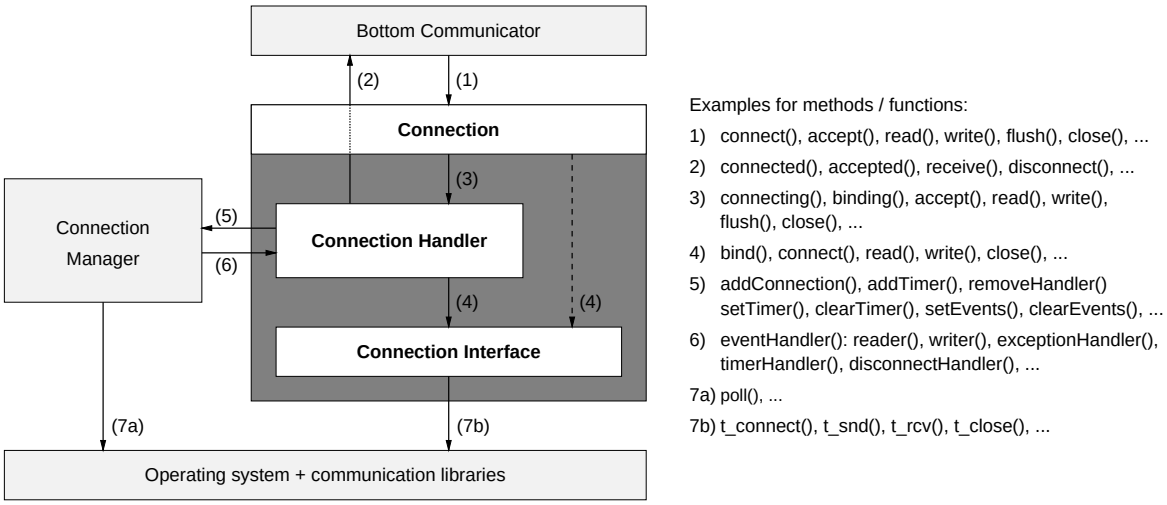


Figure 5.7: Structure of an MCS connection object

- The `Connection` object interacts with the `BottomCommunicator` — a dedicated object class of the communicator layer for interactions with the connection layer: the `Connection` object signals events by calling methods of the `BottomCommunicator`; the `BottomCommunicator` accesses any type of underlying communication service through the uniform methods of the `Connection` object which forwards the requests to the other two objects for processing as appropriate: if the function cannot be directly mapped onto a method of the `ConnectionInterface` — either because no corresponding method is available or because further actions (protocol processing, buffering, etc.) are required — the respective method of the `ConnectionHandler` is invoked (this is done for virtually all methods of all `Connections`). Otherwise, the corresponding methods of the `ConnectionInterface` are called directly.

- The `ConnectionInterface` object of a `Connection` provides a well-defined interface for accessing functions of the operating system or of communication libraries. The assumption underlying this design concept is that all communication services potentially used by MCS provide roughly the same functionality, but possibly by means of different system or library calls. It is the task of the `ConnectionInterface` object to make the communication services available through a set of common methods by internally mapping these methods onto the corresponding (sequences of) API calls to the operating system or the communication library.

- The `ConnectionHandler` is the core component of the `Connection` as it performs all the necessary functions — buffering, conversions, protocol processing, etc. — required to offer the communication service expected by the `BottomCommunicator` on top of the underlying `ConnectionInterface` and its associated (T.123 transport or IPC) connection.

  Each `ConnectionHandler` offers a uniform interface to the `Connection` object through which the latter invokes methods of the former in order to execute requests of the `BottomCommunicator`. Typically, the `ConnectionHandler` performs internal processing, optionally calls methods of the `ConnectionInterface`, and interacts with the `ConnectionMan-`

ager as necessary to register and deregister event types for the connection as well as timeouts it wants to receive notifications about.

All ConnectionHandlers also have a uniform interface for use by the ConnectionManager: the ConnectionManager dispatches events received from the operating system to the appropriate Connection by calling the respective method of the ConnectionHandler. The ConnectionHandler analyzes and processes incoming events, possibly calls methods of the ConnectionInterface (e. g. to transmit or retrieve data), and notifies the BottomCommunicator if necessary (e. g. about received data).

All sub-objects of a Connection also provide methods for retrieving state information and manipulating the behavior of the object (e. g. to implement flow control mechanisms), but those are not further described here.

Connect Ports

ConnectPort objects constitute the second object type managed by the ConnectionManager at the connection layer. While Connection objects are used for actively setting up and tearing down a connection as well as information exchange, ConnectPorts are passive objects that only wait for incoming connect requests from local applications as well as from other MCS providers. When a ConnectPort is notified about an incoming connection, it instantiates a new Connection object to handle that connection and forwards the notification along with the newly created Connection to its BottomCommunicator.

ConnectPorts look similar to Connections: they also provide an abstract interface for interactions with the communicator layer, they contain an event handler to receive notifications about incoming connect requests from the ConnectionManager and they contain an abstract interface object for dealing with incoming calls, the IncallInterface. These three parts serve similar purposes as already discussed for the Connection objects, for different functionality, though. The abstract ConnectPort interface is used by the communicator layer to specify the (transport) address to bind to as well as to activate and deactivate acceptance of incoming connections. Notifications about connection requests are signaled by the ConnectionManager to the EventHandler object of the ConnectPort. After some internal processing — which typically includes accepting the new connection at the operating system level via the IncallInterface — the ConnectPort notifies its BottomCommunicator about the incoming connection. The IncallInterface provides methods through which the ConnectPort object accesses the operating system or communication library functions for passively creating a communication endpoint, binding it to a transport address, accepting new connections on this endpoint, and releasing the address binding again.[8]

Three types of ConnectPorts are in use within the MCS: for handling connect requests to the Control MCSAP and the User MCSAP, two distinct GPCI services are registered. Furthermore, a ConnectPort object is created per supported T.123 protocol hierarchy to deal with incoming T.123 transport connections; each such ConnectPort binds to its corresponding transport protocol stack via TLI and the T.123 stack library.

[8] If TLI is used as the transport interface, the functions t_bind(), t_listen(), t_accept(), and t_unbind() among others are used for these interactions.

The Connection Manager

The `ConnectionManager` keeps track of all the `Connections` of the MCS provider as well as of the timeouts requested by the `Connection` objects and supervises the corresponding connections using mechanisms of the operating system. It offers a set of methods through which `Connections` and timers (which typically belong to a `Connection`) can be registered and deregistered. Further methods allow to define per `Connection` which events shall be supervised, and which timers shall be set, reset, or cancelled. For each event and each timer an `EventHandler` is specified (which is typically the `ConnectionHandler` of the `Connection` or the `EventHandler` of the `ConnectPort`) that is to be called by the `ConnectionManager` if any supervised events occur or timers elapse.

The `ConnectionManager` is the central point in the MCS provider where incoming events are recognized and dispatched for processing. Therefore, the control flow of a process has to be handed over to the `ConnectionManager` in regular intervals or permanently to enable it to perform its function. The `ConnectionManager` provides a dedicated method that is called from the main function to perform this hand-over: `waitForEvent()`. This method initiates processing of all outstanding events of the MCS provider and then returns the control back the caller. The method is parameterized with a timeout after which the method returns control even if no event has occurred before.

5.4. The Communicator Layer

While the connection layer provides an abstraction from the underlying networks, The communicator layer implements the actual MCS provider functionality. This includes managing an arbitrary number of domains, realizing the MCS services and protocols per domain, and handling connection establishment for the MCSAPs as well as for T.123 transport connections.

The communicator layer consists of a set of hierarchically structured communicators. A communicator is an object that implements part of the overall provider functionally. Each communicator is self-contained and has well-defined interfaces (that are common to all communicators) for exchanging information with its neighbors.[9] The root of the communicator tree is a `ControlCommunicator`, its leaves are `BottomCommunicators`; all objects in between are referred to as `StandardCommunicators`. The `ControlCommunicator` is the central management component of the MCS provider, the `BottomCommunicators` provide the interface to (objects of) the connection layer. `StandardCommunicators` implement the necessary (protocol) functionality of the MCS provider.

Figure 5.8 shows the communicators used to implement the MCS functionality as well as their relationships to one another and to objects of the connection layer. The `ControlCommunicator` is depicted as an ellipse, `StandardCommunicators` and `BottomCommunicators` are represented as rectangles with round corners and as rectangles with square corners, respectively. The communicators perform the following tasks:

[9] For these interactions, various `request()` functions are provided to convey information downwards from the `ControllerCommunicator` to the `BottomCommunicator` and several `indication()` functions are used to forward information upwards. Thus, the concept of communicators has some similarities to the STREAMS mechanism.

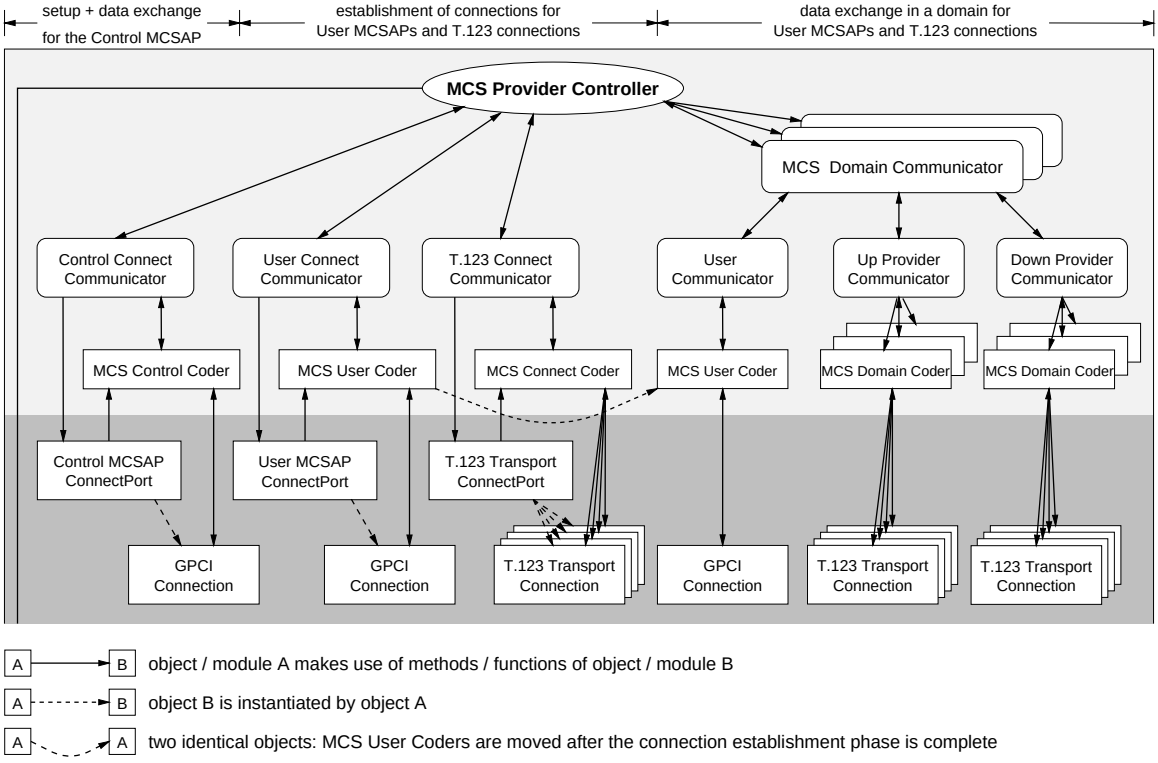


Figure 5.8: Overview of the communicator layer of the MCS

- The `MCSProviderController` is responsible for maintaining domain-independent state information within the MCS provider and handling requests from and sending indications to the Control MCSAP. It instantiates and deletes MCS domains (i. e. the `MCSDomainCommunicators`) and associates new connections (to other MCS providers as well as to local MCS users) with the domains they belong to. It also keeps track of all the utilized resources including all open connections and connect ports.

- An `MCSDomainCommunicator` is created per MCS domain that exists in an MCS provider. The `MCSDomainCommunicator` is responsible for running the MCS protocol within the domain and managing the domain-specific MCS resources: MCS User Ids, channels, tokens, the established connections between MCS providers, etc. Depending on its position in the MCS hierarchy, the `MCSDomainCommunicator` fulfills the central coordination tasks of the Top MCS Provider, a node somewhere in the middle of the hierarchy that also routes MCSPDUs up- and downtree, or acts as a leaf node.

- `ConnectCommunicators` provide means for recognizing and accepting incoming connections. They install `ConnectPorts` within the connection layer and receive indications about incoming connections via their associated `BottomCommunicator`. For each incoming connection, the corresponding `ConnectCommunicator` performs the connection establishment protocol as required by the respective connection type before the connection is handed over to the MCS domain it belongs to.[10]

[10] The exception is the `ConnectCommunicator` for the Control MCSAP that keeps this single (domain-

Three types of ConnectCommunicators are used within the MCS provider (each of which is instantiated exactly once): the ControlConnectCommunicator for the Control MCSAP, the UserConnectCommunicator for the User MCSAP, and the T123ConnectCommunicator for transport connections to other MCS providers.

- UserCommunicators — one of which exists per MCS domain — integrate user applications connected via User MCSAP into an MCS domain. After the UserConnectCommunicator has established a connection requested by an MCS user, the corresponding GPCIConnection object is passed (including the MCSUserCoder) over to UserCommunicator. The UserCommunicator acts as a multiplexer and is responsible for handling the entire information exchange with the MCS users as well as for eventually closing the connections. The UserCommunicator receives MCS SDUs destined for local user applications and routes them the intended recipient(s).

- The UpProviderCommunicator and DownProviderCommunicator are responsible for information exchange with other MCS providers via T.123 transport connections. The UpProviderCommunicator is the link to the MCS provider hierarchically higher in the MCS domain while the DownProviderCommunicator interconnects to all hierarchically lower MCS providers. Both communicator types receive MCS PDUs represented as internal data structures from the MCSDomainCommunicator and route them appropriately to the MCS providers they are destined for. The DownProviderCommunicator acts as a multiplexer to which several T123TransportConnections are attached via BottomCommunicators. UpProviderCommunicator and DownProviderCommunicator are also instantiated at most once per MCS domain.

- Four different types of coder objects act as BottomCommunicators in the MCS provider: the MCSControlCoder and MCSUserCoder for connection establishment and communication with the Control MCSAP and User MCSAP, respectively; the MCSConnectCoder for establishing T.123 transport connections and the MCSDomainCoder for subsequent communication with another MCS provider through a T.123 transport connection. As already stated above, these BottomCommunicators provide the interface to the corresponding Connections. In addition, they perform two further functions:
  - The BottomCommunicator objects perform encoding and decoding of MCSPDUs in ASN.1 (following the Basic or the Packed Encoding Rules of ASN.1, see the summary below) and of MCS IDUs in the GPCI encoding format. That is, the internal data structures are converted to or created from the external (standardized) data representation format that is transmitted across the respective connection. This function has motivated the naming of the BottomCommunicator objects as ‘‘coders’’.
  - In addition, they act as multiplexers for priorities and forward the MCS PDUs as well as IDUs according to their priority. This means that MCS PDUs are mapped to the T.123 transport connection associated with the corresponding priority and that IDUs are transmitted on the appropriate virtual channel provided by the GCPIConnection to the user application.

---

independent) connection to the controlling entity of the MCS provider throughout the connection’s lifetime. Consequently, this ConnectCommunicator is — besides connection establishment — responsible for all subsequent interactions over this Connection.

5.5. Summary

This chapter has described the Multipoint Communication Service as defined in the ITU-T recommendations T.122 and T.125 and has then presented an overview of the implementation of the MCS provider. The MCS provider has been implemented as a separate process which consists of two conceptual layers: the connection layer providing an abstraction from specifics of the operating system and communication libraries used underneath and the communicator layer that implements the MCS protocol engine and manages the operation of the MCS provider. Figure 5.9 gives a summarizing overview of all objects within the MCS implementation and their relations.[12]

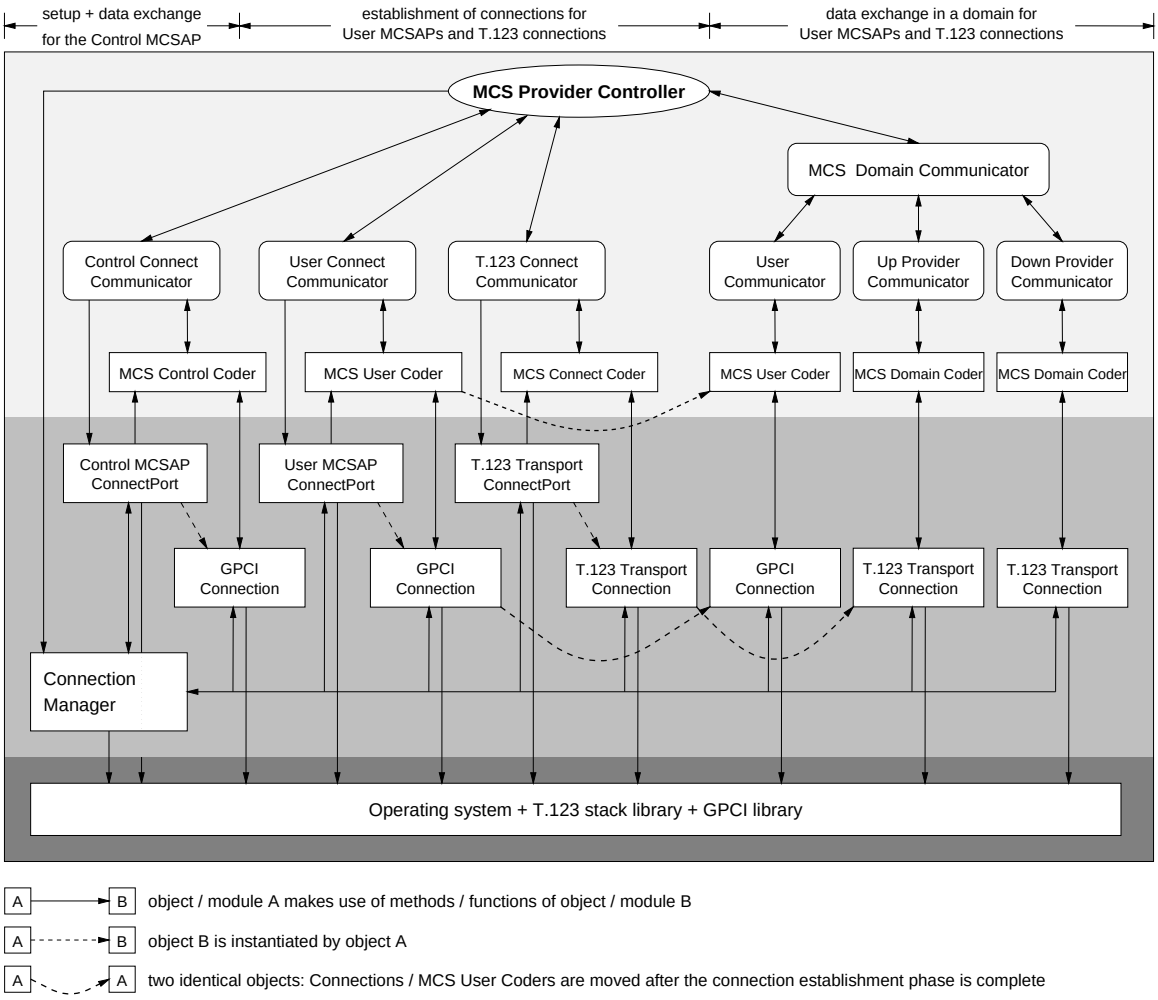


Figure 5.9: Full picture of the MCS provider

The implementation has been carried out for the UNIX (SunOS 4.1.3 and Solaris 2.5) and MS Windows (Windows 3.11, Windows 95, and Window NT) operating system environments. A few numbers on the MCS provider implementation shall be given in the following.

[12] For better readability, the many 1:n relationships are not depicted in this figure.

The MCS provider is implemented in an object-oriented fashion using C++ (GNU C compiler version 2.7.1 as well as Microsoft Visual C++ version 4.2) for the core of the provider. Some 90 C++ classes (including simple lists as well as sophisticated classes e. g. for an MCS domain) are in use in the MCS provider structure of connections and communicators. Besides C++, ANSI C has been used for environment specific extensions (such as the GPCI library, the T.123 stack library, the IMTC API library).

The MCS provider implementation consists of approximately 40,000 lines of source code, including the UNIX and the Windows specific parts, but excluding the T.123 stack library, the GPCI and IMTC API libraries, and the (generated) code for the ASN.1 encoding.

The necessary ASN.1 encoding for the MCSPDUs exchanged between MCS providers has been done with the support of two ASN.1 compilers: initially the ISODE 8.0 package [ISODE 92] has been used for ASN.1 encoding following the *Basic Encoding Rules (BER)* [ITU-T X.208] [ITU-T X.690] which has later been replaced by the commercial OSS[13] ASN.1 compiler version 4.1.2 for supporting Basic as well as *Packed Encoding Rules (PER)* [ITU-T X.691].

For encompassing and detailed descriptions of the MCS provider internals, the reader is referred to the following documents. A detailed description on the implementation concepts underlying the MCS provider and its UNIX-based prototype can be found in [Schmidt 94] with extensions and modifications described in [Nikolaus 95a]. Its MS Windows 3.11 counterpart is described in [Düwel 95], and the complete and tested implementation including the 32-bit Windows platforms (Windows 95 and Window NT) is documented in [Melcher 96].

[13] Open System Solutions, Inc.

6

Multicast Extensions for MCS

The previous chapter has described the implementation of the MCS provider that realizes the MCS protocol as specified in the T.125 ITU-T recommendation. This chapter presents an extension to the MCS provider developed by the author that allows significantly larger conferences to be accommodated by the T.120 infrastructure. The increased scalability is achieved on multicast capable networks such as corporate networks and the Internet by making use of link and network layer multicasting facilities for information distribution. The enhancements presented in this chapter do not harm interoperability with T.120 nodes running non-extended MCS providers.

It has been foreseen [Ott / Bormann 94] [Ott 95d] as well as observed [Kisor 96] that the T.120 architecture does not scale to more than a few tens of participants: test conferences of some 70 participants held over a local area network (10 MBit/s Ethernet) have congested the network with the exchange of management information resulting from state updates while nodes were joining the conference (e. g. capability negotiations) [Kisor 96]. This lack of scalability of the T.120 infrastructure is due to a variety of reasons the three most important of which are:

- T.120 is designed for point-to-point links and thus performs information exchange in a point-to-point fashion between nodes.
- Although T.122 provides an abstraction from the underlying network topology and thus its point-to-point characteristics, in particular the Generic Conference Control (GCC, T.124) creates and exploits knowledge about the MCS domain topology for use in its protocol.
- Finally, the entire T.124 protocol is not scalable due to the way in which information about status updates is exchanged (refer to the next chapter).

These observations suggest that in order to achieve a more scalable T.120 infrastructure, (service and) protocol revisions have to encompass at least the infrastructure recommendations (T.122, T.124, and T.125) including the underlying protocol stacks (T.123) and may even affect some of the application protocols [Ott 96b].

In this thesis, however, only optimizations to the MCS protocol are addressed along with the necessary extensions to T.123. The main reasons for limiting the work to the MCS protocol are that optimizations to the MCS protocol can be deployed in arbitrary parts of the MCS domain independently and that MCS optimizations provide the most benefit because they support information distribution of the GCC and of all application protocol entities equally well.

The need for optimizations to the GCC protocol is well recognized and is briefly elaborated on in section 8.3. However, these optimizations would a) have stronger requirements on which nodes have to support them and b) primarily affect setup and changes to the conference and its application sessions but not actual information transmission within application sessions.[1] Hence a non-standardized extension with the burden to interoperate with conventional T.120 nodes is not considered a worthwhile undertaking. Rather, contributions to such optimizations are directly fed into the ITU-T standardization process to the revision of T.124 (*T.124rev*) and will be incorporated into a future implementation of the revised T.124 to produce a truly scalable revision of T.124 (refer again to section 8.3).

As stated above, this chapter addresses the multicast extensions to MCS in order to optimize information exchange. First of all, the principal problems with MCS information distribution are elaborated on along with the outline of the concept for the extensions to the MCS. Then, the multicast protocol hierarchy is introduced, including descriptions of the multicast transport protocol — MTP-2 — used as the basis, the *MCS to Multicast Adaptation Protocol (MMAP)* required to enable MCS to run on top of multicast, and the rules for multicasting MCSPDUs. Finally, the implementation of MMAP and its integration into the MCS provider are described.

6.1. Concepts for the MCS Extensions

As discussed in section 4.1, the T.120 series of recommendations has originally been designed for teleconferencing scenarios involving point-to-point networks (refer to figure 4.1). With the increasing significance of corporate internetworks (intranets) and the Internet and the increasing availability of IP-based multicast services in these environments,[2] at least the two further scenarios become relevant (figure 6.1):[3]

a) Several sites in a teleconference are interconnected by means of point-to-point WAN connections and an MCU acting as Top MCS provider (which may also be located in the WAN or co-located with one of the sites). At some sites, a number of T.120 nodes are involved so that these sites use local MCUs or multiport terminals to bundle the local connections before connecting to the MCU. The local T.120 connections are most likely to be established via the (IP-based) corporate intranet.

b) All nodes involved in a T.120-based conference are directly connected to the same IP-based network — which is either a corporate intranet for internal teleconferences or the global Inter-

[1] Recall from section 2.2.2 that for the target set of teleconferences changes to the overall conference state are expected at a moderate rate only.

[2] Virtually all network interface cards for LANs such as Ethernet and FDDI available today support multicasting. Furthermore, multicast services are built into all new releases of operating systems (Solaris, SunOS, HP/UX, Windows 95, Windows NT, OS/2, etc.) or are available as patches for earlier releases. Also, software packages for upgrading a workstation into a simple multicast router are freely available. Finally, important router vendors such Cisco Systems and Bay Networks include multicast routing facilities in their router products.

[3] The foundations for the commercial importance of these two scenarios and the resulting widespread interest have been laid with the advent of the ITU-T recommendation H.323 that defines audiovisual teleconferencing in IP-based and other internetworks as well as bridging to WAN-based teleconferences based on H.320 and H.324.

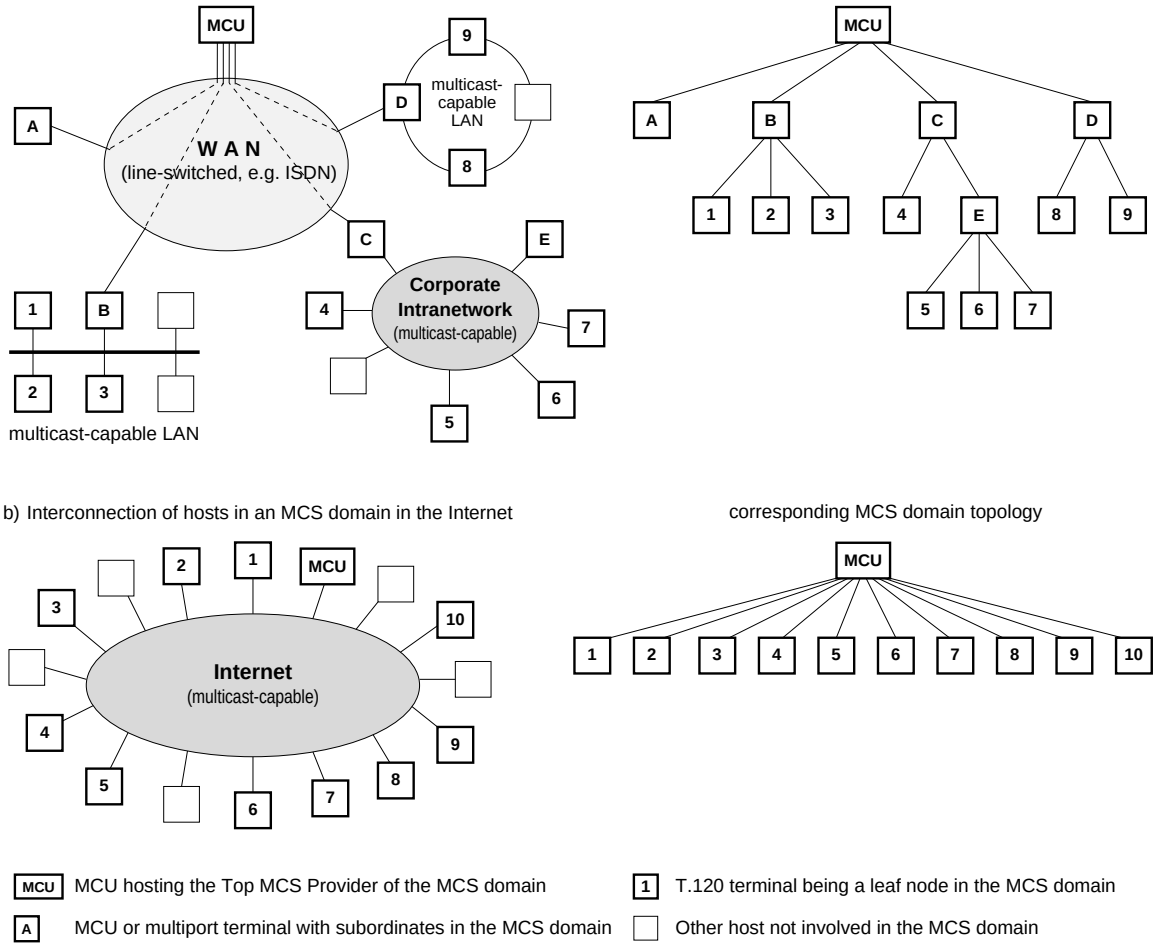


Figure 6.1: Typical interconnection scenarios including multicast networks

net. In this case, a one-level hierarchy with an (arbitrary) Top MCS provider suffices to build the MCS domain, but alternatively, an artificial multi-level hierarchy may be used as well.

From the above two scenarios, the inefficiencies of the current MCS specification when deployed in multicast-capable networks become apparent: within each of the clouds representing IP-based networks in figure 6.1, individual T.120 connections (with up to four TCP connections each) are established between the superior MCS provider for each cloud and all of its subordinates. This has the following implications:

1) All MCSPDUs exchanged between subordinates of the same superior node are routed through the superior MCS provider which introduces an unnecessary extra ‘‘hop’’ thereby increasing transmission latency.

2) With an increasing number of subordinates, the superior MCS provider may easily become a bottleneck for this level and branch of the MCS domain hierarchy because it has not only to route MCSPDUs to and from the next higher hierarchy level (if there is any), but also has to perform the forwarding of MCSPDUs exchanged between its subordinates.

3) The same MCSPDU not only has to be sent repeatedly by the superior provider but also (potentially) traverses the same physical network link(s) several times. This contributes to network congestion which increases with the number of subordinate nodes connected via the same link.

All three problems can be solved, if link and network layer multicasting in conjunction with an appropriate reliable multicast protocol on top is used as the basis for information distribution as described in the following [Ott / Bormann 94].

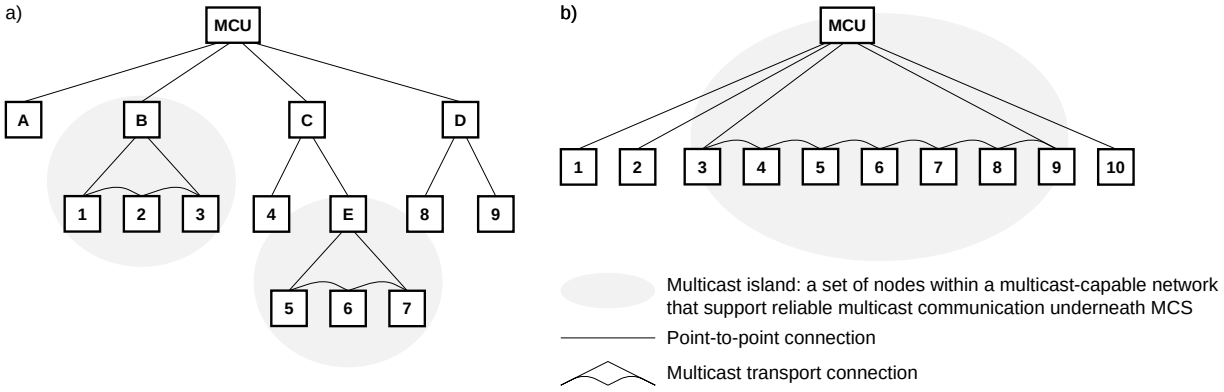


Figure 6.2: Forming multicast areas for efficient information distribution

In the aforementioned two scenarios, IP-based internetworks[4] are used for interconnecting two or more MCS providers on one or more hierarchy levels of an MCS domain. Each internetwork is potentially capable of supporting multicast information distribution — in parts of the network (e. g. within the same LAN) or network-wide. Within such a multicast-capable network part, the T.120 nodes may take advantage of this facility by forming *multicast areas* in which they create multipoint instead of point-to-point transport ‘‘connections’’ that are then used for disseminating MCSPDUs (figure 6.2). The prerequisite is that the involved MCS providers do incorporate the appropriate multicast mechanisms described in this chapter: such MCS providers are referred to as *extended* or *multicast-aware* while those not incorporating multicast are termed *non-extended* or *conventional* MCS providers.

As depicted in figure 6.2, it is determined independently for each level and each branch in the MCS domain which (if any) T.120 nodes are capable of forming a multicast area. A multicast area may be constructed between a multicast-capable superior MCS node and all its multicast-capable (and reachable via multicast) subordinate nodes. Non-extended nodes are connected to the domain via T.123 point-to-point connections, i. e. arbitrary mixtures of point-to-point and multicast transport connections are allowed.

In order not to violate the MCS service specification and to simplify synchronization of point-to-point and multicast transport connections, multicast areas may no span more than a single branch

[4] The concepts presented here are generally applicable; the design and implementation in the context of this thesis, however, have focused on IP networks and the IP multicasting facilities since this is the single widely deployed multicasting infrastructure.

and hierarchy level of an MCS domain.[5] Within a single MCS domain, any number of independent multicast areas may exist, interconnected through multicast-aware as well as conventional MCS nodes. For seamless interoperability throughout an MCS domain, the multicast areas are kept ''invisible'' to the non-extended MCS providers. This is achieved by a) having the multicast-aware nodes bridge between the two protocols and b) designing the adaption protocol in a way that neither requires changes to the MCS protocol itself nor causes violations of the MCS service specification.

By applying this concept, the scalability constraints of the MCS highlighted before are overcome within each multicast area because[6]

1) the MCSPDUs can be distributed directly via multicast to all nodes thereby eliminating the extra hop;
2) the superior node only needs to process those MCSPDUs that need forwarding uptree, that are received from its upward connection and need forwarding downtree, or that are destined for (an application entity of) the superior node itself which eliminates most of the processing for pure distribution; and
3) the multicast MCSPDUs then need only traverse each network link once because efficient distribution is ensured by the network level multicast routing algorithms, the link layer protocols, and the networking hardware.

The details of this concept and its implementation are presented in the remainder of this chapter.

6.2. Protocol Hierarchy for the Multicast-aware MCS

For the purpose of creating a multicast-aware MCS provider that implements the previously outlined MCS optimizations, a multicast-capable protocol hierarchy has to be developed. This requires extensions to the existing T.123 protocol profiles as well as a few procedural modifications the T.125 protocol [Ott / Bormann 94]:

- As MCS makes the assumption that a transport service provides reliable, flow-controlled, and sequential delivery of MCSPDUs to the receiver, a multicast transport protocol that provides similar services is needed for multicast information distribution.
- Furthermore, a control protocol has to be defined that allows to determine whether a particular peer MCS provider is multicast-aware or not and if the peer MCS provider in question is reachable via multicast.
- Finally, the procedures of the MCS protocol have to be enhanced to allow MCSPDUs to be sent via the multicast transport and it has to be defined how the MCS provider employs the control protocol to establish and tear down multicast transport ''connections''.

---

[5] Investigations have been carried out to eliminate these restrictions. However, this would have heavily increased the complexity of the required adaptation protocol. As the typical deployment scenarios for multicasting do not require multicast areas at multiple levels or multiple branches to be merged and the creation of such topologies can be avoided in the first place, the potential additional gain is considered very limited. Hence, this approach is not pursued any further.

[6] Obviously, the scalability of a multicast area is limited by the scalability of the underlying multicast transport protocol.

To fulfill these requirements, the protocol hierarchy depicted in figure 6.3 has been developed which is described in this section. This figure intentionally includes the conventional point-to-point protocol hierarchy of T.123 as well to indicate that an extended MCS provider is capable of running both protocol hierarchies simultaneously and bridging between the two thus allowing extended and non-extended MCS providers to be arbitrarily intermixed in a single MCS domain.

The multicast extensions to MCS are defined for network environments based on the Internet Protocol suite. UDP and TCP are used on top of IP: UDP provides application addressing and checksum calculation for the multicast transport protocol; TCP connections are used by the adaptation protocol to initiate connections between extended MCS providers and to negotiate the use of the multicast transport. The *Multicast Transport Protocol Version 2 (MTP-2)* provides the required reliable multicast transport service and the *MCS to Multicast Adaptation Protocol (MMAP)* performs the connection setup, protocol negotiation, and the necessary adaptation during operation of an MCS domain. The services and the protocol operation of MTP-2 are briefly outlined in subsection 6.2.1, the design of MMAP is presented in subsection 6.2.2. Finally, the way the extended MCS provider makes use of the MMAP and MTP-2 services is described in subsection 6.2.3.

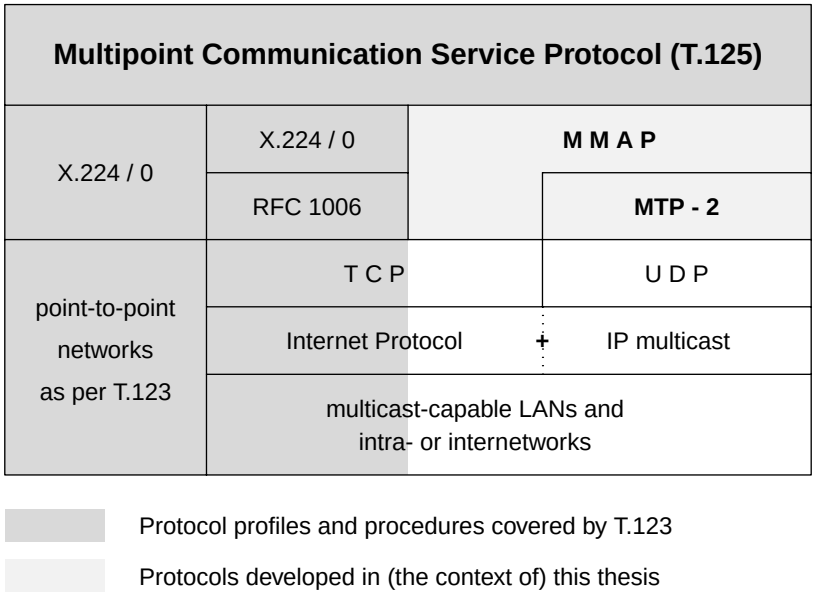


Figure 6.3: Protocol stacks for the multicast-aware MCS provider

6.2.1. The Multicast Transport Platform: MTP-2

The MCS provider expects from an underlying point-to-point transport reliable, sequential and flow-controlled transmission of information units. The same requirements apply to a multicast transport, unless the MCS protocol is to be modified to incorporate the functionality of a reliable multicast transport protocol as well. As the MCS protocol and its implementation in the MCS provider should not be burdened with this additional complexity, a separate reliable multicast transport protocol is required. As already pointed out in subsection 3.2.1, however, only a best-effort datagram transport service is standardized for multicast communication. Due to the lack of

a standardized solution, a novel reliable multicast transport protocol is used instead that meets the requirements of MMAP and the extended MCS. This protocol — MTP-2 — is introduced in this subsection: at first, an overview of the development towards MTP-2 is given, followed by a presentation of the transport services it provides and an outline of its protocol characteristics. Finally, at set of requirements for the design of MMAP is derived from the discussion of MTP-2.

Note that this section deliberately does not present any details of the MTP-2 protocol specification nor is it intended to justify the design decisions in the context of the MTP-2 development. MTP-2 is considered a means to obtain the multicast transport services required for the realization of the the MCS extensions, but the development of MTP-2 has not been focused on this particular application. Although MTP-2 has been designed and implemented in the context of this thesis, its development constitutes a research effort of its own. For more detailed information on MTP-2, the reader is referred to [Bormann *et al.* 94b] [Bormann *et al.* 94c] [Kerschat 94] [Seifert 94] [Bormann *et al.* 96b], and [Seifert 97].

Historic Development towards MTP-2

When the conceptual work on the multicast-capable MCS was started in 1993 [Ott 93], the choice of suitable reliable multicast transport protocols supporting many-to-many communications that were readily available was very limited.[7] The protocols developed until then include the work of CHANG AND MAXEMCHUK [Chang / Maxemchuk 84], the Versatile Message Transfer Protocol [Cheriton 86] [RFC 1045], the *Multicast Transport Protocol (MTP)* [Freier / Marzullo 90] [RFC 1301], and a few others. Out of these, MTP was chosen as a starting point for a reliable multicast transport because it was judged both a reasonably suitable and a well-documented protocol. Furthermore, MTP includes interesting concepts to achieve scalability [Bormann *et al.* 94b]. During the evaluation and prototype implementation of MTP [Gehrcke *et al.* 94], it became apparent that several extensions to MTP were desirable if not required to make MTP more scalable, more efficient, more robust, and more flexible. The intention was to provide a general purpose multicast transport protocol that serves equally well as transport underneath MCS as it does when used by a range of other multicast applications (e. g. [Bormann *et al.* 94a] [Krause 95] [Bormann 95]).[8] The extensions and refinements of MTP have lead to the development of the *Multicast Transport Protocol Version 2 (MTP-2)* [Bormann *et al.* 94b] [Bormann *et al.* 94c] [Seifert 94] [Kerschat 94]. At the time of writing, the MTP-2 specification is being revised to further increase the scalability of this protocol as well as to accommodate ideas and requirements that have been voiced during general discussion sessions on the topic of reliable multicast transport at the 36th and 37th IETF in summer and winter 1996, respectively. The revised version of MTP-2 is termed *self-organizing multicast transport*

[7] Note that there seems to be a correlation between the eventually increasing deployment of multicast routing protocols in intranets and the Internet (with the creation of the multicast backbone, MBone) around 1993 and the rapidly growing interest in all kinds of multicast transport protocols in the research community since then.

[8] More general applicability is of particular importance since the ultimate goal is to build the protocol stack on top of a *standardized* reliable multicast transport, and such a standard is unlikely to be tailored to this specific application. Therefore, MTP-2 is not only intended to solve the particular problem of providing a suitable multicast transport for the multicast-aware MCS provider, but also as a contribution to the development of a (set of) standardized reliable multicast transport protocol(s).

(MTP/SO); the current state of work is documented in [Bormann *et al.* 96b] and [Bormann *et al.* 97].

In parallel to the development of MTP-2, various other reliable multicast transport protocols have evolved, the more widely known of which include the following:[9]

- The *Reliable Multicast Protocol (RMP)* is a protocol that uses a virtual ring topology of group members to achieve reliability [Whetten *et al.* 94]. RMP and similarly designed protocols may easily be used as a substitute for MTP-2 in the MMAP protocol hierarchy as they provide roughly the same services. Note, however, that the virtual ring topology does not scale as well as MTP-2 does.

- *ISIS* and its successor *Horus* are two fault tolerant multipoint communication systems that were initially based on point-to-point connections, but were extended to work on top of IP multicast as well [Birman *et al.* 91] [van Renesse *et al.* 96].[10] In principle, any fault-tolerant system providing reliable message delivery on top of IP multicast may be used as a basis for MMAP, too. However, the requirements on fault-tolerant protocols developed for distributed systems are typically much higher than on a reliable multicast protocol so that the resulting protocols tend to be more heavyweight compared to MTP-2 or RMP.

- *Scalable Reliable Multicast (SRM)* [Floyd *et al.* 95] is based on the *Application Layer Framing (ALF)* approach to [Clark / Tennenhouse 90] that provides reliability of multicast communication within the application and thus tailored to the application-specific needs. Unlike the previous two protocol types, the ALF concept is not a suitable platform for the extended MCS, unless MMAP, MCS itself, or another protocol layer beneath or above MMAP would implement the retransmission strategy for MCSPDUs.

The existence of these other reliable multicast protocols is well-recognized and, as indicated above, MMAP is designed to avoid any dependency on a particular multicast transport.[11] Therefore, no or at most minor refinements would have to be made to MMAP to make it run on top of RMP, for example. The protocol that is eventually chosen to provide the reliable multicast service in a particular implementation has an impact on the quality (in terms of throughput, latency, etc.) of the resulting transport service as well as its scalability, but does not affect the basis of MMAP and the MCS extensions.

Service Description

In MTP-2, a set of entities communicate within a *web* which is comparable to a transport connection but interconnects more than two end points. A web is identified by a *web name* and has a *web id*, a multicast address, and a UDP port associated with it. A set of *web parameters* describes the maximum throughput, the degree of reliability of the web, and other variables of the web (see

[9] For an overview of publications on (reliable) multicast transport protocols refer to [Knight 97].

[10] Other protocols for fault-tolerant multipoint communication include Totem [Moser *et al.* 96], Transis [Dolev / Malki 96], and the Rampart toolkit [Reiter 96]. For overviews of communication aspects in distributed systems refer to [Mullender 89], [Mullender 93], and [Birman *et al.* 94].

[11] As pragmatic approach for the development of MMAP, a minimal required multicast transport service — which is to be easily provided by other reliable multicast protocols as well — is derived from the MTP-2 service description and presented in the context of MMAP in section 6.2.2.

the protocol description below). In a web, a designated MTP-2 protocol entity acts as the *master*. Web members that source information are termed *producers*, those only receiving information are referred to as *consumers*.[12] Data transmission in a web is done as *messages* each of which consists of one or more data packets. The various ordering and reliability properties offered by MTP-2 relate to messages rather than individual packets.

For the descriptions in the remainder of this subsection, OSI style terminology is used. The terms *MTP-2 entity* and *entity* are used to refer to a particular instance of an MTP-2 protocol implementation, the terms *peer* and *peer entity* denote MTP-2 entities in the same web communicating via the MTP-2 protocol. Applications or higher protocol layers that access MTP-2 services are termed *user process* or simply *user*. The services offered by MTP-2 are described using the naming conventions for OSI service primitives: *request*, *indication*, *response*, and *confirmation*.

Joining and Leaving a Web

MTP-2 does not support explicit and confirmed establishment and teardown of connections between MTP-2 entities in the sense that point-to-point transport connections do. Rather, the web name — plus optionally the multicast address and port number of a web — is used as a rendezvous point by all MTP-2 user processes that want to engage in an MTP-2-based multicast communication relationship.

An MTP-2 user makes its local MTP-2 provider entity become part of a web by issuing an `MT-JOIN` request in which the rendezvous information is specified and relinquishes its membership in the web by issuing a `MT-LEAVE` request. No confirmed interaction with any other user process in the same web is required for the joining or leaving user. All necessary interactions are carried out at the protocol level by the MTP-2 entities without intervention of the application processes.

In case of network failures, an MTP-2 entity may be excluded from the web through MTP-2 protocol mechanisms about which the excluded user process is notified through a (provider-initiated) `MT-LEAVE` indication.

Data Transmission

MTP-2 users send data to a single (unicast) or all other users in an MTP-2 web by using the `MT-DATA` request primitive. The MTP-2 entity of the sender transmits the data to all other MTP-2 entities that are members of the web — except for unicasting where only a single MTP-2 entity within the web receives the data. The recipients then deliver the data to their respective user process by means of an `MT-DATA` indication.

Data transmission in MTP-2 is rate-controlled with a maximum total transmission rate being agreed upon during the creation of a web — note that this rate may be adapted dynamically to changing network conditions. Adherence to the rate for the entire web is ensured by the master and the MTP-2 entities of transmitting user processes.

Messages are delivered to the receiving MTP-2 users by their respective MTP-2 entities according to the message ordering properties which may be specified individually per message. MTP-2

[12] This terminology is entirely borrowed from the original MTP.

supports per-source as well as global ordering. Furthermore, MTP-2 introduces the concept of *streams*: each message is associated with a stream, and message ordering is performed on a per stream basis. Messages belonging to different streams can overtake one another arbitrarily while messages of the same stream are ordered according to their respective ordering property. This allows application-specific ordering to be implemented on top of MTP-2.

Finally, for each message a priority is specified by the sender. A higher priority means that a message may be transmitted (and delivered) earlier compared to other messages that are sent roughly at the same time by other MTP-2 users. However, in contrast to streams, different transmission priorities do not imply independent ordering, and priorities are not indicated in the `MT-DATA` indication.

Reliability in MTP-2 is defined as follows: each message is delivered to all recipient users either error-free or not at all. For each message transmitted in an MTP-2 web, each user that does not fail during the message transmission either

- receives the message correctly (through a `MT-DATA` indication) or
- is notified that the message was not delivered to anybody in the web (by means of a `MT-DATA-NOT-ACCEPTED` indication) or
- is notified that it has missed the message (by receiving a `MT-DATA-MISSED` indication).

After the MTP-2 entity of an application process has successfully joined the web and correctly received its first message, the above condition holds true for all messages with larger message sequence numbers until the user process leaves the web. As MTP-2 is a reliable multicast transport protocol, the latter two indications may only happen as a result of network connectivity problems or network congestion that could not be repaired by the retransmission mechanism of MTP-2[13] or due to unavailability of system resources at the sending (producing), receiving (consuming), or master MTP-2 entity.

Master Changes and Web Reconfiguration

In each web, one MTP-2 entity — typically the first one joining the web — assumes the role of the master to coordinate information exchange in the web (see the protocol description below). When a new application process joins the web, it receives the identity of the current master in the `MT-JOIN` confirmation. An MTP-2 web relies on the availability and the functioning of a master. Therefore, MTP-2 automatically recovers from master failures with no intervention of the application processes being required. Also, MTP-2 allows to explicitly pass on the master role from one MTP-2 entity to another by means of the `MT-MASTER-PASS` service. When the master changes for either reason, all MTP-2 users joined to the web are informed about the new master's identity by receiving an `MT-MASTER-CHANGE` indication.

Protocol Characteristics

The MTP-2 protocol is built around the master that performs the required coordination functions: rate control, global ordering[14], handling join and leave requests, as well as dissemination of the current web state. The web state contains: the identity of the current master, the *message*

[13] Note that such problems also would lead to provider-initiated disconnection of e. g. a TCP connection.

[14] It should be noted that the master does not perform any forwarding of data packets; hence, it does not introduce extra transmission latency, nor is it subject to becoming a bottleneck.

acceptance record containing the status of the twelve most recently transmitted messages, the current web parameters, and a sequence number (that is incremented each time the web state changes). The web state is regularly distributed by the master either as a separate packet or piggybacked on other packets that are sent by the master anyway and is also carried in some packets sent by other MTP-2 entities. The information contained in the web state is used by the MTP-2 entities to decide when to deliver messages to their users, which messages to discard, when to perform error recovery, etc.

A set of four web parameters is essential to the protocol operation of MTP-2. These define the reliability level and the maximum throughput of a particular web and also allow tuning an MTP-2 web to different network conditions:

- For the operation of MTP-2, the time axis is divided into equidistant *heartbeat* intervals. The duration of a heartbeat is set initially upon creation of the web but may be modified by the master to adopt to varying network conditions. Virtually all timers within MTP-2 are defined as multiples of the current heartbeat.
- The *window size* specifies how many data packets a producer is allowed to transmit per heartbeat.
- The *packet size* specifies the maximum number of bytes that may be transmitted per packet.
- *Retention* defines the number of heartbeats a sender is required to buffer data packets of a message it has sent for later retransmission. Most counters of MTP-2 are defined in relation to the retention parameter.

Heartbeat, window size, and packet size together define the maximum throughput for the entire web as well as per sender and per message; taking all four parameters together allows to derive the maximum buffer space required for each sender (if it wants to make use of the maximum throughput).

The following paragraphs give a high-level outline of the protocol operation of MTP-2. The intention is to show that MTP-2 is sufficiently scalable and reliable to be a suitable platform for the MCS extensions as well as to point out its benefits over the usage of point-to-point connections. Therefore, the focus is on regular operation of the protocol: membership control and data transmission in a web are presented. Deliberately, no details are provided about master migration and master loss recovery, nor are internal protocol optimizations and handling of special cases addressed as such issues are considered beyond the scope of this thesis. The full specification of MTP-2 is found in [Seifert 94], further details can also be obtained from [Bormann *et al.* 94b] and [Kerschat 94].

In the following description, the notational conventions introduced in the MTP specification [RFC 1301] are re-used for MTP-2 packets: `packet-type[modifier]` denotes a packet of type `packet-type` with the `modifier` indicating whether the packet contains a request or a (positive or negative) response, or which (packet-type dependent) control flags are set for this packet. Note also that all MTP-2 packets are multicast, unless explicitly specified otherwise.

Creating, Joining and Leaving a Web

An MTP-2 entity that wants to join a particular web, multicasts a `join[request]` containing the web name to the corresponding multicast address. If it does not receive a reply after several attempts, it assumes the master role for the web that is newly created.[15] Upon creation, a web is assigned a globally unique *web id*.[16] If a master is already present for this web, it receives the `join[request]`, checks the parameters against the current web parameters and replies with either a `join[confirm]` or a `join[deny]` depending on whether or not the web parameters specified in the `join[request]` were compatible with those of the web.

An MTP-2 entity that wants to leave a web, typically sends a `quit[request]` to the master via unicast. It repeats sending the packet until it receives a `quit[confirm]` or until the retry limit is reached. Then, the entity leaves the web's IP multicast group. As an option, MTP-2 entities may also quit a web silently, i. e. without sending any notification packet.

The master may also decide to exclude a particular MTP-2 entity from the web if this entity does no longer act in conformance to the web parameters: e. g. if the entity is not capable of successfully receiving data packets with the expected throughput and therefore slows down the entire web. The master does so by unicasting a `quit[request]` packet to the MTP-2 entity in question which then leaves the web silently.

Data Transmission

When a producer wants to transmit a message, it applies for a transmission token by unicasting a `token[request]` to the master; this message contains a priority.[17] If a token is available, the master assigns a token to the requester by unicasting a `token[confirm]` packet back that also contains the global sequence number assigned to the message and marks the corresponding entry in message acceptance record as *pending*. Otherwise, the master buffers the token request and honors it as soon as new tokens become available.[18] The total number of tokens defines the numbers of simultaneously transmitted MTP-2 messages: the protocol specifies that up to twelve tokens can be used resulting in a maximum of eleven concurrent messages. If several token requests are pending, the master answers those with the highest priority first. A token assigned to a message is reclaimed by the master as soon as the message is either completed or aborted. It should be noted that, if master and sender are the same MTP-2 entity, no packet exchange to obtain a token occurs at all.

---

[15] An MTP-2 entity may also enforce creation of a new web with itself as the master provided that it is able to ensure uniqueness of the web name it has chosen. In this case, the MTP-2 entity does not transmit a `join[request]` packet but joins the multicast group and waits for other MTP-2 entities to join the web.

[16] The web id is contained in all further packets transmitted in the web and allows to have even several webs concurrently using the same IP multicast address. This is of importance for two reasons: a) because at the time of writing no reliable global allocation mechanism is available for IP multicast addresses and b) to handle temporary network partitioning after which a web is split into two webs both of which definitely use the same IP multicast address.

[17] For efficiency reasons, the `token[request]` may be piggybacked onto any data packet if the requester is already transmitting a message and needs to obtain a token for a subsequent one.

[18] In the meantime, the requester may retransmit the `token[request]`; such retransmissions are detected by means of sequence numbers and do not lead to accidental repeated token assignments to the requester.

When the sender has received a token, it starts transmitting `data[data]` packets. The packet transmission rate must not exceed the maximum rate — measured in packets per heartbeat — that is also specified by the master in the token assignment. All packets belonging to a message contain a packet sequence number and the global message sequence number among other control information. The last packet of a message is identified by an *end of message* indicator (a `data[eom]` packet). A sender in possession of a token is required to transmit at least one data packet per heartbeat. If a sender has no information to transmit within a heartbeat but the message is not yet complete, it sends a single empty `data[dally]` packet for that heartbeat.

Receivers detect missing packets by either observing a gap in the packet sequence number space of a message, by not receiving any data packets for an incomplete message during an entire heartbeat, or from the message acceptance record in the web state that is regularly announced by the master. Receivers encounter that entire messages are missing either from gaps in the global message sequence number space or from the message acceptance record. In order to request retransmission of the missed packet(s) the receivers multicast a `nak[request]` to the web.[19] Each sender buffers data packets that it has sent for *retention* heartbeats. During this time, retransmission requests are answered by the sender's re-sending the corresponding data packet. Afterwards, the sender replies with a `nak[deny]` indicating which packets cannot be re-sent because they have already been dropped. An MTP-2 entity receiving a `nak[deny]` informs its application process through the `DATA-MISSED` indication that it has missed the corresponding message.

If the master has received the entire message, it reclaims the token and marks the message in the message acceptance record as *accepted*. If the master is denied a packet it has asked for to be retransmitted, it reclaims the token and rejects the corresponding message by setting this message's entry in the message acceptance to *rejected*. As the message acceptance record is part of the web state, it is multicast to the web at least once per heartbeat so that all MTP-2 entities in the web can generate the appropriate service indications.

Comparison of MTP-2 and TCP

This final part of this subsection shows the differences between the service characteristics offered by MTP-2 and those provided by TCP. The intention of this brief comparison is twofold: first, it is to highlight the advantages of MTP-2 over the use of TCP as defined in the T.123 protocol stack when concerned with large conferences. Second, it is to provide some background on the motivations of the MMAP design: with respect to the MMAP functionality as well as the requirement to use TCP connections in addition to MTP-2. The following comparison compares an MTP-2 web on one hand and a single TCP connection — as well as a fan-out of TCP connections for those characteristics where the distinction between TCP and TCP fan-outs is relevant — on the other.

For the extensions to MCS, the issues discussed in the following are of relevance (refer also to [Bormann *et al.* 94b], [Ott / Bormann 94], and [Seifert 94]):

[19] To avoid NAK implosions, the receivers first wait for the end of the current heartbeat and then wait another random time (in the order of a fraction of a heartbeat) before they actually send their `nak[request]`. If a receiver hears one or more other retransmission requests for the same data packets while it is waiting, it does not send its request at all.

- *Connection management services*

TCP supports actively contacting a peer entity in order to establish a connection while an MTP-2 entity passively waits for its peer(s) to join the web. Furthermore, a TCP entity is able to notice that the connection to its peer has been dropped. MTP-2 does not keep membership information about the web; if an MTP-2 entity leaves the web or loses connectivity to the other members, only the (user process of) the affected MTP-2 entity itself is notified about this occurrence.

- *Reliability*

MTP-2 may be deemed less reliable compared to TCP because it employs a negative acknowledgment scheme: instead of a positive acknowledgment, a timeout is used to determine when a sender may release buffered data. Under bad network conditions combined with ill-suited web parameters (heartbeat, retention), this may lead to messages being missed by some recipients. If a reliable transport service is expected, such an occurrence requires that the connections to the web be dropped for the affected recipients and/or the sender. In order to increase the reliability level, the timeout period and thus the buffer space has to be increased.

With TCP, bad network conditions may lead to timeouts which eventually may result in closure of the corresponding connection — meaning that TCP may not be considered to be fully reliable either (note, however, that a TCP sender can adapt its timeouts according to the experienced network connectivity to its peer).

On one hand, significantly less buffer space is required for a single TCP connection compared an MTP-2 entity in order to achieve the same degree of reliability. On the other hand, the amount of buffer space needed by an MTP-2 entity is independent of the number of peer entities while it grows linearly with the number of recipients for TCP; that is, for TCP fan-outs consisting of large numbers of connections, the required memory resources finally exceed those for an MTP-2 web.

> For example, with a packet size of 1,024 bytes, a window size of 4, and a heartbeat of 200 milliseconds (i. e. a throughput of 20Kbytes/s), — which are values suitable to a LAN or well provisioned intranet — a retention of 10 means that packets are available for 2 seconds after their transmission (a round-trip time in such a network is in the order of a few tens of milliseconds). This also means that a total buffer space of 40 Kbytes is required at each continuously transmitting producer. In contrast, for each TCP connection on a SPARCstation (SunOS 4.1.3), 4 Kbytes of transmission buffer space are allocated by default, so that a fan-out of 10 TCP connections requires the same amount of buffer space.

Furthermore, while the TCP entity for each individual connection may adapt itself to match the achievable throughput to a peer, in case of TCP fan-outs, the application has to perform additional buffering if the achieved throughputs of the individual connections of a TCP fan-out differ significantly.[20]

---

[20] In this context, it needs to be pointed out again that this thesis focuses on interactive application scenarios. A point-to-point TCP connection may be able to tolerate longer delays (due to packet loss and repeated retransmissions) and still preserve the reliability properties of the connection. However, the additional delay may render the system unusable for the human conferee. That is, increased reliability at the expense of interactivity is not an option for a teleconferencing system. Rather, the chosen network infrastructure has to ensure provision of sufficient (bandwidth) resources so that significant packet loss due to network congestion does not occur in the first place. This may be achieved by means of resource reservation mechanisms (if available), by use of dedicated communication links for a teleconference, or by only using well provisioned networks (such as corporate intranetworks).

- *Scalability*

MTP-2 is more scalable compared to a fan-out of TCP connections (obviously, this attribute is not applicable to a single TCP connection). It is also more efficient with respect to network resource utilization, particularly if packet loss in the network occurs only rarely — which is at least a valid assumption for most intranets.

However, if used for point-to-point communication, the early MTP-2 prototype implementation is slightly less efficient compared to TCP, in terms of number of packets and total number of bytes sent across the network. For up to two web members, the throughput of the prototype MTP-2 implementation is lower compared to the use of a fan-out of two TCP connections and for three members it is approximately the same. For four web members and above the MTP-2 prototype's throughput exceeds that of a fan-out of TCP connections [Seifert 94].

- *Connection establishment delay*

TCP requires three packet transmissions and thus establishes a connection within about 1.5 round-trip-times assuming no packet loss. Connection setup delay for MTP-2 depends on whether the web already exists. If so, entering the web is accomplished by a single handshake with the master[21] which takes only one round-trip time.

If a new web is established, it takes the joining MTP-2 entity *retention* × *heartbeat* to determine this, unless the MTP-2 entity enforces the creation of a new MTP-2 web in which case the request completes instantaneously.

- *Data transmission latency*

Under no load, the latency experienced by data sent via TCP is ideally in the order of the propagation delay on the path to the recipient plus local processing delay (under load, various mechanisms increase this latency [RFC 0896] [Jacobson 88]). In case of TCP fan-outs, this applies to each of the TCP connections as well; however, the sender has to perform the transmission of data across all connections serially; i.e the data have to be sent repeatedly across the local network interface which may incur minor additional delay.

In contrast, MTP-2 requires an extra round trip to obtain a token for the message before the data can be sent; this results in approximately three times the latency of TCP. To reduce this overhead, a message may consist of multiple packets with only a single token request for all of them. Furthermore, MTP-2 allows piggybacking the token requests on data messages so that the latency due to the extra round trip may be eliminated entirely for *additional* messages. Finally, messages originated by the master do not experience any extra transmission delay since the token assignment protocol takes place locally.

- *Processing overhead*

The prototype implementation of MTP-2 requires approximately one order of magnitude more processing power than a TCP connection does. However, while the processing required for a fan-out of TCP connections grows linearly with the number of connections, the processing for MTP-2 webs increases less than linearly [Seifert 94] and is expected to stay roughly constant for more than a few tens of web members.

[21] Note that this does not take into account the extra delay required for joining the corresponding multicast group at the level of IP multicast and its multicast routing protocols.

The much higher processing overhead is mainly due to the actions to be carried out regularly per heartbeat in MTP-2 while TCP is almost entirely driven by data packets being sent or received (except when handling retransmissions).

Derived Requirements for MMAP

From this overview, it becomes apparent that MTP-2 is a good candidate to improve scalability of the MCS but also that MTP-2 has some limitations with respect to this particular application that need to be addressed by additional mechanisms in MMAP:

- First of all, MTP-2 does not provide a mechanism for an MTP-2 entity to actively contact another entity and forward the necessary rendezvous information to eventually make the contacted entity join a particular web. This needs to be performed out-of-band to MTP-2.
- Another problem arises because user or provider-initiated connection teardown is only signaled to the entity that has left the web or has been excluded. Therefore, MMAP needs to provide a means to detect that MTP-2 peer entities have disconnected.
- Furthermore, the potential latency for creating and joining a web induced by multicast routes building up is undesirable for interactive applications such as teleconferences. To circumvent this additional startup delay, a second transport may be used for initial communication until the web establishment is complete.
- For efficiency reasons, MTP-2 should only be used when at least three subordinate MCS providers can be included in a web. This requires that it be possible to delay the establishment and use of an MTP-2 web until a threshold number of web members is reached. Also, as the use of MTP-2 becomes more economic compared to TCP with a growing number of subordinates, the creation of an MCS domain should avoid artificial hierarchy levels, but rather connect as many subordinate MCS providers to a single superior node as possible.

These and other issues are addressed by the design of the adaptation protocol MMAP that is presented in the next subsection.

6.2.2. The MCS-to-Multicast Adaptation Protocol

In the previous subsection, a description of the transport service offered by MTP-2 has been given and these services have subsequently been analyzed with respect to the applicability of MTP-2 underneath MCS in order to implement the multicast extensions. This analysis has shown that the central transport services of MTP-2,

- `MT-JOIN`,
- `MT-LEAVE`, and
- `MT-DATA`,

are sufficient for information exchange within an MCS domain. However, it has also been pointed out that an additional point-to-point reliable transport protocol and further functionality in MMAP are required for connection setup and teardown.

This subsection introduces the concepts of the *MCS to Multicast Adaptation Protocol (MMAP)* that is used to overcome the previously identified limitations and provide a suitable platform to the MCS provider for the implementation of the multicast extensions. Besides MTP-2, TCP is used underneath MMAP for (initial) information exchange. With these protocols in place and the

requirement that the MCS protocol definition and its fundamental concepts shall not be changed, in principle, MMAP has to perform the following major tasks:

- MMAP has to provide means for establishing a connection between two MCS providers with the aim of using reliable multicast; this includes determining whether using MTP-2 as a transport is possible. If this is not the case, the calling MCS provider has to fall back to using a T.123-compliant unicast transport in order to preserve backward compatibility with non-extended MCS providers.
- Furthermore, MMAP must support the data transmission within the MCS domain and perform the potentially necessary adaptations to allow multicasting of MCSPDUs.
- Finally, MMAP has to provide means for releasing the TCP and the MTP-2 connection as well as detecting loss of connectivity to a group member in MTP-2 webs.

The MMAP protocol operation distinguishes five phases that are in detail described in the remainder of this subsection:

1) initial connection setup via TCP (the *initial MMAP connection*);[22]
2) establishment of *additional (point-to-point or multicast) MMAP connections*, in particular of an MTP-2 web;
3) data transfer across the *active MMAP connection*;
4) transition from using one MMAP connection for data transfer to another in order to use a different (more suitable) protocol and/or a different transport type (i. e. point-to-point vs. multicast); and
5) teardown of individual MMAP connections as well as the MCS connection between two MCS providers.

An MMAP-based MCS connection between two MCS providers always starts with phase one and always terminates with phase five; in-between, phases two through five may be almost arbitrarily interleaved. Figure 6.4 depicts an example for the operation of MMAP:

> The connection setup is performed by establishing the initial MMAP connection which consists of up to four TCP connections; the initial MMAP connection is also used to determine whether multicast communication is possible. The TCP-based initial MMAP connection is used for data transmission while the MTP-2 web is established as another MMAP connection. At any time during the data transfer phase, the transport protocol transition phase may be entered to change the MMAP connection being used to convey data between two MCS providers: from TCP to MTP-2. This change typically occurs as soon as a given threshold number of participants in a multicast area is exceeded so that the use of MTP-2 provides benefits over the use of a fan-out of TCP connections (which is assumed in this example). As soon as the transition to MTP-2 is complete, the TCP-based MMAP connection is closed since it is no longer needed; data exchange continues via the MTP-2 web. Finally, the MTP-2-based MCS connection between the two MCS providers is closed by the subordinate MCS provider thus disconnecting the entire MMAP-based MCS connection.

[22] Note that each MMAP connection may consists of one up to four transport connections (similar to a T.123-based MCS connection), as well as only of a single transport connection if the transport protocol is capable of multiplexing multiple priorities into a single transport connection as is the case with MTP-2. In particular, the initial MMAP connection should not be confused with the initial transport connection of the MCS; the latter is the first out of four point-to-point TCP connections that together constitute the former. An MMAP-based MCS connection may consist of an arbitrary number of parallel MMAP connections.

The overall concept of MMAP was originally introduced in [Ott / Bormann 94]. For a detailed description of the design and implementation of MMAP, refer to [Nikolaus 95a] and [Nikolaus 95b], the complete protocol specification can be found in [Nikolaus 95b].

Connection Setup Phase

The connection setup phase is initiated by the controlling entity of the MCS provider issuing an `MCS-CONNECT-PROVIDER` request. The request primitive is parameterized with the transport address of the MCS provider to connect to. From this structured address, the calling MCS provider derives whether and which T.123 protocol profile to use for the connection setup or whether an MMAP-based MCS connection shall be established instead.[23] If the MMAP hierarchy is selected, the calling MCS provider initiates the MMAP initial connection setup phase described in the following.

First of all, the calling MCS provider has to determine whether its peer is an extended MCS provider that supports MMAP. This is achieved by attempting to set up the TCP connections of the initial MMAP connection to a TCP port number which is different from the one that is used by the T.123 profile. If the peer supports MMAP, establishing the first TCP connection of the initial MMAP connection succeeds, and the operation of MMAP continues as described further down. If the connection setup fails, the peer is not an extended MCS provider and the connection establishment procedure for the Internet Protocol profile as defined in T.123 is initiated instead. In this case, the called MCS provider does not notice the first connection setup attempt (and thus is not disturbed in its operation) so that backward compatibility is provided.

If establishment of the first TCP connection of the initial MMAP connection succeeds, the two MCS providers have to find out which (multicast) transport protocols both of them support so that they can decide which of these to use for later (multicast) communication. This is achieved during the initial exchange of MMAP PDUs. The calling MCS provider transmits an `MMAP-CONNECT-INITIAL-RQ` PDU via the first TCP connection. This PDU contains a connection identifier which is used by the calling MCS provider to refer to this connection and a list of transport protocols (for multicast and unicast) supported by the sender. The called MCS provider responds with an `MMAP-CONNECT-INITIAL-CF` PDU that contains a list of the transport protocols supported by the callee as well as the callee's local identifier for the MMAP connection.[24] If the caller determines that the intersection of multicast transport protocols advertised by itself and the callee is empty, it closes the MMAP connection and reverts to the use of the T.123 compliant protocol profile. Otherwise, both MCS providers memorize their respective transport protocol capabilities as well as the connection identifiers.

---

[23] In the IMTC MCS API, the address parameter is structured into two parts (`transport-stack:transport-address`). Besides identifying the standard T.123 profiles, the first part of the address may also specify `mmap` to indicate that use of the multicast protocol hierarchy is preferred. This choice implies falling back to the T.123 profile for TCP/IP if the called provider does not support MMAP.

[24] The connection identifier is used for the transition from one (out of many) established MMAP connections to another to select the MMAP connection to switch to. The list of multicast transport protocols describes the capabilities of each MCS provider so that a commonly supported transport can be chosen. The current implementation of the extended MCS provider only supports MTP-2 as the single multicast and TCP as the single point-to-point transport protocol.

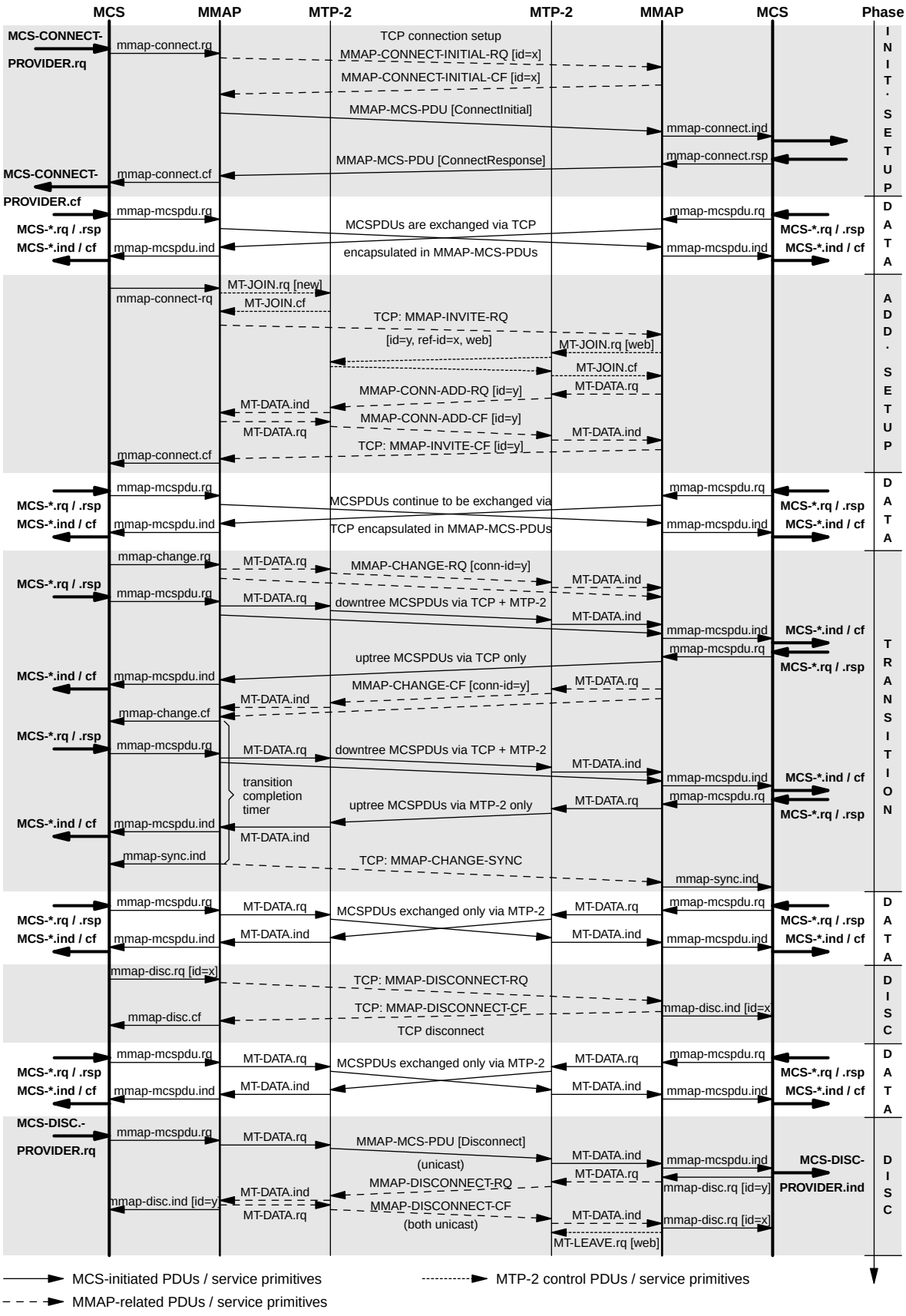


Figure 6.4: MMAP protocol operation

This completes the initial handshake across the first TCP connection. Subsequently, the MCS connection establishment protocol is carried out as described in T.125. As a result of this, up to three additional TCP transport connections may be established which together then form the initial MMAP connection which automatically becomes the *active connection* between those two MCS providers. After successful completion of the MMAP-based MCS connection setup, the two providers immediately start to exchange MCSPDUs across the initial MMAP connection which are encapsulated in MMAP PDUs (see data transmission below). In parallel, the superior of the two MCS providers determines

- whether its subordinate can be included into an MTP-2 web already in place (if any) for the respective hierarchy level, or
- if a new web needs to be created because there is not yet an MTP-2 web at this hierarchy level but using a web is justified, or
- whether the threshold for establishing and using an MTP-2 web is not yet reached.

In the latter case, creation of an MTP-2 web is delayed until enough further subordinate MCS providers supporting MMAP connect to the same superior node for the same domain so that the threshold is reached. Until then, no further actions related to the multicast transport are taken. In all three cases, for the time being MCS communication is carried out via the initial MMAP connection until the MTP-2 web is established and eventually becomes the active MMAP connection used for data transmission.

Establishing and Joining an MTP-2 Web

During the connection setup, the two newly connected MCS providers have determined that they are both capable of using MTP-2 as multicast transport protocol. However, they still have to find out whether communication via multicast is possible at all.[25] There are no mechanisms provided in MMAP to determine the availability — and possibly the quality — of multicast connectivity. Rather, this is left to MTP-2: an `MT-JOIN` request will only succeed if multicast connectivity is provided. Furthermore, an `MT-DATA-MISSED` indication or a provider-initiated `MT-LEAVE` received shortly after joining the web indicate that the connectivity is insufficient, and thus the multicast communication cannot be used. This determination process is intertwined with the setup process as described in the following.

When the superior MCS provider decides to establish an MTP-2 web to replace the initial MMAP connection, it issues an `MT-JOIN` request to its local MTP-2 entity and specifies an unused web name as well as optionally an unused multicast address,[26] and indicates that a creation of a new web shall be enforced. As soon as the web creation is complete, the MCS provider transmits an `MMAP-INVITE-RQ` PDU to those of its subordinates that shall switch to the newly established MMAP connection. This PDU contains

[25] Multicast communication may not be possible for a variety of reasons: the operating systems at the T.120 nodes may not be configured to support multicast, the physical network to which the nodes are attached may not be multicast-capable, and the routers may not provide multicast routing facilities [Ott 95e].

[26] An unused multicast address might be obtained e. g. from a multicast address allocation service (if available). A unique web name can easily be generated, e. g. from the local IP address, port number, and a time stamp.

- the transport protocol type — which is chosen out of those commonly indicated by all subordinates in the previously exchanged capability set (in this case MTP-2);[27] and
- a protocol-specific transport address to connect to and other protocol specific parameters — in the case of MTP-2, the multicast address equals the one previously picked for the web; in addition, the web name and further MTP-2 web parameters are included.

Upon reception of this PDU, the subordinate node tries to establish a connection using the transport protocol and address as specified. For MTP-2, the subordinate issues an `MT-JOIN` request to its local MTP-2 entity. If the `MT-JOIN` confirmation from the MTP-2 entity indicates that a new web has been created or that a web has been joined of which the superior node is not the master, there is no multicast connectivity to the superior node. In this case, the subordinate MCS provider issues a `MT-LEAVE` request for the web and transmits an `MMAP-INVITE-RJ` PDU to the superior. If the web indicated by the superior node is joined successfully, the subordinate node unicasts an `MMAP-CONNECT-ADDITIONAL-RQ` PDU using the MTP-2 web to the superior node that responds with an `MMAP-CONNECT-ADDITIONAL-CF`, again via MTP-2. After successful reception of the latter PDU, the subordinate node sends an `MMAP-INVITE-CF` PDU via TCP to the superior thus completing successful establishment of an MTP-2 web as a second MMAP connection belonging to the same MMAP-based MCS connection.

In principle, this procedure may be used to set up an arbitrary number of point-to-point as well as multicast MMAP connections using arbitrary protocols. The corresponding MMAP PDUs may be sent via unicast as well as multicast transport connections so that, for example, reverting from MTP-2 back to TCP is or another point-to-point transport is possible without disturbing the MCS communication.

> When an additional point-to-point connection — e. g. to use another (more suitable) transport protocol instead of TCP — shall be established, the superior node dynamically chooses a transport address upon which it waits for incoming transport connections. Then, it forwards this transport address in the `MMAP-INVITE-RQ` to the subordinate node which initiates the setup of an additional MMAP connection using the previously specified transport protocol and address and, afterwards, reports success or failure back to the superior node via the initial connection.

Out the *n* parallel MMAP connections belonging to an MMAP-based MCS connection exactly one is *active* at any point in time, all others are *inactive*. MCSPDU are only exchanged via the active connection. The procedures described in the following are used to select a new active MMAP connection and render the currently active one inactive. During this transition period, the two affected connections are *in-transit* and the rules described in the following apply to forwarding MCSPDUs across one or both of the in-transit connections.

Transport Protocol Transition Phase

Once an MTP-2 web is established and all MCS providers in a multicast area have joined it successfully, the superior MCS provider may decide to move the stream of MCSPDUs from the point-to-point MMAP connection over to the MTP-2 web: this changing of the MMAP

[27] If the intersection of the capabilities indicated by all subordinates is empty, the superior node may either decide to create several multicast transport connections with non-overlapping subsets of its subordinates, or it may set up a single multicast transport protocol for communication with all those supporting it and continue to use point-to-point connections for communication with the others.

connection is termed *transition phase*. The transition process has to be invisible to the MCS provider's protocol engine and all the other T.120 entities in the MCS domain. This means that the flow of MCSPDUs must not be delayed or even interrupted during the transition phase. As a consequence, MCSPDUs continue to flow so that the information transmission across the two connections in transit has to be synchronized in order to guarantee that MCSPDUs are neither duplicated nor lost.

In order to change the active MMAP connection to one of its subordinates, the superior MCS provider transmits an `MMAP-CHANGE-RQ` PDU both over the active MMAP connection and the MMAP connection to be activated. This MMAP PDU contains the connection identifier of the MMAP connection to switch to: this puts both connections into the *in-transit* state. After transmission of this PDU, the superior node sends MCSPDUs on both in-transit MMAP connections until the transition phase is complete.

Upon reception of the change request on both MMAP connections, the subordinate node stops sending MCSPDUs on the previously active (point-to-point) MMAP connection and transmits the MCSPDUs via the MMAP connection to be activated (e. g. the MTP-2 web) instead. As a confirmation, it sends an `MMAP-CHANGE-CF` PDU to the superior node, again across both MMAP connections in transit. This does not yet change the state of either connection; both are still in in-transit; and the subordinate accepts MCSPDUs from both of them, and eliminates duplicates locally.

When the superior node has received the change confirmation on both MMAP connections, it starts the *transition completion* timer reflecting the maximum transmission time for an MCSPDU on the web (for MTP-2, the timer is set to *retention* × *heartbeat*). This timeout mechanism is required to avoid accidental loss of MCSPDUs due to propagation delays that have been sent by other subordinates that were already part of the web.

> Consider three nodes — *A*, *B*, and *S* — with *S* being the superior node, *A* a node already part of the MTP-2 web, and *B* the node in the process of changing the MMAP connection towards the MTP-2 web. If *A* sends an MCSPDU via its active MMAP connection (i. e. the MTP-2 web) *before* B starts interpreting MCSPDUs from the multicast MMAP connection, then it must be ensured that *S* forwards this MCSPDU to *B* via the point-to-point MMAP connection; otherwise *B* would miss the MCSPDU and the MCS service guarantee would be violated. Hence, *S* must not stop forwarding MCSPDUs to *B* until it can be sure that no further MCSPDUs from *A* are outstanding to be forwarded. Due to the limited size of MCSPDUs and the resulting maximum transmission time (otherwise the MTP-2 transport has failed), this condition is satisfied after the aforementioned period of time. Refer to [Nikolaus 95b] for a more detailed description of this specific synchronization problem.

When the *transition completion* timer expires, the superior node sends an `MMAP-CHANGE-SYNC` PDU to its subordinate across the point-to-point MMAP connection, marks the MTP-2 web as active, the point-to-point connection as inactive, and stops forwarding MCSPDUs over the point-to-point connection. On receipt of the `MMAP-CHANGE-SYNC` PDU, the subordinate marks the two connections like the superior node did and stops interpreting MCSPDUs from the point-to-point transport; after the synchronization MMAP PDU, no further MCSPDUs will be arriving via this connection, anyway, so that the now inactive connection may be dropped (see ''Disconnect Handling'' below). This completes the transition phase.

The same procedure is followed to switch from the MTP-2 web back to a point-to-point MMAP connection — or, in general, to switch between any two previously established MMAP connections.

(Multicast) Data Transfer Phase

MCSPDUs to be exchanged between MCS providers interconnected via MMAP-based transport connections are encapsulated in MMAP PDUs. For all but one MCSPDU, an `MMAP-MCS` PDU is transmitted which contains one or more MCSPDUs. The single exception is the `SDrq` PDU which contains data transmitted with the `MCS-SEND-DATA` request; such MCSPDUs are encapsulated in `MMAP-MCS-DATA` PDUs for the following reason:

> In unmodified T.125, subordinates of the same superior node in an MCS hierarchy do not communicate directly with one another. `SDrq` PDUs sent from one subordinate are sent to the superior node instead, which forwards them uptree as `SDrq` PDUs and disseminates them downtree as `SDin` PDUs. Therefore, with the conventional MCS protocol, the protocol engine of an MCS provider never expects to receive an `SDrq` from a connection to an uptree node (which would constitute a protocol error).
>
> When using multicast, `SDrq` PDUs that are sent (uptree) by a subordinate node, reach the superior as well as all other subordinate nodes. To make the protocol engine of MCS work without changes, the incoming `SDrq` PDU has to be converted into the expected `SDin` PDU at all other subordinate nodes before the MCSPDU is passed to the MCS protocol engine. Using a distinct MMAP PDU type allows the recipients to identify such MCSPDUs easily without the need to do the ASN.1 decoding for all MCSPDUs. Rather, only the MCSPDU type field has to be adjusted — which may be done without decoding and re-encoding since the type field is located at a fixed position and the two PDU encodings are identical for all other fields.

For active point-to-point MMAP connections, the transmission of MCSPDUs and their interpretation is performed exactly as defined in T.125, except that MMAP is used as a transport replacing X.224/0 and RFC1006 of the T.123 profile for the Internet Protocol. If the MTP-2 multicast transport is used, transmission of MCSPDUs is done as described in the following in order to optimize network resource utilization.

The main optimization that can be achieved by transmitting MCSPDUs via a reliable multicast transport is that MCSPDUs need only be sent once per web by the sender and are no longer sent repeatedly across the same network links. The degree of network traffic reduction achieved depends on the number k out of n MCS providers in a web for which a particular MCSPDU is destined (see also section 6.2.3). This may be a single provider, a subset of the providers in the web, or all providers (except the sender):

- Transmitting an MCSPDU to a single receiver ($1 \rightarrow 1$) is accommodated by using the MTP-2 unicasting mechanism.

- Multicasting an MCSPDU to all providers in the web ($1 \rightarrow n$), is done efficiently by MTP-2 in conjunction with the underlying network multicast facilities.

- For addressing an MCSPDU to a subgroup of the providers in the web ($1 \rightarrow k$), there is a tradeoff between network (link bandwidth, number of multicast groups) and host resources (CPU and memory utilization). In principle, the following approaches are conceivable:

– Using separate webs for distinct subgroups of receivers — this approach is infeasible because of the potentially large number of multicast transport connections to be established and the resulting excessive utilization of multicast groups.[28]
– Expecting the multicast transport to handle subgroup addressing — this would do nothing but shift the problem one layer downwards. Neither MTP-2 nor any other scalable reliable multicast protocol known to the author provides such an efficient subaddressing service.
– Providing receiver-based packet filtering — this requires the receiving MCS providers to decode and interpret the MCSPDUs and discard them (at the MCS layer); thus this approach puts additional burden on the receiving side and requires the network to forward the MCSPDUs to all receivers.[29]

For MMAP, the third approach is followed. In order to reduce the processing burden on the receiving sides, the sender of an MCSPDU includes a hash value in the encapsulating MMAP PDU's header that allows the recipients to do a first approximation on whether the MCSPDU is addressed to them or not without having to perform the more expensive ASN.1 decoding of the enclosed MCSPDU.[30] At the superior MCS provider, the hash value used in MMAP is calculated from the IP addresses of the intended recipients. At the subordinates, this information is not available, so that a default hash value of all ''1''s is used instead, thus ensuring that all peer subordinates in the web will receive and interpret the MCSPDU. For details about the hash algorithm refer to [Nikolaus 95a].

Upon reception of an MMAP PDU carrying an MCSPDU, the recipients apply a specific compare function the hash value included in the MMAP header and their own IP address (of the network interface via which they are communicating with the superior). The result of this comparison indicates to each recipient whether it may immediately discard the MCSPDU. If not discarded, the MCSPDU is decoded and forwarded to the MCS protocol engine, which then decides based upon the MCSPDU type and part of its contents (e. g. the channel number) to process or to ignore the PDU.[31] The PDU is then processed as defined in T.125 with one exception: if the superior receives data PDU from one of its subordinates via the MTP-2 web, it does not forward the PDU downwards to other MCS providers connected to the same web. In order to avoid modifications to the T.125 protocol engine, forwarding such packets is suppressed by the MMAP implementation.[32]

---

[28] For *n* providers in a web, in the order of 2^n different multicast groups would be needed to accommodate all different sets of recipients.

[29] This approach may be deemed unattractive for large multicast networks such as the Internet because MCSPDUs may travel long ways through the network — potentially including congested links — just to be discarded at some receivers. On the other hand, however, burdening the multicast routers in such a network to deal with large numbers of multicast groups and the corresponding routes is no better alternative either.

[30] Another approach would have been to include the MCS channel number in the MMAP header. However, this would require that the MMAP protocol engine has access to the MCS state information base of the corresponding domain. This in turn would require intertwining the two protocols and their implementations more closely — which can be avoided following the MMAP design.

[31] Note that this does not require any changes in the MCS protocol engine: race conditions in the MCS protocol may result in an MCS provider receiving data it is not interested in so that MCS provider already has to discard unwanted MCSPDUs.

[32] This is achieved by means of the specific interface between the MCS provider and MMAP. The interface uses address lists to identify the intended recipients of an MCSPDU and also provides — in case of data MCSPDUs — information about the node from which the MCSPDU to be forwarded was received.

Disconnect Handling

With MMAP, two types of MMAP connections may exist in parallel — active and passive ones — which are disconnected for different reasons:

- Passive MMAP connections are disconnected when they are no longer needed, i. e. when the data transmission is going on via another connection and it is not expected that the flow of data will be switched back to the passive connection in question. Shutting down passive connections may only be initiated by the superior provider in order to avoid race conditions, e. g. between switching to a passive connection and closing it.
- Active MMAP connections are disconnected when the MCS connection is to be dropped; i. e. closing an active connection results from an `MCS-DISCONNECT-PROVIDER` request. If an active MMAP connection is closed, all passive connections are torn down as well. This also applies if the active MMAP connection is closed due to an error; i. e. the passive MMAP connections do not constitute a backup to deal with error situations on the active one.

At the MMAP layer, the procedures for shutting down an MMAP connection are identical for both active and passive connections but differ for point-to-point and multicast transport underlying the MMAP connections.

On a point-to-point MMAP connection, the initiating MCS provider transmits an `MMAP-DISCONNECT-RQ` PDU via the connection to be closed and sets a disconnect timer. The recipient of this PDU responds with an `MMAP-DISCONNECT-CF` and also sets a disconnect timer. If an active MMAP connection is to be disconnected, the same process is invoked in parallel for all inactive transport connections belonging to this MMAP-based MCS connection. When the initiating node receives the confirmation MMAP PDU, or when either of the timers expires, the respective provider initiates transport layer disconnects for all the transport connections of all the MMAP connections to be closed.

For an MTP-2 web underlying an MMAP connection, the exchanged MMAP PDUs are identical to those used for point-to-point connections. However, the disconnect request may be directed either to a single MCS provider or to all subordinates (if originated by the superior):

- If a single MMAP connection between the superior and a subordinate MCS provider shall be closed, the initiating provider unicasts the `MMAP-DISCONNECT-RQ` PDU to the node from which to disconnect. The receiver of this PDU responds with an `MMAP-DISCONNECT-CF`. If the initiator is the subordinate node, it sets a disconnect timer and quits the MTP-2 web after either reception of the confirmation or expiration of the timer, whichever happens first. If the initiator is the superior node, the subordinate sets its timer after responding to the `MMAP-DISCONNECT-RQ` PDU and quits the MTP-2 web when the timer expires. In both cases, the superior node does not need to perform any further actions besides sending the appropriate disconnect MMAP PDUs.

- If the superior MCS provider wants to shut down the entire web simultaneously, it multicasts the `MMAP-DISCONNECT-RQ` PDU to all subordinates. Each subordinate node then issues a `LEAVE` request to their respective MTP-2 entity and silently leaves the web without any further actions. Not transmitting the confirmation MMAP PDU avoids an acknowledgment implosion at the superior node.

A final issue to be addressed by MMAP is that MTP-2 does not provide a mechanism to detect that (the user of) an MTP-2 entity has failed or that network connectivity has been lost to (the user of) an MTP-2 entity. As the MCS protocol relies on such events being recognized by the transport, MMAP introduces a surveillance mechanism for MTP-2 webs — that might as well be used for point-to-point or other multicast transports if needed. To reduce the burden on the network, this mechanism is only applied when the MTP-2 web is the active MMAP connection and thus a failure of a node would lead to disconnection of the MMAP-based MCS connection.

The superior node supervises its subordinates by periodically multicasting `MMAP-PING-RQ` PDUs to all peers in the MTP-2 web. The frequency of these ping requests — the ping interval — is adjusted to the number of web members. After sending such an MMAP PDU, the superior node starts starts a *ping timer* equivalent to three times the maximum transmission latency plus the response interval (see below). Each `MMAP-PING-RQ` PDU contains a response filter based on IP addresses to select those subordinate providers that are expected to respond with a `MMAP-PING-CF` PDU — which is transmitted at a reserved (highest) MTP-2 priority in order to avoid that transmission of this response is blocked by other messages being exchanged in the web. The selected providers do not answer immediately but dither their responses randomly over a response interval that is also specified in the request MMAP PDU.[33] If a subordinate node fails to respond before the *ping timer* at the superior node expires, the respective subordinate is considered to have failed and the MMAP-based MCS connection to this node is closed.

Subordinate MCS providers detect failure of the superior node through the master recovery procedure of MTP-2. If the superior node — that hosts the master of the MTP-2 web — fails or becomes unreachable for one or more MTP-2 entities, these automatically invoke the master recovery procedure. Completion of the master recovery leads to a `MASTER-CHANGE` indication being issued by the respective MTP-2 entities to all affected MCS providers. The MMAP layer of an MCS provider transforms this indication into a disconnect indication for the MCS connection and thus the MCS provider is disconnected.

Summary: Utilization of Point-to-Point and Multicast Transport Services

Table 6.1 gives an overview of the services offered by TCP and MTP-2 and shows which of these are required (‘‘Req.’’) for the functioning of MMAP. Furthermore, the table shows how MMAP makes use of the additional services offered by MTP-2.

MMAP expects the same reliable point-to-point transport service interface as MCS does but it does not require packet boundaries to be preserved by the transport. Any point-to-point transport protocol offering these services may be used as alternative transport underneath MMAP. However, the use of TCP is mandated to have a common baseline for the establishment of the initial MMAP transport connection.

A reliable multicast transport service is expected to provide means for joining and leaving a multicast group as well as for data transmission. This is similar to the service offered by IP multicast, except that reliability is added as service requirement.[34]

[33] The superior MCS provider has knowledge of all of its subordinates and uses the ping interval, the response interval, and the response filter to trade off network utilization against the time required to determine that a subordinate has failed.

[34] This is a subset of the services typically offered by reliable multicast transport protocols in use or under

Protocol	Transport Service		Req.	Use by MMAP
TCP	T-CONNECT	request, indication response, confirmation	•	establishment of initial connection
	T-DISCONNECT	request indication	• •	drop a transport connection detect peer or provider-initiated disconnect
	T-DATA	request indication	•	data transmission
MTP-2	MT-JOIN	request confirmation	•	create and join a web
	MT-LEAVE	request indication	• •	leave a web detect a provider-initiated disconnect
	MT-DATA	request indication	•	data transmission
	MT-DATA-MISSED	indication		→ detect transport disconnect
	MT-DATA-NOT-ACCEPTED	indication		→ detect transport disconnect
	MT-MASTER-CHANGE	indication		→ detect transport disconnect
	MT-MASTER-PASS	request, indication		*not used*

Table 6.1: Point-to-point and multicast transport services used by MMAP

If reliability can only be guaranteed with a certain probability — as is the case with MTP-2 —, additional services are required to detect failures of the multicast transport which then are interpreted as disconnection from the transport connection: for MTP-2, these are the `MT-DATA-MISSED` and `MT-DATA-NOT-ACCEPTED` indications.

Finally, MMAP exploits the `MT-MASTER-CHANGE` indication to detect the failure of the master. If no service is available from the transport to recognize failure or disconnection of the superior node, the ping mechanism of MMAP needs to be extended to enable all subordinates to determine loss of connectivity to the superior node.[35]

6.2.3. Optimizations for the Multipoint Communication Service

This section so far has laid the foundations for running the MCS protocol on top of multicast-capable networks by providing the expected transport services for setup and teardown of MCS connections plus the reliable multicast services for data transmission. This subsection now addresses the changes to the MCS procedures required to allow the MCS provider to exploit the services provided by MMAP and MTP-2 for efficient transmission of MCSPDUs.

---

study today. Also, similar services are included in the ISO and ITU-T work on a (reliable) multicast service definition [ITU-T X.6] [Moulton 94].

[35] This is easily achieved by adding a parameter to the `MT-PING-RQ` in which the superior node announces the current ping interval. If the subordinates do no receive another ping request within this interval plus the maximum transmission latency, they must assume that the superior node has failed and locally generate disconnect indications.

The starting point is a review of the MCS protocol to determine how the various MCSPDU types are distributed and which underlying transport services have consequently to be used for their transmission. In essence, five different ways of sending MCSPDUs within an MCS domain can be distinguished (refer also to section 5.1.2):[36]

1) *Subordinate → superior node*

   A subordinate provider sends a MCSPDU to the superior node that either processes it or forwards the PDU uptree. This is typical for all request MCSPDUs — including data PDUs that are dealt with separately as items 4) and 5).

2) *Superior → 1 subordinate*

   The superior node at a hierarchy level transmits an MCSPDU to exactly one subordinate. All confirmation MCSPDUs are sent this way back to originator of a request.

3) *Superior → k subordinates*

   The superior node sends the same MCSPDU to *k* out of *n* subordinates (with $1 \leq k \leq n$). This scheme is used for virtually all indication MCSPDUs — again including data PDUs. A special case is the MCSPDU carrying information for the `MCS-DETACH-USER` indication (`DUin`) which is always sent to all *n* subordinates.

4) *Subordinate → superior → k subordinates*

   An MCSPDU for non-uniform data transmission — `SDrq` (send data request) — is transmitted to the superior which then forwards it further up and sends it — as `SDin` (send data indication) — to all subordinates (except the sender) that have members in the channel the data PDU is addressed to. Except for the PDU type, the contents of the PDU sent upwards and distributed downwards are identical.

5) *Subordinate → superior → ... → Top MCS Provider → ... → superior → k subordinates*

   An MCSPDU for globally ordered data transmission — `UDrq` (uniform data request) — is sent to the superior node and forwarded further uptree until it reaches the Top MCS Provider which then initiates the dissemination downtree — as `UDin` (uniform data indication). Again, the contents of the PDUs traveling uptree and downtree are identical except for the PDU type field.

For the first two cases, no optimizations using a multicast transport are possible: the respective MCSPDUs are transmitted using the MTP-2 unicasting service. Indication MCSPDUs (item 3) are transmitted using the MTP-2 multicast service with the hash value calculated by the IP addresses of the intended *k* recipients. Globally ordered data transmission (item 5) is handled as combination of items 1) and 3): the `UDrq` PDU is first forwarded to the Top MCS Provider via unicasting, and is then disseminated as a `UDin` PDU.[37] Thus, downtree distribution of control and

[36] Six MCSPDU types are not covered in the following discussion because these MCSPDUs are exchanged for establishing and tearing down MCS connections. Both actions constitute point-to-point interactions between directly interconnected MCS providers and hence transmission of these MCSPDUs via multicast is not supported — for connection setup this is not even possible because the multicast transport connection has not been created at the time the respective MCSPDUs are exchanged.

[37] If the entire MCS domain consists of a single MTP-2 web as in figure 6.1b or the Top MCS Provider and all its immediate subordinates are members of the same MTP-2 web, the global ordering service of

particularly uniform data indication MCSPDUs via multicast is one part of the MMAP optimizations.

The major optimization is achieved for data transmissions via `MCS-SEND-DATA` requests by means of the following procedure:

- A subordinate node transmits data contained in an `SDrq` MCSPDU to the web and uses a hash value of all '1's. As described in the previous subsection, all subordinate receivers of `SDrq` PDUs convert it locally to an `SDin` PDU before handing it to their MCS protocol engine.
- The superior node receiving an `SDrq` PDU from within MTP-2 web forwards the MCSPDU uptree and (as an `SDin` PDU) to those subordinates that are connected through point-to-point connections or joined to a different web. It does not re-send the MCSPDU to the same web.
- The superior node transmits data originated from outside the web[38] contained in an `SDin` MCSPDU via multicast to all subordinates in the web indicating the intended recipients by means of the hash value as done for item 3).

Table 6.2 on the next page gives an overview for how the individual MCSPDUs are transmitted using the MMAP protocol hierarchy.

6.3. Integration with the MCS Provider Implementation

The previous section has described the components of the multicast transport protocol hierarchy depicted in figure 6.3 on page 178: TCP, the multicast transport protocol MTP-2, the adaptation protocol MMAP, and the refinements to the transmission and forwarding procedures for MCSPDUs. This section addresses how these extensions are integrated into the MCS provider.

As already pointed out in subsection 6.2.1, the MTP-2 protocol engine is implemented as a separate entity — a STREAMS module or a separate process — and its services are accessed through a well-defined API which resembles the BSD *socket* interface [Leffler *et al.* 89]. From an MCS provider implementation point of view, MTP-2 is in principle like any other transport protocol, except that it supports multicasting and multiple priorities. As the interface to the MTP-2 API, a new connection object class that deals with the specifics of MTP-2 is introduced at the connection layer of MCS: the `MTPConnection`. In contrast to MTP-2, the MMAP protocol engine is not implemented as a separate entity but is rather fully integrated into the MCS provider, for efficiency reasons and because MMAP is closely tied to the MCS protocol anyway. The integration of MMAP affects the MCS provider at both the connection and the communicator layer: at the connection layer, a new `TCPConnectPort` object is required to accept incoming MMAP connection setup requests; and new `TCPConnection` and `MTPConnection` objects are needed for the exchange of MMAP PDUs. At the communicator layer, an `ConnectMMAPCommunicator` and the corresponding `ConnectMMAPCoder` are introduced for handling setup of initial and additional MMAP connections. For communication within an MCS domain, two further new objects

MTP-2 could be used for even more efficient communication. However, this optimization is not included in MMAP in order not to require global ordering from the reliable multicast transport and thus keep it applicable on top of more multicast transport protocols.

[38] This includes data originated from one of its local T.120 applications, data received from the next higher node in the MCS domain hierarchy, and data received from a subordinate connected via a point-to-point connection or via a different web.

MCSPDU	MCSPDU Name	Direction		Transmission via
PDin	Plumb domain indication	downtree	1→*k*	multicast
EDrq	Erect domain request	uptree	1→1	unicast
MCrq	Merge channel request	uptree	1→1	unicast
MCcf	Merge channel confirmation	downtree	1→1	unicast
PCin	Purge channel indication	downtree	1→*k*	multicast
MTrq	Merge token request	uptree	1→1	unicast
MTcf	Merge token confirmation	downtree	1→1	unicast
PTin	Purge token indication	downtree	1→*k*	multicast
AUrq	Attach user request	uptree	1→1	unicast
AUcf	Attach user confirmation	downtree	1→1	unicast
DUrq	Detach user request	uptree	1→1	unicast
DUin	Detach user indication	downtree	1→*n*	multicast
CJrq	Channel join request	uptree	1→1	unicast
CJcf	Channel join confirmation	downtree	1→1	unicast
CLrq	Channel leave request	uptree	1→1	unicast
CCrq	Channel convene request	uptree	1→1	unicast
CCcf	Channel convene confirmation	downtree	1→1	unicast
CDrq	Channel disband request	uptree	1→1	unicast
CDin	Channel disband indication	downtree	1→*k*	multicast
CArq	Channel admit request	uptree	1→1	unicast
CAin	Channel admit indication	downtree	1→*k*	multicast
CErq	Channel expel request	uptree	1→1	unicast
CEin	Channel expel indication	downtree	1→*k*	multicast
SDrq	Send data request	uptree	1→*k*	multicast
SDin	Send data indication	downtree	1→*k*	multicast
UDrq	Uniform send data request	uptree	1→1	unicast
UDin	Uniform send data indication	downtree	1→*k*	multicast
TGrq	Token grab request	uptree	1→1	unicast
TGcf	Token grab confirmation	downtree	1→1	unicast
TIrq	Token inhibit request	uptree	1→1	unicast
TIcf	Token inhibit confirmation	downtree	1→1	unicast
TVrq	Token give request	uptree	1→1	unicast
TVin	Token give indication	downtree	1→1	unicast
TVrs	Token give response	uptree	1→1	unicast
TVcf	Token give confirmation	downtree	1→1	unicast
TPrq	Token please request	uptree	1→1	unicast
TPin	Token please indication	downtree	1→*k*	multicast
TRrq	Token release request	uptree	1→1	unicast
TRcf	Token release confirmation	downtree	1→1	unicast
TTrq	Token test request	uptree	1→1	unicast
TTcf	Token test confirmation	downtree	1→1	unicast

Table 6.2: Transmission of MCSPDUs with MMAP

are needed: the `DomainMMAPCommunicator` implements the MMAP protocol engine and the `DomainMMAPCoder` performs MCS and MMAP PDU encoding.

Finally, some modifications are necessary to a few object classes of the MCS provider to allow for the integration of the new objects. These modification are done carefully in order not to affect the basic functioning of the MCS protocol engine. The changed object classes are the `BaseCommunicator` object class — from which all other communicator classes are derived — and the `T123ConnectCommunicator` — the modified version which is termed `MCSConnectCommunicator`.

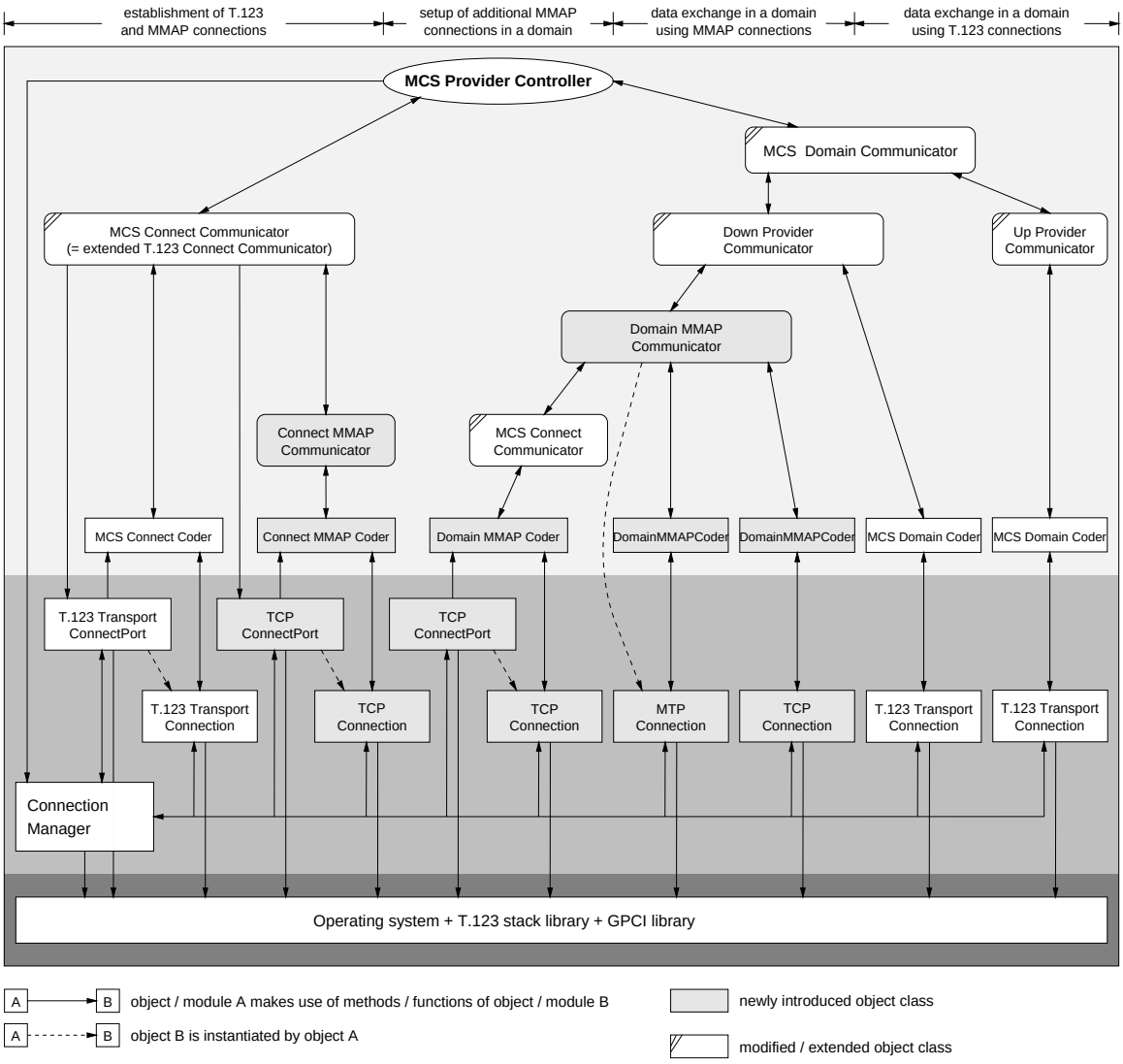


Figure 6.5: Modifications and extensions to the MCS provider

Figure 6.5 gives an overview of the structure of the modified MCS provider. MMAP-specific object classes are grey-shaded instead of white, modified object classes are marked with two diagonal lines in their upper left corner. In order to keep the figure simple, communicators and connection objects only needed for local communication are deliberately not shown because they are

not affected at all by the extensions. For the same reason, the corresponding connection and communicator objects are not considered at all in this subsection.

The following two subsections provide an overview of the changes and enhancements to the connection and the communicator layer of the MCS provider, respectively.

6.3.1. Connection Layer

The tasks to be performed at the connection layer for the multicast extensions include a) creating outgoing and accepting incoming MMAP connections via TCP and b) creating, joining, and leaving MTP-2 webs, and exchanging MMAP PDUs c) through MTP-2 webs as well as d) across TCP connections. The objects responsible for these four tasks are:

a) The `TCPConnectPort` — this object class is largely identical to the `T123ConnectPort` used for the T.123 Internet profile with two exceptions: different TCP port numbers are used and a different transport stack is instantiated when a connection is established.[39] Because of this similarity, the `TCPConnectPort` is not described any further.

b,c) The `MTPConnection` — This object class is newly introduced to deal with multicast communication on the basis of MTP-2. An `MTPConnection` consists of an `MTPConnectionInterface`, an `MTPConnectionHandler`, and an `MTPConnectionDemultiplexer` (figure 6.6b).

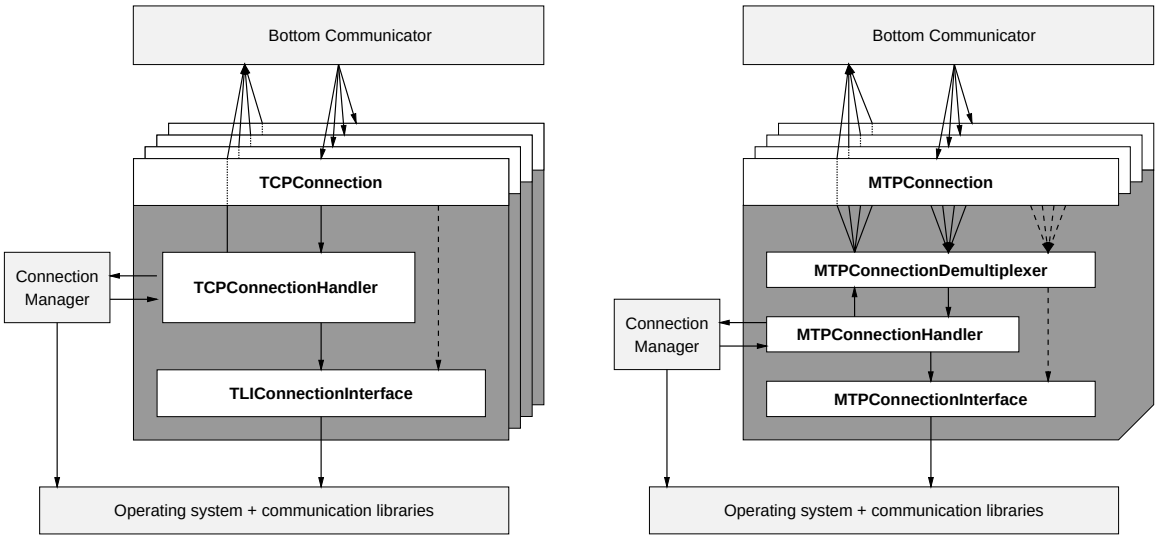
d) The `TCPConnection` — Like the `TCPConnectPort`, this object is largely similar to its counterpart (the `T123TransportConnection` used for communicating via a T.123 protocol profile). The `TCPConnection` object provides the methods to access the TCP services through the `TLIConnectionInterface` object. Furthermore, it buffers incoming and outgoing data in the `TCPConnectionHandler` object which also interfaces to the `ConnectionManager` (figure 6.6a).

Figure 6.6 depicts the internal structure of the two `Connection` objects used by MMAP. Because of the similarity between the structures of the `TCPConnection` and the generic `Connection` object structure presented in section 5.3, no further explanations the are given here. The remainder of this subsection is devoted to the description of the concepts of the `MTPConnection`.

The main difference between the objects representing an MTP-2 web as a transport connection and those representing conventional T.123 transport connections is that `MTPConnections` have been extended to be capable of handling multiple priorities. As depicted in figure 6.6b, the `MTPConnection` object is partially similar to a conventional `Connection` object in that it provides the same interface to the `BottomCommunicator` for communication with the communicator layer. However, internally it does neither interface directly to the `MTPConnectionHandler` nor to the `MTPConnectionInterface` but interacts with a new object, the `MTPConnectionDemultiplexer`, instead.

---

[39] In contrast to the conventional T.123 transport hierarchy for TCP/IP, the one used underneath MMAP does not include the X.224 protocol and the packet framing defined in [RFC 1006].



a) Connection object for TCP-based MMAP communication

b) Connection object for MTP-2-based MMAP communication

Figure 6.6: Connection objects for MMAP-based communication

In the MCS provider, each `BottomCommunicator` manages as many `Connection` objects for a single MCS connection as different priorities are negotiated for the respective MCS domain. This is required since none of the T.123 transport protocol hierarchies supports multiplexing several priorities into the same transport connection. With MTP-2, however, data of all four priorities is transmitted through the same web and thus only a single MTP-2 transport ‘‘connection’’ is needed. The `MTPConnectionDemultiplexer` is introduced to combine up to four virtual `MTPConnections` and to map interactions with these connections onto a single MTP-2 web:

- data transmission initiated by the `BottomCommunicator` via any of the four `MTPConnections` is mapped onto an MTP-2 data transmission request using one MTP-2 stream per MCS priority and also requesting the MTP-2 token according to the MCS priority;
- incoming data units are forwarded according to their MCS priority (which is determined from the MTP-2 stream) to the corresponding `MTPConnection` object; and
- all other events received from the MTP-2 protocol entity or from the `ConnectionManager` are either handled within the `MTPConnectionHandler` and thus need no forwarding, or are mapped to an indication on the highest priority `MTPConnection` — which is sufficient for all other event types.

Thereby, the `MTPConnectionDemultiplexer` hides the aggregation of up to four priorities into a single MTP-2 web from the `BottomCommunicator`.

Like the `MTPConnectionDemultiplexer`, the remaining two objects belonging to an `MTPConnection` deal with multiple priorities. The `MTPConnectionHandler` performs buffering and flow control of incoming and outgoing data units for up to four priorities and forwards notifications about received data or flow control situations on a per priority basis to the demultiplexer which then notifies the respective `MTPConnection`. Finally, the `MTPConnectionInterface` abstracts from the interface offered by the MTP-2 API library and provides a similar interface to MTP-2 as the other `ConnectionInterface` objects do for point-to-point connections; the

major difference being that the `MTPConnectionInterface` supports the notion of multiple priorities for all communication methods.

If the MCS provider is to be extended to run MMAP on top of other point-to-point or multicast transport protocols, new object classes interfacing to (or even entirely implementing) the respective protocol have to be provided. These object classes are likely to be of a similar structure as the `TCPConnection` or the `MTPConnection`.

6.3.2. Communicator Layer

While the connection layer provides all the necessary objects for establishing MMAP connections and exchanging MMAP PDUs, the MMAP protocol engine and its embedding into the MCS provider structure are implemented at the communicator layer. The following functions related to MMAP have to be performed:

1) initiating and accepting *initial* point-to-point MMAP connections (that signal the establishment of a new MCS connection);
2) initiating and accepting *additional* point-to-point MMAP connections with a different (more suitable) transport protocol as a replacement for the currently active point-to-point or multicast MMAP connection;
3) joining MTP-2 webs or other multicast transport connections with a different (more suitable) transport protocol as a replacement for the currently active point-to-point or multicast connection; and
4) executing the MMAP protocol within an MCS domain to exchange MCSPDUs as well as to handle setup, changes, and shutdown of connections for both point-to-point and multicast connections.

Referring to figure 6.5 on page 203, the changes and extensions to the communicator layer affect the left hand side dealing with connection setup (for item 1) as well as the right hand side handling communications within an established MCS domain (items 3 and 4). Furthermore, support for connection setup within the context of an established domain has been added to accept additional connections paralleling already existing ones (item 2).

These changes are elaborated on in the following: first, the objects dealing with the establishment of initial MMAP connections are described (item one); then, those objects required for the remaining MMAP protocol operation within a domain are introduced (handling items two through four).

Establishment of Initial MMAP Connections

As described in section 5.4, the establishment of new MCS connections is controlled by the `T123ConnectCommunicator` which initiates outgoing connections, accepts incoming ones, and executes the MCS connection establishment protocol. For use with MMAP, the `T123Connect-Communicator` has been extended to support other protocol hierarchies besides those defined in T.123. The resulting extended connect communicator is capable of instantiating child communicators other than `MCSConnectCoder`; which communicator type is actually created as a child for further use with an incoming or outgoing transport connection being established depends on the transport protocol to be used: for T.123 transport connections, the `MCSConnectCoder` continues to be used; for MMAP transport connections, a `ConnectMMAPCommunicator` is created instead.

As the modified connect communicator is fully generic with respect to the transport protocols on which it is able to establish connections, it is termed `MCSConnectCommunicator`.

When instantiated by the `MCSConnectCommunicator`, the `ConnectMMAPCommunicator` firstly creates a `ConnectMMAPCoder` as its interface to the connection layer to which the previously established `TCPConnection` is attached. Then, the `ConnectMMAPCommunicator` executes those parts of the MMAP protocol necessary to set up initial MMAP connections.[40] This includes the initial exchange of `MMAP-CONNECT` PDUs and the capability negotiation for the supported transport protocols underneath MMAP. Furthermore, it forwards the initial MCSPDUs across the transport connections. The `ConnectMMAPCommunicator` conceals the MMAP protocol actions from the `MCSConnectCommunicator` so that the latter runs the MCS setup protocol across MMAP-based transport connections as it does on top of transport connections conforming to a T.123 protocol profile.

The `ConnectMMAPCoder` provides the interface to the connection layer and performs ASN.1 encoding of those MCSPDUs used during connection setup — as does the `MCSConnectCoder` from which it is derived. In addition, the `ConnectMMAPCoder` implements the encoding functions for MMAP PDUs including the encapsulation of MCSPDUs in MMAP PDUs.

MMAP Protocol Operation within an MCS Domain

All communication within an MCS domain is under control of the `MCSDomainCommunicator` which also contains the MCS protocol engine. Communication with non-extended superior and subordinate MCS providers is done via the `UpProviderCommunicator` and the `DownProviderCommunicator`, respectively, which eventually interface to the connection layer via the `MCSDomainCoder` objects. While only a single (uptree) MCS connection may be attached to the `UpProviderCommunicator`, the `DownProviderCommunicator` is responsible for an arbitrary number of (downtree) connections. The MCS protocol engine emits and forwards MCSPDUs by passing them via the `UpProviderCommunicator` and/or the `DownProviderCommunicator` and the `MCSDomainCoder` objects to the `T123Connections`.

With the introduction of MMAP, different MCS connections may use different types of transport connections (T.123 vs. MMAP). Furthermore, an MMAP-based transport connection either is used to communicate with a single MCS provider if it is a point-to-point (TCP) connection; or, it is used to communicate with an arbitrary number of peer MCS providers provided that it is an multicast connection (an MTP-2 web). Finally, several transport connections may connect the same two MCS providers in parallel for the same MCS domain but still represent a single MCS connection: in this case, either one connection is active and all others are passive or two connections are in-transit.[41] These extensions require additional procedures to be implemented in the MCS provider;[42] this integration is described in the following.

---

[40] The `ConnectMMAPCommunicator` also supports setting up additional MMAP connections in the context of an MCS domain and is re-used in conjunction with the `DomainMMAPCommunicator`. This is discussed below.

[41] This applies to the uptree connection attached to the `UpProviderCommunicator` as it does for the `DownProviderCommunicator` with the exception that — from an MCS protocol point of view — at most one uptree connection exists connecting to exactly one superior MCS provider.

[42] Note that the aforementioned procedures for dealing with T.123 transport connections are not affected.

Two conceptual modifications have been made to the communicator base class in order to accommodate the functionality added by MMAP.

- First, the method for instantiating child communicators has been extended to base the actions taken for dealing with a new MCS connection — e. g. to select the communicator object to be created (if any) — on the protocol type of the transport connection; this allows to handle T.123 and MMAP-based transport connections differently.
- In addition, a method for forwarding MCS protocol elements has been added that allows addressing groups of subordinate providers in a single request rather than only individual ones; this provides a means to have multiple MCS providers connected via a single MTP-2 web and address an MCSPDU to an arbitrary subset of those.

As all communicators are derived from the communicator base class, these additions affect all objects at the communicator layer except for the `MCSProviderController`. Apart from modifications to make use of the new communicator features, the `MCSDomainCommunicator`, `DownProviderCommunicator`, and `UpProviderCommunicator` have not been changed at all. All other extensions required for the integration of MMAP — such as handling multiple parallel MMAP connections as part of the same MCS connections — are implemented by adding new object classes: the `DomainMMAPCommunicator`, the `DomainMMAPCoder`, and the `ConnectMMAPCommunicator` the latter of which is also used for setting up initial MMAP connections and has already been introduced.

While for T.123 transport connections, the `MCSDomainCoder` object interfacing to the connection layer is directly connected to the `UpProviderCommunicator` or the `DownProviderCommunicator`, for MMAP connections, an additional communicator in-between is needed: the `DomainMMAPCommunicator`. At most two such communicators may exist per domain, one for the uptree connections and one for all downtree connections.[43] The corresponding `DomainMMAPCommunicator` is created as soon as the first MMAP transport connection is established in the respective direction. This and all further MMAP transport connections are then attached to the `DomainMMAPCommunicator`.

The `DomainMMAPCommunicator` implements the MMAP protocol engine and initiates all protocol actions as described in subsection 6.2.2 except the setup of initial MMAP connections: establishment of additional transport connections, changing from one transport connections to another, transmission of MCSPDUs, synchronization of streams of MCSPDUs when changing the transport type, and teardown of active or passive MMAP connections.[44] In particular, the `DomainMMAPCommunicator` deals with multiple parallel MMAP transport connections to a peer MCS provider and realizes a simple configurable policy for selecting one of these for actual data transmission. It entirely conceals the existence of multiple parallel transport connections per MCS connection so that no general extensions to communicator objects are required to accommodate this feature.

---

[43] Note that figure 6.5 (page 203) depicts the `DomainMMAPCommunicator` only attached to the `DownProviderCommunicator` (because the uptree connection in the figure is a conventional T.123 connection).

[44] Note that, following the MMAP protocol specification, the behavior of the MMAP protocol engine depends on whether it is the superior or the subordinate node at the MCS hierarchy level. If the `DomainMMAPCommunicator` is attached to the `UpProviderCommunicator` it acts as the subordinate, otherwise as the superior node.

For executing the MMAP protocol, the DomainMMAPCommunicator interacts with three different types of objects:

- a ConnectMMAPCommunicator is used for establishing additional TCP connections;
- MTPConnection objects are instantiated directly to create and join MTP-2 webs; and
- DomainMMAPCoder objects are used for information exchange via both TCP connections and MTP-2 webs.

Internally, the DomainMMAPCommunicator groups TCP and MTP-2 transport connections according to the peer MCS provider(s) they connect to and keeps track of their respective state (active, passive, in transit).

The MCSConnectCommunicator, which has been described above in conjunction with the setup of initial MMAP connections, is also used to establish additional point-to-point MMAP connections. It instantiates a single TCPConnectPort object that is used to accept incoming and create outgoing TCP connections. This TCPConnectPort is parameterized to use a different port number than is used for setup of initial MMAP connections. For each incoming or outgoing connection, the TCPConnectPort instantiates a TCPConnection object and the MCSConnectCommunicator creates a DomainMMAPCoder (which is only used for the setup and deleted afterwards). The DomainMMAPCommunicator then executes the MMAP setup protocol for additional transport connections. If the setup is successful, the DomainMMAPCommunicator then attaches the TCPConnection to an (optionally newly created) DomainMMAPCoder and adds this initially passive connection to the existing (group of) connection(s) leading to the same peer MCS provider. Thereby, the new connection becomes available for later switching to it for transmission of MCSPDUs.

If a new multicast transport connection is to be established or an existing one is to be joined, the DomainMMAPCommunicator instantiates a new MTPConnection object and then attaches it to the optionally newly created DomainMMAPCoder to be associated with the MTP-2 connection. If a superior node wants to create an MTP-2 based MMAP connection to a peer but use an existing MTP-2 web — which is the case for all but the first MTP-2-based MMAP connection to a subordinate node — these steps are not needed and the connection ‘‘establishment’’ consists only of the creation of internal state. In both cases, like for TCP-based MMAP connections, then the DomainMMAPCommunicator executes the MMAP connection setup protocol, marks the MTP-2 connection as passive, and adds it to the (group of) existing connection(s) to the same peer.

The final component needed are DomainMMAPCoder objects one of which is instantiated for each MMAP transport connection. These are BottomCommunicators and thus provide an interface to the connection layer. They perform the ASN.1 encoding of MCSPDUs and the encoding of MMAP PDUs including the encapsulation of MCSPDUs.

Distributing MCSPDUs to the subordinate nodes is done by the DomainMMAPCommunicator based on the state of the various transport connections attached to it via the DomainMMAPCoder objects: data is distributed to all subordinate MCS providers specified by the MCSDomainCommunicator and their respectively active or in-transit connections is used for transmission. MCSPDUs received from the peers are filtered according to the MMAP protocol specification to eliminate duplicates before they are eventually passed to the MCSDomainCommunicator via the Up- or DownProviderCommunicator.

6.4. Summary

This chapter has introduced the multicast extensions that have been developed for the Multipoint Communication Service of the T.120 series of ITU-T recommendations in order to increase the scalability of the T.120 infrastructure when applied in multicast-capable network environments. The adaptation protocol MMAP enables an MCS provider to run on top of a reliable multicast transport protocol without requiring changes to the MCS protocol and without harming backward compatibility with non-extended MCS providers. As at the time of writing no standardized reliable multicast transport protocol is available — and in the near future none is expected to be —, the multicast transport protocol MTP-2 developed in the context of this thesis is used as the transport platform for MCS since it approximately meets the needs of MMAP.

The protocols required for the MCS extensions are in part implemented separately and in part integrated into the MCS provider itself. MTP-2 is implemented as a separate process and is also available as a STREAMS module with an API similar to the BSD *socket* interface [Leffler *et al.* 89]. All other components are incorporated into the object hierarchy of the MCS provider: additional objects at the connection layer provide the necessary interfaces to MTP-2 and TCP, objects at the communicator layer implement the MMAP protocol engine as well as the PDU encoding for MMAP, and some modifications to other object classes enable integration of conventional T.123 connections and MMAP-based ones. The implementation of MMAP comprises a total of approximately 9,000 lines additional C++ code in the MCS provider (for both connection and communicator layer) plus the changes to some object classes of the original MCS provider to adapt to the conceptual changes. The implementation of MTP-2 amounts to roughly 15,000 lines of code for both the application process and the API library.

The concept of multicast extensions to the MCS presented in this chapter has been designed and implemented in a proprietary fashion [Ott / Bormann 94] [Nikolaus 95a] [Nikolaus 95b] as no standardized mechanisms — neither for MTP-2, nor for MMAP, nor for the MCS modifications — have been in place. However, the approach described in this chapter has been fed into the standardization process within ITU-T [Ott 95d] [Ott 96b] [Ott 96c] to initiate work on a standardized (set of) protocol(s) — which has finally succeeded.

Within the T.120 working group of the ITU-T, the need for multicast extensions to the MCS (as well as for a more scalable T.124 protocol) has been officially recognized in summer 1996 [DeGrasse / Lyons 96a] [DeGrasse / Lyons 96b] and revisions of the recommendations T.123 and T.125 as well as the required adaptation protocol, T.MAP [Galvin 97], are being worked on. At the time of writing, these efforts are nearing completion from a conceptual point of view; nevertheless, a lot of details still need to be worked out.

As soon as the standardization work is finalized in the ITU-T (the goal is a draft ready for decision by the end of March 1997), the proprietary mechanisms in MCS will be replaced by the standardized protocols. The internal structure of the MCS provider does not need to be redesigned for this change because all necessary mechanisms are already in place and the MCS implementation is sufficiently generic. In particular, the protocols under standardization are largely based on the ideas and designed along the lines of MMAP and thus their concepts are largely similar to the ones developed here.

7

Implementation of the GCC Provider

The previous two chapters have dealt with the design and implementation of the MCS provider as well as with extensions to the MCS protocol to make it more efficient in multicast environments. The MCS service provides multipoint communication and synchronizations services and thus covers most of the conferencing-independent parts of the teleconferencing infrastructure. This chapter now addresses the implementation of the GCC service provider that offers the conference-specific control and internal management functionality within the T.120 series of recommendations. As in chapter five, a more detailed overview of the T.124 services and protocol characteristics is given first. Afterwards, the structure of the GCC provider is outlined, followed by detailed descriptions of its main components. Finally, a summary including some numbers on the implementation concludes this chapter.

7.1. The ITU-T Recommendation T.124

This section first describes the services offered by T.124 to the T.120 applications and then the protocol mechanisms employed underneath to realize the services in order to provide the foundation for the subsequent description of the implementation details.

7.1.1. GCC Service Overview

As already introduced in section 4.1.3, the GCC provider makes use of the MCS service for communications with its peers, and provides conference control and internal management functionality. Like MCS, the GCC employs a centralized model with a Top GCC Provider performing central coordination functions to avoid race conditions. The Top GCC Provider of a GCC conference is co-located with the Top MCS Provider and typically resides on an MCU, except in case of point-to-point calls where no MCU is involved at all. The Top GCC provider maintains several databases containing static and dynamic information about the conference (see below), arbitrates conference resources, admits participants to the conference, and ensures adherence to the conference policy.

Four types of data structures are maintained by GCC per conference: the *conference profile* that defines the general characteristics of the conference; the *conference roster* that contains information about all the conference participants; the *application roster* that keeps track of the conference applications and their capabilities; and the *application registry* that is used to identify resources utilized by applications and to store parameters of application sessions.

Creating, running, and terminating the conference as well as access to the databases is done through a total of 39 services defined in the T.124 specification. These GCC services are grouped into six functional classes: conference establishment and termination, conference roster, application roster, application registry, conference conductorship, and miscellaneous functions. Out of these, conference establishment and termination as well as parts of conference conductorship and miscellaneous functions are user-visible conference control services as defined in section 3.4; all others are internal management services.

In the remainder of this subsection, the services of the various service classes are presented along with the respective data structures. Beforehand, an introduction is given to the GCC conference profile, the parameters of which impact several of the service classes.

The GCC Conference Profile

The characteristics of a GCC conference are defined by the *conference profile* in which all the parameters relevant to GCC are specified. The conference profile is defined upon creation of the conference and may not be changed later during the conference. The conference profile contains the following pieces of information:

- The conference is identified by its (alpha)numeric *name* (together with an optional *modifier* to ensure uniqueness) with a textual conference *description* intended for human perception.
- In the conference profile, a *password* may be specified in order to prevent unauthorized access to the conference.[1] Furthermore, a conference may be defined to be *unlisted* which means that the conference and its profile are invisible to all who have not already joined the conference — the contrary are *listed* conferences information about which can be obtained upon request (e. g. from the MCU with the Top GCC provider).
- A conferences is either *conductible* meaning that a participant may become the conference conductor or *non-conductible* meaning that a conductor role does not exist for the conference.
- The *termination method* defines whether the conference remains in place until manually terminated (even if no participants are joined to it) or whether the conference ceases to exist as soon as the last participant leaves (automatic termination).
- Finally, several privilege lists defining permissions to invoke certain GCC services (otherwise only available to the convener) may be specified. These are defined independently for conducted and non-conducted mode of operation: for the conductor (if available) and all other participants, respectively.

[1] Note that the current (first) revision of T.124 does not provide for the standardized specification of authentication algorithms other than plain text passwords. Nevertheless, the corresponding services for joining a conference already provide hooks for *n*-way authentication schemes, and revision two of T.124 is expected to address these open issues. The IMTC has defined some authentication algorithms in its API specification for GCC [Braun 96a] to be used in the interim.

The conference profile is created along with the conference and is destroyed when the conference terminates. The GCC profile deliberately does not include any type of information relevant to advance reservation and scheduling of conferences, nor does it reference any other personal identification information, e. g. in order to provide services for personalized permissions or role assignments.[2] Referring to personal information would require widely deployed and trusted directory and public key management services which are virtually non-existent at the time of writing. As T.124 is orthogonal to such infrastructure services, T.124 conferences can be deployed without these services in place but will benefit from their availability — on a private or public basis — in the future.

Note also that the conference policies that can be expressed based on the plain GCC conference profile are very limited — which is again partially due to the non-personalizability. However, GCC provides several hooks that allow to implement more elaborate conference policies on top of GCC.

Conference Establishment and Termination Services

GCC conferences are created in either of two ways: at a remote T.120 node[3] upon establishment of a T.120 connection by using `GCC-Conference-Create`; this service automatically joins the calling and the called node into the newly created conference; or at the local GCC provider without the involvement of GCC communication services and without including a second party into the conference upon creation.[4] The initiator of the conference creation is referred to as the conference *convener* and gains the privileges for terminating the conference, inviting and excluding participants, as well as locking and unlocking a conference (see below). Upon creation of a conference, the conference profile is specified and a convener password may be defined allowing the convener to leave and re-enter the conference without loosing her privileges. Finally, it is specified whether the conference is initially *locked* or *unlocked*.

Two ways are defined for entering a T.124 conference: nodes either join a conference on their own volition, or they are invited into the conference by a privileged conference participant. Nodes may join a conference by means of the `GCC-Conference-Join` service; however, only unlocked conferences can be joined, and a joining node has to specify the correct password during the join process. In order to be able to join a conference, a node has to find out about the existing conferences and their profiles: this is done via the `GCC-Conference-Query` service that returns the listed conferences active at the queried node one of which can then be joined. Regardless of whether or not the conference is locked, nodes may be invited into the conference by means of the `GCC-Conference-Add` and `GCC-Conference-Invite`[5] services; for invitations, the password protection is overridden.

---

[2] The work on the draft ITU-T Recommendation T.RES [Ceccaldi 97] is expected to cover some of these issues.

[3] A node is either a terminal, a multiport terminal, or an MCU implementing the T.120 protocol suite (refer to section 4.1).

[4] As this is a purely local interaction, a corresponding creation service is not defined in T.124. However, the IMTC GCC API includes a corresponding function call definition.

[5] The invitation of a new node into a conference may be triggered by any privileged node invoking the `GCC-Conference-Add` service. This leads to either the Top GCC provider or another MCU or multiport terminal establishing a connection to the node to be invited and then using the `GCC-Conference-Invite` service to actually include the invited node into the conference.

To leave a conference voluntarily, a node makes use of the `GCC-Conference-Disconnect` service; this service is also used to indicate a provider-initiated disconnect from the conference, e. g. due to a failure of an MCS connection. A privileged node may expel another participant from a conference by invoking the `GCC-Conference-Eject-User` service and may also request other nodes to change from one conference into another (`GCC-Conference-Transfer`). Finally, a privileged participant may terminate the entire conference by means of `GCC-Conference-Terminate`; in this case, all other nodes are notified by receiving a `GCC-Conference-Disconnect` indication.

Any privileged node can alter the lock state of a conference: by issuing a `GCC-Conference-Lock` request a conference is locked, thereby preventing subsequent joins to the conference; a `GCC-Conference-Unlock` request is used to release the lock again and allow other nodes to join the conference. All participants are informed about changes to the lock state by means of `GCC-Conference-Lock-Report`.

Conference Roster Services

The conference roster is a data base that contains information about the nodes involved in a GCC conference and their characteristics. Per T.120 node, the node type (MCU, multiport terminal, terminal), the MCS user id of its GCC provider for the respective conference (the *Node ID*), the name(s) of (human) participant(s) at this node, and other node-specific information are maintained. This information is made available by each node upon entry to the conference (and may be updated at any time) by means of the `GCC-Conference-Announce-Presence` service. Whenever the conference roster changes — due to newcomers, updates, or nodes leaving — the `GCC-Conference-Roster-Report` is used by the GCC providers to notify all node controllers at all participating nodes. Any node controller or application protocol entity at any node may use the `GCC-Conference-Roster-Inquire` service to obtain the current conference roster from its GCC provider. Following the T.124 specification, a node is not considered to be part of a conference unless its corresponding entry is included in the conference roster.

Application Roster Services

The application roster is a data base that contains information about available application protocols and active application protocol entities per node as well as on-going application protocol sessions in a conference. Each entry in the application roster consists of three parts:

- a *session key* that identifies the application protocol and — for active application sessions — also distinguishes multiple sessions of the same application protocol;
- an *application record* kept per application protocol entity containing a flag indicating whether the entity is active or not, the entity's MCS user id channel, a flag showing whether it is capable of acting as conducting entity in an application session, and a list of application capabilities (the *non-collapsing capabilities*) that are not to be merged with other application protocol entity's capabilities; and
- the *application capabilities list*. For each application protocol entity enrolled with a GCC provider, this list defines the features supported by the individual application protocol (entity) present at this node; this information belongs to the *local application roster* (see below). For each application protocol session in the *global application roster*, this list indicates the least common denominators of the capabilities (termed *collapsed capabilities*) of all application

protocol entities enrolled at all T.120 nodes of the conference per application protocol or per active application protocol session.

A *local application roster* is created by each GCC provider: its local T.120 applications register their supported application protocols and the respective capabilities with the GCC provider. In addition, each time they are instantiated to participate in a particular application protocol session, they register again for this particular session and declare their session-specific capabilities. The GCC provider gathers this information from all local application protocol entities (APEs) to form a local application roster and forwards (updates to) this roster to the Top GCC provider which compiles the *global conference application roster* from the input received from all nodes in the conference. The conference application roster is distributed to all GCC providers and updated whenever changes occur so that the data base is kept consistent at all nodes.

The application roster data base serves two purposes: it allows individual nodes to indicate the supported application protocols and the respectively supported parameters to other nodes, thereby providing the basis for interoperability; and it allows each application protocol entity to find out about ongoing application protocol sessions and their parameters, thus enabling it to participate in a session.

The local application roster is created in three steps: firstly, the application protocol entities attach to a GCCSAP of their local GCC provider via some local means. When a new conference is established at a node, the GCC provider notifies the application protocol entities with a `GCC-Application-Permission-To-Enroll` about the new conference and invites them to participate; the same service primitive is also used to revoke the invitation. Then, the APEs register their capabilities — i. e. the respectively supported application protocol and the protocol parameters — with the GCC provider by means of `GCC-Application-Enroll`; they also use the same primitive to indicate their participation in a particular application session as well as to deregister. When the groupware applications have announced their capabilities to the GCC provider for a specific conference, they are referred to as *inactive* APEs, as soon as they enroll for a particular application session, they do become *active* APEs. After exchanging application roster information with its peers as well as upon each change in the application roster, the GCC provider informs each locally enrolled APE about those parts of the new (changed) application roster that are relevant to the application protocol the APE has registered. This is done by issuing a `GCC-Application-Roster-Report`. An application protocol entity can obtain information about the current roster from the local GCC provider through `GCC-Application-Roster-Inquire`. If a new application session is to be created, a node may decide to request the instantiation of the corresponding application protocol entities at one or more remote nodes via the `GCC-Application-Invoke` service.

Application Registry Services

The application registry is a database that contains information about the resources (MCS channels and tokens) utilized by application sessions as well as about session-specific parameters (which could be e. g. the file name of the document in a joint editing session). The purposes of the GCC application registry are a) to allow registering resources and thus avoid conflicting use of the same resource; and b) to find out about the resources that are used in a specific application session. As a side effect, the application registry may be used to obtain hints about which application protocol session to join if several similar ones do exist.

Channel ids are acquired by means of MCS services and may subsequently be registered with the GCC application registry through the `GCC-Registry-Register-Channel` service. In contrast to channels, for tokens no assignment mechanism that would guarantee exclusive use of a token for a particular purpose is provided by MCS. To circumvent this shortcoming, `GCC-Registry-Assign-Token` atomically assigns an unused token id to the requester and registers this token for the purpose specified in the request. `GCC-Registry-Set-Parameter` may be used to create and modify arbitrary session-specific identifiers and assign values and modification rights to them. Any of the above three types of information may be retrieved from the application registry by issuing a `GCC-Registry-Retrieve-Entry` request. Entries in the registry may be deleted by their creators using `GCC-Registry-Delete-Entry`. Any application protocol entity may monitor the contents of the registry by selecting the entries to be monitored with the `GCC-Registry-Monitor` service; changes to the value or the modification rights of a monitored entry as well as its deletion induce a notification service primitive being issued to all APEs monitoring this particular entry. Finally, the application registry provides a mechanism to generate conference-widely unique 32 bit values — *handles* — with the `GCC-Registry-Allocate-Handle` service.

Conference Conductorship

GCC supports the notion of conducted-mode conferences in which a dedicated node performs central coordination functions — for the GCC as well as for all application sessions:

- With respect to GCC services, in conducted conferences the node acting as conference conductor may have been assigned different privileges compared to the other nodes, while in non-conducted conferences all nodes share the same set of privileges. Note that in either mode, the conference convener has privileges to invoke all GCC services.
- For all other application protocol sessions, it is up to the respective application protocol to define the difference between conducted and non-conducted mode of operation. The behavior defined in the two application protocols standardized so far in the ITU-T can summarized as follows: in non-conducted mode, all application protocol entities may invoke any kind of operations — e. g. a file transfer in T.127. If the conference operates in conducted mode, some or all operations are only permissible for the application protocol entities at the node acting as conductor. All other APEs have to inquire the corresponding APE at the conducting node to obtain permission for the invocation of such operations; the permission may or may not be granted and, if granted, the permission may be revoked at any time.

The prerequisite for running a conference in conducted-mode is that the conference profile specifies that the conference is conductible. If so, the conference may arbitrarily switch between conducted and non-conducted mode of operation;[6] any node may become conference conductor and the conductor role may be passed dynamically between nodes. Conductorship is a possible way to technically represent the role of the conference chairperson in a tightly controlled conference; however, the conductorship services can also be used to implement a variety of other conference policies.

[6] It is up to higher level conference policies to define which nodes may actually become conference conductor and to ensure that this policy is adhered to. T.124 only provides mechanisms that facilitate enforcement of such policies.

If a GCC conference is conductible, a node becomes conductor when its node controller issues a GCC-Conductor-Assign request, unless another node is already conference conductor. If the conference was in non-conducted mode beforehand, assignment of conductorship places the conference in conducted mode. The conductor role is relinquished again by means of the GCC-Conductor-Release request, thus putting the conference into non-conducted mode. GCC-Conductor-Give may be used by the current conductor to pass the conductor role directly to another node; the conference remains in conducted mode and no third party can intervene with this change. The GCC-Conductor-Please service may be used to request conductorship from the current conductor who may or may not pass it on using GCC-Conductor-Give. With GCC-Conductor-Inquire, the node controller of a node and all application protocol entities are able to determine whether the conference is currently in conducted mode and, if so, who is the current conductor. Finally, GCC-Conductor-Permission-Ask is used by application protocol entities to request permission for certain actions for the specific application session from their peer or the node controller at the conducting node. As reaction to such requests, permission is granted, denied, or revoked for one or more nodes simultaneously by means of the GCC-Conductor-Permission-Grant service.

Miscellaneous Functions

Two further services are defined in T.124: handling of timed conferences and conference assistance.

While GCC itself has no knowledge of reservation systems and scheduling, it provides hooks to interact with such systems (via the node controller of the conference convener's node) from within a GCC conference through a set of standardized services.[7] GCC-Conference-Time-Remaining may be used by the convener to indicate the remaining time available in a conference. GCC-Conference-Time-Inquire may be issued by any node to query for the remaining time and GCC-Conference-Extend to ask the node controller of the conference convener to extend the duration of the conference. Both requests trigger a GCC-Conference-Time-Remaining indication issued by the convener which includes the (possibly updated) remaining time for the conference.

The GCC-Conference-Assistance service provides a means for interaction with a conference operator (if available) and allows passing opaque data carrying higher level information. A possible response may be — although this is not specified in T.124 — a GCC-Text-Message which allows sending textual information to a single or all nodes in a conference. The usage of both services is not specified any further in T.124.

7.1.2. GCC Protocol Characteristics

The GCC protocol [ITU-T T.124] defines the interactions between GCC providers to establish and run a conference as well as to maintain and distribute the GCC state and the contents of the GCC information bases as required for offering the GCC services. The GCC protocol makes use

[7] Referring back to section 3.4.1, one possible scenario is that the MCU always acts as conference convener. The MCU's GCC provider may be controlled by a reservation system that starts and terminates conferences according to some externally defined schedule.

of most MCS services — except for services relating to private channels — for creating GCC conferences and exchanging information within the boundaries of a conference.

The GCC provider acts as the controlling entity of the MCS provider and attaches to the Control MCSAP to control setup and teardown of MCS connections and thereby to create and destroy MCS domains upon which GCC conferences are built. Furthermore, for information exchange within a conference, a GCC provider attaches to exactly one User MCSAP of the MCS domain that corresponds to the GCC conference; the MCS user id which the GCC provider receives upon attaching to the MCS domain is referred to as *Node ID* of the respective node for this specific conference. In T.124, two static MCS channels — the *GCC-Broadcast-Channel* and the *GCC-Convener-Channel* — and one MCS token (the *Conference-Conductorship-Token*) are assigned for use by GCC.

In most cases, GCC providers communicate with one another by exchanging protocol data units defined in T.124: GCCPDUs.[8] The GCCPDUs for initially contacting another GCC provider are carried in the user data portion of an `MCS-CONNECT-PROVIDER`. All other GCCPDUs are transmitted using the MCS services `MCS-SEND-DATA` and `MCS-UNIFORM-SEND-DATA` specifying one of the highest two MCS priorities (*Top* and *High*) and addressing the PDU to one of the static GCC channels or to a GCC Node Id channel. GCCPDUs are encoded using the aligned variant of the Packed Encoding Rules of ASN.1 [ITU-T X.680] [ITU-T X.691].

Like the MCS, the GCC employs a centralized model of control: the Top GCC Provider maintains consistency of the conference state, is responsible for processing many of the GCC service requests, and acts as the central arbiter for T.120 resources. In particular, the Top GCC Provider makes the ultimate decision whether or not to admit newcomers into a conference, computes the conference-global conference and application rosters, maintains the application registry, and checks the validity of all requests against the conference profile. All GCC providers at non-leaf nodes perform aggregation functions when gathering and forwarding information of the potentially voluminous data bases (see the conference and application roster description below). Finally, each GCC provider is responsible to validate all local requests as well as all incoming GCCPDUs with respect to its current understanding of the conference state (e. g. the whether or not the conference is conducted and, if so, who is the conductor) and with respect to the permissions defined in the conference profile.

The GCC protocol uses roughly five different mechanisms to implement the GCC service classes that are described in the remainder of this subsection:[9]

- the initial information exchange between two GCC providers to create and join a conference;
- invocation and execution of privileged operations, i. e. services reserved for use by the convener and nodes that have been designated by the convener in the conference profile;
- the procedures for creating, updating, and distributing the conference and the application rosters;

[8] The exceptions are most services related to conductorship management. These are mapped onto token management services of the MCS.

[9] Note that the following descriptions are simplified to provide a concise overview of the way the GCC protocol works; for the fully detailed specification refer to the ITU-T Recommendation [ITU-T T.124].

- the maintenance of the application registry; and
- the handling of GCC conductorship.

Initial Communication to Create and Join Conferences

Four services in T.124 cause the GCC provider to initiate the establishment of an MCS connection via the Control MCSAP of the MCS provider: `GCC-Conference-Create`, `GCC-Conference-Query`, `GCC-Conference-Join`, and `GCC-Conference-Invite`. In all four cases, the initial exchange of GCCPDUs — at least one request and the corresponding response — is carried out using the user data part of the `Connect-Initial` and the `Connect-Response` MCSPDUs to transport the GCCPDUs: the plain OCTET STRING of the user data field of these two MCSPDUs is structured as defined in T.124 into a set of type-value pairs: for the GCCPDU, the type is set to indicate the ASN.1 Object Identifier assigned to T.124, the value is again an OCTET STRING that contains the ASN.1 encoded GCCPDU.

Depending on the GCC service requested, several courses of actions may be taken during the setup process:[10] [11]

- *Conference Query*

T.124 specifies that when contacting another T.120 node, the GCC provider of the calling node should at first issue a `GCC-Conference-Query` to determine the type of the called node, the available conferences and their profiles, and what the called node wants it to do. From this information, the caller is able to derive its behavior for the following conference establishment process: the caller may be allowed to create a new conference; may be informed that it is supposed to join a particular conference; may be requested to wait for an invitation from the called GCC provider; may be left with a choice of conferences it has to choose one from; etc. The `GCC-Conference-Query` is a simple client-server-like interaction between the calling and the called GCC provider: the caller encapsulates the request GCCPDU in the user data field of the `MCS-CONNECT-PROVIDER` request that is always refused by the called GCC provider which places the response GCCPDU in the user data field of the response MCSPDU.

- *Conference Creation*

Conference creation may be invoked either locally or at a remote T.120 node. In case a node controller wants to create a conference locally, it issues a conference create request to its GCC provider which creates a new MCS domain (again locally) and becomes the first and so far single member of the conference. Note again that local creation of GCC conferences and MCS domains is not standardized and is mentioned here for completeness only. For a conference creation at a remote node, the calling node sends the corresponding GCCPDU — which is provided as user

[10] In the following description, the terminology introduced in section 5.1.2 is re-used: the GCC provider that has initiated the MCS connection setup — i. e. has sent the first GCCPDU — is referred to as *calling GCC provider* or *caller*, the other is termed *called GCC provider* or *callee*. Note that in some cases the roles of caller and callee may change after the initial exchange of GCCPDUs due to the involved node types and their characteristics. However, the details and implications of this fine distinction are beyond the scope of this protocol overview.

[11] The precise behavior of the calling and the called node also depends on the respective node types, already existing conferences, and other information. For the full specification of the procedures the reader is referred to the current version of the T.120 implementor's guide [Bernstein 96].

data to the MCS-CONNECT-PROVIDER service primitives as is the response — to the called node which either accepts or rejects the request. If accepted, a new GCC conference is created, the MCS connection is established and attached to this domain, and the two GCC providers are joined to the newly created conference with called node becoming the Top GCC Provider. If the conference creation is rejected, the MCS connection setup is refused and the contained GCCPDU indicates the reason for the refusal.

- *Conference Join*

If a conference join is attempted, the caller uses the user data of the MCS-CONNECT-PROVIDER request to forward the corresponding GCCPDU to the callee. However, the decision whether or not to accept the newcomer is exclusively taken by the Top GCC Provider. This means that if the newcomer is not connected directly to the Top GCC provider, the called GCC provider forwards the received join request GCCPDU to the Top GCC provider which processes the request and sends the response back to the called GCC provider. This latter information exchange takes place using the MCS-SEND-DATA service. Eventually, the called GCC provider forwards the join response GCCPDU to the caller as user data of the MCS-CONNECT-PROVIDER response.

Upon reception of a join request GCCPDU, the Top GCC Provider may 1) admit the newcomer to the conference; 2) reject it; or 3) initiate an *n-way* authentication procedure to verify the caller's identity. In the first case, the MCS connection is established and attached to the GCC conference by the called GCC provider. In the other two cases, the MCS connection setup is refused: in the second case, this refusal terminates the attempt to join a conference while in the third case, the response GCCPDU indicates that the Top GCC Provider requests further information from the calling GCC provider. This leads to one or more additional handshakes between the two GCC providers which are again carried out via the user data part of the MCS-CONNECT-PROVIDER service with optional forwarding through MCS-SEND-DATA. After an arbitrary number of further PDU exchanges — that depends on the authentication algorithm — the authentication procedure finally results in acceptance or refusal of the newcomer. The newcomer is only admitted to the conference if the conference specified in the join request does exist, is not locked, and the (n-way) authentication of the caller succeeds; otherwise, the calling node is rejected.

- *Invitation to a conference*

To invite a node into a conference, the node to which the new node shall be connected in the MCS hierarchy[12] (the caller) issues an MCS-CONNECT-PROVIDER with the invite request GCCPDU contained in the user data. If the called GCC provider accepts the invitation, it is included in the conference. Note that for an invitation, no authentication is performed by the calling or called GCC provider. Again, the GCCPDUs are passed as user data to the MCS-CONNECT-PROVIDER service primitives.

When a new MCS connection for a GCC conference is established between two nodes (in case of a successful creation, joining, or invitation), one of the two nodes becomes the subordinate in the MCS hierarchy (the caller in case of a create or a join) and one becomes the superior node (the caller in case of a invitation). During the initial exchange of GCCPDUs, the subordinate node is

[12] The execution of the invitation process may have been triggered by the Top GCC provider or another privileged node of the conference issuing a GCC-Conference-Add request. In this request, besides the node to be invited, also the node that shall perform the invitation may be specified.

informed about the Node IDs of the superior node and that of the Top GCC provider — both of which are needed for later GCC protocol operation. As soon as the MCS connection setup for the MCS domain corresponding to the GCC conference is completed, the subordinate node's GCC provider attaches to the MCS domain to obtain its GCC Node ID for the new conference. Then, the GCC provider joins its Node ID channel, the GCC broadcast channel, and, if it has created the conference, the GCC convener channel. It finally informs the GCC provider of its superior node of its GCC Node ID thus completing the protocol procedures. Figure 7.1 illustrates the creation of a GCC conference at a remote node.[13]

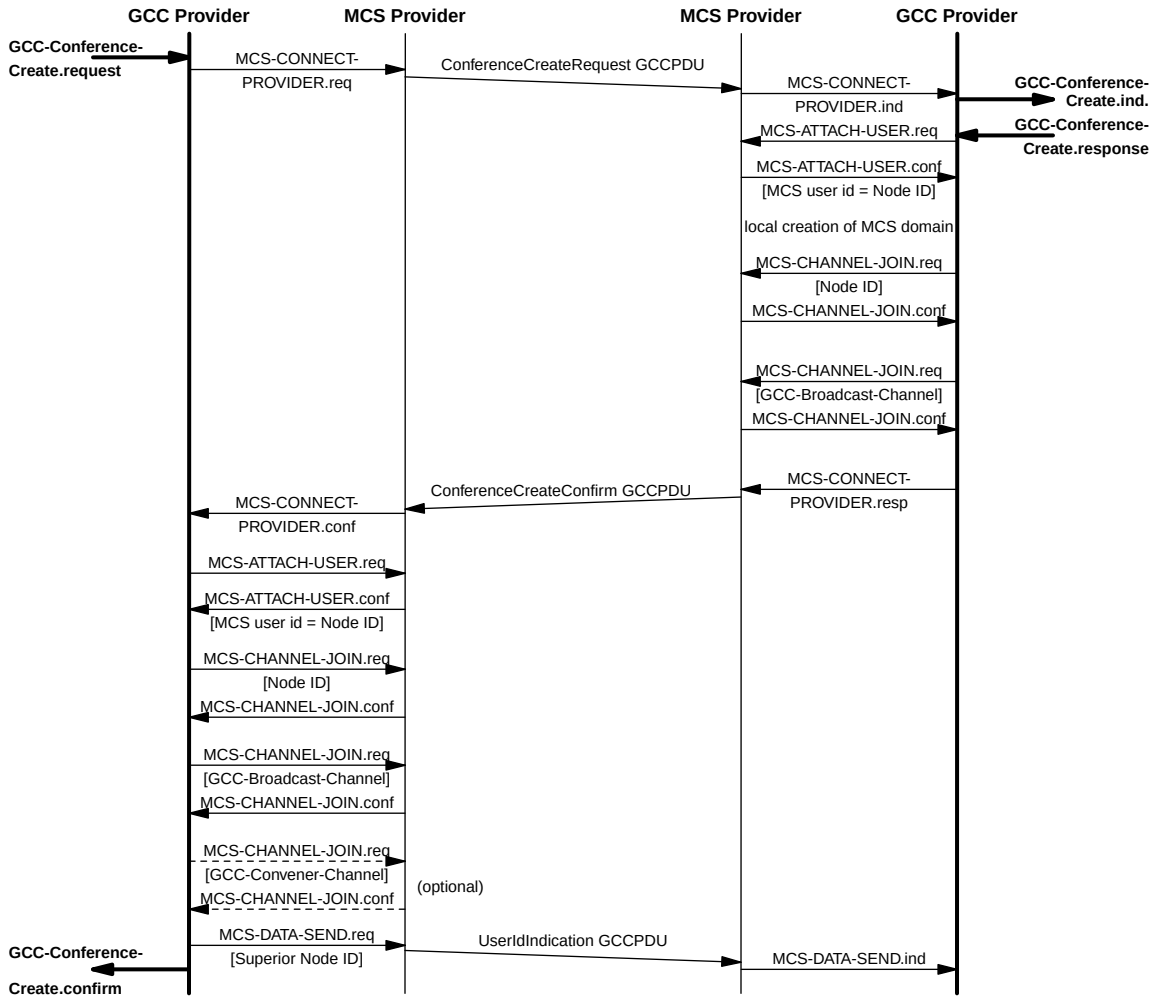


Figure 7.1: Example for the creation of a GCC conference

[13] In case of a `GCC-Conference-Create` the GCC provider of the superior node (which becomes the Top GCC provider) creates the MCS domain and the GCC conference locally, attaches to the MCS domain, and joins all the predefined GCC channels as well as its Node ID channel before sending the reply indicating successful conference creation. Afterwards, the Top GCC provider is able to supply its Node ID (for the Top GCC provider as well as the immediately superior node in the MCS hierarchy) in the response sent back to the requester.

Note that through this exchange of Node IDs, the GCC protocol creates partial knowledge about the MCS interconnection topology (from which the MCS provides an abstraction) as each GCC provider determines the Node IDs of its immediately superior and subordinate nodes. This knowledge is exploited for parts of the GCC protocol operation, such as the roster maintenance described below.

Privileged Services

Privileged services include triggering the invitation (adding) of new nodes to the conference, locking and unlocking the conference, terminating the conference, ejecting a node from a conference,[14] and requesting the transfer of a node to another conference. If a node controller issues one these service requests to the Control GCCSAP of the GCC provider, the provider locally checks the validity of the request and, if the request is permissible, sends a corresponding GCCPDU to the Top GCC provider using its Node ID channel. Upon receipt of such a request, the Top GCC provider double checks whether the requesting GCC provider is allowed to invoke the respective service depending on the conference profile and the conference state. Execution of the requested services only proceeds if

- the service is requested from the convener;
- the service request has been originated from the current conference conductor and the convener has granted the corresponding permission to the conductor node; or
- the corresponding permissions are granted by the convener in the conference profile for all nodes for the current conference mode (conducted or non-conducted).

Otherwise, the request is refused with an appropriate GCCPDU being sent back to the originator. If the request is permissible, a positive response is sent to the originator, and processing the service request proceeds as follows:

- Adding a node to the conference asks the Top GCC provider either to carry out the invitation procedure itself or to initiate that the procedure be carried out by the node indicated in the `ConferenceAddRequest` GCCPDU by forwarding the request to this very node.
- Locking and unlocking the conference results in the corresponding state change in the Top GCC provider and leads to a report about the new state being disseminated to all nodes on the GCC-Broadcast-Channel.
- Ejecting nodes from the conference causes the eject GCCPDU indicating the nodes to be ejected being broadcast to the GCC-Broadcast-Channel. The ejected nodes and their superior nodes then take the required actions to make the request take effect.
- Transferring nodes to another conference is achieved by forwarding the corresponding GCCPDU — which includes the Node IDs to be transferred — on the GCC-Broadcast-Channel. The indicated nodes then disconnect from the current conference and join the indicated conference using the regular GCC protocol for joining a conference. Note that from the GCC protocol point of view there is no shared state between the two conferences, and that therefore the initiator of a conference transfer does not obtain any confirmation on success or failure of the conference transfer service.

[14] Note that a GCC provider does not require convener-assigned privileges to eject a node that is one of its immediate subordinates in the MCS hierarchy.

- Terminating a conference leads to a `ConferenceTerminateIndication` being distributed by the Top GCC provider via the GCC-Broadcast-Channel to all nodes in the conference. This causes all GCC providers in the conference to disconnect from the conference: at first, the leaf nodes in the MCS domain hierarchy disconnect and when all children of previous non-leaf nodes have left, these node themselves disconnect. This recurs until only the Top GCC Provider is left alone in the conference which then terminates the conference by deleting the conference state.

Conference and Application Roster Maintenance

The conference and the application roster are maintained as a distributed database. Each GCC provider stores a copy of the currently valid conference-global rosters as well as a local application roster containing information about all locally enrolled application protocol entities. GCC providers of non-leaf nodes in the MCS hierarchy in addition keep a copy of the rosters they have received from each of their subordinate nodes. Conference-wide distribution and updating of the conference and application rosters is done by exchanging a single type of GCCPDU: the `RosterUpdateIndication`.

Requests from the node controller or application protocol entities that cause changes to the conference or the application roster — such as enrolling and un-enrolling application protocol entities — are first applied to the local roster within the GCC provider. The GCC provider then initiates conference-wide distribution of the changes by combining its local roster with those of its subordinates (if any) and sending the resulting changed entries in a `RosterUpdateIndication` GCCPDU to the GCC provider of its superior node using this GCC provider's Node ID channel. The process of aggregating local roster information and roster information received by each subordinate and forwarding the aggregated information set to the superior node recurs until the `RosterUpdateIndication`s reach the Top GCC provider. The Top GCC provider also aggregates the rosters received from all its subordinates and computes the global conference and application rosters. Finally, the Top GCC provider broadcasts the updated conference and/or application roster on the GCC-Broadcast-Channel to all nodes in the conference.[15] Each roster announcement of the Top GCC provider is tagged with a monotonically increasing (modulo 2^{16}) sequence number to allow synchronizing all nodes and their applications with respect to a particular instance of the global conference and application roster.

By means of this protocol, all GCC providers in a conference are regularly updated with the current conference and application roster. This allows all requests for information about the rosters from local application protocol entities to be handled locally (and immediately) without the need for further exchange of GCCPDUs.

Application Registry Maintenance

As stated above, the application registry is a centralized database that is kept at the Top GCC provider with no parts being replicated at other nodes. Each GCC service request related to the application registry is processed by the Top GCC provider in a client-server style:

[15] Note that on the way upwards to the Top GCC provider it is up to the discretion of each GCC provider whether to send the entire roster or whether to forward only the changes in the roster. When the Top GCC provider finally distributes the update roster, always the entire roster is transmitted.

- Requests to modify the contents of the registry — registering channels, assigning tokens, and storing parameters as well as deleting registry entries — are encapsulated in the corresponding GCCPDUs and then sent to the Top GCC provider. The Top GCC provider verifies the permissions of requesting application protocol entity: for example, only the owner of registry entries for channels and tokens is allowed to overwrite or delete them. If either the requester is allowed to perform the intended operation or the requested registry entry is currently not owned or unused, the Top GCC provider modifies the registry entry according to the contents of the request GCCPDU. If the entry is unused or not owned, the requesting application protocol entity is recorded as the (new) owner.[16] In case of a `GCC-Registry-Assign-Token` request, beforehand a new token id is allocated to be stored in the registry. Regardless of the requested operation and regardless of whether or not it has been performed successfully, the Top GCC provider returns the contents of the GCC registry entry in question to the initiator[17] along with the result of the request.

- Retrieving information from the application registry is done by sending a request GCCPDU containing the registry key identifying the registry entry to the Top GCC provider. The Top GCC provider returns the state of the entry and, if it is not unused, all information stored for the specified entry.

- Monitoring the application registry is done through a single flag for each registry entry at the Top GCC provider indicating whether or not the respective entry is being monitored. The request to monitor an entry of the registry is forwarded to the Top GCC provider. If the entry in question exists, it is marked to be monitored and a successful response is generated. Otherwise the request is refused. If an entry is monitored, for each modification to this entry — change in value, in ownership, or permissions as well as deletion of the entry — an indication containing all information about the entry is sent to the GCC-Broadcast-Channel. It is up to each GCC provider to keep track of the locally enrolled application protocol entities that need to be notified about such event because they have previously requested to monitor a particular entry. If all local APEs of a GCC provider have stopped monitoring a particular registry entry, the GCC provider sends a corresponding GCCPDU to the Top GCC provider indicating that it no longer requires notifications about changes to this particular entry.[18]

Assignment of conference-globally unique handles is handled roughly in the same fashion as is the assignment of tokens. However, no registry key is associated with an assigned handle. A GCC provider may obtain blocks of handles through a single request.

---

[16] If an application protocol entity detaches from the conference, the Top GCC provider notices this from the change in the application roster and removes ownership for all entries in the application registry previously owned by this very APE so that these entities become *not owned*; the contents of the entries, however, are preserved.

[17] This is equivalent to an atomic *test-and-set* operation and simplifies synchronization of application protocol entities.

[18] Note, however, that more than the single flag per registry entry specified in T.124 is needed in the Top GCC provider to effectively allow disabling the monitoring of an entry, unless the corresponding entry is deleted.

Conference Conductorship Handling

Conference conductorship is managed by means of the MCS token mechanism and is only applicable if the conference profile specifies that the conference is conductible. GCC defines a static token — the `Conference-Conductorship-Token` — for the purpose of controlling conductorship. Exclusive possession of this token is associated with the conductor role; if a GCC provider owns the conductor token the conference is in conducted mode. If the token is not in use, the conference is non-conducted. Shared possession is not defined for the conductor token (and cannot occur since the `GCC-Conductorship-Token` is only accessed through the GCC conductorship services).

If the invocation of any conductorship-related GCC service is requested, the GCC provider first determines from the conference profile whether the conference is conductible and refuses all such requests immediately if it is not. Afterwards the processing for obtaining and releasing conductorship is as follows:

- A `GCC-Conductor-Assign` request is mapped onto an `MCS-TOKEN-GRAB` request for the `Conference-Conductorship-Token` after checking whether the node is already the conference conductor (in which case no further action would be taken). If the MCS request succeeds, the requesting GCC provider becomes the new conference conductor and the conference is placed in conducted mode. Otherwise, another node is already conference conductor, and a negative result is returned in the confirmation for the request.

- Conductorship may be requested from the current conductor by means of `GCC-Conductor-Please` which is mapped onto an `MCS-TOKEN-PLEASE`. This service notifies the GCC provider currently acting as conference conductor about the request which then asks its node controller whether or not to pass on conductorship. If the node controller does not want to give up conductorship, it need not react at all on this request; otherwise, it issues a `GCC-Conductor-Give` request to its Control GCCSAP.

- A `GCC-Conductor-Give` request is used to pass on conductorship to a particular other node; invocation of this service may have been triggered by a `GCC-Conductor-Please` indication. Passing the conductor token is achieved by issuing an `MCS-TOKEN-GIVE` which succeeds if the receiving node's GCC provider accepts the conductor token. Otherwise, the conductorship state remains unchanged.

To notify the other conference participants of the change in conductorship,[19] the new conference conductor offers the conductor token to the Top GCC provider using `MCS-TOKEN-GIVE`. If the donor of the conductorship token is not recorded as current conference conductor, the Top GCC Provider refuses the offered token and stores the Node ID of the donor as new conference conductor. Then, the Top GCC provider broadcasts the identity of the new conductor on the `GCC-Broadcast-Channel` to make all nodes aware of the state change. If the donor of the token is already known as current conference conductor, the Top GCC Provider considers the offer as a `GCC-Conductor-Give` request and processes accordingly.

[19] It is a bug in the MCS specification that is circumvented by this procedure: a) changes to token states cannot be monitored within MCS, and b) only the status of a token (unused, inhibited, or grabbed) but not its current owner(s) can be determined via `MCS-TOKEN-TEST`.

- If the conference conductor releases the conductor token, the conference falls back to non-conducted mode. The releasing GCC provider broadcasts a `ConductorReleaseIndication` GCCPDU on the `GCC-Broadcast-Channel` to notify all other GCC providers of the state change.

Broadcasting conductorship state changes to the entire conference allows all GCC providers to keep a copy of the current state so that they are able to validate requests and to answer inquiries related to conductorship locally. The GCC providers also forward changes in the conference conductorship to their locally enrolled conductorship-aware application protocol entities.

If the conference operates in conducted mode, application protocols may define restricted behavior for nodes that are not the current conference conductor. These nodes may apply to the conference conductor for conducted-mode permissions — the precise meaning of which is independently defined per application protocol. Asking for conducted-mode permissions is done by sending the corresponding GCCPDU to the GCC provider of the node that is the current conference conductor where an indication is signaled to the node controller via the Control GCCSAP. The node controller may react on the request by issuing a `GCC-Conductor-Permission-Grant` request primitive which leads to a `ConductorPermissionGrantIndication` GCCPDU being broadcast on the `GCC-Broadcast-Channel`. This GCCPDU includes a list of all the nodes that are granted conducted-mode permission and a list of `GCC-Conductor-Permission-Ask` requests that are still outstanding so that each GCC provider can determine whether its request has been accepted, rejected, or still needs to be processed. Each time a new `ConductorPermissionGrantIndication` GCCPDU is broadcast, this one supersedes the previously granted permission.

If the conference conductor is disconnected from the conference — which is determined by receiving a `MCS-DETACH-USER` indication containing the Node ID of the current conference conductor — the conference falls back to non-conducted-mode, and all conducted-mode permissions are revoked.

7.2. Structure of the GCC Provider

Like the MCS provider, the GCC provider is implemented as a separate process. The GCC provider process uses the IMTC API defined for the MCS (which is mapped onto GPCI for message passing) to access the MCS services and provides its service to the node controller and the user applications via the GPCI message passing mechanism (which is concealed on the client side by the IMTC API for GCC). That is, the GCC interacts with other system components exclusively through the GPCI mechanism. The protocol engine of the GCC provider is independent of the underlying operating system. Employing a single standardized communication mechanism allows to keep the interactions with other components of the T.120 infrastructure OS-independent, too. The sole parts of the GCC provider differing significantly between OS platforms are its `main()` function doing the system-specific process initializations and its main loop that recognizes incoming events — signaling and control of which is done in an OS-specific manner — and initiates their processing.

Despite some general similarities between the MCS and the GCC providers — which are reflected in the fact that the overall structure of the GCC provider (figure 7.2) implementation roughly resembles that of the MCS provider — the GCC provider has to perform very different

types of tasks: the GCC protocol is much more complex compared to the MCS protocol with respect to the number interactions between different nodes required to carry out some of the GCC services, the amount of state information to be kept per conference, etc. The internal structure developed for the GCC provider implementation is tailored to meet the specific needs of the GCC protocol, which consists to a large degree of (sometimes complex) sequences of sending out (one or more) requests and collecting (one or more) responses. In order to deal with the request-response operation and also to handle temporary congestion of communication paths, the core concept of the GCC provider implementation includes the simulation of *threads* the program code and state of which are represented by objects (see below) as well as a *sleep/wakeup* mechanism that allows threads to asynchronously wait for an event (*sleep*) and continue to be executed as soon as the event occurs (*wakeup*) [Radig 96].

The internal structure of the GCC provider implementation as outlined in figure 7.2 consists of four main components:

- *Service Access Point (SAP) objects* provide an abstraction from interactions with other T.120 components and the operating system and implement part of the sleep/wakeup mechanism for GCC state objects;
- *GCC state objects* (which contain the state information plus the program code of GCC threads) implement the T.124 protocol;
- the *GCC information base* contains GCC provider-global as well as GCC conference-specific information; and
- the *event dispatcher* (which is integrated with extensions to the underlying GPCI communication library) recognizes incoming events and forwards them to the appropriate SAP objects for further processing and implements part of the sleep/wakeup mechanism for SAP and GCC state objects.

The GCC provider is mainly implemented in the C++ programming language. The following three sections describe the GCC information base as well as the implementation of the SAP and the GCC state objects. The dispatcher is briefly discussed in the context of the GCC main function and event processing in the second to last section of this chapter. The detailed description of many implementation concepts underlying the GCC provider and its UNIX-based prototype can be found in [Degener 95] with several concept enhancements and refinements done by RADIG. The complete and tested implementation for 32-bit MS Windows platforms (Windows 95 and Window NT) is documented in [Radig 96].

7.3. GCC State Information Base

The GCC provider keeps two types of state information: local management information specific to the GCC provider itself and information related to a particular conference. Local management information include

- the service access point objects for the IPC connections to the Control SAP of the MCS provider as well as to the node controller;
- the SAP objects of all locally attached application protocol entities;
- the Timer and Memory objects for maintaining timers and performing memory management functions;

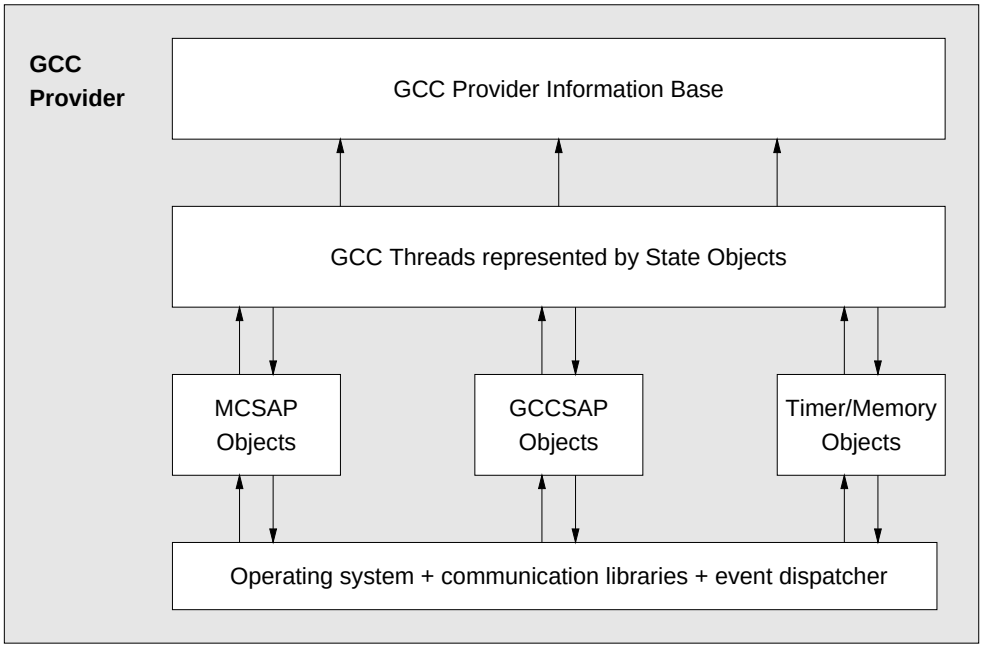


Figure 7.2: Structure of the GCC provider

- global lists of all SAP and GCC state objects;
- a list of GCC conferences currently active at the provider; and
- further control information for logging and debugging as well as hooks to enable the integration of network management information.

All other state information is conference-dependent and is therefore maintained in relation to the respective GCC conference. All conferences active at the GCC provider are maintained in a list and are accessible through hash functions. Within the GCC provider implementation, information about a conference is identified either through the textual or numeric conference name (including the conference name modifier) or through its MCS domain selector. Per conference, the following set of information is kept:

- *MCS-related information* — this includes the local MCS domain selector, the MCS domain parameters, and the MCS connections belonging to the conference. Connections to subordinate nodes in the MCS domain hierarchy are grouped and distinguished from the one to the superior node. For each MCS connection, the following data is maintained:
  - the MCS connection handle,
  - the Node ID of the directly connected node, and
  - the conference and application rosters most recently reported by that node.
- *Conference profile and state information* — the conference name, the GCC conference profile, Node IDs of the local GCC provider, the Top GCC Provider, the current conductor (if any), and the conference convener.

- *Conference and application roster information* — an entry of the conference roster contains exactly the set of information defined in T.124 for conference roster entries (which include node name and type, network address, names of participants, etc.). The application roster consists of a set of application session entries and a set of application records:
  - Application sessions are identified by a session key. A session entry stored in the GCC provider contains the collapsed capabilities of the session as well as lists of unchanged, added, deleted, and changed application records (if any) for this session as received in the most recent update via the MCS connection the roster entry belongs to or from the Top GCC Provider.
  - Application records contain the Node ID of the node to which the corresponding application protocol entity belongs and the APE's entity id as assigned by its respective local GCC provider. This pair uniquely identifies the record. Per record, the respective APE's non-collapsing capabilities are maintained as are the state information defined in T.124.

    For application protocol entities enrolled locally with the GCC provider, registered collapsing capabilities and further management information — such as registry entries monitored by the APE — are kept in addition.

  Both conference and application rosters are identified by an instance number that is incremented upon each update by the GCC provider originating the roster.

  As stated above, one set of roster information is kept per subordinate connection; in addition, three further roster instances are stored per conference in the GCC provider:
  - one roster instance contains the local site information and the application roster of the locally enrolled application protocol entities,
  - a temporary roster variable is used for the process of merging the local rosters and those of the subordinates in order to compute the roster update to be sent uptree, and
  - the third roster instance holds the global conference and application roster most recently broadcast by the Top GCC provider.

- *Information about GCC state objects* — state objects contain the program code and the current status of execution of the threads that perform protocol operations within a GCC provider. For each conference, the GCC provider keeps a list of all threads currently instantiated to process (part of) a GCC service for this particular conference.

For many of the aforementioned information objects, the number of instances to be kept per conference may grow very large. This may apply e. g. to application sessions, application records, capability lists, and registry entries as well as (lists of) state objects. As many of these object types are frequently accessed, they are either stored sorted as binary trees or made accessible through hash functions to allow efficient search and retrieval of information.

7.4. Service Access Points and Scheduling

The operation of the GCC provider is — like the MCS provider and other communication applications — driven by external events. When an incoming event is recognized, the event is dispatched to the responsible object which does the processing and finally produces a result. This result may comprise changes to the internal state information base and one or more pieces of information being sent to other entities: in the form of GPCI messages to local clients or as

GCCPDUs via the MCS provider to other GCC providers. How incoming events are noticed depends on the operating system environment and is hidden within the dispatcher; OS-specific transmission and reception of information is concealed by the GPCI library and the IMTC API for the MCS provider. Internally, the GCC provider uses the abstraction of *Service Access Point (SAP) objects* to deal with incoming events as well as transmitting and receiving information. The primary task of SAP objects is to define an interface to the dispatcher and provide a generic mechanism implementing the sleep/wakeup paradigm known from most operating systems: Threads (see next section) indicate that they are waiting for a particular event to occur (*sleep*) and are scheduled as soon as the requested event occurs (*wakeup*). For this purpose, the base SAP object class defines a sleep/wakeup mechanism that is uniformly employed by all derived object classes.

In addition, most SAP objects perform various tasks related to their nature as communication interfaces to other entities. These tasks include

- encoding and decoding of information as needed for transmission;
- transmission or reception of interface data units (IDUs) including segmentation and reassembly if needed;
- handling flow control including blocking and unblocking other objects that want to access a service access point object; and
- providing interfaces that abstract from the underlying (communication) services and that simplify interactions with the MCS provider as well as GCC clients.

The GCC provider uses the services of the MCS provider — of the Control MCSAP and one additional User MCSAP per GCC conference — and offers its services to the node controller (at the Control GCCSAP) and to application protocol entities (via the User GCCSAP). Therefore, two different types of SAP objects for interactions with other T.120 system components are defined:

- *MCSAP objects* are used for communication with the MCS provider via the IMTC API. A single MCSAP object is instantiated for communication with the Control MCSAP of the MCSAP provider. In addition, for each GCC conference, one UserMCSAP is object is required which contains four interrelated MCSAP objects in the GCC provider. This approach is taken because the four MCS priorities are independently flow-controlled and therefore four independent SAP objects are needed to implement the sleep/wakeup mechanism per priority.

  The MCSAP objects perform ASN.1 encoding and decoding of the GCCPDUs before sending and after receiving them, respectively, through the MCS IMTC API. They also carry out segmentation and reassembly of GCCPDUs perform exception handling such as dealing with interrupted message transfers and invalidated MCS attachments.

- *GCCSAP objects* implement the interactions with the node controller and local application protocol entities. A dedicated GCCSAP object is used to accept the attachment requests — from the node controller as well as from application protocol entities — and then instantiates required GCCSAP objects for further information exchange related to the particular attachment. Exactly one GCCSAP object is used for communication with the node controller, but there is no one to one correspondence between application protocol entities and GCCSAP entities. Rather, one GCCSAP object is instantiated per GPCI connection to a groupware

application process. This process may implement one or more APEs and may be attached to one or more GCC conferences, but all its communication with the GCC provider is handled through a single GCCSAP entity. This construction is required by the way in which the IMTC GCC API maps T.120 application processes and APE instances to GCC provider attachments [Braun 96a].

The GCCSAP objects perform similar functions to the MCSAP objects: GPCI encoding and decoding of the GCC IDUs before sending and after receiving them by means the GPCI library, respectively, as well as segmentation and reassembly of IDUs. Furthermore, GCC-SAP objects implement multiplexing of a single GPCI connection for GCC IDUs transmitted on *Top* and on *High* MCS priority. Finally, to accommodate the IMTC API, the GCCSAP objects multiplex GCC IDUs based on the conference and APE identifiers.

The MCSAP and GCCSAP objects use the sleep/wakeup mechanism to allow waiting for incoming PDUs or IDUs as well as for dealing with flow-control situations on IPC connections.

Besides the aforementioned SAP objects for local inter-process communication, two further object classes are derived from the base SAP object class (which do not provide communication functionality like the other SAP objects do, though):

- The *Timer object* provides an abstraction for timers required for GCC protocol handling. It provides methods to set, reset, and cancel timers. When a timer elapses, the timer object generates a timeout event and wakes up the state object waiting for the timer.
- The *Memory object* provides internal memory management functions for GCC (which are of particular importance in the MS Windows environment). The memory object allows dealing with temporary memory allocation failures: it provides a mechanism to generate an event when memory resources have become available again.

Timer and Memory objects are similar to SAP objects (and hence share a common base class) in that — like the other SAP objects — they are capable of asynchronously generating events for which threads may be waiting: for timeouts and for availability of memory resources. Thereby, a uniform interface to the sleep/wakeup mechanism is provided for all kinds of events.

7.5. State Objects and Threads

State objects are the central component in the implementation of the GCC protocol. To execute any GCC service, a (GCC) thread[22] is instantiated that performs all necessary actions to carry out the service request and is deleted after completing its task. All information about the thread is maintained in its associated state object. A base object class defines the common parts of all state objects. For each (group of) GCC service(s), an individual object class is derived from this base class, and for each incoming GCC IDU or GCCPDU, a (set of) processing function(s) is defined in the object class for the respective GCC service (group).

Figure 7.3 shows the structure common to all state objects. Each state object contains a reference to the conference for which it has been instantiated so that the conference state information base

[22] The GCC threads do not make use of e. g. a multi-threading mechanism provided by an operating system. Rather, the GCC threads resemble co-routines and are implemented based on the internal scheduling mechanisms provided internally within the GCC provider.

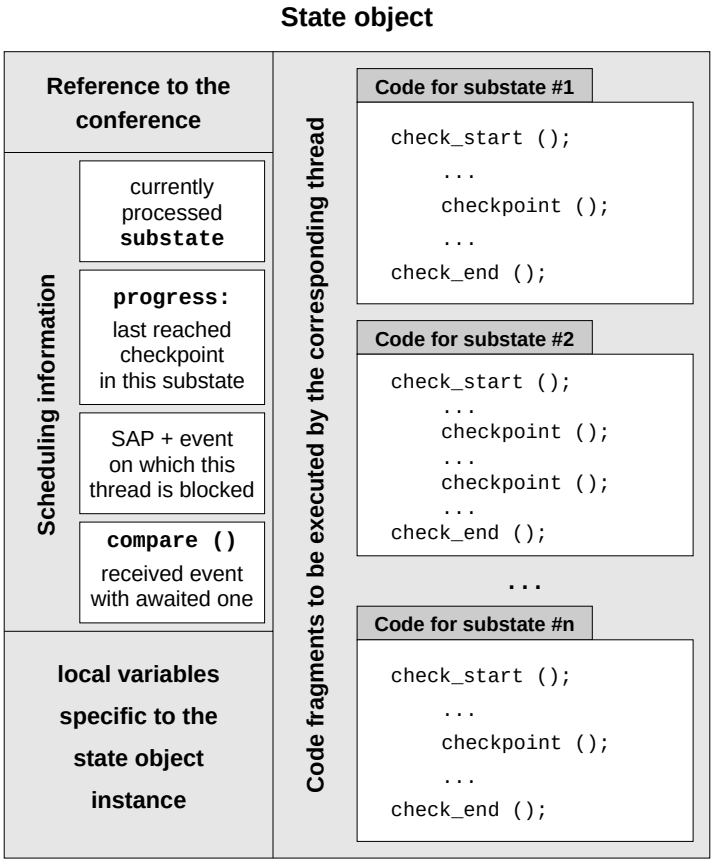


Figure 7.3: State object of the GCC provider

can be accessed. In addition, it contains the code (see below) and the local variables required for the execution of the corresponding thread. The remaining information in the state object describes the execution state of the thread:

- The code part of the state object contains the instructions to be executed by the thread carrying out the GCC protocol to provide the requested GCC service. The code is broken down into `substate`s each of which implements an ‘‘atomic part’’ of the protocol procedures. An atomic part is a sequence of actions that can be performed without the thread having to wait for information from other GCC providers or local entities. For example, after a request GCCPDU has been sent to the Top GCC provider, the local GCC provider may have to wait for the response before it can continue. In this case sending the PDU would be the last action of `substate` *n* and receiving the response the first action of `substate` *n+1*.

  Each `substate` may contain one or more checkpoints that mark points in the executed program code where temporary failures may occur that require the corresponding actions have to be retried later on. If a retry is needed, the thread blocks on the event it waits for and is scheduled as soon as this event occurs. For example, each attempt to transmit a GCCPDU may fail due to congestion of the MCSAP. In most cases, this failure condition is of temporary nature so that a retry is likely to succeed. Hence the thread puts itself to sleep and indicates that it is waiting for a particular MCSAP to unblock. As soon as the congestion is gone, the MCSAP object wakes up the waiting thread which then continues its execution.

- The `substate` component in the state object indicates the program code of which substate is currently executed by the corresponding thread, and the `progress` component contains a reference to the checkpoint within the current `substate` most recently passed by the thread. If the thread is blocked — either because it is waiting for a response from another entity during a transition from one substate to the next or because a protocol action has temporarily failed — the third component of the *scheduling information* describes on which event (of which SAP object) the thread is currently blocked. From the outside, this information is accessed through the `compare()` method of the state object. This method compares an event description passed as an argument to the event(s) the corresponding thread is waiting for and thus provides a mechanism to determine whether or not to wake up the thread when an event occurs. Calling this method for all threads blocked on a SAP object allows the SAP object to choose the thread(s) to schedule after e. g. a message has been received.

The GCC protocol consists of request, response, and indication GCCPDUs which are generated and transmitted as a result of a GCC service primitive being invoked or as reaction to another GCCPDU received before. As stated above, the entire GCC protocol is implemented as a set of threads executing the program code of state objects. The following discussion gives an overview of how the program code contained in state objects is structured into substates and which rules do apply for creating and terminating threads as well as for proceeding from one substate to the next.

- *Processing of requests from local clients*

For each request issued by a local entity (i. e. the node controller or any application protocol entity), a new state is instantiated. If the request can be handled locally — e. g. by retrieving information from the local GCC state information base for the conference — the request state object contains only a single substate: the thread processes the request, sends back a reply, and terminates virtually atomically. Examples are `GCC-Conference-Roster-Inquire` and `GCC-Conductor-Inquire`. Only a single substate is also needed for all requests that transmit a GCCPDU but do not require a response from a remote entity; examples are `GCC-Text-Message` and `GCC-Conference-Assistance`. If one or more information exchanges with other GCC providers or the local MCS provider are required, the state object contains multiple substates, one for each sequence of actions between interactions with other entities. This applies to most other GCC service request primitives.

- *Processing of requests and indications from remote providers*

As for local requests, for each remote request, a new state object is instantiated. If the request can be handled without interaction of the node controller, an application protocol entity, or with another GCC provider, the state object contains only a single substate: the thread processes the request, optionally generates a reply, and terminates in a single run. Examples are processing registry requests at the Top GCC provider. Incoming GCCPDUs that cause internal state updates and optionally lead to one or more indications being forwarded to the node controller or locally attached application protocol entities need only a single substate as well. This applies essentially to all indication GCCPDUs. If interactions with other local or remote T.120 system components are required, several substates are needed in the state object. Examples here are most conference establishment services such as `GCC-Conference-Join` and `GCC-Conference-Lock`.

- *Processing of responses*

When processing incoming responses — from another GCC provider or a locally attached entity — a state object that induced this response is expected to exist. If no corresponding state object is found, this indicates an error and the incoming GCCPDU or GCC IDU is ignored. If the state object is found, it is woken up and continues its processing (in the next substate). Typically, this leads to a either a confirm indication being issued to one of the local entities or a response GCCPDU being sent to the GCC provider that has originated the request. Afterwards, the state object is deleted.

The following example of a request for registering a channel with the GCC registry using the `GCC-Registry-Register-Channel` service illustrates the implementation of the corresponding T.124 protocol procedures with state objects and threads.

> On the requester side, upon receipt of the `GCC-Registry-Register-Channel` request, a new thread and its state object are instantiated. In substate #1 of the state object, the thread performs the local validity checks for the requested operation and then sends a `RegistryRegisterChannelRequest` GCCPDU via the MCSAP object to the Top GCC provider. Afterwards, the thread puts itself to sleep indicating that it is awaiting a `RegistryResponse` GCCPDU from the MCSAP object and that processing shall continue in substate #2 when the response arrives.
>
> At the Top GCC provider, a new thread and state object are instantiated upon receipt of the `RegistryRegisterChannelRequest`. The code for this thread is contained in a single substate. At first, the incoming GCCPDU is validated, then the requested register operation is performed on the registry database belonging to the conference in the context of which the request has been issued. The result of the operation is encapsulated in a GCCPDU which is sent back to the requester via the corresponding MCSAP object. This terminates the thread, and the state object is deleted.
>
> When the `RegistryResponse` GCCPDU is received by the requesting GCC provider, the MCSAP object is notified by the event dispatcher about an incoming message. The MCSAP object reads the message from the MCS API and determines the thread to be awakened by means of the `compare()` function — which matches for the thread created before. The thread then interprets the contents of the GCCPDU and generates a `GCC-RegistryRegisterChannel` confirmation by sending the corresponding GCC IDU to the GCCSAP of the application protocol entity that originated the request. Then, the thread terminates and the corresponding state object is deleted.

7.6. The GCC Main Function and Event Handling

When the GCC provider process is started, it reads a local configuration file and initializes itself accordingly. Then, it binds to the Control SAP of the MCS provider and establishes a GCCSAP to wait for attachment requests from the node controller and local groupware applications. All further actions of the GCC are triggered by external events: messages received from locally attached entities, GCCPDUs received from peer GCC providers, or timeout events. The prerequisite for any further action is that the node controller attaches to the Control GCCSAP. After doing so, the node controller informs the GCC provider on which addresses the MCS provider shall listen for incoming T.120 connections. The GCC provider performs the required actions to configure the MCS provider via the Control MCSAP.[23]

[23] Note that all these actions are purely local operations and therefore not covered by the MCS or GCC recommendations of the ITU-T but rather a matter of specific implementations.

After the setup is completed, the GCC is fully operational and may be used to actively create conferences as well as to accept conference creation requests from other GCC provider. All subsequent activity is triggered by events causing the corresponding SAP objects (including Timer and Memory objects) to become active. Events are recognized in by the dispatcher in the ‘‘GCC main loop’’ through operating system-specific means such as the `poll()` system call in UNIX and reception of window messages in Windows 95. The dispatcher determines the responsible SAP object based on the IPC connection it occurred on and dispatches the processing of the event to this SAP object.

Events triggered by the node controller or an application protocol entity part of a local groupware application process activate one of the GCCSAP objects. Events caused by another GCC provider lead to activation of one of the MCSAP objects. Timeout events enable the Timer object. If the event indicates that a message has been received, the corresponding SAP object reads the message, reassembles it (if necessary), and then decodes the message using the appropriate decoding functions. In some cases — such as a timeout, the indication of an incoming local GPCI connection, or a disconnect indication — no message interpretation is done. Based on the event type, the conference it relates to (if any), the message type (if any), and (parts of) the message contents, the SAP object checks for all threads that are blocked on this particular SAP object, whether one of the state objects indicates that a thread is waiting for this event. If so, it wakes up the corresponding thread and passes on all information about the received event. If no thread is waiting two courses of action are possible: the event is discarded if it would have required an already existing thread for further processing (e. g. because the event contains a response to a former request). Otherwise, a new thread is created for processing this particular event — i. e. a state object of the corresponding object class is instantiated — and all information about the event is passed to the new thread. The woken up or newly created thread processes the event; this typically includes updating the data base and sending messages to GCCSAP or MCSAP objects. In addition, timers may be started or stopped through the Timer object. Finally, the thread either puts itself to sleep or (if it has completed its task) terminates thereby deleting the corresponding state object. In either case, the thread returns control to the dispatcher which then waits for a new event to occur.

7.7. Summary

This chapter has described the Generic Conference Control Service as defined in the ITU-T recommendation T.124 and has introduced the implementation of the GCC provider. Like the MCS provider, the GCC provider has been realized as a separate process that communicates locally via the GPCI message passing mechanism. It accesses the services of the MCS provider via the IMTC MCS API and offers its services to the node controller and groupware application via the IMTC GCC API which conceals the underlying GPCI mechanisms on the client side. Internally, the GCC provider consists of four main components: the SAP objects (plus Timer and Memory objects) that represent inter-process communication relationships of the GCC provider; the threads and their associated state objects that implement the T.124 protocol; the GCC state information base that contains GCC provider-global as well as per conference information and is manipulated primarily by state objects; and the dispatcher that is responsible for recognizing and dispatching incoming events to SAP objects. The base SAP object class provides an abstraction from specifics of the operating system and communication libraries used for (local) interactions.

The implementation has been carried out for the UNIX (SunOS 4.1.3 and Solaris 2.5) and MS Windows (Windows 3.11, Windows 95, and Window NT) operating system environments. The GCC provider is implemented in an object-oriented fashion using C++ (GNU C compiler version 2.7.2 as well as Microsoft Visual C++ version 4.2) for the core of the provider. Roughly 100 C++ classes are used by the GCC provider. Besides C++, ANSI C has been used for environment specific extensions: for manually implemented ASN.1 encoding functions, the GPCI library which is shared with the MCS provider, and the IMTC GCC API library. The GCC provider implementation consists of approximately 33,000 lines of source code, including the UNIX and the Windows specific parts as well as the hand-coded ASN.1 routines, but excluding the GPCI and IMTC API libraries.

Encompassing and detailed descriptions of the concepts of the GCC provider and its internal data structures can be found in [Degener 95] for the initial UNIX-based prototype and in [Radig 96] for the final multi-platform version of the GCC provider.

Finally, it should be explicitly noted that, while the MCS provider has been extended to make use of multicast-capable networks in order to increase its efficiency and scalability, the author has not developed comparable optimizations to GCC as part of this thesis. In the IETF, the author has been working on a scalable conference control protocol that provides services which are functionally compatible to those offered by GCC [Bormann *et al.* 96a], and many of the concepts developed there would in principle be applicable to T.124 as well. The major reason for not integrating optimizations into the GCC provider is that — in contrast to the MCS extensions which may be applied in self-contained multicast islands anywhere in the MCS domain hierarchy (and did not affect the overall MCS protocol at all) —, effective GCC protocol extensions would need to be supported by most nodes, in particular the Top GCC provider as well as the other MCUs and multiport terminals. That is, the crucial requirement of backward-compatible applicability is not satisfied for GCC extensions. Instead of pursuing non-standard approach, the author has participated in the (still ongoing) refinement of the GCC protocol in the ITU-T standardization process thereby contributing to backward-compatible efficiency improvements for T.124.

8

Conclusion

In this thesis, the development of an architecture for multipoint data communication in multimedia teleconferences has been described, and an infrastructure that follows the concepts of this architecture has been implemented based on international standards. The first two sections of this final chapter briefly review the results of this thesis: the conceptual achievements comprising contributions to the modeling of teleconferences and the design of the MCL; and the engineering work resulting in an implementation of the MCL based upon the T.120 series of recommendations and its integration with the EURO.VISION DMC system.

As part of the research carried out for this thesis, the author has contributed research findings as well as experience gained from the implementation to the process of international standardization in the ITU-T and the IETF. This has significantly influenced the future direction of T.120 standardization the author's view of which is outlined in section 8.3. In the final sections, the author presents his perspective on current and future (technological) developments in the area of multimedia teleconferencing, with respect to the standardization as well as to the market, respectively.

8.1. Conceptual Achievements

Following a rough outline of the research subject and the definition of the most relevant terms in the introduction to this thesis, chapter two has provided an overview of the research area of groupware and teleconferencing systems. It has laid the technical and terminological foundations for the subsequent chapters, and it has set the scope for the development of the teleconferencing infrastructure in this thesis.

In particular, the author has developed an encompassing model describing teleconferences as well as the associated terminology and has identified the technically relevant characteristics of (tele)conference settings. Based on these considerations, the author has defined the target set of business teleconferences for this thesis as well as the functionality to be covered by a DMC system. These functional requirements together with the need to base DMC systems on international standards serve to define the frame to be filled out by the multipoint data communication infrastructure developed in this thesis.

Chapter three has described the basic conceptual design efforts carried out for this thesis: the development of a conceptual framework for a multipoint data communication infrastructure for teleconferencing systems. Based on past experience in the development of teleconferencing systems as well as on a review of communication architectures for teleconferencing systems, the author has developed the concept of the Multipoint Communication Layer (MCL). The functional subdivision of the MCL services into four sublayers — which were developed by the author well in advance to most of the corresponding standardized architectures — largely matches the concepts that eventually appeared in the relevant international standards in this area. The MCL's functional range is more complete with respect to the requirements of groupware applications (and the fact that the standards bodies meanwhile have identified that some of the additional MCL services are actually missing in the T.120 architecture confirms this observation).

8.2. Engineering Results

The *Data Protocols for Multimedia Conferencing* as defined in the ITU-T T.120 series of recommendations approximately meet the requirements outlined in the MCL architecture and are the single sufficiently complete suite of standards addressing these issues as well as widely accepted in the industry. Therefore, in the EURO.VISION project, this suite of standards has been chosen as the basis for the implementation of the multipoint data communication services in the DMC system and the underlying DMC toolkit. Chapter four has introduced the T.120 series of recommendations as well as the overall system architecture of the EURO.VISION system and the desktop multimedia teleconferencing software development kit. In particular, the integration of the T.120 components and the implementation of the relevant interfaces have been outlined.

In chapters five, six, and seven the author has described the specific implementation of the major T.120 infrastructure components: MCS and GCC. Chapters five and seven have introduced the services and the protocol characteristics as well as the implementation concepts of the MCS and GCC providers, respectively. Chapter six has presented the multicast extensions for the MCS which the author has developed in order to fill in the most important piece of the MCL not covered by T.120: a transport protocol hierarchy for multicast-capable networks the use of which significantly enhances the scalability of the T.120 infrastructure (refer also to the next section).

Altogether, the engineering efforts in the context of this thesis have produced a fully tested implementation of all the T.120 infrastructure components, and this implementation has been integrated with the architecture of the DMC SDK underlying the EURO.VISION system in order to enhance the standard-conforming audiovisual communication functionality by mechanisms for elaborated conference control and data communications. The standard-conformance of the T.120 infrastructure implementation has been verified through extensive and structured testing.[1] Moreover, the implementation has proven its interoperability with implementations from other vendors in various test events organized by the International Multimedia Teleconferencing Consortium (e. g. [IMTC 96]).

[1] It should be mentioned that a test environment has been developed in the DMC SDK project forming the context of this thesis. This test environment has also been used for extensive testing of the implementations of the T.123 protocol hierarchies as well as the MCS and GCC providers [TELES 96].

8.3. Prospects in T.120 Standardization

Much of the research for this thesis has been done in close coordination with international standardization, in the ITU-T as well as in the IETF. Because of the focus of this thesis, the T.120 standardization in the ITU-T has been of primary interest, with accompanying participation in the IETF for harmonization purposes. The author has actively contributed to the T.120 standardization process, with the aim of achieving richer functionality, more complete services, and more efficient protocols. In particular, the author's extensions to T.123 and the MCS protocol described in chapter six and related contributions have initiated the debate about achieving scalability in the T.120 infrastructure and significantly influenced the solution being developed.

The new work items brought up in 1996 in the T.120 working group primarily address improvements to the infrastructure recommendations T.122, T.123, T.124, and T.125. In particular, the current standardization efforts are undertaken

- to improve the scalability of the T.120 infrastructure by one or two orders of magnitude: from a few tens to several hundreds or even thousands of participants;[2] and
- to extend the services provided by the T.120 recommendations in order to make T.120 suit a wider range of groupware applications as well as to fix bugs in the first revision of the T.120 series.

Both these aspects are detailed in the subsequent two subsections.

Additional efforts are ongoing in the T.120 working in order group to provide additional application services, namely for application sharing, for the integration of audiovisual control, and for advance reservation of conferences. See section 4.1 for a snapshot of these recommendations in progress.

The author looks forward to further contributing to the entire T.120 refinement process as designated editor for the next revision of the T.124 recommendation.

8.3.1. Scalability Issues

As already pointed out in chapter six, increasing the scalability of the T.120 infrastructure requires extensions and modifications at several layers. While, in this thesis, the author has focused on the MCS layer for reasons of backward compatibility, the refinement of T.120 now being pursued in the T.120 working group encompasses all T.120 infrastructure recommendations. This endeavor is outlined subsequently.

A new protocol profile to be added to T.123 is being worked on since October 1996 which essentially follows the concepts introduced in chapter six. This profile specifies the use of MCS on top of a reliable multicast transport, an adaptation protocol T.MAP (originally named T.MT until the January 1997 meeting) [Galvin 97], and the required procedural modifications and extensions to the MCS protocol. At the time of writing, the service requirements for the multicast transport are being specified. However, no transport protocol has yet been chosen due to the lack of a standardized solution; the two candidates currently discussed in the T.120 working group are MTP-2's successor MTP/SO [Bormann *et al.* 97] and RMP [Whetten *et al.* 94]. Consensus is

[2] Solving this problem is a prerequisite to (better) accommodate the use of T.120 within the Internet.

emerging that if the IETF fails to produce a standardized reliable multicast transport protocol in the near future, the T.120 working group is going to pick a suitable candidate and do the standardization on its own. Finally, the multicast extensions to the transport profiles also have to be reflected in the MCS protocol description (the MCS services are not modified in the course of these changes): an annex to T.125 is being defined that describes how the MCS service provider makes use of the services provided by T.MAP.

The multicast extension to MCS now being standardized in the ITU-T is conceptually equivalent to the approach which the author has introduced in chapter six: a reliable multicast transport plus an adaptation protocol provide a suitable platform for the MCS providers. Some of the MCS provider's connection control and data transmission procedures are refined to make use of the additional services. Both approaches may use identical multicast transport protocols. The only significant differences are at the adaptation protocol layer, between MMAP and T.MAP: a) T.MAP relies on *additional* point-to-point connections for transmitting control MCSPDUs and for determining loss of connectivity (which may also be provided by the multicast transport, however) while MMAP does not require these; and b) in T.MAP switching from unicast to multicast occurs individually per peer MCS provider within a multicast island which avoids complex synchronization phases, at the expense of requiring much more state information to be kept per MCS provider. These differences imply (minor) differences in the modifications to be applied to the MCS provider in both approaches.

Besides revisions and extensions to T.123 and MCS, two important refinements for the GCC specification are currently being worked out [Pulito 97]:

1) The process of updating the GCC conference and application roster is streamlined; instead of transmitting the full roster information to all nodes whenever a change occurs, only information about the added, deleted, or changed entries is communicated.

2) Two new types of T.120 nodes are introduced to reduce the size of the roster, the amount of information to be transmitted, and the number of nodes to receive the roster information: *anonymous nodes* and *counted nodes*.[3] The new node types represent participants that are anonymous to the conference (such as members of the audience in a panel discussion). The respective participant's names and other conference roster information are not disseminated to the conference, nor are the node's capabilities. T.120 application protocol entities at nodes of either type may actively participate in application sessions of the conference to the degree their capabilities match the required ones.

All the previously listed items are short term issues. Except for T.123, the respective new recommendations as well as the required changes to existing ones are intended to be technically stable by March 1997 and shall be ratified by the ITU-T Study Group at the beginning of 1998 [DeGrasse / Lyons 97]. As stated before, the schedule of T.123 depends on the availability of a standardized multicast transport protocol.

[3] The single difference between those two node types is that the number of counted nodes participating in an application session becomes known to the regular GCC nodes while anonymous nodes do not show up at all. Counted nodes are required for certain application protocol scenarios that, however, are not further discussed here.

8.3.2. Service Extensions to the Infrastructure Components

Besides the extensions described in the previous section which address the most immediate needs, a variety of issues is considered for the mid-term development of the T.120 series of recommendations. The following list contains a subset of the work items that have been identified at the T.120 working group meeting in January 1997 for further discussion and potential inclusion within existing or new recommendations of the T.120 series.

- One of the most pressing requirements is to provide means for dynamically changing the Top MCS and Top GCC Providers — upon request as well as to recover from crashes of the Top Providers — to overcome that a Top Provider constitutes a single point of failure. In conjunction with this, dynamic balancing of the tree constituting an MCS domain is also being discussed.

- A variety of extensions are highly desirable to be added to the data transmission services of MCS:
  - First of all, MCSPDUs need to be scheduled for forwarding according to their respective priority; in addition, it should be guaranteed that an MCSPDU can never be overtaken by one of a lower priority. This requirement helps in synchronizing the flow of PDUs in application protocols. In conjunction with this, the usefulness of explicit mechanisms for cross channel and cross priority synchronization is investigated.
  - Furthermore, the MCS should incorporate means for domain-wide rate and flow control on a per channel and/or per application session basis. The range of ideas discussed includes the assignment of maximum data rates to an MCS priority, to an application session, and maybe even to an application protocol entity. The agreed on maximum throughput rates should then be enforced by the MCS providers.
  - Also, the MCS should support compression as well as encryption of the payloads of MCS PDUs at least on a per channel basis.
  - Finally, new service primitives are considered for introduction: unreliable data transmission, reliable data transmission without sequencing, and support of data transmission with aggregated acknowledgments — the latter of which has already been discussed in early drafts of T.122.

- Ideas for extensions to further increase scalability include, in analogy to the anonymous node type used in GCC, allowing groupware applications not sourcing data to attach to the MCS domain locally without obtaining an MCS user id.

- Another feature considered useful but not yet provided by MCS is a service to provide state information about a domain upon request: how many tokens are assigned, how many and which channels are in use, etc. In particular, the token management of MCS should be enhanced to allow an application protocol entity to determine who is owner of a token.

Finally, although the current focus is on MCS, several extensions specific to GCC are being discussed, too. These include providing a standardized way for mutual authentication when joining a conference, mechanisms for assigning and exchanging encryption keys, and extensions to the conference control services to accommodate the needs of the T.130 recommendations dealing with audiovisual control in teleconferences.

Draft text for recommendations covering these issues is expected to emerge within 1997 so that a first overview of these extensions can be presented at the first ITU-T Study Group meeting in 1998 [DeGrasse / Lyons 97].

8.4. Teleconferencing Standardization

Throughout this thesis, the author has repeatedly pointed out the importance of international standardization to the entire research area of multimedia teleconferencing and particularly to the author's design and implementation efforts.[4] As mentioned above, in principle, standardization work is being done because any type of telecommunication technology requires a critical mass of users to eventually become accepted, and because this critical mass is generally only achievable if products from different vendors that serve the same purpose do interoperate:[5] *at first*, the market has to be created; *then*, the vendors can compete for market shares. This insight is well established among all those participating in international standardization, and the number of participants is increasing. While, some ten years ago, standardization was dominated by the PTTs, within the last few years manufacturers of (what the PTTs consider to be) customer equipment become more and more relevant, particularly in the area of multimedia communications. Even large companies such as IBM, Intel, and Microsoft — that have been *setting* standards in the past — now are *negotiating* on them to achieve a consensus on which they can base their products and which significantly contribute to interoperability.[6] However, although

- the need for blind interoperability is well recognized as are
- the economic disadvantages for all competitors if multiple competing solutions finally come into existence (because each company then has to implement all approaches to achieve interoperability);

and although

- the industry agrees on the concept of achieving a consensus for which
- the international standardization bodies provide a suitable vehicle,

today, to many telecommunications problems there are often several ''standardized'' solutions. As TANENBAUM puts it: ''The nice thing about standards is that you have so many to choose from; (...)'' [Tanenbaum 81, p. 168].

This is typically due to the fact that there are several standards bodies — such as ISO, ITU-T, IETF, ETSI, and many others — each of which potentially has one or more groups working (independently) on the same or related topics. Often, working groups are not aware of one another: either because their members do not know about the other group's existence, or because they do not want to know and do not care. The former problem is overcome as soon as the groups get to know about each other (e. g. through individuals or organizations) informing one group about the other(s), and subsequently liaisons between the groups can be used to coordinate their activities. The latter problem is more severe because this means that the majority of the group members

[4] The author has started his standardization activities in 1993 and has been participating in an average of ten standards meetings per year for the last two years in several working groups of the ITU-T and the IETF as well as in two industry fora.

[5] The most prominent examples of successful standardization are the telephone and telefax systems.

[6] Product differentiation is achieved through e. g. quality, support of additional (optional) functionality, integration with other services, user interface design, etc.

deliberately ignores the other group's existence and thus hinders collaborations from the beginning. This behavior may have various reasons including the following:

- The *not invented here phenomenon* describes the fact that a group rejects to adopt (parts of) work others have already done because the members of the group believe they can do a better job, do not fully understand the rationale and the background of the already existing approach, or are simply approaching the problem from a different angle.
- Frequently, the policies of the involved companies request that (new) standards to be defined match the company's products (if there are such) as close as possible. This may even mean that the standardization process is delayed if the desired match cannot be achieved or if the own product development is not sufficiently far advanced. As a result, two groups dominated by different companies are likely to produce incompatible outcomes.
- Large enterprises have further means besides domination of standards groups: if in no important group sufficient influence can be obtained, such a company may found a specially focused industry consortium in the context of which it further develops its favored approach.[7]
- Similar politics may also be pursued by individuals: if one standards group does not like a proposal or an idea, the (group of) person(s) tries to convince the next group, until eventually a group is found that is willing to adopt the proposal. Such behavior is particularly observed when e. g. editorship of a document increases the persons' reputation or the final standards documents do even carry the names of the author(s).
- Finally, at least in UN or EU related bodies — such as ISO, ITU-T, and ETSI — standardization is also much of a political business of the governments of the participating nations. Each countries' delegates try to establish competitive advantage for their local industry. Political disputes between delegations of different countries may heavily affect (the acceptance of) the outcome of technical work.[8]

From the viewpoint of standardization as well as from that of the industry as a whole, such trends are obviously undesirable because different and incompatible solutions contribute to confusion of potential customers and may impede the evolution of the market.

In the particular example of teleconferencing, the development of the T.120 series of recommendations is only one out of many standardization activities — that are coordinated to some degree but are not yet entirely in alignment. As already described in section 3.1, the important activities besides T.120 comprise the ITU-T work carried out for the H.300 series of recommendations (i. e. audiovisual communication) as well as the IETF efforts in the AVT and MMUSIC working groups.[9] [10]

---

[7] Particularly in the area of teleconferencing, a variety of consortia has been in existence over the last years. Finally, all of them merged to form the International Multimedia Teleconferencing Consortium (IMTC). The most recent attempt to introduce a new ''standard'' — competing with the already completed work standardized in the ITU-T — has been undertaken by some companies founding the VoIP forum in 1996. However, its members eventually accepted the ITU-T standards and the VoIP forum was integrated into the IMTC in the end of 1996.

[8] For example, the dispute between France and Germany about a common programming interface for call control services in ISDN led to the fact that eventually two standards were decided by the ITU-T: the French PCI and the German CAPI.

[9] AVT is short for *Audio-Visual Transport*, MMUSIC is the abbreviation of *Multiparty Multimedia Session Control*.

[10] In addition, several groups in the ITU-T and the ISO standardize audio and video coding algorithms.

These various groups have evolved from and initially concentrated on distinct focuses of work, including the following:

- circuit-switched networks vs. packet-switched (inter)networks vs. cell-switched (ATM);
- networks with guaranteed vs. networks without guaranteed quality of service;
- point-to-point vs. multipoint communications; and
- real-time communications vs. data communications vs. control aspects.

Originally, multimedia communication work within the ITU-T has focused on circuit-switched and ATM networks, for all of which a guaranteed quality of service was assumed. Study Group 15 of the ITU-T dealt with real-time communication and control for point-to-point connections (H.300 series of recommendations). Three different subgroups (in the ITU-T, such working groups are termed *Questions*) addressed ISDN, PSTN, and ATM, respectively. Question 10 of Study Group 8 worked on multipoint communication and elaborate conference control (T.120).

The IETF is focused on IP-based (i. e. packet-switched) internetworks for which initially no assumptions about service quality, throughput, or latency are made,[11] and which support point-to-point as well as multicast communication. While the AVT group addresses real-time communication aspects in the Internet environment; control issues are dealt with in the MMUSIC working group.

In the recent years, the rapid progress in multimedia conferencing standardization has created at least two areas of overlap:

- In the area of elaborate multipoint conference control as well as control of real-time streams and devices, between the relevant Questions in Study Group 15 on one side and Question 10 of Study Group 8 on the other: in addition to the elaborate framework for dealing with multipoint conference and audiovisual control developed in the T.120 series, SG15 has developed its own (point-to-point) control protocol which is being expanded to cover simple multipoint settings as well. Before 1995, not too much collaboration took place towards harmonization of the two approaches.
- For Internet-teleconferencing and Internet telephony between the ITU-T and the IETF: Study Group 15 has developed H.323 describing teleconferencing for IP-based (intra)networks, the original aim of which was to provide interoperability between IP-based systems on one side and the traditional PSTN, ISDN, and ATM-based systems on the other. During the development of H.323, this scope was expanded to include usage of the protocol in the entire Internet, for teleconferencing as well as for Internet telephony.

In both cases, collaboration between the respective working groups has eventually been achieved through individuals that actively attended most of the meetings, thereby conveying information and bringing people from all sides together. Some of the achievements resulting from the collaboration and the mutual acceptance of the various groups should be mentioned here:

- Despite initial resistance within the H.323 working group, the Internet Proposed Standard for a Real-time Transport Protocol (RTP) has become standardized part of H.323.

[11] As already stated in subsection 3.1.6, various groups in the IETF address the issues relevant for providing quality of service guarantees in the Internet.

- The H.323 and the T.120 working groups decided on a road map towards the harmonization of their respective protocols for audiovisual and conference control during a joint meeting in October 1996.
- The T.120 working group has acknowledged the expertise of the IETF in the area of multicast protocols and does not intend to develop a reliable multicast transport protocol for use underneath MCS on their own — unless the IETF fails to define a suitable protocol within a reasonable time frame (see also section 8.3.1).
- One of the most recent developments in the H.323 group — the work on panel conferences with a small core group of participants and a huge audience [Kumar 97] — makes use of all the existing IETF protocols as appropriate to avoid duplication of functionality, and to achieve backwards compatibility with existing Internet conferencing tools that exclusively based on IETF protocols.

The author was one of the first individuals going to and contributing to IETFs, T.120, H.323, and IMTC meetings, thereby contributing to the creation of links between those standardization bodies of most importance to teleconferencing. Meanwhile, other individuals have joined this endeavor, and also the IMTC supports the effort to make relevant ITU-T and IETF groups become more aware of and talk to one another.

Overall, the industry has accepted the need for worldwide collaboration in this area, and the resulting collaboration between all standardization and industry groups seems to be leading towards a single approach towards (tightly-coupled) teleconferencing:

> The baseline of tightly coupled conferencing as well as Internet telephony is the ITU-T Recommendation H.323, with well-defined gateways provided to conventional telephony and multimedia conferencing over PSTN, ISDN, and ATM [Reid 97].
>
> Groupware applications such as application sharing, shared whiteboard, file transfer, etc. are based on T.120 as is elaborate conference control. Rudimentary audiovisual control services are covered by H.323, for more elaborate mechanisms T.130 services will be used.
>
> Transmission of real-time information in packet-switched networks (such as the Internet and intranets) is based on the respective IETF standards as is the entire domain of loosely-coupled conferencing. For panel-style conferences, a new H.323-related recommendation ties together the existing ITU-T and the IETF standards without much additional specification being needed.

The growing awareness of the need for close collaboration in the entire area of teleconferencing in particular and multimedia communication in general is now also being reflected in the organizational structure of the probably most important standardization body in this area: the ITU-T. In the past, there was no specific place within the ITU-T where standardization of multimedia services was to be performed so that activities related to multimedia communications independently appeared everywhere. The ITU-T has recognized the urgent need for harmonization of all the related activities, and has created a new Study Group specifically for multimedia services: Study Group 16. This new Study Group — which is to constitute itself during its first meeting in March 1997 — comprises all those Questions from various Study Groups that are related to multimedia communications. The includes not only the aforementioned Questions from Study Group 8 and 15, but also those dealing with related issues: modem technology, network and terminal equipment, and general service requirements. Finally, it should be noted that the ITU-T is in the process of accepting the IETF as a standardization body to which official liaisons may exist and the documents of which may be referenced (still in 1996, for T.120 and H.323 standardization, the

contents of relevant IETF documents had to be reproduced as part of the respective ITU-T recommendations).

The global consensus about teleconferencing reached within the industry, together with the refined standardization procedures which are expected to help avoiding future divergence of efforts up front, form the fundamental organisatorial prerequisites for more successful deployment of teleconferencing technology in the future — compared to the poor results achieved in the past.

8.5. About the Future of Teleconferencing

The previously described convergence of the industry towards a (set of) standardization solutions instead of proprietary products is only one out of many trends that could be observed during the historic development of the various types of teleconferencing systems as outlined in subsection 2.1.4. Summarizing what has been stated before in this thesis, during the evolution from early telecommunication means such as the telephone over the variety of teleconferencing system types to the current state-of-the-art technology, namely DMC systems, at least the following in part interrelated trends could be observed in addition to standardization:

- from ‘‘mono’’media to multimedia communication — from systems initially only capable of pure audio or text communication, via audiovisual and audiographics systems, to multimedia conferencing systems that even include groupware applications;
- from conference rooms to the desktop — from the initial requirement for specifically equipped conference rooms for teleconferencing via rollabout systems to desktop systems (stand-alone appliances as well as computer-based systems);
- from point-to-point to multipoint communication — from point-to-point communication to MCU-based multipoint conferencing with the prospect of (present and) future networks increasingly supporting multipoint communication as an inherent feature, thereby eliminating the need for dedicated central systems;
- from stringent to relaxed requirements on networks — from required high bandwidth and isochronous transmission services (that could only be met by a few dedicated networks at that time) to the capability of running teleconferences on virtually any commonly deployed network type;
- from stand-alone components to integrated systems — from separate rooms over separate rollabout and desktop devices to systems that are integrated into the everyday working environment: the workstation computer; and
- from very expensive (100,000+ DM for room-based systems and some 40,000 DM for early video telephones) to inexpensive systems (less than 500 DM for upgrading a workstation to a simple DMC system).

These developments together with the fact that the base technology is now maturing provide a much better technological and economic foundation for widespread deployment of DMC systems compared to room and rollabout conferencing systems in the past — which have never succeeded in the market place in spite of the repeated bright market forecasts for this technology starting in the early 1970s.[12] That is, at least from the technical perspective, DMC systems as described in

[12] Refer e. g. to [Snyder 71], [Lineback 82], [Frost and Sullivan 83], [Bohm / Templeton 84], [Hudetz *et al.* 89], [Funkschau 91], and Arthur D. Little and Frost and Sullivan quoted in [Schindler 94b].

this thesis seem to be the right approach towards eventually making teleconferencing technology succeed. And market forecasts promising a bright future of teleconferencing continue to appear, now with the focus on desktop multimedia teleconferencing (e. g. [Rügenheimer 96], [Zillich 96]).

However, the observed failures in the past of teleconferencing and other kinds of groupware systems and the — obviously legitimate — doubts about their (wide) applicability have shown that there is more to computer-supported collaboration and in particular teleconferencing than just technology. In the author's opinion, one important key to success of teleconferencing, that has not received so much attention in the past, is the broad education of the public, because many people are potential users. Teleconferencing (DMC) systems should no longer be perceived as high end business equipment even most computer-literate people do not know about. Rather, people should get familiar with this technology (in everyday life) so that they eventually rate DMC systems as convenient means for communication and collaboration — as they now rate the telephone.

> In industrialized countries, virtually every household has a telephone and at least one TV set; furthermore, the number of people using answering and telefax machines as well as computers at home is rapidly increasing. The generation of those now being in the twenties has grown up with telephone and TV could barely imagine life at home without this technology. In offices, telefax machines and (networked) workstations are meanwhile considered essential equipment.
>
> Similarly, a next generation growing up (and thus being sufficiently familiar) with computers, multimedia technology, and also DMC systems is likely to find the usage of these tools natural in daily life, at home as well as in the office.[13]

However, particularly with respect to wide availability of DMC technology, the world is undergoing a significant change since 1996, compared to the 1970s, 1980s, and even early 1990s. DMC technology is now available to every PC user at virtually no cost; and the DMC systems are increasingly integrated with software packages that the user is already familiar with.

> For example, the German-based TELES AG has integrated a fully standards-compliant ISDN-based DMC system with their extensive ISDN software package, and in addition offers virtually free downloads for PSTN and IP-based systems.
>
> Microsoft and Intel are offering free downloads of IP-based DMC systems from their respective WWW sites [Microsoft 96b] [Intel 96]. Netscape is planning to integrate DMC technology with their essentially free-of-charge WWW browser, and Microsoft is likely to follow suit.
>
> Since early 1997, the US company 8x8 is selling for around US$ 500 a stand-alone device that connects to telephone line and the television set and converts the latter to a standard-conforming video telephone.

As a result of these efforts, unparalleled throughout the history of teleconferencing, the number of people getting in touch with this technology is rapidly increasing,[14] so that the first step towards a broader technological education of the population has been taken. This may pave the way for

[13] Children who are used to audiovisual communications with their parents — e. g. because the parents do have videophones at home and in their office and use this technology to talk to their children daily — do no longer judge a voice-only telephone call with a parent to be suitable for a real conversation [Baker 96].

[14] Intel has reported more than 50,000 downloads of their *iphone* within the first week of its availability on their WWW site in 1996. Microsoft is expecting that about 20 million copies of their product NetMeeting will be in use by summer 1997.

other (private) uses of (desktop) multimedia conferencing technology during leisure time and for social purposes as envisioned by SCHINDLER [Schindler 92a] and briefly described in chapter three of this thesis.

- For example, ubiquitously available teleconferencing technology may offer new ways for the integration of elderly or disabled persons and may provide them with means for social interactions that were simply not conceivable before.

- Also, DMC technology may enable new types of leisure time activities. Networked multi-user games are already gaining popularity; and combining e. g. adventure games with natural ways of interpersonal communications between the players — and possibly virtual reality — would provide a new dimension of group entertainment.

The variety of possible and useful areas of application for DMC and other teleconferencing systems emphasizes the effects that this technology may have on everyday life. Its broad applicability, particularly in social and private areas, paired with its easy and inexpensive availability may favorably influence the knowledge about and acceptance of this technology within our society. This time, market analysts may eventually be right and teleconferencing technology may actually take off to become a ubiquitously available means for interpersonal communication and collaboration — not just in business work groups.

9

References

[Abdel-Wahab / Jeffay 92]
Hussein M. Abdel-Wahab and Kevin Jeffay, “Issues, Problems, and Solutions in Sharing X Clients on Multiple Displays,” Technical Report (TR92-042), UNC-Chapel Hill, December 1992.

[Altenhofen 90] Michael Altenhofen, “Erweiterung eines Fenstersystems für Tutoring-Funktionen,” Diplomarbeit, Universität Karlsruhe., 1990.

[Americana 62] *The Encyclopedia Americana*, 7, Rand McNally & Company, 1962.

[Arango *et al.* 93] Arango et al., “The Touring Machine System,” *Communications of the ACM*, vol. 36, no. 1, pp. 68-77, The Association for Computing Machinery, Inc., January 1993.

[Baecker 93] Ronald M. Baecker (ed), *Readings in Groupware and Computer-Supported Cooperative Work*, Morgan Kaufmann Publishers, Inc., San Mateo, USA, 1993.

[Baker 96] Richard Baker, PictureTel Corporation, personal communication, 1996.

[Ballardie *et al.* 96]
A. Ballardie, S. Reeve, and N. Jain, “Core Based Trees (CBT) Multicast — Protocol Specification,” Internet Draft draft-ietf-idmr-cbt-spec-06.txt, Work in Progress, September 1996.

[Bastian 92] Jan Bastian, “Entwurf und Implementierung eines Eingabe-Anpassungsmoduls für VPAX und X-Windows,” Studienarbeit, Technische Universität Berlin, 1992.

[Bates / Segal 90] P. Bates and M. Segal, “Touring Machine: A video telecommunications software testbed,” in *Proceedings of the First International Workshop on Network and Operating System Support for Digital Audio and Video*, Berkeley, California, 1990.

[Berners-Lee *et al.* 92]
T. J. Berners-Lee, R. Cailliau, J.-F. Groff, and B. Pollermann, "World-Wide Web: The Information Universe," *Electronic Networking: Research, Applications, and Policy*, vol. 2, no. 1, pp. 52-58, Spring 1992.

[Bernstein 96] Jeff Bernstein (ed), "T.120 Implementor's Guide," ITU-T Study Group 8, Question 10, 1996.

[Berresheim 92] Alexander Berresheim, "Redesign/-implementierung einer Videokonferenz-Benutzeroberfläche," Studienarbeit, Technische Universität Berlin, September 1992.

[Beyer 91] Ulrich Beyer, "Optimierung des X-Servers Version 11 Release 4 und Erweiterung um ein Protokoll zur Unterstützung BERKAPS-spezifischer Fenstertypen," Studienarbeit, Technische Universität Berlin, Juli 1991.

[Beyer 92] Ulrich Beyer, "Redesign/-implementierung eines Konferenzmanagement-Systems," Diplomarbeit, Technische Universität Berlin, 1992.

[Birkicht 96a] Bernd Birkicht, "GPCI Message Definitions for the GCC API," Internal Project Documentation, TELES AG, June 1996.

[Birkicht 96b] Bernd Birkicht, "Specification of the T.121 API," Internal Project Documentation, TELES AG, November 1996.

[Birman *et al.* 91] Kenneth Birman, André Schiper, and Par Stephenson, "Lightweight Causal and Atomic Group Multicast," *ACM Transactions on Computer Systems*, vol. 9, no. 3, pp. 272-314, August 1991.

[Birman *et al.* 94] K. P. Birman, F. Mattern, and A. Schiper (eds), *Theory and Practice in Distributed Systems*, Springer Verlag, 1994.

[Birman / Joseph 87]
K. Birman and T. Joseph, "Reliable Communication in the Presence of Failures," *ACM Transactions on Computer Systems*, vol. 5, no. 1, pp. 47-76, February 1987.

[Bohm / Templeton 84]
R. J. Bohm and L. B. Templeton, *The Executive Guide to Video Teleconferencing*, Artech House, Inc., Dedham, MA, 1984.

[Borenstein / Thyberg 88]
Nataniel S. Borenstein and Chris A. Thyberg, "Cooperative Work in the Andrew Message System," in *CSCW'88: Proceedings of the Conference on Computer Supported Cooperative Work*, pp. 306-315, The Association for Computing Machinery, Inc., Portland, 26-28 September 1988.

[Borenstein / Thyberg 90]
Nathaniel S. Borenstein and Chris A. Thyberg, "Power, ease of use and cooperative work in a practical multimedia message system," in *Readings in Groupware and Computer Supported Cooperative Work assisting human-human collaboration*, ed. Ronald M. Baecker, pp. 485-500, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1993.

[Bormann 91] Ute Bormann, “Internationale Standards für Informationssysteme — Systemarchitektur,” Skript zur Lehrveranstaltung ISIS 1, Technische Universität Berlin, Fachgebiet Kommunikations- und Betriebssysteme, 1985, 87, 91.

[Bormann *et al.* 94a] Carsten Bormann, Jörg Ott, and Gero Hoffmann, “First Experience with Multicasting the X Protocol,” in *Broadband Islands Third International Conference*, Elsevier, 1994.

[Bormann *et al.* 94b] Carsten Bormann, Jörg Ott, Hans-Christian Gehrcke, Torsten Kerschat, and Nils Seifert, “MTP-2: Towards Achieving the S.E.R.O. Properties for Multicast Transport,” in *Proceedings of the 1994 International Conference on Computer Communications and Networks (ICCCN ’94)*, 1994.

[Bormann *et al.* 94c] Carsten Bormann, Jörg Ott, Hans-Christian Gehrcke, Torsten Kerschat, and Nils Seifert, “Multicast Sockets: A Reliable Multicast Transport Service Interface and its Implementation,” in *Proceedings of the Workshop on New Protocols for Multimedia Systems, Technische Universität Berlin*, 1994.

[Bormann 95] Carsten Bormann, “The Newscaster Experiment,” Research Note, Universität Bremen, October 1995.

[Bormann *et al.* 96a] Carsten Bormann, Jörg Ott, and Christoph Reichert, “Simple Conference Control Protocol,” Internet Draft draft-ietf-mmusic-sccp-00.txt, Work in Progress, June 1996.

[Bormann *et al.* 96b] Carsten Bormann, Jörg Ott, and Nils Seifert, “MTP/SO: Self-Organizing Multicast,” Internet Draft draft-bormann-mtp-so-00.txt, Work in Progress, November 1996.

[Bormann *et al.* 97] Carsten Bormann, Jörg Ott, and Nils Seifert, “Service description for the MTP/SO reliable multicast transport protocol,” T120C-116, contribution to the Rapporteur’s meeting of ITU-T Study Group 8, Question 10, 13-17 January 1997.

[Bostrom *et al.* 91] R. Bostrom, R. Anson, and V. Clawson, “Group Facilitation and Group Support Systems,” Working Paper, Department of Management, University of Georgia, 1991.

[Braden *et al.* 96] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, “Resource ReSerVation Protocol (RSVP) — Version 1 Functional Specification,” Internet Draft draft-ietf-rsvp-spec-14.txt, Work in Progress, November 1996.

[Braun 95a] Stefan Braun, “Mapping of Generic PCI Specifications for Teleconferencing Terminals to Operating Systems Environments,” Technical Report, TELES AG, 13 January 1995.

[Braun 95b] Stefan Braun (ed), “Message Exchange Mechanisms for the Realisation of Teleconferencing APIs according to prETS 300 470, Draft Version 0.5,” Technical Report, TELES AG, 29 March 1995.

[Braun 96a] Stefan Braun (ed), “GCC API Specification, Version 1.0,” Technical Report, International Multimedia Teleconferencing Consortium, API & Protocols Activity Group, 1996.

[Bullen / Bennett 90] Christine V. Bullen and John L. Bennett, “Learning From User Experience With Groupware,” in *Proceedings of the Conference on Computer Supported Cooperative Work*, pp. 291-302, The Association for Computing Machinery, Inc., Los Angeles, 1990.

[Bullinger 90] Hans-Jörg Bullinger (ed), *Handbuch des Informationsmanagements im Unternehmen*, C. H. Beck’sche Verlagsbuchhandlung, München, 1991.

[Bullinger *et al.* 93] Hans-Jörg Bullinger, Hans-Peter Fröschle, and Werner Bretteich-Teichmann, “Informations- und Kommunikationsinfrastrukturen für innovative Unternehmen,” *zfo. Zeitschrift für Führung und Organisation*, no. 4, pp. 225-234, 1993.

[Bush 45] Vannevar Bush, “As we may think,” *Atlantic Monthly*, vol. 176, no. 1, pp. 101-108, June 1945.

[Buxton 94] W. A. Buxton, “Ubiquitous Media and the Active Office,” Ontario Telepresence Project, 1994.

[CAPI 2.0] Common ISDN API Working Group, “Common ISDN API, Version 2.0,” Final Draft, Deutsche Telekom, Projekt ROLAND, January 1994.

[Casner / Deering 92] Stephen Casner and Stephen Deering, “First IETF Internet Audiocast,” *ACM SIGCOMM Computer Communications Review*, vol. 22, no. 3, July 1992.

[Ceccaldi 97] Bruno Ceccaldi (ed), “User-to-Reservation System Transactions within T.120 Conferences,” Draft for ITU-T Recommendation T.RES.1, ITU-T Study Group 8, Question 10, January 1997.

[Chamber 55] *Chamber’s Encyclopedia,* 3, Hazell Watson & Viney Ltd., 1995.

[Chang / Maxemchuk 84] J. Chang and N. F. Maxemchuk, “Reliable broadcast protocols,” *ACM Transactions on Computer Systems*, vol. 2, no. 3, p. 251−273, The Association for Computing Machinery, Inc., August 1984.

[Cheriton 86] David R. Cheriton, “VMTP: A transport protocol for the next generation of communication systems,” in *Proceedings of the SIGCOMM’86 Conference*, pp. 406-415, August 1986.

[Clarke 90] A. M. Clarke, “Is the failure of videoconferencing uptake due to a lack of human factors or poor market research?,” in *Human Factors in Telecommunications, Proceedings of the 13th International Symposium*, pp. 133-140, Turin, 1990.

[Clark / Tennenhouse 90]
D. Clark and D. Tennenhouse, “Architectural Considerations for a New Generation of Protocols,” in *Proceedings of the ACM SIGCOMM ’90 Conference*, pp. 201-208, September 1990.

[Collier 94] *Collier’s Dictionary*, Simon & Schuster, Inc., 1994.

[Cool *et al.* 92] C. Cool, R. S. Fish, R. E. Kraut, and C. M. Lowery, “Iterative Design of Video Communication Systems,” in *CSCW’92 — Sharing Perspectives: Proceedings of the Conference on Computer Supported Cooperative Work*, pp. 25-32, The Association for Computing Machinery, Inc., Toronto, 31 October - 4 November 1992.

[Crowley *et al.* 90]
Terrence Crowley, Paul Milazzo, Ellie Baker, Harry Forsdick, and Raymond Tomlinson, “MMConf: An Infrastructure for Building Shared Multimedia Applications,” in *CSCW’90: Proceedings of the Conference on Computer Supported Cooperative Work*, pp. 329-342, The Association for Computing Machinery, Inc., Los Angeles, 07-10 October 1990.

[CSCW 88] *CSCW’88: Proceedings of the Conference on Computer Supported Cooperative Work*, The Association for Computing Machinery, Inc., Portland, 26-28 September 1988.

[CSCW 90] *CSCW’90: Proceedings of the Conference on Computer Supported Cooperative Work*, The Association for Computing Machinery, Inc., Los Angeles, 07-10 October 1990.

[CSCW 92] *CSCW’92 — Sharing Perspectives: Proceedings of the Conference on Computer Supported Cooperative Work*, The Association for Computing Machinery, Inc., Toronto, 31 October - 4 November 1992.

[Darby 90] George E Darby, “Compound Documents, Multiprocessor PCs, and the Marketing of Videoconferencing,” in *PTC’90, Pacific Telecommunications Council, Weaving the Technological and Social Fabric, Proceedings of the 12th Annual Conference*, pp. 505-509, Honolulu, 14-17 January 1990.

[Deering 91] Stephen Deering, “Multicast Routing in a Datagram Internetwork,” Ph.D. Thesis, Stanford University, 1991.

[Deering *et al.* 96]
Stephen Deering, Deborah L. Estrin, Dino Farinacci, Van Jacobson, Ching-Gung Liu, and Liming Wei, “The PIM Architecture for Wide-Area Multicast Routing,” *IEEE/ACM Transactions on Networking*, vol. 4, no. 2, pp. 153-161, April 1996.

[Degener 95] Jutta Degener, “Design and Implementation of a Conference Management System for Teleconferences based on the ITU-T Recommendation T.124,” Diplomarbeit, Technische Universität Berlin, September 1995.

[DeGrasse / Lyons 96a]
Bruce DeGrasse and Terry Lyons, “Report of meeting 04-08 August 1996 in Santa Rosa, California,” ITU-T Study Group 8, Question 10, August 1996.

[DeGrasse / Lyons 96b]
Bruce DeGrasse and Terry Lyons, “Report of meeting 30 September - 04 October 1996 in Ismaning, Germany,” ITU-T Study Group 8, Question 10, October 1996.

[DeGrasse / Lyons 97]
Bruce DeGrasse and Terry Lyons, “Report of meeting 13-17 January 1997 in Irvine, California,” ITU-T Study Group 8, Question 10, January 1997.

[DeSanctis / Gallupe 85]
G. DeSanctis and R. B. Gallupe, “Group Decision Support Systems: A New Frontier,” *DATA BASE*, vol. 16, no. 2, pp. 3-10, 1985.

[Dolev / Malki 96]
Danny Dolev and Dalia Malki, “The Transis Approach to High Availability Cluster Communication,” *Communications of the ACM*, vol. 39, no. 4, pp. 64-70, April 1996.

[Dorros 69] Irwin Dorros, “Picturephone,” *Bell Laboratories Record*, pp. 136-141, May/June 1969.

[Douglas 89] Scott Douglas, “Why travel when you can call?,” *Telephony*, April 1989.

[Dourish 93] Paul Dourish, “Culture and Control in a Media Space,” in *ECSCW’93: Proceedings of the Third European Conference on Computer Supported Cooperative Work*, ed. Giorgio de Michelis, Carla Simone, and Kjeld Schmidt, pp. 125-137, Kluwer Academic Publishers, Milan, Italy, 13-17 September 1993.

[Dourish / Bellotti 92]
Paul Dourish and Victoria Bellotti, “Awareness and Coordination in Shared Workspaces,” in *CSCW’92 — Sharing Perspectives: Proceedings of the Conference on Computer Supported Cooperative Work*, pp. 107-114, The Association for Computing Machinery, Inc., Toronto, 31 October - 4 November 1992.

[Dourish / Bly 92]
Paul Dourish and Sara Bly, “Portholes: Supporting Awareness in a Distributed Work Group,” in *Proceedings of CHI’92*, pp. 541-547, The Association for Computing Machinery, Inc., 1992.

[Düwel 95] Henning Düwel, “Portierung eines standardisierten Mehrpunkt-Kommunikationsdienstes nach MS Windows,” Diplomarbeit, Technische Universität Berlin, April 1995.

[Dzwillo 91] Barnim Dzwillo, “Flexibles Scheduling von Bewegtbildübertragung im VKAPS — Analyse und Synthese,” Diplomarbeit, Technische Universität Berlin, Juli 1991.

[ECSCW 91] L. Bannon, M. Robinson, and K. Schmidt (eds), *ECSCW’91: The 2nd European Conference on Computer Supported Cooperative Work*, Amsterdam, The Netherlands, 24-27 September 1991.

[ECSCW 93] Giorgio de Michelis, Carla Simone, and Kjeld Schmidt (eds), *ECSCW'93: Proceedings of the Third European Conference on Computer Supported Cooperative Work*, Kluwer Academic Publishers, Milan, Italy, 13-17 September 1993.

[Edlich 95] Stefan Edlich, "Kooperationsmechanismen synchroner Groupwaresysteme — Entwurf, Realisierung und Einsatz eines Werkzeugsatzes für replizierte Softwarekooperation," Dissertation, Technische Universität Berlin, 1995.

[Egido 88] Carmen Egido, "Videoconferencing as a Technology to Support Group Work: A Review of its Failure," in *CSCW'88: Proceedings of the Conference on Computer Supported Cooperative Work*, pp. 13-24, The Association for Computing Machinery, Inc., Portland, 26-28 September 1988.

[Egido 90] Carmen Egido, "Teleconferencing as a Technology to Support Cooperative Work: Its Possibilities and Limitations," in *Intellectual Teamwork: Social and Technical Foundations of Cooperative Work*, ed. J. Galegher, J. Kraut, and C. Egido, pp. 351-371, Lawrence Erlbaum Associates, 1990.

[Ellis *et al.* 91] C. A. Ellis, S. J. Gibbs, and G. L. Rein, "Groupware — Some Issues and Experiences," *Communications of the ACM*, vol. 34, no. 1, pp. 38-58, January 1991.

[Ellis / Gibbs 89] C. A. Ellis and S. J. Gibbs, "Concurrency Control in Groupware Systems," in *Proceedings for the ACM SIGMOD 1989 Conference on the Management of Data*, pp. 399-406, 1989.

[Ellis / Nutt 80] Clarence A. Ellis and Gary J. Nutt, "Office Information Systems and Computer Science," *Computing Surveys*, vol. 12, no. 1, pp. 27-60, The Association for Computing Machinery, Inc., 1980.

[ETS 300101] ETSI, "International Digital Audiographic Teleconference," Draft prETS 300 101, European Telecommunication Standards Institute, September 1990.

[ETS 300383] ETSI, "File Transfer over the ISDN — EUROFILE transfer profile," Final Draft prETS 300 383, European Telecommunication Standards Institute, August 1994.

[ETS 300470] ETSI, "General Architecture for Programming Communication Interfaces," Draft prETS 300 470, European Telecommunication Standards Institute, November 1994.

[Fenner 97] W. Fenner, "Internet Group Management Protocol, Version 2," Internet Draft draft-ietf-idmr-igmp-v2-05.txt, Work in Progress, October 1996.

[Fischer / Gupta 90] Dell Fischer and Ramesh Gupta, "Taking a second look at videoconferencing," in *Data Communications International*, pp. 85-92, June 1990.

[Flores *et al.* 88] Fernando Flores, Michael Graves, Brad Hartfield, and Terry Winograd, "Computer Systems and the Design of Organizational Interaction," *ACM Transaction on Office Information Systems*, vol. 6, no. 2, pp. 153-172, The Association for Computing Machinery, Inc., April 1988.

[Floyd *et al.* 95] Sally Floyd, Van Jacobson, Steven McCanne, Ching-Gun Liu, and Lixia Zhang, *A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing*, Lawrence Berkeley Laboratories, University of California, Berkeley, August 1995.

[Fox 91] Edward A. Fox, "Advances in Interactive Digital Multimedia Systems," *IEEE Computer*, vol. 24, no. 10, pp. 9-21, October 1991.

[Freier / Marzullo 90] Alan O. Freier and Keith Marzullo, "MTP: An Atomic Multicast Transport Protocol," Technical Report TR 90-1141, Cornell University, Department of Computer Science, July 1990.

[Frivold / Lang 94] Thane Frivold and Ruth Lang, "Meeting Report of the MMUSIC Working Group," Proceedings of the Thirty First Internet Engineering Task Force, pp. 587-606, San Jose, CA, December 1994.

[Frost and Sullivan 83] Frost and Sullivan, Inc., "The Teleconferencing Market in the U.S.," Report No. 1112, Frost and Sullivan, Inc., New York, 1983.

[Funkschau 91] "Boom dank Krieg," *Funkschau*, no. 20, 1991.

[Galvin 97] Pat Galvin (ed), "T.120 Multicast Adaptation Protocol for MCS," Draft Recommendation T.MAP, ITU-T Study Group 8, Question 10, January 1997.

[Gaver 92] William W. Gaver, "The Affordances of Media Spaces for Collaboration," in *CSCW'92 — Sharing Perspectives: Proceedings of the Conference on Computer Supported Cooperative Work*, pp. 17-24, The Association for Computing Machinery, Inc., Toronto, 31 October - 4 November 1992.

[Gehrcke *et al.* 94] Hans-Christian Gehrcke, Torsten Kerschat, and Nils Seifert, "Abschlußdokumentation zur Implementierung von MTP (RFC 1301)," Telekooperationsprojekt I/II, Technische Universität Berlin, August 1994.

[Gehrcke 95] Hans-Christian Gehrcke, "Entwurf und Implementierung einer Shared Whiteboard-Applikation auf der Grundlage der ITU-T-Empfehlung T.126 (Still Image Conferencing)," Diplomarbeit, Technische Universität Berlin, Mai 1995.

[Gerfen 86] W. Gerfen, *Videokonferenz. Eine Alternative für weltweite geschäftliche Kommunikation — ein Leitfaden für Anwender*, Heidelberg, 1986.

[Greenberg 91] Saul Greenberg, "Personalizable groupware: Accommodating individual roles and group differences," in *ECSCW'91: The 2nd European Conference on Computer Supported Cooperative Work*, ed. L. Bannon, M. Robinson, and K. Schmidt, pp. 17-31, Amsterdam, The Netherlands, 24-27 September 1991.

[Green / Hansell 84] D. Green and K. J. Hansell, "Videoconferencing," *Business Horizons*, no. 6, pp. 57-61, 1984.

[Greif 88] Irene Greif (ed), *Computer-Supported Cooperative Work: A Book of Readings*, Morgan Kaufman Publishers, Inc., San Mateo, 1988.

[Grudin 88] Jonathan Grudin, "Why CSCW Applications Fail: Problems in the Design and Evaluation of Organizational Interfaces," in *CSCW'88: Proceedings of the Conference on Computer Supported Cooperative Work*, pp. 85-93, The Association for Computing Machinery, Inc., Portland, 26-28 September 1988.

[Grudin 90] Jonathan Grudin, "Groupware and Cooperative Work: Problems and Prospects," in *The Art of Human Computer Interface Design*, ed. Brenda Laurel, Apple Computer, Inc., 1990.

[Grudin 94a] Jonathan Grudin, "CSCW: History and Focus," in *IEEE Computer Magazine*, pp. 19-26, May 1994.

[Grudin 94b] Jonathan Grudin, "Eight Challenges for Developers," *Communications of the ACM*, vol. 37, no. 1, pp. 93-105, January 1994.

[Handley 96] Mark Handley, "SAP: Session Announcement Protocol," Internet Draft draft-ietf-mmusic-sap-00.txt, Work in Progress, November 1996.

[Handley *et al.* 96a] Mark Handley, Jon Crowcroft, and Carsten Bormann, "The Internet Multimedia Conferencing Architecture," Internet-Draft draft-ietf-mmusic-confarch-00.txt, Work in Progress, February 1996.

[Handley *et al.* 96b] Mark Handley, Henning Schulzrinne, and Eve M. Schooler, "SIP: Session Initiation Protocol," Internet Draft draft-ietf-mmusic-sip-01.ps, Work in Progress, December 1996.

[Handley / Jacobson 96] Mark Handley and Van Jacobson, "SDP: Session Description Protocol," Internet Draft draft-ietf-mmusic-sdp-03.txt, Work in Progress, November 1996.

[Handley / Wakeman 94] Mark Handley and Ian Wakeman, "CCCP: Conference Control Channel Protocol — A scalable base for building conference control applications," Technical Report, University College London, Department of Computer Sciences, 1994.

[Herrtwich / Hommel 89] Ralf-Guido Herrtwich and Günther Hommel, *Kooperation und Konkurrenz*, Springer Verlag, 1989.

[Hollan / Stornetta 92] Jim Hollan and Scott Stornetta, "Beyond Being There," in *Groupware and Computer-Supported Cooperative Work*, ed. Ronald M. Baecker, pp. 842-848, Morgan Kaufmann Publishers, Inc., 1992.

[Hudetz *et al.* 89] W. Hudetz et al., "Das Bildtelefon in der geschäftlichen Kommunikation — Endbericht," Studie des FhG-ISI (Abt. Telematik) für das Unternehmen PKI, 1989.

[IBM 94] The IBM Lakes Team, *Lakes — An Architecture for Collaborative Networking*, R Morgan Publishing, Chislehurst, UK, 1994.

[IEEE 802.3] "Local and metropolitan area networks — Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications," ANSI/IEEE Standard 802.3, 1996.

[IEEE 802.9] "Local and metropolitan area networks — Specific requirements — Part 9: Integrated Services (IS) LAN," ANSI/IEEE Standard 802.9, 1996.

[IMTC 96] International Multimedia Teleconferencing Consortium (IMTC), "IMTC's Event-120 Accelerates Availability of Interoperable Data Conferencing Applications," Press Release, San Ramon, CA, 24 September 1996.

[Intel 96] Intel Corporation, "New Version Of Intel Internet Phone Available Free From Intel's World Wide Web Site — First Standards-based Internet Phone Enhanced With New Capabilities," Press Release, Santa Clara, CA, 20 September 1996.

[Ishii 90] Hiroshi Ishii, "TeamWorkStation: Towards a Seamless Shared Workspace," in *CSCW'90: Proceedings of the Conference on Computer Supported Cooperative Work*, pp. 13-26, The Association for Computing Machinery, Inc., Los Angeles, 07-10 October 1990.

[Ishii *et al.* 93] Hiroshi Ishii, Kazuho Arita, and Takashi Yagi, "Beyond Videophones: TeamWorkStation-2 for Narrowband ISDN," in *Proceedings of The European Conference on Computer Supported Cooperative Work*, ed. G. De Michelis, C. Simone, and K. Schmidt, pp. 325-340, September 1993.

[Ishii / Miyake 91] Hiroshi Ishii and Naomi Miyake, "Towards an Open Shared Workspace: Computer and Video Fusion Approach of TeamWorkStation," *Communications of the ACM*, vol. 34, no. 12, The Association for Computing Machinery, Inc., December 1991.

[ISO 7498] International Organization for Standardization, "Information Processing Systems — Open Systems Interconnection — Basic Reference Model," International Standard ISO 7498, 1984.

[ISO 8571] International Organization for Standardization, "Information processing systems — Open Systems Interconnection — File Transfer, Access, and Management," International Standard ISO 8571, Part 1-4, 1988.

[ISODE 92] *The ISO Development Environment: User's Manual, Version 8.0*, ISODE Consortium, June 1992.

[ISO / IEC 92] International Organization for Standardization, "Proposal for a New Work Item: Computer-supported Voting and Polling as a Group Communication Task," contribution to ISO/IEC JTC1/SC18, December 1992.

[ITU-T F.701] ITU-T Recommendation F.701, "Teleconference service," International Telecommunication Union, Telecommunication Standardization Sector, 1993.

[ITU-T F.710] ITU-T Recommendation F.710, “General principles for audiographic conference service,” International Telecommunication Union, Telecommunication Standardization Sector, 1991.

[ITU-T F.720] ITU-T Recommendation F.720, “Videotelephony service — General,” International Telecommunication Union, Telecommunication Standardization Sector, 1993.

[ITU-T F.730] ITU-T Recommendation F.730, “Videoconference service — General,” International Telecommunication Union, Telecommunication Standardization Sector, 1993.

[ITU-T F.740] ITU-T Recommendation F.740, “Audiovisual interactive service,” International Telecommunication Union, Telecommunication Standardization Sector, 1994.

[ITU-T G.701] ITU-T Recommendation G.701, “Vocabulary of digital transmission and multiplexing, and pulse code modulation (PCM) terms,” International Telecommunication Union, Telecommunication Standardization Sector, March 1993.

[ITU-T H.221] ITU-T Recommendation H.221, “Frame structure for a 64 to 1920 kbit/s channel in audiovisual teleservices,” International Telecommunication Union, Telecommunication Standardization Sector, July 1995.

[ITU-T H.223] ITU-T Recommendation H.223, “Multiplexing protocol for low bit rate multimedia communication,” International Telecommunication Union, Telecommunication Standardization Sector, March 1996.

[ITU-T H.233] ITU-T Recommendation H.233, “Confidentiality system for audiovisual services,” International Telecommunication Union, Telecommunication Standardization Sector, July 1995.

[ITU-T H.234] ITU-T Recommendation H.234, “Encryption key management and authentication system for audiovisual services,” International Telecommunication Union, Telecommunication Standardization Sector, November 1994.

[ITU-T H.242] ITU-T Recommendation H.242, “System for establishing communication between audiovisual terminals using digital channels up to 2 Mbit/s,” International Telecommunication Union, Telecommunication Standardization Sector, March 1996.

[ITU-T H.245] ITU-T Recommendation H.245, “Control protocol for multimedia communication,” International Telecommunication Union, Telecommunication Standardization Sector, March 1996.

[ITU-T H.323] ITU-T Recommendation H.323, “Visual telephone systems and equipment for local area networks which provide a non-guaranteed quality of service,” International Telecommunication Union, Telecommunication Standardization Sector, November 1996.

[ITU-T Q.922] ITU-T Recommendation Q.922, “ISDN data link layer specification for frame mode bearer services,” International Telecommunication Union, Telecommunication Standardization Sector, February 1992.

[ITU-T Q.933] ITU-T Recommendation Q.933, “Signalling specification for frame mode basic call control,” International Telecommunication Union, Telecommunication Standardization Sector, October 1995.

[ITU-T T.120] ITU-T Recommendation T.120, “Data protocols for multimedia conferencing,” International Telecommunication Union, Telecommunication Standardization Sector, July 1996.

[ITU-T T.121] ITU-T Recommendation T.121, “Generic application template,” International Telecommunication Union, Telecommunication Standardization Sector, July 1996.

[ITU-T T.122] ITU-T Recommendation T.122, “Multipoint communication service for audiographic and audiovisual conferencing, service definition,” International Telecommunication Union, Telecommunication Standardization Sector, March 1993.

[ITU-T T.123] ITU-T Recommendation T.123, “Protocol stacks for audiographic and audiovisual teleconference applications,” International Telecommunication Union, Telecommunication Standardization Sector, November 1994.

[ITU-T T.124] ITU-T Recommendation T.124, “Generic Conference Control,” International Telecommunication Union, Telecommunication Standardization Sector, August 1995.

[ITU-T T.125] ITU-T Recommendation T.125, “Multipoint communication service, protocol specification,” International Telecommunication Union, Telecommunication Standardization Sector, April 1994.

[ITU-T T.126] ITU-T Recommendation T.126, “Multipoint still image and annotation protocol,” International Telecommunication Union, Telecommunication Standardization Sector, August 1995.

[ITU-T T.127] ITU-T Recommendation T.127, “Multipoint binary file transfer protocol,” International Telecommunication Union, Telecommunication Standardization Sector, August 1995.

[ITU-T T.130] John Boucher and Trevor Peers (eds), “Real-time Architecture for Multimedia Conferencing,” Draft for ITU-T Recommendation T.130, ITU-T Study Group 8, Question 10, October 1996.

[ITU-T T.190] ITU-T Recommendation T.190, “Cooperative Document Handling (CDH) - Framework and basic services,” International Telecommunication Union, Telecommunication Standardization Sector, August 1995.

[ITU-T T.431] ITU-T Recommendation T.431, “Document Transfer And Manipulation (DTAM) - Services and protocols,” International Telecommunication Union, Telecommunication Standardization Sector, September 1992.

[ITU-T T.434] ITU-T Recommendation T.434, “Binary file transfer format for the telematic services,” International Telecommunication Union, Telecommunication Standardization Sector, July 1996.

[ITU-T V.80] ITU-T Recommendation V.80, “In-band DCE control and synchronous data modes for asynchronous DTE,” International Telecommunication Union, Telecommunication Standardization Sector, August 1996.

[ITU-T X.200] ITU-T Recommendation X.200, “Information technology - Open Systems Interconnection - Basic reference model: The basic model,” International Telecommunication Union, Telecommunication Standardization Sector, July 1994.

[ITU-T X.208] CCITT Recommendation X.208, “Specification of Abstract Syntax Notation One (ASN.1),” in *Blue Book Fascicle VIII.4*, Comité Consultatif International Télégraphique et Téléphonique (CCITT), Geneva, 1988.

[ITU-T X.214] ITU-T Recommendation X.214, “Information technology - Open Systems Interconnection - Transport service definition,” International Telecommunication Union, Telecommunication Standardization Sector, November 1995.

[ITU-T X.215] ITU-T Recommendation X.215, “Information technology - Open Systems Interconnection - Session service definition,” International Telecommunication Union, Telecommunication Standardization Sector, November 1995.

[ITU-T X.217] ITU-T Recommendation X.217, “Information technology - Open systems interconnection - Service definition for the association control service element,” International Telecommunication Union, Telecommunication Standardization Sector, April 1995.

[ITU-T X.218] ITU-T Recommendation X.218, “Reliable transfer: model and service definition,” International Telecommunication Union, Telecommunication Standardization Sector, March 1993.

[ITU-T X.219] CCITT Recommendation X.219, “Remote operations: Model, notation and service definition,” in *Blue Book Fascicle VIII.4*, Comité Consultatif International Télégraphique et Téléphonique (CCITT), Geneva, 1988.

[ITU-T X.220] ITU-T Recommendation X.220, “Use of X.200-Series protocols in CCITT applications,” International Telecommunication Union, Telecommunication Standardization Sector, March 1993.

[ITU-T X.224] ITU-T Recommendation X.224, “Information technology - Open systems interconnection - Protocol for providing the connection-mode transport service,” International Telecommunication Union, Telecommunication Standardization Sector, November 1995.

[ITU-T X.227] ITU-T Recommendation X.227, “Information technology - Open Systems Interconnection - Connection-oriented protocol for the association control service element: Protocol specification,” International Telecommunication Union, Telecommunication Standardization Sector, April 1995.

[ITU-T X.228] CCITT Recommendation X.228, “Reliable transfer: Protocol specification,” in *Blue Book Fascicle VIII.5*, Comité Consultatif International Télégraphique et Téléphonique (CCITT), Geneva, November 1988.

[ITU-T X.229] CCITT Recommendation X.229, “Remote operations: Protocol specification,” in *Blue Book Fascicle VIII.5*, Comité Consultatif International Télégraphique et Téléphonique (CCITT), Geneva, November 1988.

[ITU-T X.400] ITU-T Recommendation X.400, “Message handling system and service overview,” International Telecommunication Union, Telecommunication Standardization Sector, July 1996.

[ITU-T X.500] ITU-T Recommendation X.500, “Information technology - Open systems Interconnection - The directory: Overview of concepts, models and services,” International Telecommunication Union, Telecommunication Standardization Sector, November 1993.

[ITU-T X.6] ITU-T Recommendation X.6, “Multicast service definition,” International Telecommunication Union, Telecommunication Standardization Sector, March 1993.

[ITU-T X.680] ITU-T Recommendation X.680, “Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation,” International Telecommunication Union, Telecommunication Standardization Sector, July 1994.

[ITU-T X.681] ITU-T Recommendation X.681, “Information technology - Abstract Syntax Notation One (ASN.1): Information object specification,” International Telecommunication Union, Telecommunication Standardization Sector, July 1994.

[ITU-T X.682] ITU-T Recommendation X.682, “Information technology - Abstract Syntax Notation One (ASN.1): Constraint specification,” International Telecommunication Union, Telecommunication Standardization Sector, July 1994.

[ITU-T X.683] ITU-T Recommendation X.683, “Information technology - Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications,” International Telecommunication Union, Telecommunication Standardization Sector, July 1994.

[ITU-T X.690] ITU-T Recommendation X.690, “Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER),” International Telecommunication Union, Telecommunication Standardization Sector, July 1994.

[ITU-T X.691] ITU-T Recommendation X.691, “Information technology - ASN.1 encoding rules - Specification of Packed Encoding Rules (PER),” International Telecommunication Union, Telecommunication Standardization Sector, April 1995.

[ITU-T X.851] ITU-T Recommendation X.851, “Information technology - Open Systems Interconnection - Service definition for the commitment, concurrency and recovery service element,” International Telecommunication Union, Telecommunication Standardization Sector, November 1993.

[ITU-T X.852] ITU-T Recommendation X.852, “Information technology - Open Systems Interconnection - Protocol for the commitment, concurrency and recovery service element: Protocol specification,” International Telecommunication Union, Telecommunication Standardization Sector, November 1993.

[ITU-T X.860] ITU-T Recommendation X.860, “Open Systems Interconnection - Distributed transaction processing: Model,” International Telecommunication Union, Telecommunication Standardization Sector, September 1992.

[ITU-T X.861] ITU-T Recommendation X.861, “Open Systems Interconnection - Distributed transaction processing: Service definition,” International Telecommunication Union, Telecommunication Standardization Sector, September 1992.

[ITU-T X.862] ITU-T Recommendation X.862, “Open Systems Interconnection - Distributed transaction processing: Protocol specification,” International Telecommunication Union, Telecommunication Standardization Sector, November 1993.

[ITU-T X.880] ITU-T Recommendation X.880, “Information technology - Remote Operations: Concepts, model and notation,” International Telecommunication Union, Telecommunication Standardization Sector, July 1994.

[ITU-T X.881] ITU-T Recommendation X.881, “Information technology - Remote Operations: OSI realizations - Remote Operations Service Element (ROSE) service definition,” International Telecommunication Union, Telecommunication Standardization Sector, July 1994.

[ITU-T X.882] ITU-T Recommendation X.882, “Information technology - Remote Operations: OSI realizations - Remote Operations Service Element (ROSE) protocol specification,” International Telecommunication Union, Telecommunication Standardization Sector, July 1994.

[Ives 30] Herbert E. Ives, “Two-Way Television,” *Bell Laboratories Record*, vol. 8, no. 9, pp. 393-404, May 1930.

[Jacobson 88] V. Jacobson, “Congestion Avoidance and Control,” in *Proceedings of the ACM SIGCOMM ’88 Conference*, The Association for Computing Machinery, Inc., August 1988.

[Jacobson / McCanne 92] Van Jacobson and Steven McCanne, “vat,” UNIX Manual Pages, part of ftp://ftp.lbl.ee.gov/sun-vat.tar.Z, Lawrence Berkeley Laboratory, Berkeley, CA, February 1992.

[Jacobson / McCanne 93] V. Jacobson and Steven McCanne, “Using the LBL Network ‘whiteboard’,” Presentation slides, part of ftp://ftp.lbl.ee.gov/wb/sun-wb.tar.Z, Lawrence Berkeley Laboratory, Berkeley, CA, February 1993.

[Jakab 93] Istvan Jakab, “Kommunikationsorientierte Untersuchung zur Wirtschaftlichkeit der Bürokommunikation,” *io Management Zeitschrift*, vol. 62, no. 4, pp. 45-50, 1993.

[Jalote 94] Pankaj Jalote, *Fault Tolerance in Distributed Systems*, Prentice Hall International, Inc., 1994.

[Johansen 88] R. Johansen, *Groupware: Computer Support for Business Teams*, The Free Press, New York, 1988.

[Johnson-Lenz / Johnson-Lenz 82]
Peter Johnson-Lenz and Trudy Johnson-Lenz, “Groupware: The Process and Impacts of Design Choices,” in *Computer-Mediated Communication Systems*, ed. Elaine B. Kerr and Star Roxanne Hiltz, Academic Press, New York, 1982.

[Johnstone 96] Jim J. Johnstone (ed), “MCS API Specification,” International Multimedia Teleconferencing Consortium, API & Protocols Activity Group, April 1996.

[Kaplan *et al.* 92] Simon M. Kaplan, William J. Tolone, Douglas P. Bogia, and Celsina Bignoli, “Flexible, Active Support for Collaborative Work with ConversationBuilder,” in *CSCW’92 — Sharing Perspectives: Proceedings of the Conference on Computer Supported Cooperative Work*, pp. 378-385, The Association for Computing Machinery, Inc., Toronto, 31 October - 4 November 1992.

[Karsenty *et al.* 93]
Alain Karsenty, Christophe Tronche, and Michel Beaudouin-Lafon, “GroupDesign: Shared Editing in a Heterogeneous Environment,” *Computing Systems*, vol. 6, no. 2, p. 167−192, Spring 1993.

[Kedzierski 82] Beverli I. Kedzierski, “Communication and Management Support in System Development Environments,” in *Proceedings of the Conference on Human Factors in Computer Systems*, The Association for Computing Machinery, Inc., Gaithersburg, MD, 1982.

[Kerkhoff 97] Matthias Kerkhoff, “Entwurf und Implementierung eines Dienstes zum Dateitransfer in Mehrpunkt-Telekonferenzen auf der Basis der ITU-T-Empfehlung T.127,” Studienarbeit, Technische Universität Berlin, 1997.

[Kerschat 94] Torsten Kerschat, “Integration eines Transportprotokolls für Mehrpunktkommunikation auf der Basis von IP-Multicast in den Kernel von Solaris 2,” Diplomarbeit, Technische Universität Berlin, September 1994.

[Kirsche *et al.* 93]
T. Kirsche, R. Lenz, H. Lührsen, K. Meyer-Wegener, H. Wedekind, M. Bever, U. Schäffer, and C. Schottmüller, “Communication support for cooperative work,” *Computer Communications*, vol. 16, no. 9, pp. 594-602, Butterworth-Heinemann Ltd., September 1993.

[Kirstein *et al.* 93]
Peter T. Kirstein, Mark J. Handley, and M. Angela Sasse, “Piloting of Multimedia Integrated Communications for European Researchers (MICE),” in *Proceedings of INET ’93*, 1993.

[Kisor 96] Gregory H. Kisor, “Suggested changes to T.120 Series Recommendations,” T120C-006, contribution to the Rapporteur’s meeting of ITU-T Study Group 8, Question 10, April 1996.

[Knight 97] Jon Knight, “Multicast Transport Protocols,” URL http://hill.lut.ac.uk/DS-Archive/MTP.html, 1997.

[Knister / Prakash 93]
Michael Knister and Atul Prakash, “Issues in the Design of a Toolkit for Supporting Multiple Group Editors,” *Computing Systems*, vol. 6, no. 2, p. 135−166, Spring 1993.

[Koifman / Zabele 96]
Alex Koifman and Stephen Zabele, “RAMP: A Reliable Adaptive Multicast Protocol,” URL http://www.tasc.com:80/simweb/papers/RAMP/ramp.htm, TASC, Inc., 1996.

[Kolrep *et al.* 90] Harald Kolrep, Mohammad Arif, Klaus Hopf, and Lothar Mühlbach, “Mehrpunkt-Videokonferenzen per Selbstwahl — Ergebnisse einer Nutzeruntersuchung,” *ntz*, vol. 43, no. 7, pp. 520-525, 1990.

[Kratz 90] Peter Kratz, “Entwurf und Implementierung von Moduln zur Realisierung von Schutzmechanismen und virtuellen Prozessorumgebungen in AX unter Berücksichtigung der Konzepte und Modelle des i80386,” Diplomarbeit, Technische Universität Berlin, 1990.

[Krause 95] Wolfgang Krause, “Untersuchungen von Gruppenkommunikationsmechanismen zur Entwicklung eines verteilten Graphikeditors,” Diplomarbeit, Rheinisch-Westfälische Technische Hochschule Aachen, Mai 1995.

[Kraut *et al.* 90] Robert E. Kraut, Robert S. Fish, Robert W. Root, and Barbara L. Chalfonte, “Informal Communication in Organisations: Form, Function, and Technology,” in *People’s Reactions to Technology in Factories, Offices and Aerospace*, ed. S. Oskamp and S. Spacepan, pp. 145-199, Sage Publications, Inc., 1990.

[Kremar 92] Helmut Kremar, “Computerunterstützung für die Gruppenarbeit — Zum Stand der Computer Supported Cooperative Work Forschung,” *Wirtschaftsinformatik*, vol. 34, no. 4, pp. 425-437, August 1992.

[Kubicek / Rolf 86]
Herbert Kubicek and Arno Rolf, *Mikropolis: Mit Computernetzen in die Informationsgesellschaft, 2. Auflage*, VSA-Verlag, Hamburg, 1986.

[Kumar 97] Vineet Kumar (ed), “Visual Telephone Systems and Equipment for Local Area Networks — Loosely Coupled Conferencing,” Draft for ITU-T Recommendation H.Loosely-coupled, February 1997.

[Lamport 78] Leslie 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.

[Lange 92] Beth Marcia Lange, “Electronic Group Calendaring: Experiences and Expectations,” in *Groupware ’92*, ed. D. Coleman, pp. 428-432, Morgan Kaufmann Publishers, Inc., 1992.

[Lautz 95] Alexander Lautz, “Videoconferencing. Theorie und Praxis für den erfolgreichen Einsatz im Unternehmen,” Dissertation, Institut für Medienentwicklung und Kommunikation GmbH, 1995.

[Lauwers *et al.* 90]
J. Chris Lauwers, Thomas A. Joseph, Keith A. Lantz, and Allyn L. Romanow, "Replicated Architectures for Shared Window Systems: A Critique," in *Groupware and Computer-Supported Cooperative Work*, ed. Ronald M. Baecker, pp. 651-662, Morgan Kaufmann Publishers, Inc., 1990.

[Lauwers / Lantz 90]
J. Chris Lauwers and Keith A. Lantz, "Collaboration Awareness in Support of Collaboration Transparency: Requirements for the Next Generation of Shared Window Systems," in *Groupware and Computer-Supported Cooperative Work*, ed. Ronald M. Baecker, pp. 663-671, Morgan Kaufmann Publishers, Inc., 1990.

[Leffler *et al.* 89] Samuel J. Leffler, Marshall Kirk McKusick, Michael J. Karels, and John S. Quaterman, *The Design and Implementation of the 4.3BSD UNIX Operating System*, Addison-Wesley Publishing Company, Inc., 1989.

[Leland *et al.* 88] Mary D. P. Leland, Robert S. Fish, and Robert E. Kraut, "Collaborative Document Production using Quilt," in *CSCW'88: Proceedings of the Conference on Computer Supported Cooperative Work*, The Association for Computing Machinery, Inc., Portland, September 1988.

[Lineback 82] J. R. Lineback, "Video Conferencing Growing Up," *Electronics*, pp. 105-106, September 1982.

[Lin / Paul 95] John C. Lin and Sanjoy Paul, "RMTP: A Reliable Multicast Transport Protocol," in *Proceeding of IEEE INFOCOM '96*, pp. 1414-1424, 1996.

[Linton *et al.* 91] M. A. Linton, P. R. Calder, J. A. Interrante, S. Tang, and J. M. Vlissides, *Interviews Reference Manual*, Stanford University, 1991.

[Lubich 89] Hanns Peter Lubich, "MultimETH: Ein Beitrag zur Konzeption eines Echtzeit-Multimedia-Konferenzsystems," Dissertation, Eidgenössische Technische Hochschule (ETH) Zürich, 1989.

[Lynne Markus / Connolly 90]
M. Lynne Markus and Terry Connolly, "Why CSCW Applications Fail: Problems in the Adoption of Interdependent Work Tools," in *CSCW'90: Proceedings of the Conference on Computer Supported Cooperative Work*, pp. 371-380, The Association for Computing Machinery, Inc., Los Angeles, 07-10 October 1990.

[Maciejewski 91] Paul G. Maciejewski, "Keine Kompromisse," *net*, vol. 45, no. 6, pp. 247-249, 1991.

[Maeno *et al.* 91] Kazutoshi Maeno, Shiro Sakata, Toyoko Ohmori, Kazuo Watabe, and Hideyuki Fukuoka, "Distributed Desktop Conferencing System (MERMAID) Based on Group Communication Architecture," *IEICE Transactions*, vol. E 74, no. 9, September 1991.

[Malm 94] Pål S. Malm, "The unOfficial Yellow Pages of CSCW. Groupware, Prototypes, and Projects," Frequently Asked Question of the Usenet Newsgroup ''comp.groupware''.

[Malone *et al.* 89]
Thomas W. Malone, Kenneth R. Grant, Kum-Yew Lai, Ramana Rao, and David A. Rosenblitt, "The Information Lens: An Intelligent System for Information Sharing and Coordination," in *Technological Support for Work Group Collaboration*, ed. M. H. Olson, pp. 65-88, Lawrence Erlbaum Associates, Hillsdale, NJ, 1989.

[Mantei *et al.* 91] Marilyn M. Mantei, Ronald M. Baecker, Abigail J. Sellen, William A. S. Buxton, and Thomas Milligan, "Experiences in the Use of a Media Space," in *Groupware and Computer-Supported Cooperative Work*, ed. Ronald M. Baecker, pp. 803-808, Morgan Kaufmann Publishers, Inc., 1991.

[Marmolin / Sundblad 91]
Hand Marmolin and Yngve Sundblad, "An analysis of Design and Collaboration in a Distributed Environment," in *ECSCW'91: The 2nd European Conference on Computer Supported Cooperative Work*, ed. L. Bannon, M. Robinson, and K. Schmidt, pp. 147-161, Amsterdam, The Netherlands, 24-27 September 1991.

[Martin *et al.* 94] James Martin, Kathleen Kavanagh Chapman, and Joe Leben, *Local Area Networks — Architecture and Implementations, 2nd Edition*, Prentice Hall International, Inc., New Jersey, 1994.

[Mauthe *et al.* 95]
A. Mauthe, G. Coulson, D. Hutchison, and S. Namuye, "Group Support in Multimedia Communications Systems," in *Teleservices and Multimedia Communications*, ed. D. Hutchison, H. Christiansen, C. Coulson, and A. Danthine, pp. 1-18, November 1995.

[McCanne / Jacobson 95]
Van Jacobson and Steven McCanne, "vic: A flexible framework for packet video," Proceedings of ACM Multimedia'95, Berkeley, CA, November 1995.

[Medina-Mora *et al.* 92]
Raúl Medina-Mora, Terry Winograd, Rodrigo Flores, and Fernando Flores, "The Action Workflow Approach to Workflow Management Technology," in *CSCW'92: Proceedings of the Conference on Computer Supported Cooperative Work*, The Association for Computing Machinery, Inc., Toronto, 1992.

[Mee 98] Arthur Mee, "The Pleasure Telephone," *The Strand Magazine*, pp. 339-369, 1898.

[Melcher 96] Arne Melcher, "Erweiterung eines Transportdienstes für Mehrpunktkommunikation in Telekonferenzen auf der Basis der ITU-T-Empfehlungen T.122/T.125," Diplomarbeit, Technische Universität Berlin, April 1996.

[MERCI 95] "MERCI — Multimedia European Research Conferencing Integration," Project Plan of the MERCI Project in the Telematics Programme of the European Union, 1995.

[Mühlbach *et al.* 89]
Lothar Mühlbach, Mohammad Arif, Klaus Hopf, and Götz Romahn, "Mehrpunkt-Telekonferenzen — Nutzeruntersuchungen mit verschiedenen Varianten," *ntz*, vol. 42, no. 1, pp. 8-12, 1989.

[Microsoft 96a] Microsoft Corporation, *Microsoft NetMeeting 2.0 Resource Kit*, 1996.

[Microsoft 96b] Microsoft Corporation, "Microsoft Delivers First Beta Release of NetMeeting 2.0; Integrated Support of ITU H.323 Standard Paves the Way For Rich Communication on the Internet," Press Release, Redmont, WA, 14 October 1996.

[Miller *et al.* 97] K. Miller, K. Robertson, A. Tweedly, and M. White, "StarBurst Multicast File Transfer Protocol (MFTP) Specification," Internet Draft draft-miller-mftp-spec-02.txt, Work in Progress, January 1997.

[Modarressi 91] Farhad Modarressi, "Entwurf und Realisierung der Konferenzfunktionalität des KBS-Vision Projektes," Diplomarbeit, Technische Universität Berlin, Januar 1991.

[Molnar 69] Julius P. Molnar, "Picturephone Service — A New Way of Communicating," *Bell Laboratories Record*, pp. 134-135, May/June 1969.

[Moser *et al.* 96] L. E. Moser, P. M. Melliar-Smith, D. A. Agarwal, R. K. Budhia, and C. A. Lingley-Papadopoulos, "Totem: A Fault-Tolerant Multicast Group Communication System," *Communications of the ACM*, vol. 39, no. 4, pp. 54-63, April 1996.

[Moulton 94] James Moulton (ed), "Application of the Multicast Taxonomy to the Proposed Draft Amendment to X.214 (OSI Transport Service Definition) for the Inclusion of Multicast Services," Contribution to ITU-T Study Group 7, Geneva, 7-18 February 1994.

[Mullender 89] Sape Mullender (ed), *Distributed Systems*, ACM Press, New York, 1989.

[Mullender 93] Sape Mullender (ed), *Distributed Systems, 2nd Edition*, ACM Press, New York, 1993.

[Nakajima 93] Amane Nakajima, "Telepointing issues in desktop conferencing systems," *Computer Communications*, vol. 16, no. 9, pp. 603-610, September 1993.

[Narayanaswamy / Goldman 92]
K. Narayanaswamy and Neil Goldman, "'Lazy' Consistency: A Basis for Cooperative Software Development," in *CSCW'92 — Sharing Perspectives: Proceedings of the Conference on Computer Supported Cooperative Work*, pp. 257-264, The Association for Computing Machinery, Inc., Toronto, 31 October - 4 November 1992.

[Neuwirth *et al.* 90]
Christine M. Neuwirth, David S. Kaufer, Ravinder Chandhok, and James J. Morris, "Issues in the Design of Computer Support for Co-authoring and Commenting," in *Readings in Groupware and Computer Supported Cooperative Work assisting human-human collaboration*, ed. Ronald M. Baecker, pp. 537-549, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1993.

[Nikolaus 95a] Boris Nikolaus, “Entwurf und Implementierung eines Adaptionsprotokolls zur Integration von MTP-2 in den ITU-T Multipoint Communication Service,” Studienarbeit, Technische Universität Berlin, Januar 1995.

[Nikolaus 95b] Boris Nikolaus, “Weiterentwicklung und Optimierung des Adaptionsprotokolls MMAP zum effizienten Einsatz des ITU-T Multipoint Communication Service in multicastfähigen IP-Netzen,” Diplomarbeit, Technische Universität Berlin, November 1995.

[Ott 91a] Jörg Ott, “Einsatz der X-Protokolle bei der Realisierung von Mechanismen der verteilten Gruppenarbeit,” Studienarbeit, Technische Universität Berlin, Januar 1991.

[Ott 91b] Jörg Ott, “Der Synchronisationsdienst im IVICO-System auf der Basis des OSI-Referenzmodells,” Diplomarbeit, Technische Universität Berlin, September 1991.

[Ott 92a] Jörg Ott, “The Required Synchronization Services within MIMIS — A Basis For Further Discussion,” R2025/WP4/TELES/002/A, contribution to the MIMIS Project (R2025) of the European Union, 15 April 1992.

[Ott 92b] Jörg Ott, “Architecture of the prETS 300 101,” R2025/WP7/TELES/005/A, contribution to the MIMIS Project (R2025) of the European Union, 28 April 1992.

[Ott 92c] Jörg Ott, “Concept for the Multipoint Service,” R2025/WP7/TELES/006/A, contribution to the MIMIS Project (R2025) of the European Union, 02 May 1992.

[Ott 92d] Jörg Ott, “Integration of a Multipoint Communication Architecture into the OSI Reference Model,” R2025/WP4/TELES/004/A, contribution to the MIMIS Project (R2025) of the European Union, 12 June 1992.

[Ott 92e] Jörg Ott, “Draft Specification for the Integration of Protocol Stacks,” R2025/WP4/TELES/008/A1, internal report of the MIMIS Project (R2025) of the European Union, 28 September 1992.

[Ott 92f] Jörg Ott, “Draft Specification of Synchronization Mechanisms,” R2025/WP4/TELES/007/A, internal report of the MIMIS Project (R2025) of the European Union, 29 September 1992.

[Ott 93] Jörg Ott (ed), “Specification of the Multipoint Services for MM2,” R2025/WP7/TELES/026/B, internal report of the MIMIS Project (R2025) of the European Union, 10 March 1993.

[Ott *et al.* 94] Jörg Ott, Carsten Bormann, and Ute Bormann, “Service Interoperability in Teleconferences,” published as Common Functional Specification L313, RACE Industrial Consortium, February 1994.

[Ott 95a] Jörg Ott, “Einsatz von Telekooperationstechnik im Unternehmen — technisch-wirtschaftliche Rahmenbedingungen,” Diplomarbeit, Technische Fachhochschule Berlin, März 1995.

[Ott 95b] Jörg Ott, “Implementation Guidelines for the MCS GPCI Library,” Internal Report, Technische Universität Berlin, 25 June 1995.

[Ott 95c] Jörg Ott, “MCS API Draft Version 0.4a,” Working document of the European ISDN User Forum (EIUF), 10 August 1995.

[Ott 95d] Jörg Ott, “Transmission of T.120 information in LAN environments,” AVC-797, contribution to the Rapporteur’s meeting of ITU-T Study Group 15, Questions 2 and 3, 15-18 May 1995.

[Ott 95e] Jörg Ott, “Usability of Today’s LAN Multicast Environments for ITU-T Teleconferencing,” AVC-831, contribution to the Rapporteur’s meeting of ITU-T Study Group 15, Questions 2 and 3, 24-27 October 1995.

[Ott 96a] Jörg Ott, “Comparison: EUROFILE vs. Multipoint Binary File Transfer,” Report for the European Union, TELES AG, 1996.

[Ott 96b] Jörg Ott, “Considerations on the Extension of the T.120 Series towards using the Multicast Capabilities of Networks,” T120C-42, contribution to the Rapporteur’s meeting of ITU-T Study Group 8, Question 10, 05-09 August 1996.

[Ott 96c] Jörg Ott, “Requirements for a multicast transport to be applicable as a platform for MCS,” T120C-79, contribution to the Rapporteur’s meeting of ITU-T Study Group 8, Question 10, 30 September - 04 October 1996.

[Ott / Bormann 94] Jörg Ott and Carsten Bormann, “Multicasting the ITU MCS: Integrating Point-to-Point and Multicast Transport,” in *Proceedings of the 1994 International Conference on Communication Systems*, Singapore, 1994.

[Oxford 89] *The Oxford English Dictionary, 2nd Edition*, 3, Clarendon Press, Oxford, 1989.

[Pagani / Mackay 93] Daniel S. Pagani and Wendy E. Mackay, “Bringing Media Spaces into the Real World,” in *Proceedings of The European Conference on Computer Supported Cooperative Work*, ed. G. De Michelis, C. Simone, and K. Schmidt, pp. 341-356, September 1993.

[Pankoke-Babatz / Prinz 89] Uta Pankoke-Babatz and Wolfgang Prinz, “Support for Coordinating Group Activities,” in *Message Handling Systems and Distributed Applications*, ed. E. Stefferund, O.-J. Jacobsen, and P. Schicker, pp. 345-358, Elsevier Science Publishers B. V., North Holland, 1989.

[Paszkowsky 94] Ingo Paszkowsky, “Der Computer wird zur Video-Plattform,” *Tagesspiegel*, Berlin, 18. Oktober 1994.

[Patel / Kalter 93a] Dorab Patel and Scott D. Kalter, “Low overhead, loosely coupled communication channels in collaboration,” in *ECSCW’93: Proceedings of the Third European Conference on Computer Supported Cooperative Work*, ed. Giorgio de Michelis, Carla Simone, and Kjeld Schmidt, pp. 203-218, Kluwer Academic Publishers, Milan, Italy, 13-17 September 1993.

[Patel / Kalter 93b]
Dorab Patel and Scott Kalter, “A UNIX Toolkit for Distributed Synchronous Collaborative Applications,” *Computing Systems*, vol. 6, no. 2, pp. 105-133, Spring 1993.

[Patterson *et al.* 90]
John F. Patterson, Ralph D. Hill, Steven L. Rohall, and W. Scott Meeks, “Rendezvous: An Architecture for Synchronous Multiuser Applications,” in *CSCW’90: Proceedings of the Conference on Computer Supported Cooperative Work*, pp. 317-828, The Association for Computing Machinery, Inc., Los Angeles, 07-10 October 1990.

[Peng 93] Chengzhi Peng, “Survey on Collaborative Drawing Support Tools,” in *Computer Supported Cooperative Work (CSCW)*, vol. 1, pp. 197-228, Kluwer Academic Publishers, The Netherlands, 1993.

[Petrovic 92] Otto Petrovic, “Groupware — Systemkategorien, Anwendungsbeispiele, Problemfelder und Entwicklungsstand,” *Information Management*, no. 1, pp. 16-22, 1992.

[Pulito 97] Brian Pulito (ed), “Generic Conference Control, Revision 2,” Draft for ITU-T Recommendation T.124, ITU-T Study Group 8, Question 10, January 1997.

[Pusateri 96] T. Pusateri, “Distance Vector Multicast Routing Protocol, Version 3,” Internet Draft draft-ietf-idmr-dvmrp-v3-03.txt, Work in Progress, September 1996.

[Radig 96] Stefan Radig, “Concept for the Implementation of the GCC Provider,” Internal Project Documentation, TELES AG, April 1996.

[Reid 97] Mark Reid, “Interworking of H.series multimedia terminals (Draft H24i),” AVC-1133, contribution to the Rapporteur’s meeting of ITU-T Study Group 15, Questions 2 and 3, 18-21 February 1997.

[Reiter 96] Michael K. Reiter, “Distributing Trust with the Rampart Toolkit,” *Communications of the ACM*, vol. 39, no. 4, pp. 71-74, April 1996.

[Resnick 93] Paul Resnick, “Phone-Based CSCW: Tools and Trials,” *ACM Transactions on Information Systems*, vol. 11, no. 4, pp. 401-424, October 1993.

[RFC 0793] J. Postel, “Transmission Control Protocol,” Internet RFC 0793, September 1981.

[RFC 0821] J. B. Postel, “Simple Mail Transfer Protocol,” Internet RFC 0821, August 1982.

[RFC 0822] D. Crocker, “Standard for the format of ARPA Internet text messages,” Internet RFC 0822, August 1982.

[RFC 0896] J. Nagke, “Congestion control in IP/TCP internetworks,” Internet RFC 0896, January 1984.

[RFC 0959] J. Postel and J. K. Reynold, “File Transfer Protocol,” Internet RFC 0959, October 1985.

[RFC 1006] Marshall T. Rose and Dwight E. Cass, “ISO Transport Service on top of the TCP, Version 3,” Internet RFC 1006, May 1987.

[RFC 1036] M. R. Horton and R. Adams, “Standard for interchange of USENET messages,” Internet RFC 1036, December 1987.

[RFC 1045] D. R. Cheriton, “VMTP: Versatile Message Transaction Protocol: Protocol specification,” Internet RFC 1045, February 1988.

[RFC 1057] Sun Microsystems, “RPC: Remote Procedure Call Protocol specification: Version 2,” Internet RFC 1057, June 1988.

[RFC 1075] D. Waitzman, C. Partridge, and S. Deering, “Distance Vector Multicast Routing Protocol,” Internet RFC 1075, November 1988.

[RFC 1112] S. Deering, “Host Extensions for IP Multicasting,” Internet RFC 1112, August 1989.

[RFC 1235] J. Ioannidis, Jr. and G. Q. Maguire, “Coherent File Distribution Protocol,” Internet RFC 1235, June 1991.

[RFC 1301] S. Armstrong, A. Freier, and K. Marzullo, “Multicast Transport Protocol,” Internet RFC 1301, February 1992.

[RFC 1421] J. Linn, “Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures,” Internet RFC 1421, February 1993.

[RFC 1521] N. Borenstein and N. Freed, “MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies,” Internet RFC 1521, September 1993.

[RFC 1522] K. Moore, “MIME (Multipurpose Internet Mail Extensions) Part Two: Message Header Extensions for Non-ASCII Text,” Internet RFC 1522, September 1993.

[RFC 1584] J. Moy, “Multicast Extensions to OSPF,” Internet RFC 1584, March 1994.

[RFC 1633] R. Braden, D. Clark, and S. Shenker, “Integrated Services in the Internet Architecture: an Overview,” Internet RFC 1633, June 1994.

[RFC 1736] J. Kunze, “Functional Recommendations for Internet Resource Locators,” Internet RFC 1736, February 1995.

[RFC 1737] K. Sollins and L. Masinter, “Functional Requirements for Uniform Resource Names,” Internet RFC 1737, December 1994.

[RFC 1738] T. Berners-Lee, L. Masinter, and M. McCahill, “Uniform Resource Locators (URL),” Internet RFC 1738, December 1994.

[RFC 1825] R. Atkinson, “Security Architecture for the Internet Protocol,” Internet RFC 1825, August 1995.

[RFC 1826] R. Atkinson, “IP Authentication Header,” Internet RFC 1826, August 1995.

[RFC 1827] R. Atkinson, “IP Encapsulating Security Payload (ESP),” Internet RFC 1827, August 1995.

[RFC 1831] R. Srinivasan, “RPC: Remote Procedure Call Protocol Specification Version 2,” Internet RFC 1831, August 1995.

[RFC 1889] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications,” Internet RFC 1889, January 1996.

[RFC 1890] H. Schulzrinne, “RTP Profile for Audio and Video Conferences with Minimal Control,” Internet RFC 1890, January 1996.

[RFC 1911] G. Vaudreuil, “Voice Profile for Internet Mail,” Internet RFC 1911, February 1996.

[Rügenheimer 96] Hannes Rügenheimer, “ISDN-Test — Von Angesicht zu Angesicht,” *connect*, pp. 94-98, October 1996.

[Rodden *et al.* 92] T. Rodden, J. A. Marini, and G. Blair, “Supporting Cooperative Applications,” *Computer Supported Cooperative Work (CSCW)*, vol. 1, no. 1/2, pp. 41-67, Kluwer Academic Publishers, The Netherlands, 1992.

[Rodden / Blair 91] Tom Rodden and Gordon Blair, “CSCW and Distributed Systems: The Problem of Control,” in *ECSCW’91: The 2nd European Conference on Computer Supported Cooperative Work*, ed. L. Bannon, M. Robinson, and K. Schmidt, pp. 49-61, Amsterdam, The Netherlands, 24-27 September 1991.

[Romano 97] Pat Romano (ed), “Application Sharing,” Draft for ITU-T Recommendation T.SHARE, ITU-T Study Group 8, Question 10, January 1997.

[Root 88] Robert W. Root, “Design of a multimedia vehicle for social browsing,” in *Proceedings of the 1988 Conference on Computer Supported Cooperative Work*, pp. 25-38, ACM Press, New York, 1988.

[Roseman / Greenberg 92] Mark Roseman and Saul Greenberg, “GroupKit — A Groupware Toolkit for Building Real-time Conferencing Applications,” in *CSCW’92 Proceedings*, pp. 43-50, The Association for Computing Machinery, Inc., Toronto, 31 October - 4 November 1992.

[Sarin / Greif 85] S. Sarin and I. Greif, “Computer-based real-time conferencing systems,” *IEEE Computer*, vol. 18, no. 10, pp. 33-45, 1985.

[Sathi *et al.* 86] Arvind Sathi, Thomas E. Morton, and Steven F. Roth, “Callisto: An Intelligent Project Management System,” *AI Magazine*, pp. 34-52, American Association for Artificial Intelligence, 1986.

[Schatz 92] Bruce R. Schatz, “Building an Electronic Community System,” *Journal of Management Information Systems*, vol. 8, no. 3, pp. 87-107, M. E. Sharpo, Inc., 1992.

[Schindler 90] Sigram Schindler, *Open Security Technology and Electronic Data Interchange*, TELES GmbH / Technische Universität Berlin, Berlin, 1990.

[Schindler 91] Sigram Schindler, *Das TELES.VISION-System — Philosophie und Technologie*, TELES GmbH, 1991.

[Schindler 92a] Sigram Schindler, *Videoconferencing and TELES.VISION — Concepts and Contexts*, TELES GmbH / Technische Universität Berlin, 1992.

[Schindler *et al.* 93]
Sigram Schindler, Stefan Edlich, and Jörg Ott, "Videokonferenzen und Groupware," in *Groupwareeinsatz in Organisationen*, pp. 7-24, Verlag GMD mbH, Oktober 1993.

[Schindler *et al.* 94]
Sigram Schindler, Stefan Braun, Barnim Dzwillo, Peter Kratz, and Jörg Ott, *Desktop Videoconferencing with EURO.VISION*, TELES GmbH, 1994.

[Schindler 94a] Sigram Schindler, *Im Fokus: Desktop-Multimediakonferenz-Systeme*, TELES GmbH / Technische Universität Berlin, 1994.

[Schindler 94b] Sigram Schindler, *Technische Basis-Konzepte verteilter Büro-/Verwaltungskooperation — Checkliste für Desktop Videokonferenzsysteme*, TELES GmbH / Technische Universität Berlin, 1994.

[Schindler *et al.* 95]
Sigram Schindler, Stefan Braun, Barnim Dzwillo, Peter Kratz, and Jörg Ott, "Software Architecture for Multimedia Conference Terminals based on prETS 300 740," in *Broadband Islands Fourth International Conference*, Elsevier, 1995.

[Schindler / Heidebrecht 89]
Sigram Schindler and Carsten Heidebrecht, *Future IBC-Based Teleconference Workstations*, TELES GmbH / Technische Universität Berlin, 1989.

[Schindler / Heidebrecht 90]
Sigram Schindler and Carsten Heidebrecht, "Broadband Technology within the DIDAMES Project (RACE 1060)," in *Proceedings of the International Conference on Integrated Broadband Services and Networks*, pp. 148-153, London, 15-18 October 1990.

[Schindler / Ott 97]
Sigram Schindler and Jörg Ott, "Definition of 'Applications Sharing'," in *Lexikon Informatik und Kommunikationstechnik, 2. Aufl.*, Springer Verlag, in press.

[Schmidt 94] Andreas Schmidt, "Analyse, Erweiterung und Implementierung eines Transportdienstes für Mehrpunktkommunikation in Telekonferenzen auf der Basis der ITU-T-Empfehlung T.122," Diplomarbeit, Technische Universität Berlin, November 1994.

[Schmidt / Bannon 92]
Kjeld Schmidt and Liam Bannon, "Taking CSCW Seriously," *Computer Supported Cooperative Work (CSCW)*, vol. 1, no. 1/2, pp. 7-40, Kluwer Academic Publishers, The Netherlands, 1992.

[Schmitt 94] Hans-Jochen Schmitt, "Gast-Wirt-schaft. Fünf Windows-Fernsteuerungen im Vergleich," *c't*, no. 9, pp. 196-206, 1994.

[Schneier 96] Bruce Schneier, *Applied Cryptography*, John Wiley and Sons, Inc., 1996.

[Schooler 91] Eve M. Schooler, “A Distributed Architecture for Multimedia Conference Control,” Research Report ISI/RR-91-289, University of Southern California, Information Sciences Institute, November 1991.

[Schooler 93] Eve Schooler, “Multiparty Multimedia Session Control (MMUSIC) Working Group Meeting Report,” in *Proceedings of the Twenty-Seventh Internet Engineering Task Force*, pp. 419-430, Amsterdam, 1993.

[Schulte 93] Regine Schulte, *Substitut oder Komplement — die Wirkungsbeziehungen zwischen der Telekommunikationstechnik Videokonferenz und dem Luftverkehrsaufkommen deutscher Unternehmen,* Bonner Geographische Abhandlungen, Ferd. Dümmlers Verlag, Bonn, 1993.

[Schumann 96] Manfred Schumann, “Cooperative Document Handling in the Multipoint Environment,” Delayed Document D364/G, ITU-T Study Group 8, Questions 8, 10, and 15, February 1996.

[Seifert 94] Nils Seifert, “Analyse, Erweiterung und Implementierung eines Transportprotokolls für Mehrpunktkommunikation in Telekonferenzen auf der Basis von IP-Multicast,” Diplomarbeit, Technische Universität Berlin, September 1994.

[Seifert 97] Nils Seifert, “Definition, Entwicklung und Analyse eines Multicast-Transportprotokolls auf der Basis von IP-Multicast,” Dissertation, Technische Universität Berlin, In preparation.

[Shenker *et al.* 95] Scott Shenker, Abel Weinrib, and Eve M. Schooler, “Managing Shared Ephemeral Teleconferencing State: Policy and Mechanism,” Internet Draft draft-mmusic-agree-00.ps, Work in Progress, 1995.

[Short *et al.* 76] John Short, Ederyn Williams, and Bruce Christie, “Visual Communication and Social Interaction,” in *The Social Psychology of Telecommunications*, pp. 43-60, John Wiley & Sons, 1976.

[Simm 95] Jörg Simm, “Implementierung eines Vermittlungsschichtprotokolls auf der Basis der ITU-T-Empfehlung T.123 unter STREAMS,” Diplomarbeit, Technische Universität Berlin, November 1995.

[Snyder 71] F. W. Snyder, “Travel Patterns: Implications for New Communication Facilities,” Bell Laboratories Memorandum, 1971.

[Stefik *et al.* 87a] M. Stefik, D. G. Bobrow, G. Foster, S. Lanning, and D. Tatar, “WYSIWIS Revised: Early Experiences with Multiuser Interfaces,” in *Groupware and Computer-Supported Cooperative Work*, ed. Ronald M. Baecker, pp. 585-595, Morgan Kaufmann Publishers, Inc., 1987.

[Stefik *et al.* 87b] Mark Stefik, Gregg Foster, Daniel G. Bobrow, Kenneth Kahn, Stan Lanning, and Lucy Suchman, “Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings,” *Communications of the ACM*, vol. 30, no. 1, pp. 32-47, The Association for Computing Machinery, Inc., 1987.

[Steinle 94] Claus Steinle, “Das Büro als Lean Office — Komplexes Management für eine schlanke Büroproduktionswelt,” *zfo. Zeitschrift für Führung und Organisation*, no. 2, pp. 78-85, 1994.

[Straßburger 90] Franz Xaver Straßburger, “ISDN — Chancen und Risiken eines integrierten Telekommunikationskonzeptes aus betriebswirtschaftlicher Sicht,” Dissertation, Verlag V. Florentz GmbH, 1990.

[STREAMS 90] Sun Microsystems, *STREAMS Programming*, Sun Microsystems, Inc., 1990.

[Swineheart 91] D. C. Swineheart, “The Connection Architecture of the Etherphone System,” Technical Report CSL 91-8, Xerox Palo Alto Research Center, August 1991.

[Tagyos 85] P. R. de Tagyos, “Teleconferencing — the human side of office automation,” in *Teleconferencing*, ed. T. B. Cross, pp. 61-66, Berkshire, 1985.

[Tanaka 96] Toshiaki Tanaka, “Requirements of T.M-RPC for multipoint document communications,” Temporary Document 0087, ITU-T Study Group 8, Questions 8, 10, and 15, February 1996.

[Tanenbaum 81] Andrew S. Tanenbaum, *Computer Networks*, Prentice Hall International, Inc., 1981.

[Tanenbaum 89] Andrew S. Tanenbaum, *Computer Networks, 2nd Edition*, Prentice Hall International, Inc., 1989.

[Tang / Isaacs 93] John C. Tang and Ellen Isaacs, “Why Do Users Like Video? Studies of Multimedia-Supported Collaboration,” in *ECSCW’93: Proceedings of the Third European Conference on Computer Supported Cooperative Work*, ed. Giorgio de Michelis, Carla Simone, and Kjeld Schmidt, pp. 163-196, Kluwer Academic Publishers, Milan, Italy, 13-17 September 1993.

[TELES 95] *TELES.VISION-Handbuch, Version 2/1995*, TELES GmbH, 1995.

[TELES 96] *TELES.DMC SDK Specification, Version 0.5*, TELES AG, January 1996.

[TLI 88] *X/Open Specification: Networking Services, Issue 3*, 7, X/Open Company, Ltd., August 1988.

[Toga 97] James Toga (Ed), “Security and Encryption for H.series (H.323 and other H.245 based) multimedia terminals,” AVC-1124, Draft for ITU-T Recommendation H.Secure, ITU-T Study Group 16, 18-21 February 1997.

[Tonnemacher 88] Jan Tonnemacher, “Zur Akzeptanz von Videokonferenzen in Dienstbesprechungen der DBP,” *ntz*, vol. 41, no. 7, pp. 396-399, 1988.

[Trevor *et al.* 93] Jonathan Trevor, Tom Rodden, and Gordon Blair, “COLA: A Lightweight Platform for CSCW,” in *ECSCW’93: Proceedings of the Third European Conference on Computer Supported Cooperative Work*, ed. Giorgio de Michelis, Carla Simone, and Kjeld Schmidt, Kluwer Academic Publishers, Milan, Italy, 13-17 September 1993.

[Turletti 93] T. Turletti, “H.261 software codec for videoconferencing over the Internet,” Research Report No. 1834, INRIA, Sophia Antipolis, France, 1993.

[van Renesse *et al.* 96]
Robbert van Renesse, Kenneth P. Birman, and Silvano Maffeis, "Horus: A Flexible Group Communication System," *Communications of the ACM*, vol. 39, no. 4, pp. 76-83, April 1996.

[Völz 93] Hartmut Völz, "Fernbedienung von MS-Windows-Anwendungssystemen unter OS/2 — grafische Einbettung in ein PC-integriertes Videokonferenzsystem," Diplomarbeit, Technische Universität Berlin, 1993.

[Vogel *et al.* 88] D. Vogel, J. Nunamaker, J. George, and A. Dennis, "Group Decision Support Systems: Evolution and Status at the University of Arizona," in *Organizational Decision Support Systems, Proceedings of the IFIP Conference*, ed. R. M. Lee, A. M. McCosh, and Migliarese, pp. 287-303, North Holland, 1988.

[Wang *et al.* 91] Xingwei Wang, "GRPC: A Communication Cooperation Mechanism in Distributed Systems," Research Report, Northeast University of Technology, Shenyang, P. R. China, 1991.

[Watabe *et al.* 90] Kazuo Watabe, Shiro Sakata, Kazutoshi Maeno, Hideyuki Fukuoka, and Toyoko Ohmori, "Distributed Multiparty Desktop Conferencing System: MERMAID," in *CSCW'90: Proceedings of the Conference on Computer Supported Cooperative Work*, pp. 27-38, The Association for Computing Machinery, Inc., Los Angeles, 07-10 October 1990.

[Webster 86] *Webster's Third New International Dictionary,* I (A-G), Merriam Webster, Inc., 1986.

[Weinrib 94] Abel Weinrib, "Meeting Report of the MMUSIC Working Group," Proceedings of the Thirtieth Internet Engineering Task Force, Toronto, Canada, July 1994.

[Wenger 95] Stephan Wenger, "ISDN-basierte Desktop-Multimedia-Konferenzsysteme," Dissertation, Technische Universität Berlin, 1995.

[Whetten *et al.* 94]
Brian Whetten, Todd Montgomery, and Simon Kaplan, "A High Performance Totally Ordered Multicast Protocol," in *Theory and Practice in Distributed Systems*, ed. K. P. Birman, F. Mattern, and A. Schiper, Springer Verlag, September 1994.

[Wilde 92] Michael Wilde, "PC ferngesteuert," *c't*, no. 7, pp. 96-109, 1992.

[Winsock 96] "Winsock API Specification," Current information available in the Microsoft Developer's Network, 1996.

[Woollett 97] Andrew Woollett (ed), "Remote Device Control," Draft for ITU-T Recommendation T.RDC, ITU-T Study Group 8, Question 10, January 1997.

[XTI 92] *X/Open Specification: Networking Services, Issue 3,* X/Open Company, Ltd., February 1992.

[Yager 93] Tom Yager, "Better than being there," *BYTE*, p. 129, March 93.

[Yakemovic / Conklin 90] K. C. Burgess Yakemovic and E. Jeffrey Conklin, “Report on a Development Project Use of an Issue-Based Information System,” in *CSCW’90: Proceedings of the Conference on Computer Supported Cooperative Work*, ed. F. Halasz, pp. 105-118, The Association for Computing Machinery, Inc., Los Angeles, October 1990.

[Zhang *et al.* 93] Lixia Zhang, Stephen Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala, “RSVP: A New Resource ReSerVation Protocol,” *IEEE Network*, vol. 7, no. 5, p. 8−18, September 1993.

[Zillich 96] Christian Zillich, “Videokonferenzlösungen auf dem Weg zum Massenmarkt,” *LANline*, pp. 14-20, November 1996.