Design Considerations for a Multicast Communication Framework

Daniel Bauer, Erik Wilde, Bernhard Plattner
Swiss Federal Institute of Technology (ETH Zürich)
Computer Engineering and Networks Laboratory
CH – 8092 Zürich
{bauer,wilde,plattner}@tik.ee.ethz.ch

Abstract

In the last years, networked multimedia multipoint applications have been developed in conjunction with emerging broadband networks. Experiences have shown that existing transport systems support these applications only insufficiently, since they offer no assistance for real-time multimedia and multipoint applications.
In this paper, we propose a Multicast Communication Framework (MCF) which satisfies the needs of multimedia multipoint applications. MCF covers both transportation and presentation of multimedia data. It guarantees quality of service (QoS) for the complete path between multimedia sources and multimedia sinks. Furthermore, it offers a high-level abstraction of multicast communication services that hides the details of the underlying endsystems and networks.

1. Introduction and Motivation

Research in advanced communications has led to the development of complex networked multimedia applications. Teleconferencing systems are challenging examples of multimedia multipoint applications; they consist of different tools each of them having individual requirements on the communication system. The components of a typical conferencing system are: a picturephone that allows the participants to see each other and to talk to each other, a shared whiteboard or an application sharing tool for joined viewing and editing, and a telemarker tool for pointing into the shared whiteboard. In the following paragraphs, each components's requirements on the communication system are described.

The picturephone produces a continuous stream of audio samples and video frames that have to be transported with low delay and a guaranteed throughput. If the delay is too high, a face to face communication is not possible. The presentation of video frames and the playback of audio samples in the endsystem have to take place at a fixed rate. If the frames are not displayed in time, the video quality degrades. In the case of audio, variations in the playback rate makes it difficult or even impossible to understand a remote speaker. Applications like the picturephone are often refered to as 'real-time' applications [Steinmetz 95].

The shared whiteboard needs a reliable transport for its bulk data. Although the main concern of the shared whiteboard is reliability, the delay must not exceed some hundred miliseconds. Otherwise, the interaction between the participants becomes cumbersome.

The telemarker tool allows a user to move a cursor in the shared whiteboard. This cursor is seen by all participants. The cursor position is periodically multicast to all participants. The messages are small and have to be transported with low delay, such that each participant sees the cursor at the same place.

An example of a teleconferencing application is JVTOS [Dermler et al. 93]. JVTOS runs on top of a standard UNIX operating system. The current implementation uses TCP and UDP as transport protocols and IP over ATM at the network layer. Experiences in using JVTOS have shown that the system and the underlying transport environment have major drawbacks. Firstly, the reliable multicast transport had to be emulated by several point-to-point TCP connections. Secondly, the endsystem does not guarantee that the real-time multimedia data are displayed at a fixed rate. The picturephone UNIX process is vulnerable to changes in the system load. If the endsystem is unter heavy load, the quality of the audio playback and video presentation is severly degraded. Thirdly, the transport protocols do not offer the needed flexibility. Each component of JVTOS needs its own type of transport service. The picturephone needs a guaranteed throughput and a low delay. TCP and UDP do not offer throughput guarantees, even if the underlying ATM network does.

The above mentioned drawbacks can also be identified in other multimedia multipoint applications. In order to overcome these drawbacks, a communication system has to include the following two features:

- **Endsystem performance guarantees:**
  The endsystem has to guarantee that real-time multimedia data is processed and displayed at the re-

quired rate. For this tasks, resources in the endsystems such as CPU, memory, and multimedia devices have to be reserved. Resource reservation in endsystems together with resource reservation in the network (e.g. bandwidth reservation in ATM) allow for a guaranteed services that covers the whole path from multimedia source to multimedia sink.

- **Flexible multicast transport services:**
  The transport system has to offer multicast transport services that are configurable to the needs of the application. The wide range of different application requirements makes it impossible to use a single multicast protocols, e.g. a protocol that offers reliable, causally ordered delivery of messages is not suited for real-time multimedia, since the causal order introduces delays. However, a unified interface to all this protocols is needed in order to simplify the design of the applications.

Both endsystem performance guarantees and flexible transport services are topics of ongoing research. Systems that provide endsystem performance guarantees have been developped only recently: Nahrstedt's "QoS Broker" [Strayer et al. 92][Nahrstedt et al. 95/B] manages the endsystem resources that are needed for the application and the transport protocol. Furthermore, it negotiates with the network resource manager and also with other QoS Broker entities. The QoS-A system which is developed at Lancaster University [Campbell et al. 94] spans both endsystems and network by integrating a range of QoS configurable protocols and mechanisms. Gopalakrishnan [Gopalakrishnan et al. 95] describes a framework for QoS guarantees within endsystems. It provides QoS mapping and support for real-time protocols.
The above mentioned systems are being developed for point-to-point applications and contain no or only marginal support for multipoint applications.

Flexible transport services have been studied for several years. The tenet suite [Ferrari et al. 92] provides communication protocols for multimedia communication. It includes a multicast transport protocol that is QoS configurable. The tenet protocols are only concerned with data transportation and do not cover the playback of real-time multimedia data. In the F-CSS framework [Zitterbart et al. 93], protocol functions are configured to protocol machines. However, F-CSS has no support for multicasting.

In this paper, we propose a multicast communication framework (MCF) that provides endsystem performance guanrantees and flexible multicast transport services. It is based on the Da CaPo framework [Plagemann 94/B][Gotti 94][Vogt et al. 93] which allows protocols to be dynamically configured. This configuration is accomplished by combining predefined building blocks. The resulting protocols are not only used to transport data, but also to display video frames and to play audio samples.

The design goals of MCF are:

- **Support for real-time multimedia data**
  For real-time multimedia data streams, performance guarantees from source to sink has to be provided. This covers the endsystem of the sending side, the network and the endsystem of the receivers. A guanranteed service is achieved by reserving the resources that are needed. In the endsystems, CPU and memory are reserved for protocol execution. At the network level, bandwidth is allocated.
- **Flexible multicast transport**
  MCF has to support a wide range of applications by letting the applications specify their needs using QoS parameters. Using the QoS parameters, a multicast protocol is configured that fulfills the application requirements without adding overhead in form of unnecessary functionality.
- **Support for dynamic join/leave**
  Participants must have the possibility to join or leave an application while data is already being exchanged. This contrasts to unicasting, where both of the two participants are present during the whole data exchange phase.
- **Support for heterogeneous receivers**
  The receviers in a multipoint application are often heterogeneous, with differing endsystems and network-access bandwidth. This heterogeneity has to be taken into account when real-time multimedia data is distributed. Each endsystem should receive the amount of data that it is capable of handling, i.e. for each endsystem the highest possible quality is strived for.

In the following chapters, the design of MCF is described: Chapter 2 presents the multicasting model that is used in MCF. It introduces the concept of MCF flows and application level parameters which are used

to describe flow properties. The operations that are defined for manipulating flows are also presented in chapter 2. In chapter 3, the realisation of MCF using the existing Da CaPo system is described. The results are discussed in chapter 4.

2. Model of MCF

2.1 Overview

The multicast communication framework (MCF) is located between the application and the operating system (see Figure 1). MCF offers configurable protocols for data transportation and presentation of multimedia data.
Real-time multimedia data is directly taken from multimedia sources such as cameras, framegrabbers and audio devices and delivered to multimedia sinks such as framebuffers and loudspeakers. Applications never touch the real-time data, they only control the flow of the data. The time critical data handling is done inside MCF, which means that the applications do not have to cope with real-time data handling. Non real-time multimedia data are forwarded form the appliacation to MCF, from where it is transported over the network.

MCF reserves endsystem resource in order to to guarantee the application requirements. Memory and CPU are reserved in the operating system. It is assumed that the operating system offers real-time scheduling capabilities. Each protocol runs in a separate task. For each protocol task, the scheduling priorities and memory consumption are computed from the application requirements.

MCF protocols run on top of generic communication infrastructure that provide end-to-end connectivity and the possibility of bandwidth reservation as for instance in ATM/AAL5.

MCF contains an interface to the group management system GMS. GMS is used to administer information about users such as their names, their addresses, access rights etc. It is used by MCF as a specialized directory service where administrative information can be stored and distributed. Further information about GMS can be found in [Wilde 95]

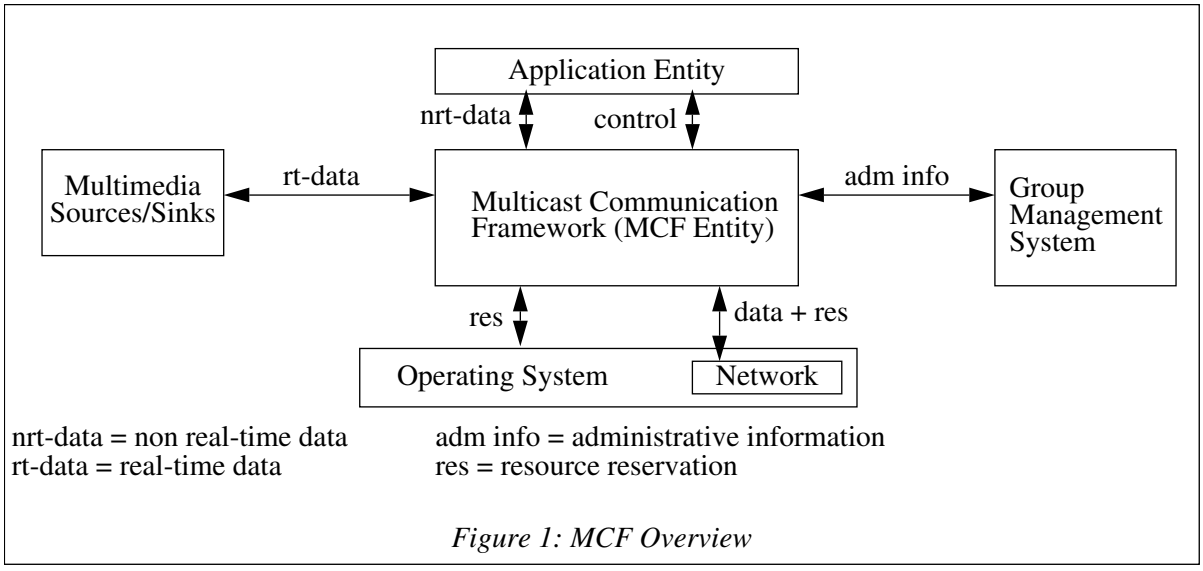


Figure 1: MCF Overview

MCF denotes the whole multicast communication framework, whereas individual users participate via an *MCF entity*. In analogy, the multicast application comprises all *application entities*.

2.2 MCF Flows

A multicast association as seen by the application is called an *MCF flow* or more simply a *flow*. A flow transports data in one direction either from application or multimedia sources to other applications or multimedia sinks. MCF flows abstract from network and endsystem details in the following points:

- **Topology:** Each MCF flow is configured to use one of the following topologies: point-to-multipoint, multipoint-to-multipoint, and as a trivial case point-to-point. The MCF flow topology is independent of the topology at the network level. MCF maps the application configured topology to the topology at the network level. For example, the topology provided by ATM/UNI 3.1 offers only point-to-multipoint connections. Thus, multipoint-to-multipoint flows are built by a combination of point-to-

multipoint ATM connections, with an ATM connection starting at each sender. There are several other possibilities of how the topology mapping could be done, e.g. with one or several multicast servers. However, these possibilities have not been considered yet.

- **Join/leave operations:** The details of join and leave operations heavily depend on the properties of the used network layers. Basically, two modes are known: sender oriented and receiver oriented operations. In the sender oriented mode (as it is used in ATM/UNI 3.1), the sender actively adds or drops members. The receiver oriented approach is used in IP multicasting. In this model, a new participant informs the routing system that it wants to receive data for a flow. Data is then forwarded to the new participant, the senders are not involved in the join opeartion and are even not aware of the new participant.
  The sender oriented approach needs an interaction between the sender and the new participant. When serveral senders are taking part in a flow as it is the case in multipoint-to-multipoint topologies, one interaction for each sender is needed. The sender oriented approach is therefore not suited as a general abstraction. MCF uses the receiver oriented approach for all flows and maps it to the whatever the underlying network level provides.
- **QoS parameters:** QoS parameters are used at different levels inside the MCF system. The lowest level is the network and endsystem level, the QoS parameters at this level describe properties such as bandwidth, delay, memory and scheduling. The middle level is the protocol level with parameters such as throughput, maximum packet size, end-to-end delay, bit-error rate, packet error rate, and loss rate. The highest level is formed by the MCF flows, which build the interface to the application. Therefore, the parameters at this level are called application level QoS parameters. These parameters form the highest abstraction level and are the only one seen by the applications. The lower layer parameters are derived from the application level parameters. A survey of QoS parameters in general is given in [Vogel et al. 95].

2.3 Application Level QoS Parameters

Applications that create a new flow characterize the desired service by specifying application level QoS parameters. This description of the flow is - along with other attributes - stored within the GMS when the flow is created. Application level QoS parameters consist of two parts. The first part describes media and data characteristics whereas the second part characterizes data independent, qualitative service aspects.

2.3.1 Media characteristics

MCF separates media and data characteristics in different classes [Gopalakrishnan et al. 95]. Each media class uses its own set of parameters. Applications have to specify the media class they want to use and the additional parameters. Media classes are defined for standard situations, as shown in Table 1. For a video data stream, the application selects the video media class and specifies the frame rate, the frame size and the delay. Each media class implicitly defines hidden parameters. For instance, video frames in the video media class are displayed in the same order as they are captured.

Media class	**Parameters**
Video	frame rate, average and maximum frame size in pixels, delay
Audio	frequency range, signal-to-noise ratio, delay
Bulk data	bandwidth, delay
Control data	delay, multicast ordering relations

Table 1: Examples of Media Classes

The predefined media classes of Table 1 do not cover every possible application. Additional media classed can be defined and added to MCF. This is done by adding a new media class description to the MCF profile database as described in section 3.2.

2.3.2 Qualitative Service Description

The qualitative service description is used to describe those aspects that are independent of the media. It contains the following elements:

- **Type of guarantees:** MCF offers three types of service commitments: *guaranteed service*, *statistical guarantee* and *best effort* (see also [Danthine et al. 92])
  In the case of *guaranteed service*, MCF assures that the media parameters are never violated, except when a component in the endsystem or the network fails. This is done by reserving all the resources based on the worst case. For instance, memory and bandwidth requirements in the video media class are then allocated based on the maximum frame size. This means that not all of the reserved resources are used all the time, resource efficiency is therefore often rather low.
  Not all applications need guaranteed service. In most cases a *statistical guarantee* is sufficient. Here, resources are allocated in such a way that the media characteristics are kept 'on the average'. This also means that the service will no longer be guaranteed in the worst case.
  For the *best effort* service, no resources are reserved at all. Data is only processed as resources are available.
- **Topology**: The application can choose among different topologies: point-to-point, point-to-multipoint, multipoint-to-multipoint. This parameter is used to describe the topology as it is perceived by the application, independent of the network layer topology.
- **Number of senders/receivers**: In order to calculate the resource requirements, MCF has to know how many senders and receiver participate in a flow. As mentioned in [Zhang et al. 93] and [Gupta et al. 95], there are applications with a large number of senders where only a small number of them transmit data simultaneously. An example is an audio conference where everybody is a candidate speaker, but at any moment in time only one person speaks. It is therefore sufficient to allocate bandwidth for a single speaker only.
  Some applications support a virtually unlimited number of receivers. Therefore it is possible to specify an "unbound" value of receivers. However, this will reduce the functionality of MCF because it is no longer possible to negotiate QoS parameters with a large number of receivers.
- **QoS negotiation:** This parameter describes the participants that are allowed to carry out QoS negotiations.

2.4 Flow Setup

In unicasting, the lifetime of a connection can be separated in three consecutive phases. One partner passively waits until being contacted by the initiating partner. At this phase, QoS parameters are negotiated between the two partners. After the connection is established, data is being exchanged. The last phase is connection release, which can be done by either of the two partners.

In multicasting, however, the different phases are no longer in sequential order. While some participants are already exchanging data, new participants join in or existing ones leave. In the proposed MCF model, a sender of multicasting data does not contact directly its receivers (or vice versa) but rather joins a flow. Before a flow can be joined, it has to be created.

2.4.1 Flow Creation

The only operation carried out when a flow is created is the definition of application level QoS parameters and an appropriate protocol specification. An application creates a flow by defining the properties of the flow. For this purpose, the applications negotiates the application level QoS parameters with its MCF entity. A suitable protocol, fulfilling the QoS requirements, is configured in the MCF entity. This protocol will be used by all participants of the flow. Both the application level QoS parameters and the protocol specification are stored in the GMS, as indicated by the arrow 1 in Figure 2a.

2.4.2 Join Operation

Before data can be exchanged, participants, both senders and receivers, have to join the flow. Applications join either as sender or as receiver. If an application wants to receive and transmit data simultaneously, it has to join the flow twice as a receiver and a sender.

The join operation is carried out in two steps. First, the MCF obtains the flow's application level QoS parameters and the protocol specification from the GMS, as shown by arrow 2 in Figure 2b. Using these

parameters, the MCF instantiates the specified protocol, decides on the resource requirements, and reserves the resources. If this step is successful, the application can participate in the flow, otherwise the join fails.

2.4.3 Leave Operation

Leaving a flow is straightforward. The application informs the MCF entity that it wants to leave the flow. The framework then frees all allocated resources.

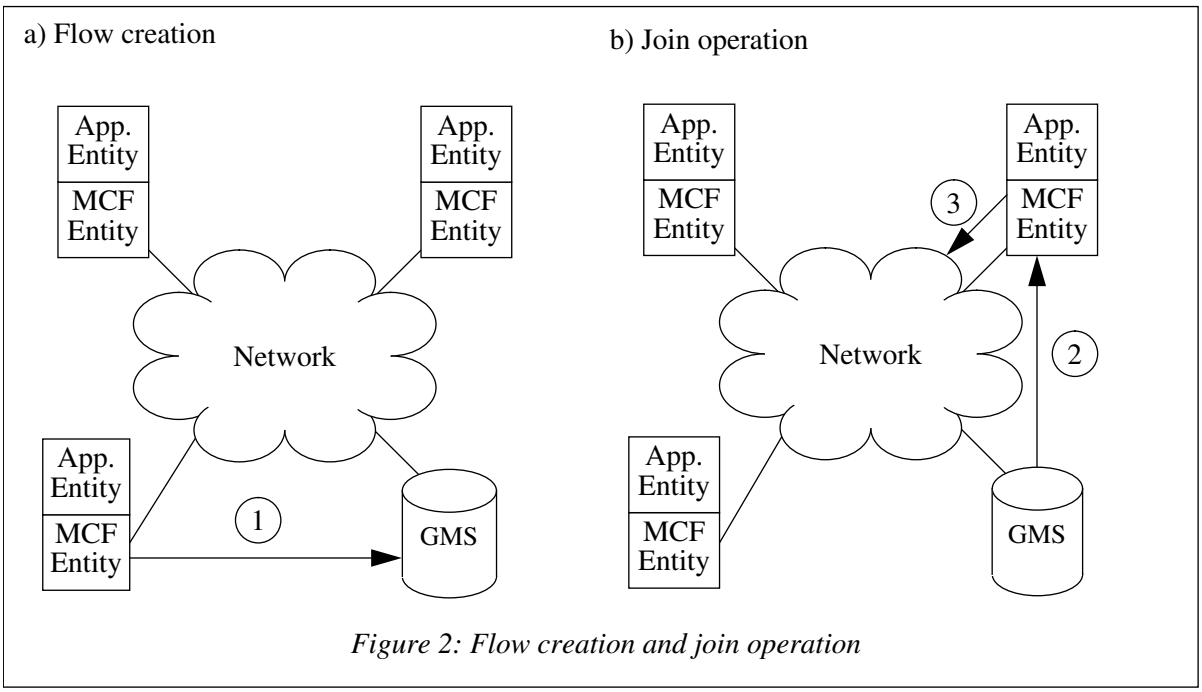


Figure 2: Flow creation and join operation

2.5 QoS Negotiation

QoS negotiation can be separated in layer-to-layer negotiation and peer-to-peer negotiation. Layer-to-layer negotiation is used inside an endsystem between the application, the MCF entity and the operation system and network, whereas peer-to-peer negotiation involves all MCF entities of a flow. QoS negotiation takes place in three different situations.

- **Flow creation:** The application level QoS parameters are negotiated between the application entity and the MCF entity. The application and the MCF entity agree on the values of the application level QoS parameters for which a suitable protocol can be configured.
- **Join Operation:** Peer-to-peer QoS negotiations involving a large number of peers are very time consuming. Therefore, during the join operation only a layer-to-layer negotiation is carried out which checks whether enough resources are available to setup the protocol for the flow. In this case, the negotiation is reduced to a ‘take it or leave it’ approach. Exceptions are discussed in section 2.6.
- **Renegotiation:** Peer-to-peer QoS renegotiation is used for changing the media characteristics of an existing flow. All participants of this flow must take part, which can be very time consuming. This peer-to-peer renegotiation automatically involves layer-to-layer negotiations within each endsystem.

Since the QoS renegotiation is a costly operation it is carried out in exceptional situations only:

- An application requests QoS renegotiation as a result of a user request.
- A component fails and some resources become unavailable. The affected MCF entity then requests a QoS renegotiation.

The MCF entity that requests the change of media characteristics carries out the QoS renegotiation. This involves four steps:

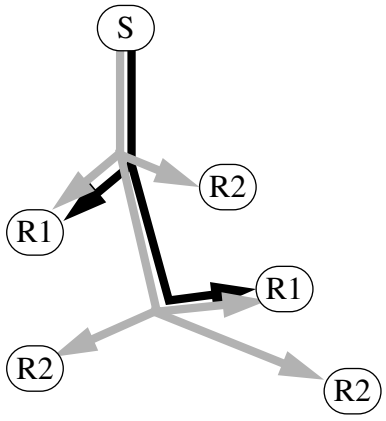
- The requesting MCF entity, called negotiator, changes the application level QoS parameters, configures a new protocol if necessary and multicasts the resulting information to all participants of the flow.

- The MCF entity of each participant checks whether it can adapt to the new QoS parameters. This is in fact the same operation which is done when an application entity joins an existing flow. The result is sent back to the negotiator.
- The negotiator collects all answers. The negotiation is only successful if all participants agree on the new specification. On success, the negotiator issues a “commit” to all participants, on failure, a “roll-back”.
- The MCF entities receive the result of the negotiation. Upon receiving a commit, they change to the new media characteristics and instantiate the new protocol. The new application level QoS and protocal parameter is then stored in the GMS.

2.6 Accommodating Heterogeneous Receivers

For some applications, not all receivers need to experience the same service quality. For example, the media characteristics of a video flow specifies a frame rate of 15 frames/s. If the processing resources in an endsystem allow only for a frame rate of 10 frames/s, it is still useful for this endsystem to join the flow, since every third frame can be discarded. Receivers that join a flow are therefore allowed to *downgrade* the media characteristics. When doing so, the other participants of the flow are not affected.

This kind of downgrading of media characteristics is mainly possible if local endsystem resources can not cope with the incoming media. Another problem arises if an application wants to join a video flow but the network only provides a part of the needed bandwidth. In this case, MCF allows the video flow to be split into multiple sub-flows. For instance, the video flow of 20 frames/s is transported in two sub-flows of 10 frames/s each (see Figure 3). Applications with enough bandwidth join both sub-flows and thus receive the full quality. However, they are forced to synchronize and possibly order the frames before displaying it. Applications with limited bandwidth join only one flow and thus receive only 10 frames/s. A generalization of this solution is given in [Shacham 92].



S: Sender
R1: Receiver getting full frame rate
R2: Receiver getting reduced frame rate

Figure 3: Flow splitting

3. System Architecture

3.1 MCF Components

MCF is a multicast extension to the Da CaPo (dynamic configuration of protocols) system which has been implemented at our lab [Plagemann 94/B][Gotti 94][Vogt et al. 93]. Its basic idea is to offer high flexibility and efficiency by configuring end-system protocols at runtime. Figure 4 gives an overview of the components that are used in an MCF entity.

- **QoS Mapper:** The QoS mapper translates application level QoS parameters to protocol level QoS parameters. It uses mapping functions that are specific for each media class.

- **Protocol Configurator:** The protocol configurator is used to configure a protocol that satisfies the protocol level QoS parameters. The result of the protocol configuration is stored in the GMS and used to instantiate the protocol.
- **Runtime Environment:** The runtime environment is responsible for initiating and executing a protocol. When a protocol is initiated, the runtime environment allocates the resources that are needed for the execution.

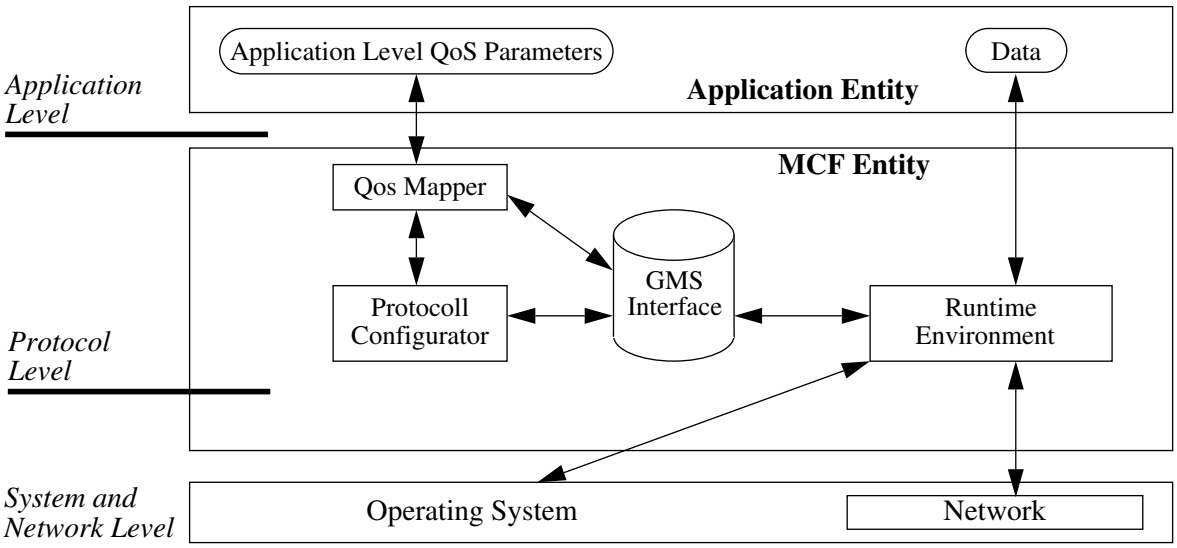


Figure 4: MCF Components

3.2 QoS Hierarchy

Three different abstraction levels can be distinguished in MCF: application level, protocol level and systen and network level. Each level comprises its own set of QoS parameters.

Application level QoS parameters are used by the applications to specify their requirements, as shown in section 2.3.

Protocol level QoS parameters describe throughput, maximum packet size, end-to-end delay, bit-error rate, packet error rate, and loss rate which are used for protocol configuration.

At the system and network level, parameters such as memory consumption, CPU usage and bandwidth are used to reserve the needed resources.

3.3 Protocol Configuration

The basis for protocol configuration (see [Plagemann et al. 94/A]) is a so-called *protocol graph*, which defines a set of protocol functions and their relations. Protocol functions are implemented in modules. Each module encapsulates a typical task such as error control, flow control or encryption and decryption. Modules are not limited to the classical protocol functions, they are also used for displaying video frames and playing audio samples. Each module is characterized by its effect on the protocol level QoS parameters. Normally, there are several modules available for a single protocol function. For example, error detection can be implemented using a CRC method or with a simple parity bit. A protocol is configured by selecting modules for each protocol function such that the protocol level QoS parameters are satisfied. The result of the protocol configuration is a *module graph* which can be executed by the runtime environment after it is initiated. Figure 5 gives an example of a protocol graph and a module graph. The abstract protocol functions like ‘Error Detection’ have been replaced by a specific module like ‘CRC8’.

The properties of a media class are reflected in its protocol graphs. In particular, the protocol graph also defines how heterogeneous receivers are to be handled. For example, the protocol graph for the receiver in the media class ‘video’ contains a filter function. Modules which implement the filter function can be instructed to throw away video frames when they are instantiated at the join operation.

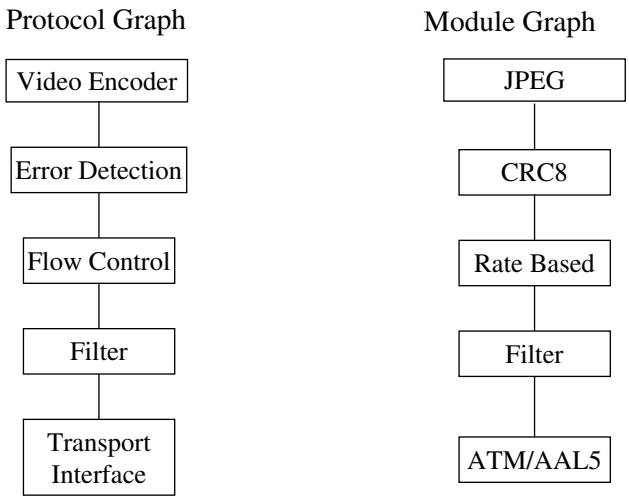


Figure 5: Protocol Graph and Module Graph

3.4 Protocol Instantiation and Protocol Execution

When an application entity joins a flow, its MCF entity gets the application level QoS parameters and the module graph for the flow from the GMS. The protocol described in the module graph is instantiated by allocating resources for the modules. If not enough resources are available to initiate the protocol, protocol instantiation fails. In this case, the only possibility for the application entity is to request a QoS renegotiation. After all the modules and resources are allocated, the protocol is executed by the runtime environment.

4. Conclusion

Existing communication frameworks support multipoint multimedia applications only insufficiently. They are inflexible and lack support for real-time multimedia.

In this paper, the design of a multicast commuication framework (MCF) is presented that allows multicast protocols to be configured according to the application's requirements. MCF flows are defined as multicast associations as seen by the application. They provide a high-level abstraction of the underlying services. This abstraction comprises independence of the network topology, generalized join and leave operations as well as application level QoS paramenters which are used to specify the characteristics of MCF flows.

The QoS parameters are used to compute resource usage and conduct resource reservation in both the endsystem and the network. Real-time multimedia streams can therefore be transported from multimedia source to multimedia sink with a guaranteed performance.

A fundamental component of MCF is the dynamic configuration of protocols, which has been successfully implemented in the Da CaPo project. Future work concentrates on the implementation of the MCF flow concept.

5. Acknowledgements

We would like to thank Bettina Bauer, Germano Caronni and Christian Conrad for their helpful comments and suggestions on this work. In particular, we like to thank Prof. Guru Parulkar and Dr. Thomas Plagemann for their help and encouragements.

6. References

[Beadle 95] H.W. P. Beadle: "Experiments in Multipoint Multimedia Telecommunication", *IEEE Multimedia, Summer 1995, pp. 30 - 40*

[Campbell et al. 94] A. Campbell, G. Coulson, D. Hutchison: "A Quality of Service Architecture", *Computer Communication Review, Vol. 1, No. 2, Apr. 1994, pp 6-27*

[Danthine et al. 92] A. Danthine, Y. Baguette, G. Leduc, L. Léonard: "The OSI 95 Connection-Mode Transport Service - The Enhanced QoS", *4th IFIP conference on high performance networking, Liège, December 1992*

[Dermler et al. 93] G. Dermler, T. Gutekunst, B. Plattner, E. Ostrowski, F. Ruge, M. Weber: "Constructing a Distributed Multimedia Joint Viewing and Tele-Operation Service for Heterogeneous Workstation Environments". *Proceedings, Fourth IEEE Workshop on Future Trends of Distributed Computing Systems, pp. 8-15. Lisbon, 1993*

[Ewald 95] J. Ewald: "Development of an API for Dynamic Protocols", *Diploma Thesis at the Computer Engineering and Networks Lab, Swiss Federal Institute of Technology, Zurich, August 1995*

[Ferrari et al. 92] D. Ferrari, A. Banerjea, H. Zhang: "Network Support for Multimedia - A Discussion of the Tenet Approach", *Technical Report TR-92-072, International Computer Science Institute, Berkeley, CA, October 1992.*

[Ferrari et al. 93] D. Ferrari, A. Gupta. "Resource Partitioning for Real-time Communication", *Proceedings of the First IEEE International Symposium on Global Data Networking, Cairo, Egypt, December 1993.*

[Gopalakrishnan et al. 95] R. Gopalakrishnan, G. M. Parulkar: "A Framework for QoS Guarantees for Multimedia Applications within an Endsystem", To be published in *GISI 95, Jahrestagung der deutschen Gesellschaft für Informatik und der Schweizer Informatiker Gesellschaft, Zürich, 18.-20. Sept. 1995*

[Gotti 94] A. Gotti: "The Da CaPo Communication System", *Technical Report, Computer Engineering and Networks Lab, Swiss Federal Institute of Technology, Zurich (http://www.tik.ee.ethz.ch/~dacapo)*

[Gupta et al. 95] A. Gupta, W. Howe, M. Moran, and Q. Nguyen, "Resource sharing for multi-party real-time communication", *Proc. IEEE INFOCOM`95, Boston, MA, April 1995.*

[Gutekunst et al. 95] T. Gutekunst, D. Bauer, G. Caronni, Hasan, B. Plattner: "A Distributed and Policy-Free General-Purpose Shared Window System". *IEEE/ACM Transactions on Networking, Vol. 3, No. 1, pp. 51 - 62. IEEE Computer Society Press, 1995*

[Mathy et al. 94] L. Mathy, O. Bonaventure, "QoS Negotiation for Multicast Communications", *International COST 237 Workshop on Multimedia Transport and Teleservices, Vienna, November 1994*

[Mauthe et al. 94] A. Mauthe, D. Hutchison, G. Coulson, S. Namuye, "From Requirements to Services: Group Communication Support for Distributed Multimedia Systems", *Proceedings of Multimedia: Advanced Teleservices and High-Speed Communication Architectures, second International Workshop, IWACW '94, Heidelberg, Germany, September 94*

[Metzler et al. 93] B. Metzler, I. Miloucheva, K. Rebensburg: "Multimedia Communication Platform: Specification of the Broadband Transport Protocol XTPX", *CIO Deliverable 14b, 1993.*

[Nahrstedt et al. 95/A] K. Nahrstedt, R. Steinmetz: "Resource Management in Networked Multimedia Systems", *IEEE Computer, May 1995.*

[Nahrstedt et al. 95/B] K. Nahrstedt, J. M. Smith: "The QoS Broker", *IEEE Multimedia, Spring 1995, pp. 53 - 67*

[Plagemann et al. 94/A] T. Plagemann, A. Gotti, B. Plattner: "CoRA - A Heuristic for Protocol Configuration and Resource Allocation", *IFIP Workshop on Protocols for High-Speed Networks, Vancouver, August 1994.*

[Plagemann 94/B] T. Plagemann: "A Framework for Dynamic Protocol Configuration", Dissertation no. 10830, Swiss Federal Institute of Technology, 1994

[Shacham 92] N. Shacham: "Multipoint Communication by Hierarchically Encoded Data", *IEEE INFOCOM'92, pp. 2107-14 vol.3, May 1992*

[Steinmetz 95] R. Steinmetz: "Analyzing the Multimedia Operating System", *IEEE Multimedia, Spring 1995.*

[Strayer et al. 92] W. T. Strayer, B. J. Dempsey, A. C. Weaver: "XTP The Xpress Transfer Protocol", *Addison Wesley, ISBN 0-201-56351-7*

[Vogel et al. 95] A. Vogel, B. Kerhervé, G. von Bochmann, J. Gecsei: "Distributed Multimedia and QoS: A Survey", *IEEE Multimedia, Summer 1995, pp. 10 - 18*

[Vogt et al. 93] M. Vogt, B. Plattner, T. Plagemann, T. Walter: "A Run-Time Environment for Da CaPo", *Proceedings INET'93.*

[Wilde 95] E. Wilde: "Group Management and Communication Support for Collaborative Applications". *Internal Report, available from the author.*

[Zhang et al. 93] Lixia Zhang, Stephen Deering, Deborah Estrin, Scott Shenker, Daniel Zappala: "RSVP: A New Resource ReSerVation Protocol". *IEEE Network Magazine, Vol. 7, No. 5, Sept. 1993. pp. 8-18*

[Zitterbart et al. 93] M. Zitterbart, B. Stiller, A.N. Tantawy: "A Model for Flexible High-Performance Communication Subsystems", *IEEE Journal on Selected Areas in Communications, May 1993, Vol. 11, No. 4*