A TRANSPORT-INDEPENDENT COMPONENT FOR A GROUP AND SESSION MANAGEMENT SERVICE IN GROUP COMMUNICATIONS PLATFORMS

Erik Wilde, Murali Nanduri, Bernhard Plattner

Computer Engineering and Networks Laboratory (TIK)
Swiss Federal Institute of Technology (ETH Zürich)
CH – 8092 Zürich
E-mail: wilde@tik.ee.ethz.ch

ABSTRACT

Group communications is an area of research which has received a lot of attention recently. This paper focuses on a model and the architecture of a system which supports group communications by providing group and session management functionality. This system is an extension of directory services which are used with unicast communications. New functionality is needed for the dynamics of group communications (members of a connection may change over the lifetime of the connection) and increased complexity of relations. A model is described which defines six object types which represent the relevant objects. Users and groups represent real world users and their relations. Sessions and flows describe ongoing group communications. Flow templates and certificates provide mechanisms for management and security issues. The architecture presented in this paper is transport-independent, ie it can be used within different group communication platforms. A short sketch of the implementation is given in the last section.

1. INTRODUCTION

Over the past years, surveys by various authors [1, 2, 3] have shown the increasing need for multipoint communications and the requirements for platforms supporting this type of communications. Additionally, the requirement for multimedia data transport is also applied to multipoint communications. One common approach for designing a platform which supports multipoint multimedia data transport services is the separation into different planes, one dealing with the actual data transfer and another being responsible for management issues. This paper presents an approach to the management plane which is based on the assumption that many problems to be solved are independent from the transport infrastructure being used and should therefore be addressed inside a component which is reusable for different transport platforms. This approach has the following advantages.

- *Reduced implementation costs.* Because of the transport-independency of such a component, it could be used in different transport platforms without having to implement the functionality for each platform. This leads to a reduction of implementation costs for new platforms using this component.

- *Transport-independent naming.* The name to address mapping is one of the tasks of the management plane. If naming is implemented by a transport-independent component, then it is possible to use the same names for addressing for different transport platforms. This would eliminate the situation of today, where each collaborative application has its own name space so that the use of more than one collaborative application can become a very complicated process in terms of user and group management.

- *Session directory functionality.* A session directory similar to the well-known mbone session directory would be possible, and it would be a more general directory. It would not only list the sessions and the respective applications, but also the transport infrastructure being used. This way it would be easy to implement collaborative applications which are able to use different transport infrastructures.

The approach taken in this paper illustrates that there is functionality which can be separated from transport-dependent issues and implemented in a separate and reusable component. In order to elaborate this hypothesis, the rest of the paper is structured as follows. Section 2 gives a short overview over other work in this area and related activities. Section 3 then describes our model of how such a component could be used inside a transport infrastructure. Moving on to a model of how such a component could be designed, section 4 describes our architecture of such a model. Section 5 gives a short overview over implementation issues, and section 6 then concludes the paper.

2. RELATED WORK

Group communications has become a very active field in the last years. However, most of the work concentrates on multicast protocols (ie the task of transmitting data over a network to more than one receiver without wasting network resources), especially multimedia multicast protocols, and high-level issues, such as toolkits for programming distributed multimedia applications.

Developments in the area of group and session management for group communication frameworks are not so wide-spread, although there are research activities and also standardization efforts. The rest of this section will describe some of these projects. A more detailed description has recently been published by Mauthe et al. [4].

2.1. Research activities

The BERKOM Multimedia Collaboration Service (MMC) described by Altenhofen et al. [5, 6] is an example for an application oriented development of a communications platform. MMC is a part of the BERKOM multimedia teleservices, which also include a Multimedia Mail Service (MMM) and a Multimedia Transport Service (MMT). MMT

is a pure transport service and does not offer any group management services. It is based on a variant of XTP which is called XTP-Lite. MMC contains various components, of which Conference Interface Agents (CIAs), Conference Managers (CMs), and a Conference Directory (CD) are relevant for the scope of this paper. Communication between these components is realized by two protocols, the CD Access Protocol (CDAP) and the CM access protocol (CMAP). These protocols are RPC-based and built on top of the ISO ROSE mechanisms. The architecture of MMC is similar to the architecture described in this paper. However, the service provided is application specific and therefore can be regarded as one possible application of the general concept of a group and session management service described in this paper.

GCommS by Mauthe et al. [7, 8] is a project at Lancaster University which is concerned with the support of group communication, especially for multimedia data. One of the components of the architecture proposed for GCommS is the QoS architecture QoS-A described by Campbell et al. [9]. The transport aspects of GCommS are defined in detail, but the group management services are only described on a very abstract level. Currently, no specification is available, although a transport service (the so-called M-Connection service) has been implemented. GCommS therefore can be regarded as one of the many projects where group management is necessary to provide a well-functioning platform, but is not specified in detail. Other examples for platforms where group management is identified as an important component but not investigated in detail are the xAMp architecture described by Rodrigues and Veríssimo [10], PRISM by the Toutains [11], or Tenet described by Ferrari et al. [12, 13].

CIO multi-peer communications as described by Henckel [14, 15] is a very interesting concept in terms of functionality. The transport group management defined for the CIO transport service has many similarities to the model we describe in this paper. A user of the CIO multi-peer transport service uses two different components for accessing the transport service and the transport group management service. Communications are handled with two completely separate protocols. However, CIO transport group management is limited to one communications platform (ie depends on the usage of the CIO transport service), and it has not been implemented. Furthermore, since the X.500 directory service is proposed as a basis for the transport group management service, it may be impossible to have notifications sent to users.

2.2. Standardization bodies

ITU's T.120 series of recommendations [16] is an example for a standardized application architecture which also incorporates group and session management functionality. The basis of the T.120 infrastructure are the network specific transport protocols defined in T.123 [17], which at the moment support data transfer using integrated services digital networks (ISDN), circuit switched digital networks (CSDN), public switched digital networks (PSDN), and public switched telephone networks (PSTN). Extensions to include future broadband networks are under study. T.123 is used by the multipoint communications service T.122/T.125 [18, 19], which defines a network independent service with flexible data transfer modes (broadcast and request/response), multipoint addressing (one to all, one to sub-group, and one to one), and multipoint routing (shortest paths to each receiver and uniform sequencing). Recommendation T.124 [20]
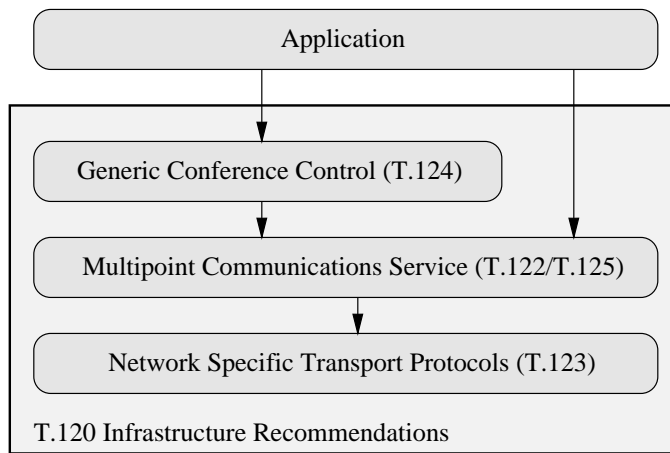
Figure 1: Architecture of the ITU T.120 infrastructure recommendations

then defines a generic conference control which uses T.122's multipoint communications service. The abstract services of the conference control include create, query, join, invite, add, lock, unlock, disconnect, terminate, eject user, and transfer services. These services provide a powerful environment for implementing conferencing applications, which then use T.124 and T.122 services. However, the applicability of these recommendations is limited because of the concentration on conferencing. The T.120 series of recommendations can therefore be regarded as a specific example which should be kept in mind when designing more general group and session management services.

In ISO, group communication is dealt with in different committees and working groups. We will only consider the work going on in JTC1/SC6, which is currently the most active with respect to group communications. ISO's multipeer taxonomy [21] (which is also described in a paper by Mathy et al. [22]) gives a general architectural model for group communications, which may be used within different OSI layers. Concepts defined in this taxonomy describe a group (a set of entities), group memberships (entities belonging to a group), population characteristics (a set of attributes a group may have), group associations (data transfer relationships), and conversations (the basic components of group associations). These concepts can be combined to get a model which is very flexible in terms of group communications. However, as with many other standardization activities, no implementations of these concepts exist, so it is not possible to experimentally evaluate the usability of this model. The ISO draft document on an enhanced communications transport service [23] incorporates most concepts described in the multipeer taxonomy. The naming and addressing model defined in this draft is more complex than in point-to-point transport services, because conversations and group associations have to be taken into account.

3. A MODEL FOR GROUP AND SESSION MANAGEMENT

Group and session management can be identified as one component which is necessary in every system supporting group communications. Figure 2 shows how such a component may be integrated into the communications platform (the component is labeled GUA, a name which will be explained later). The model is independent from the
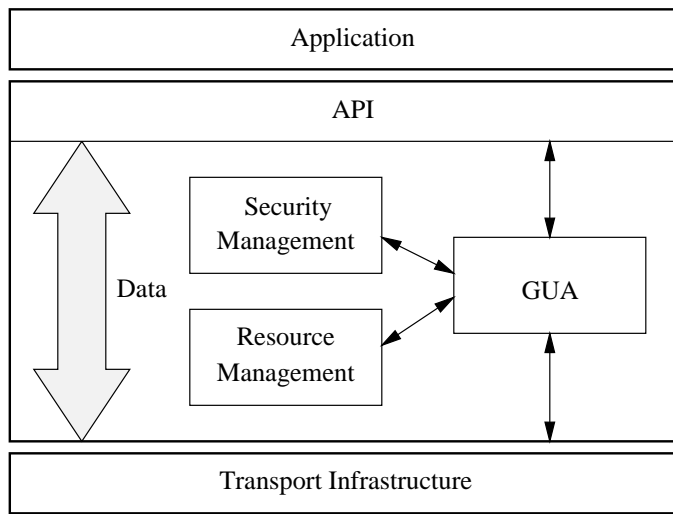
Figure 2: Group and session management inside a communications platform

communications platform being used, although it is mainly influenced by discussions within the Da CaPo project. This project, where the main goals are efficient communication protocols achieved by dynamic configuration, developed the need for group and session management after broadening the model from point-to-point communications to point-to-multipoint connections.

A group and session management component is used for different tasks within a group communications platform. These tasks may be divided into security issues, name to address mapping, QoS issues, and the core functionality, group and session management. The following paragraphs will discuss these issues in greater detail.

- *Security issues* are addressed in the form of identification, authentication, authorization, and certification. These are the security issues which are important in the context of group and session management and the provision of certified software components. Security issues may also be addressed for communications itself via QoS parameters, where authentic and/or private communications may be required. However, this type of security falls into the topic of QoS issues.

- *Name to address mapping* is the functionality which enables users of any communications system to use names instead of addresses, ie to use an abstract form of naming entities (communication endpoints). This makes it possible to use logical names independent from locations and connections which are mapped onto addresses which can be used for addressing in a given transport infrastructure. Saltzer [24] describes the concept of naming, addressing, and binding for the Internet.

- *QoS issues* are important when speaking of actual connections. The well-known multimedia QoS parameters like throughput, error rate, delay, or jitter, can be seen as one special case of a general characterization which may be defined for each connection. The parameters being used should be as versatile as possible to allow the utilization of yet to be defined types. Szyperski and Ventre [25]

and Carle et al. [26] describe this concept of QoS parameters for multipoint communications in greater detail.

- *Group and session management* is necessary for making the naming space (ie the names which may be used) flexible and configurable. Typically, groups of users are changing over time, so it must be possible to create new groups, to change group members, and to delete groups. Furthermore, the communications inside a group (which we call a session) should also be manageable by users of the communication platform. It should also be possible to create special sessions, eg anonymous sessions (where the senders and/or receivers are not known) or sessions which are open to any participant (open sessions). Another important functionality in this area is the provision of authorizations for group and session modifications.

The Architecture we propose for this functionality is that of a directory service, where each user of the directory service uses a component which is implementing the access protocol to the directory service. With respect to this design, our proposed group and session management service is similar to the Internet domain name system (DNS) defined by Mockapetris [27, 28], or the directory service standardized by ISO and ITU, also known as the X.500 service [29, 30]. However, because in our case we need more than a pure lookup service (where the directory system only reacts and never acts), and special functionality (such as operations for joining and leaving groups and sessions), a new service is required. This new service is provided by the group and session management system (GMS) and is available through a GMS user agent (GUA).

4. GMS – GROUP AND SESSION MANAGEMENT SYSTEM

The general GMS model invented in the previous chapter may be further refined to define an architecture which may then be implemented and tested. The following sections describe different topics of the architecture. After a general description of the architecture, the GMS object types are specified. QoS issues are the topic of the last section, where the different steps and definitions of QoS usage are explained.

4.1. Architecture

The architecture of the GMS is similar to that of other directory services, in the sense that the GMS service is available through a GMS access protocol (GAP) which is used by GMS user agents (GUA), and is implemented by the GMS system protocol (GSP), which is used by the GMS system agents (GSA) to communicate and thus implement the GMS. The overall view of this architecture is shown in figure 3.

- *GMS user agent (GUA)*. GUAs are the components outside the GMS which are used to access the GMS. Typically, a GUA is not much more than a protocol machine which implements the GMS access protocol. As such, the GUA provides an interface which may be used by other components of the communication platform. It performs encoding and decoding of protocol messages and also implements some local parameter checking. The GUA is not directly accessible to
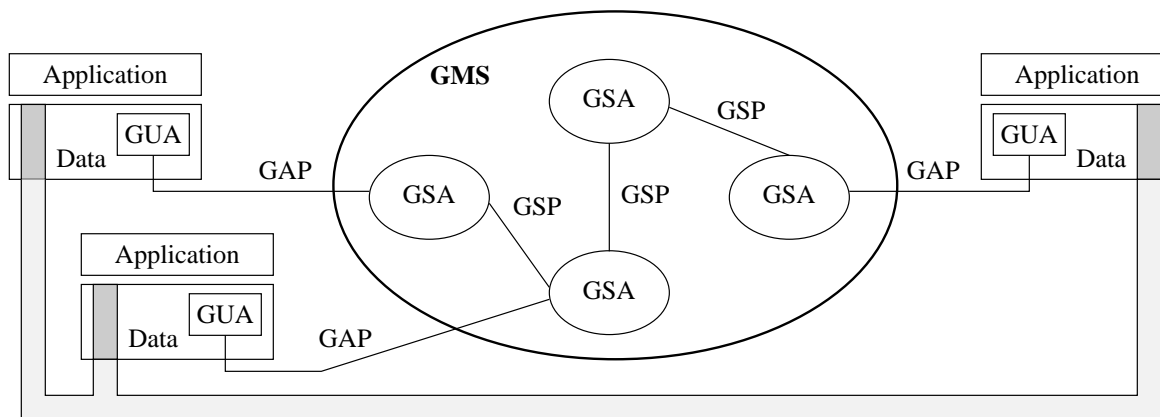
Figure 3: GMS architecture

application programmers, because the communication platform designers should be free to define the platform's API in a coherent way without having to accept a predefined GUA API for applications.

- *GMS system agent (GSA)*. GSAs are the components which, in their entirety, make up the GMS. Each GSA is a component of this distributed system and communicates with either other GSAs (using the GMS system protocol), or GUAs (using the GMS access protocol). In the GMS system, data storage is distributed, so that each GSA may store some local data which may be queried by another GSA. GSAs are grouped into domains, where GSAs communicating inside one domain use the intra-domain GMS system protocol, while communications crossing domain boundaries use the inter-domain GMS system protocol.

The two main components of the GMS architecture, GMS user and system agents, are communicating using special protocols. These protocols are used for accessing the GMS (via GUAs) respectively for communications inside the GMS. The following paragraphs will give a more detailed description of these two protocols.

- *GMS access protocol (GAP)*. GUAs are used for accessing the GMS using entry points provided by GSAs. These entry points must be accessed using GAP, which is a asymmetrical protocol. The complete specification [31] contains definitions of used object types (which are described in section 4.2 of this paper), relations between objects, PDU definitions, and state transition diagrams describing the behavior of the two communicating agents. Figures 4 and 5 show the state transition diagrams of the GUA and GSA side of the protocol. The notation of events and actions used for the transitions is taken from ITU's ACSE specification [32].

Because the number of states is much bigger than the ones shown in these figures, only the topmost level of the diagrams is shown, which illustrates connection setup and tear-down. Double bounded boxes represent states which contain complete state diagrams themselves. Dashed lines separate state diagrams which are executed in parallel. Furthermore, it is possible that there may be several parallel entities of the GUA bound state, each representing one user's connection to the
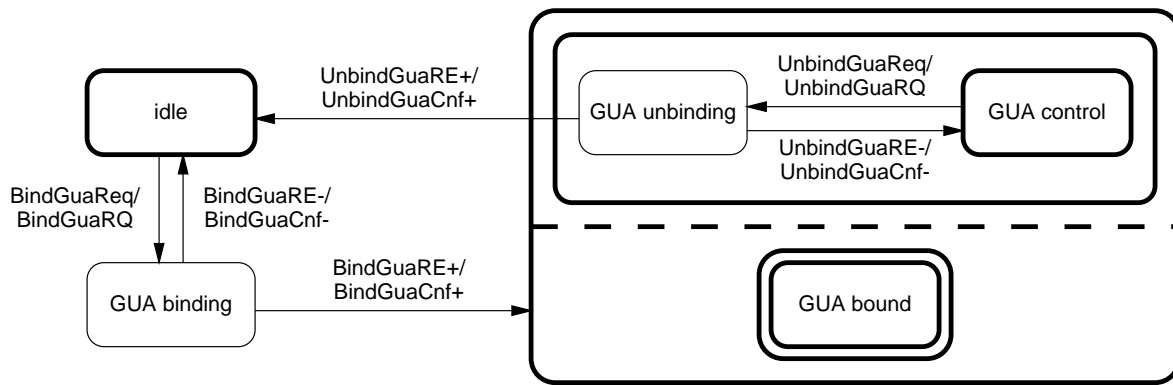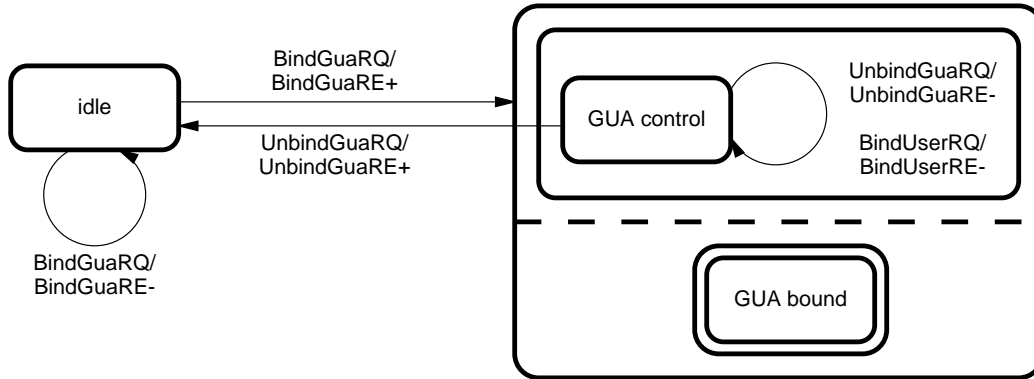
Figure 4: GAP/GUA state diagram

Figure 5: GAP/GSA state diagram

GMS. This way it is possible that several users can be connected to the GMS using only one GUA, which is necessary for communication frameworks where only one GUA exists for each machine.

GAP itself can be separated into three phases. The first phase is illustrated in the two figures, representing connection setup and tear-down between GUA and GSA. The second phase is the authentication phase, where a user is identified and authenticated. Authentication is handled in a very generic way, so that weak and strong mechanisms (from no or standard Unix passwords up to RSA challenges with multiple iterations) can be used. After successful authentication, the third phase, the user bound state (not shown in the figures), is entered and services for accessing the GMS's data can be used. These services include create, modify, delete, join, and leave for the appropriate objects. Most GAP requests are initiated by the GUA (ie the user), but there are also some situations where the GSA becomes active (eg when sending notifications, invitations, or renegotiation requests).

- *GMS system protocol (GSP)*. GSAs are components of the GMS which store data and provide access to the GMS. GSAs communicate using the GSP. The GSP is made up of two different parts, an intra-domain protocol and an inter-domain protocol. Depending on the relation of two GSAs, the communication between

them is using one of the two variants. Because the GSP is not visible for users of the GMS service, it is not described in greater detail.

These descriptions of GMS components (GUA and GSA) and the protocols used for communications between these components give an overview of the GMS system. The following two sections will deal with two aspects of the data being stored and exchanged, which are the object types available and the way QoS issues are handled.

4.2. GMS object types

All definitions of the object types available inside the GMS are specified in ASN.1. Detailed descriptions are given in the GAP specification [31], here we will only give an overview of the object types, their attributes, possible relations between objects, and the general concept behind each type.

- *User.* A user is a person or entity using the GMS. Each user has a identity (a name) and one or more ways of authenticating himself. This authentication may vary from no authentication at all (it is sufficient to use the right name) to sophisticated, hardware-oriented authentication schemes with multiple challenge iterations. The authentication method being used is important for some operations, which may only be performed if a certain level of authentication has been used when binding to the GMS.

  A user object contains information about a user, such as his real world name, a description, his email address, and a list of the bindings of a user, ie the list of active GMS connections a user has. The relations of a user object describe which objects the user owns, of which groups and/or sessions the user is a manager of, of which groups he is a member of, of which sessions he is a participant of, and for which flows he is a sender and/or receiver.

- *Group.* Group communications need a flexible way of handling groups of users. GMS groups may consist of users and/or groups, depending on the definition of the group. Joining and leaving a group is depending on the group's join policy and authentication requirements. Joins and leaves may be notified to a group's managers and/or members. Groups may also be static (opposed to dynamic), which means that no joins or leaves are possible (group members then must be defined at time of the group's creation).

  Each group object may contain a group's real world name (eg the name of a company or a company's department), a description of the group, a group's mail address, and the access rights, which determine who is authorized to modify the attributes of the group. Possible relations for a group specify an owner of the group, managers of the group, members of the group, and sessions associated with that group.

- *Flow Template.* For several applications and communication platforms it is useful to have a number of predefined possibilities to set up connections. Flow templates contain information about data types which may be carried by a flow of that type, the necessary transport service, data which is needed to set up a flow of that type,

information about uni- or bidirectional services, and a set of QoS parameters, which can be used to give a description of the flow template. However, flows may also be created without using a flow template.

- *Flow.* A flow is one connection for data transport. Depending on the flow's definition, it is either uni- or bidirectional, has a limited number of senders and/or receivers, and a renegotiation policy, which determines who is authorized to initiate QoS renegotiations for that flow. Flows are created when a session is created and are deleted when a session is deleted. Joining a flow takes place when a session is joined, and a flow is left when the session of a flow is left.

  Each flow has relations which describe the users sending and receiving that flow (the addressing information is also part of the flow and differs depending on the type of addressing being used), dependencies with other flows (in case data is being sent which needs data being sent on other flows for being interpreted correctly, such as hierarchically coded data like MPEG-2 [33]). It is also possible to create a relationship between flows which denotes a synchronization between flows (a popular example for this are individually transmitted audio and video data which must be synchronized for playback), however, the implementation of the synchronization must be done inside the communications platform.

- *Session.* The main metaphor for data transfer is a sessions. Each session is used to logically group a number of flows and to create an abstraction for management, authorization, and admission control for flows. The flows of a session are created when the session is created and deleted when the session is deleted, ie there is no possibility to dynamically add or remove flows from an existing session. However, when joining a session, not all flows of the sessions must be joined, so users can choose which flows to use (provided dependencies between flows are respected).

  Sessions may have application specific information, which consists of an application identification and application specific data, which may be interpreted by the application. Furthermore, the duration of a session may be given with either start or end times or both. In addition, it is possible to specify which authentication level a user must have to successfully join a session (provided he is authorized sufficiently). Authorization is based on the session's join policy which may be open (everyone may join), group (only members of the group associated with the session may join), or managed, which may be either relative (a given percentage of managers must confirm) or absolute (a given number of managers must confirm).

  A session's relations describe the owner and managers of a session, which are important for modifying the session object and determining who may join the session (if the join policy is set to managed). The participants of a session are all users who have joined the session (except the users who have left already). A session's flows are all flows which are part of this session. A last relation determines, which group (if any) the session is associated with.

- *Certificate.* Applications with special security requirements may have the need to store certificates inside the GMS, which are used for checking data identity

and integrity. Certificates include the type (which may be a predefined type or any other type), the name type (which also has a number of predefined values and the possibility to define own types), the certificate's validity, a simple name, and data and signatures, which contain the informations which is necessary for checking the data. The only relation a certificate has is the one with its owner.

These object types are available for usage inside the GMS and may be created, modified, and deleted with GAP services. However, because a number of attributes and relations (such as the addressing information attribute and the senders relation of flows) needs to be modified according to their semantics, they are implicitly modified by using GAP services (such as the *join session* service, which would add a sender's address to a flow he joined and modify the senders relation accordingly).

4.3. QoS issues

QoS is a very important aspect of multipoint communications, especially if multimedia is also taken into account. GMS must be able to support QoS aspects in a way which make it suitable for the support of multimedia multipoint communications. Two main issues can be identified here, the QoS parameter types (defining how QoS can be defined) and the procedures available to manipulate parameters of these types (defining how QoS can be applied).

GMS has four QoS parameter types, which are *unsorted values*, *sorted values*, *integer values*, and *real values*. Unsorted Values are a set of predefined values, which are not in any particular order (an example for this is a QoS parameter which defines a coding algorithm, where it is not possible to arrange the different algorithms in any order). Sorted Values are also predefined values, but these values are given as a sequence, because it is possible to arrange them in an order (an example for this type of QoS parameter is the selection of a error detection algorithm, which may be given as a sequence of none, CRC8, CRC16, and some more sophisticated algorithms). Integer and real values represent the two basic types of numbers which may be used (which may be used for throughput or error probability numbers).

Any parameter defined for a flow template or flow may be of one of these types. The interpretation of the parameter's values depends entirely on the parameter's name. It is therefore important for GMS users to agree on names for QoS parameters to avoid misinterpretations. For this reason, a number of predefined QoS parameters is available, where the semantics of a parameter are clearly defined (taken from the ISO draft document for a QoS framework [34]). Additional parameters may be used at any time, although it is strongly recommended to use the predefined types whenever possible.

The second aspect of QoS, as mentioned above, are the procedures available for manipulating QoS parameters. Given the object types of the GMS (as described in section 4.2), four different events where QoS come into play can be described.

- *Flow template creation.* When a flow template is created, it may also be given a number of QoS parameters. Depending on the transport infrastructure, the number of parameters already necessary or even known at this point in time may

differ a lot. For the different types of QoS parameters, values (resp. limits) may already be defined, if possible.

- *Session creation (flow creation).* Every data transfer is represented by a flow, which is created when the session it is a part of is created. Each flow's QoS parameters are defined by their name and type and at least a default value (which is the value used for joining participants if no local modifications are requested). Optionally, a weakest limit for joining the flow and strongest and weakest limits for renegotiation may be defined. For unordered values QoS parameters, there is no order of the values, therefore the join and renegotiation values must be given as sets of values.

- *Session joining.* When joining a session, the QoS parameters being used are taken from the flow's QoS definitions. According to the user's requirements, a weaker value than the default may be selected, as long as it does not fall short of the weakest limit for joining the flow. Because the actual QoS establishment lies outside the scope of the GMS (which is only responsible for storing the values), it is required that transport infrastructures using the GMS control the QoS parameters according to the values provided by the GMS.

- *QoS renegotiation.* QoS renegotiation is the process of defining new default values and weakest limits for QoS parameters of any renegotiable flow. This renegotiation may be limited by the renegotiation limits of a QoS parameter, if present. All participants (senders and receivers) of a flow are informed of QoS renegotiations of that flow.

A detailed and complete description of the QoS parameters and their manipulation along with the ASN.1 definitions can be found in the GAP specification [31]. However, it should always be kept in mind that QoS establishment and renegotiation are tasks to be performed by the transport infrastructure, while the GMS is only used for distribution of QoS values and renegotiation notifications.

5. IMPLEMENTATION

The implementation of the architecture described in section 4 is one of the key aspects of the GMS work. So far, research has either concentrated on implementing transport protocols, on implementing applications together with application-specific management components, or on merely specifying management services. GMS is being implemented at the time of writing and will be integrated in at least two communication platforms. These will be the extended Da CaPo platform developed at our lab, which has been modified to include group communications, and a multicast communications framework (MCF, described by Bauer et al. [35]), which aims at implementing multicast protocols with QoS guarantees. Furthermore, very small group communication platforms based on IP multicast and ATM/UNI multicast facilities should be implemented to prove the applicability of the GMS concept to different transport services. We hope that the application of the GMS concept within different communication platforms as well as the development of applications on top of these communication platforms will lead
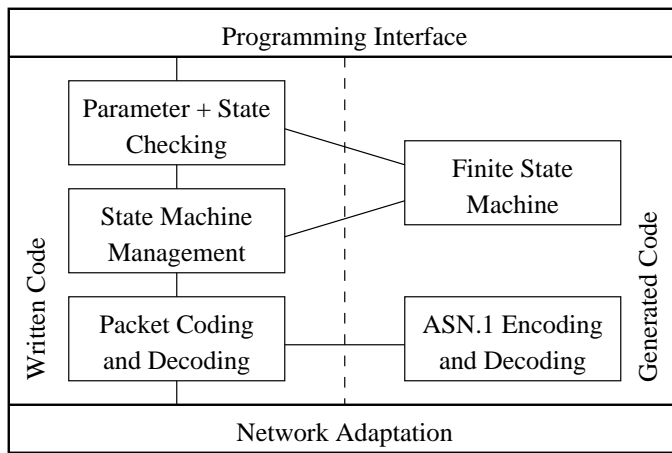
Figure 6: GMS user agent (GUA) design

us to a better understanding of the requirements for a group and session management service in terms of protocols, object types, and performance issues.

The implementation of a GUA is currently being done in a Unix environment (Solaris 2) using C. Two tools are being used, one is the ASN.1 compiler snacc described by Sample and Neufeld [36, 37], which is performing all the coding and decoding of incoming PDUs, the other tool is the commercial software StateMate, which is being used for implementing the state machine which handles the protocol. The programming interface (which is used by other components of the communication platform according to figure 2) and the interface to the transport system are subject to change for integration of the GUA within different communications platforms, therefore these components are isolated from the rest of the code and easily replaceable.

6. CONCLUSIONS

In this paper we have presented the model and architecture for a transport-independent group and session management system (GMS). GMS consists of two components, GMS user agents (GUAs), and GMS system agents (GSAs). These components communicate using the GMS access protocol (GAP, for GUA-GSA communications) respectively the GMS system protocol (GSP, for GSA-GSA communications). The GMS service is designed as a distributed name service with additional functionality, such as notifications. There is a set of defined object types which may be used to model information about users, groups, sessions, and flows. Mechanisms for authentication and authorization are available and make the design of secure communication platforms possible.

The implementation of the GUA component permits the easy integration into communication platforms. Because the design of object types and operations is independent from a specific transport service, the GMS service may be used by various communication platforms. The current implementation will be integrated into two platform, the Da CaPo system developed at our institute, and a multicast communications framework, which will also be developed at our institute. Future plans include the incorporation of the GUA component into different platforms. It is also planned to extend the GUA with regard to the transport infrastructure being used.

We believe that the GMS service in general and the GUA as a component for the inclusion into communication platforms will permit the design and implementation of group communication platforms with a more abstract service in terms of naming, addressing, name and address management, and authentication. We also believe that the implementation of such a system and its usage inside actual group communication platforms will lead us to a better understanding of which services are required from an application point of view and which services should be provided by the GUA component.

7. REFERENCES

[1] T. Rodden, J. A. Mariani, and G. Blair. Supporting Cooperative Applications. *Computer Supported Cooperative Work*, 1(1–2):41–67, 1992.

[2] Tom Rodden and Gordon S. Blair. Distributed systems support for computer supported cooperative work. *Computer Communications*, 15(8):527–538, 1992.

[3] Neil Williams and Gordon S. Blair. Distributed multimedia applications: A review. *Computer Communications*, 17(2):119–132, 1994.

[4] Andreas Mauthe, Geoff Coulson, David Hutchison, and Silvester Namuye. Group Support in Multimedia Communications Systems. In Hutchison et al. [38], pages 1–18.

[5] Michael Altenhofen, Jürgen Dittrich, Rainer Hammerschmidt, Thomas Käppner, Carsten Kruschel, Ansgar Kückes, and Thomas Steinig. The BERKOM Multimedia Collaboration Service. In *Proceedings of ACM Multimedia 93*, pages 457–463, Anaheim, California, 1993. ACM Press.

[6] Michael Altenhofen, Joachim Schaper, and Susan Thomas. The BERKOM Multimedia Teleservices. In Steinmetz [39], pages 237–250.

[7] Andreas Mauthe, David Hutchison, Geoff Coulson, and Silvester Namuye. From Requirements to Services: Group Communication Support for Distributed Multimedia Systems. In Steinmetz [39], pages 266–279.

[8] José F. de Rezende, Andreas Mauthe, David Hutchison, and Serge Fdida. M-Connection Service: A Multicast Service for Distributed Multimedia Applications. In Hutchison et al. [38], pages 38–58.

[9] Andrew Campbell, Geoff Coulson, and David Hutchison. A Multimedia Enhanced Transport Service in a Quality of Service Architecture. In D. Shepherd, G. Blair, G. Coulson, N. Davies, and F. Garcia, editors, *Proceedings of the 4th International Workshop on Network and Operating System Support for Digital Audio and Video*, volume 846 of *Lecture Notes in Computer Science*, pages 124–137, Lancaster, UK, November 1993. Springer-Verlag.

[10] Luís Rodrigues and Paulo Veríssimo. xAMp: a Multi-primitive Group Communications Service. In *Proceedings of the 11th Symposium on Reliable Distributed Systems*, pages 112–121, Houston, Texas, 1992. IEEE Computer Society Press.

[11] François Toutain and Laurent Toutain. Network Support for Multimedia Communications Using Distributed Media Scaling. In Hutchison et al. [38], pages 139–158.

[12] Anindo Banerjea, Domenico Ferrari, Bruce A. Mah, Mark Moran, Dinesh Verma, and Hui Zhang. The Tenet Real-Time Protocol Suite: Design, Implementation, and Experiences. Technical Report TR-94-059, International Computer Science Institute, Berkeley, California, November 1994.

[13] R. Bettati, D. Ferrari, A. Gupta, W. Heffner, W. Howe, M. Moran, Q. Nguyen, and R. Yavatkar. Connection Establishment for Multi-Party Real-Time Communication. In *Proceedings of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 255–266, Durham, New Hampshire, April 1995.

[14] Lutz Henckel. Multipeer Transport Services for Multimedia Applications. In S. Fdida, editor, *Proceedings of the IFIP TC6/WG6.4 Fifth International Conference on High Performance Networking*, volume C-26 of *IFIP Transactions*, pages 167–186, Grenoble, France, June 1994. Elsevier.

[15] Lutz Henckel. Multipeer Connection-mode Transport Service Definition based on the Group Communication Framework. Technical report, GMD FOKUS, Berlin, June 1994.

[16] International Telecommunication Union. Data Protocols for Multimedia Conferencing. Draft Recommendation T.120, 1995.

[17] International Telecommunication Union. Protocol Stack for Audiographics and Audiovisual Teleconference Applications. Recommendation T.123, 1993.

[18] International Telecommunication Union. Multipoint Communication Service for Audiographics and Audiovisual Conferencing – Service Definition. Recommendation T.122, 1993.

[19] International Telecommunication Union. Multipoint Communication Service – Protocol Specification. Recommendation T.125, 1994.

[20] International Telecommunication Union. Generic Conference Control for Audiovisual and Audiographic Terminals. Draft Recommendation T.124, 1995.

[21] International Organization for Standardization. Draft Text on the Subject of ”Multi-peer Taxonomy”. ISO/IEC JTC1/SC6 N9161/IV, March 1995.

[22] Laurent Mathy, Guy Leduc, Olivier Bonaventure, and André Danthine. A Group Communication Framework. In Wulf Bauerfeld, Otto Spaniol, and Fiona Williams, editors, *Broadband Islands '94: Connecting with the End-User*, pages 167–178, Hanburg, Germany, June 1994. North-Holland.

[23] International Organization for Standardization. First Draft of Enhanced Communications Transport Service Definition. ISO/IEC JTC1/SC6, March 1995.

[24] J. Saltzer. On the Naming and Binding of Network Destinations. Internet RFC 1498, August 1993.

[25] Clemens Szyperski and Giorgio Ventre. Efficient Support for Multiparty Communication. In Hutchison et al. [40], pages 185–198.

[26] Georg Carle, Jochen Schiller, and Claudia Schmidt. Support for High-Performance Multipoint Multimedia Services. In Hutchison et al. [40], pages 219–240.

[27] P. Mockapetris. Domain Names – Concepts and Facilities. Internet RFC 1034, November 1987.

[28] P. Mockapetris. Domain Names – Implementation and Specification. Internet RFC 1035, November 1987.

[29] International Telecommunication Union. The Directory – Overview of Concepts, Models and Services. Recommendation X.500, March 1995.

[30] Bohdan Smetaniuk. Distributed Operation of the X.500 directory. *Computer Networks and ISDN Systems*, 21:17–40, 1991.

[31] Erik Wilde. Specification of GMS Access Protocol (GAP) Version 1.0. Technical Report TIK-Report No. 15, Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology, Zürich, March 1996.

[32] International Telecommunication Union. Association Control Protocol Specification. Recommendation X.227, 1988.

[33] International Organization for Standardization. Information technology – Generic coding of moving pictures and associated audio information. ISO/DIS 13818, 1995.

[34] International Organization for Standardization. Information technology – Quality of Service – Framework. ISO/CD 13236, July 1995.

[35] Daniel Bauer, Erik Wilde, and Bernhard Plattner. Design Considerations for a Multicast Communication Framework. In *Proceedings of the Tenth Annual Workshop on Computer Communications*, Eastsound, Washington, September 1995.

[36] Michael Sample. Snacc 1.1: A High Performance ASN.1 to C/C++ Compiler. Technical report, University of British Columbia, Vancouver, July 1993.

[37] Michael Sample and Gerald Neufeld. Implementing Efficient Encoders and Decoders For Network Data Representations. In *Proceedings of the IEEE INFOCOM '93 Conference on Computer Communications*, pages 1144–1153, San Francisco, 1993. IEEE Computer Society Press.

[38] D. Hutchison, H. Christiansen, G. Coulson, and A. Danthine, editors. *Teleservices and Multimedia Communications – Proceedings of the Second COST 237 Workshop*, volume 1052 of *Lecture Notes in Computer Science*, Copenhagen, Denmark, November 1995. Springer-Verlag.

[39] Ralf Steinmetz, editor. *Proceedings of the Second International Workshop on Advanced Teleservices and High-Speed Communication Architectures*, volume 868 of *Lecture Notes in Computer Science*, Heidelberg, Germany, September 1994. Springer-Verlag.

[40] D. Hutchison, A. Danthine, H. Leopold, and G. Coulson, editors. *Multimedia Transport and Teleservices – Proceedings of the International COST 237 Workshop*, volume 882 of *Lecture Notes in Computer Science*, Vienna, November 1994. Springer-Verlag.