Levels of Quality of Service in CINEMA

Ingo Barth, Gabriel Dermler, Walter Fiederer

University of Stuttgart
Institute of Parallel and Distributed High-Performance Systems (IPVR)
Breitwiesenstraße 20-22, D-70565 Stuttgart, Germany
{barth, dermler, fiederer}@informatik.uni-stuttgart.de

Abstract. Distributed multimedia systems require quality of service at various abstraction levels. At the application level, QoS describes communication in terms of application specific processing units, e.g. frames. At the transport level QoS is used to describe the communication load for the transport system. An architecture integrating the various levels is presented. The architecture provides the framework for performing negotiation of QoS and resource reservation. It includes the specification of a programmer interface and a flowspec carrying QoS information between communication nodes.

1 Introduction

Distributed multimedia systems require resource reservation and quality of service (QoS) handling because of scarce resources ([ATW+90]). In the past, substantial work has been done in the field of QoS handling within transport systems and networks. QoS specification at this level is derived from the need of transport systems to handle communication load in an application independent way. Negotiation between application entities using the transport system is supported by passing higher level QoS between them. End-to-end in the context of a transport system applies to the communication endpoints of the transport system itself.

Distributed multimedia applications expose further reaching requirements. First, distributed application entities have to agree on communication characteristics beyond transport level communication load. Application level parameters are required describing media specific QoS properties of communication streams. Such parameters are required since there is no unique coupling between a specific communication load and specific value selections for media specific parameters (e.g. small picture/high video rate and large picture/low video rate may yield the same communication load). Also, media specific parameters allow application users to express their QoS requirements in a more natural way.

Second, end-to-end in the context of distributed multimedia applications does not necessarily mean a communication link connecting two end-systems. Generally, it means the interconnection of several endsystems in a possibly very application specific topology, as is the case for the connection of several audio sources to a central mixer from which distribution takes place to a number of networked loudspeakers. Such an application will require appropriate (negotiation) control mechanisms considering both transport and application level parameters and spanning the whole application topology from data sources to sinks.

CINEMA, a configurable integrated multimedia architecture, is a system attempting to identify and solve these problems. It was conceived as a development and run-time support platform for distributed multimedia applications. It supports QoS handling at both application and transport level and integrates these levels into an appropriate QoS architecture. The description of this architecture is the topic of this contribution. It is to highlight the main abstractions used in *CINEMA* for constructing and running distributed applications and the way these abstractions handle

QoS at various levels including mapping between the levels. In particular, this includes a client QoS specification interface and a flowspec description for negotiating QoS between distributed parts of *CINEMA*.

Unlike transport level QoS, application level QoS is different depending on the type of media employed by a distributed application. *CINEMA* introduces a scheme for defining application level QoS based on the notion of stream types. This scheme is also presented in this paper and described with respect to QoS handling.

The paper is structured as follows. First, we give a brief overview of related work. In Section 3, we present the abstractions of the *CINEMA* system in which the QoS architecture is integrated. This is followed by the QoS architecture containing the different QoS abstraction levels and mappings between them. Section 5 describes the QoS description used for the application level. Finally we discuss and summarize the presented QoS architecture.

2 Related Work

Substantial work has been done on multimedia transport systems and description for QoS to negotiate the possible QoS within the transport or network system and to reserve the needed resources. For instance HeiTS ([HHS+91]) provides an interface to specify the required QoS for a transport connection. Within HeiTS, QoS is negotiated and resources are reserved. The QoS specification of a transport system is mostly derived from the QoS specification which is given by the underlying network level reservation protocols, e.g. SRP [AHS90] or ST-II [Topo90]. Another popular reservation protocol RSVP [ZDE+93] defines no QoS specification so far.

[TaHo94], [SMHW95] and [KrLi94] describe the need for application specific QoS. Their QoS description is based on the media type of a stream. [TaHo94] uses a management information base to map application specific QoS to transport system QoS. [SMHW95] uses a QoS manager tool to do the mapping. The QoS manager tool is called with the selected kind of media encoding and the desired QoS class and returns the transport level (XTPX-based) QoS data structure. [KrLi94] uses a single application parameter which can be mapped onto transport level parameters. In this paper, we motivate a different approach for the mapping from application specific to transport system QoS.

In IMA [IMA93], QoS is declared as format descriptions which are assigned to ports. The IMA system selects an appropriate format for each connection. This format selection is done for each end-system to end-system connection on its own. An end-to-end QoS handling is not possible at application level (i.e. from data sources to sinks).

A quality of service architecture (QoS-A) for a transport service over high-performance ATM-based networks is explained in [CCH94]. The defined architecture contains three planes, the protocol plane, the QoS maintenance plane, and the flow management plane. The description solely relates to the transport layer and the mechanisms to realize such a transport interface.

3 *CINEMA* System

In this section, we describe the basic abstractions of the *CINEMA* system allowing clients to construct distributed multimedia applications. For a detailed description refer to [RBH94] and [RoHe94]. The basic abstractions are components, ports, links, clocks and sessions.

Components encapsulate processing of multimedia data, e.g. for generating, presenting or manipulating data. To provide a uniform data access point for the components, ports are used that provide data units to the component (input port) or take the data units from the component (output port). A client constructs an application by specifying a topology of components inter-

connected via links. A link provides an abstraction from underlying communication mechanisms which may be used to perform the transport of data units.

Components are activated by the system to handle data units of data streams taken from input ports and produce data units at their output ports. The timing of the component is controlled by the system using clocks. A clock maps a media time system to the real time according to the specified clock parameters.

With each component port a set of stream types is associated indicating the media types supported by the component at that port. The stream type concept will be detailed in section 5.

Before using an application, a client has to indicate desired QoS to *CINEMA*. For this, *CINEMA* offers the concept of a session. A session is the unit of resource reservation allowing the client to specify the (parts of an) application to be instantiated and the QoS expected. Section 4 details the client's view to QoS specification.

Figure 1 shows an example topology composed of the mentioned core abstractions: two source components generating two audio streams which are mixed by a central mixer and distributed to two loudspeaker components.

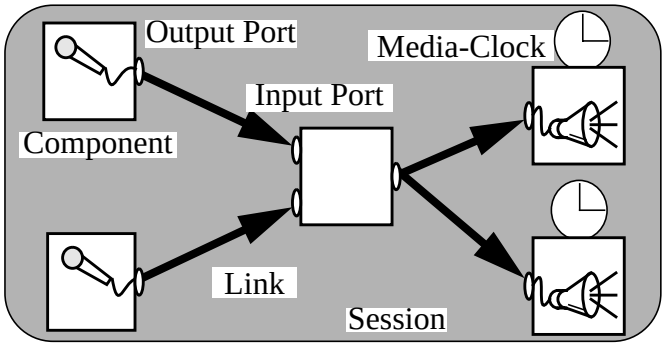

Figure 1 : Example of a CINEMA *multimedia application topology*

4 QoS Architecture

This section describes the QoS abstraction levels used in *CINEMA*, the mappings between them and the architectural aspects of the *CINEMA* QoS negotiation scheme.

4.1 Transport and application level QoS

Since multimedia streams exhibit a strong relationship with time, new transport systems have been designed which are able to handle massive multimedia related communication load within temporal bounds. For providing such guarantees, they need a description of the communication load. Based on this, they can determine the amount of resources required. Deriving such load descriptions is therefore an important task for each distributed multimedia system (see Section 4.2).

[ATW+90] introduced such a description with the LBAP model. Other known transport systems use similar descriptions which differ in some detail, but typically contain the size of data units, rate of data units, and the burstiness of data units. Depending on the capabilities of the transport system error control characteristics are included as well.

The description of QoS is independent of the type of data contained in a continuous stream. For instance, the parameters for a video stream are the same as for an audio stream.

Distributed application components have to exchange data according to commonly agreed media specific parameters. Examples of such parameters for video streams are video size and rate, color depth and compression quality. For audio, sample size and rate are commonly used. *Cinema* uses the concept of stream types to encapsulate the parameter set implied by a specific medium.

Application level QoS parameters are not just used to describe media specific communication between component ports. They are also the parameters employed by the application client when specifying desired QoS for a session. In this way, *Cinema* offers a basic level of QoS specification. On top of this, more abstract QoS layers may be built, which, however, are outside the scope of *Cinema*. For instance, a *Cinema* client may specify for a stream type "video" the tuple (video size, video rate) as required at a sink component's port.

Initially (i.e. before a session is set up by a client), each component port is associated with one or more stream types which can be supported by the component functionally. For instance, a source component may be able to support two stream types JPEG video and MPEG video. Each of these could allow the specification of values for parameters such as video size, video rate and compression quality. For each parameter values are specified which are supported functionally by the component, either as a list of discrete values (e.g. CIF, QCIF) or as a value range between a minimal and maximal value.

It is the task of the *Cinema* negotiation procedure to determine for each port which of possibly several stream types have to be supported in a session and with which parameter values.

4.2 Mapping from application level QoS to transport level QoS

The mapping is required to calculate the transport level QoS from application level QoS. The application level QoS is selected initially by the client and propagated through the topology by the negotiation protocol.

Several approaches to perform the mapping are possible. Taking the approach from [TaHo94] or [SMHW95] in *Cinema* would have meant to define an additional mapping entity containing entries for each stream type supported and defining corresponding mappings to transport level QoS.

We expect that a *Cinema* component consuming or producing data streams of a certain stream type has anyway to be aware of the characteristics of the stream type in order to be able to process it. In addition, the component must have the ability to compute its resource demands from a given application level specification. From this we concluded that a component in most cases has to possess the knowledge to do the mapping. Therefore we anchored the mapping functionality inside a component and did not assign it to a separate mapping entity.

A second reason for this approach was that in some cases the processing capabilities of the components influence the mapping. For instance, a video sink which is able to fill gaps in a video image will require less error prevention at the transport level than a video sink which cannot fill gaps.

We have introduced the abstraction of a link to relieve a client building an application from the need to be exposed to transport system specific questions. The same has to apply to component programmers as well. When defining the mapping to transport level QoS inside a component, the component programmer should not bother with features of a specific transport mechanism. Therefore we introduced an application flowspec (AFS) that contains a specific part carrying the application specific QoS parameters and a generic part carrying the communication load parameters which abstract from peculiarities of any employed transport mechanism. This generic part

consists of parameters commonly used in transport systems like size, rate and burst of data units, delay, jitter and error rate.

In order to do the mapping from the communication load parameters to the QoS parameters of a specific transport system, which are carried by the transport flowspec (TFS), each transport system is encapsulated within a specific link. Whenever a new transport system is introduced, a new link has to be realized in *CINEMA*. A link offers to *CINEMA* the desired uniform communication load oriented interface while linking to the transport system through its specific transport interface. We expect that several transport systems will coexist in the near future in multimedia systems, for instance for adapting to wireless or wirebound communications. On the other hand, we do not expect this number to be too high and transport systems to appear too often. The list of required links will therefore be rather low and static. Links are inserted by *CINEMA* in a transparent way to the client. The selection of the link is done by *CINEMA* based on the type of transport system.

Figure 2 shows the QoS architecture for communicating multimedia data using the *CINEMA* abstractions. At the application level and at the transport level flowspecs are defined containing the corresponding QoS information. These flowspecs are propagated during the negotiation phase, the AFS between components and links of an application, the TFS inside links.

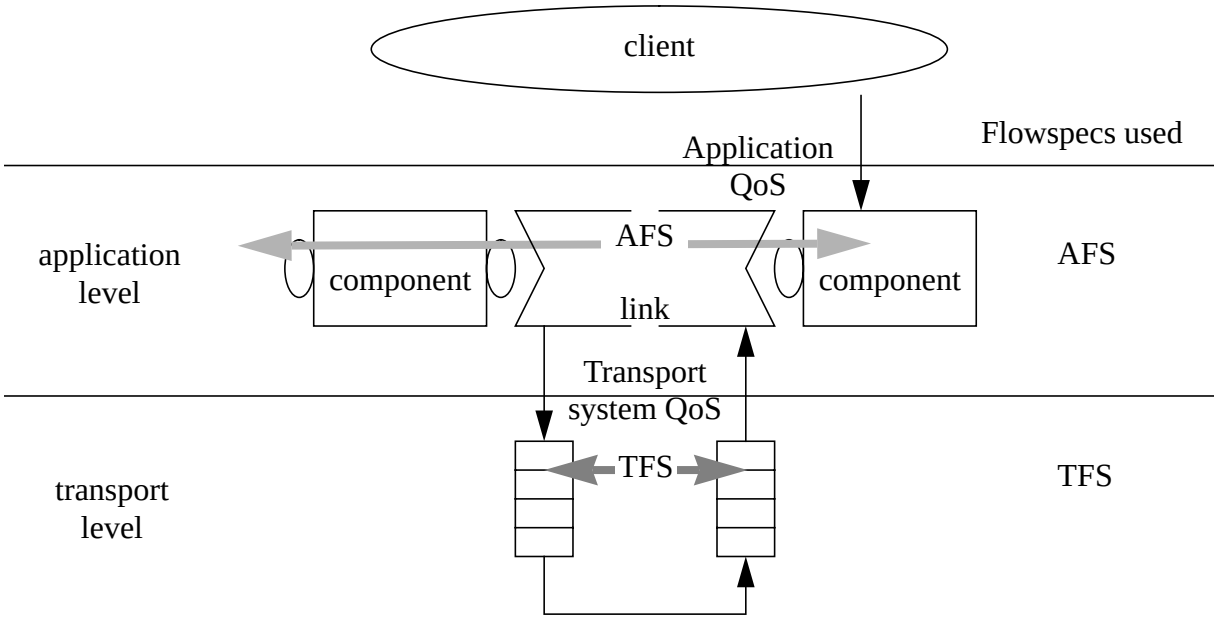

Figure 2 : QoS Architecture used in CINEMA

4.3 Negotiation of QoS

A *CINEMA* client wishing to initiate a session, first has to specify the QoS it desires. The client relates this specification to perceivable quality, i.e. to the quality of the final visible presentation, e.g. the quality of a video viewed. Usually, this quality can be described at the sinks of a topology.[1] In *CINEMA* the client is expected to provide a QoS description for each sink port.

Before QoS specification, the client can query the port for possible stream types the port can support and selects an appropriate one. Since a stream type can allow various parameter settings (see Section 5), the client has also the opportunity to exclude values (or value ranges) which are allowed by the stream type, but not desired by the client.

---

1 Only these cases are considered here.

After having done the selections described, the client initiates a session. The client has to specify which part of a topology has to be built up and (for each sink port separately) what end-to-end parameters have to be applied to the session. Then, the *CINEMA* negotiation process is triggered.

The next step is to assemble a flowspec (application flowspec, AFS) from the QoS specification given by the client and completed by sink components. This flowspec is propagated by *CINEMA* through the interlinked topology. Whenever the AFS traverses a transport system, the generic part of the AFS is used to calculate the corresponding transport system flowspec by the corresponding link.

From a point of the component topology view, the propagation takes place at the link, i.e. *CINEMA* takes the AFS from a component's incoming port and passes it to the link connected to that port. The link passes the (possibly adjusted) AFS back to *CINEMA* in the context of the outport originating that link. In turn, *CINEMA* passes this AFS to the component carrying the outport, such that the component can calculate (a) the AFS of its incoming ports and (b) the resources required. The procedure is repeated by *CINEMA* until all component ports were given an appropriate AFS.

From an architectural point of view, the propagation takes place - as seen in Figure 2 - at the application level (containing components and links) and at the transport level. Whenever a link is invoked with an AFS, it checks whether it serves local or remote communication. For the latter, the link calculates a transport system flowspec from the generic part of the AFS. TFS is handed over to the transport system, such that it can initiate resource reservation at the transport and network level. The AFS part is handed over as transport user data. At the other end of the transport connection, AFS and the (possibly) adapted TFS are assembled by the destination part of the link by adjusting the generic parts of AFS. The link passes this AFS to the *CINEMA* system for further propagation at the application level.

We described here the negotiation procedure only as far as architectural issues concerning QoS are touched. A detailed description of the *CINEMA* QoS negotiation and resource reservation protocol is outside the scope of this paper and will be given in [BDFR95].

5 Stream Types

Application level QoS is organized based on the notion of stream types. A stream type consists of different parameters, which describe characteristic features of the medium. For each parameter the semantics is defined by the stream type. The stream type hierarchy allows the definition of stream types that have the same parameters with the same semantics but with different sets of possible values. A stream type can be looked at as a class definition in an object-oriented environment. A QoS description for a media stream is then an instantiation of an element of the given stream type. According to the object-oriented approach we can derive a new stream type out of an existing one by specializing the values for one or more parameters. To assure that the parent stream type can always be used together with the derived stream type we must assure, that all parameter values from derived stream types must be part of the possible parameter values from the parent stream type. Furthermore, derived stream type must not have a parameter that the parent stream type has not.

This leads to stream type hierarchies where a base type defines the possible parameters and all derived stream types restrict the possible values for parameters. With these hierarchies two ports can be connected when the stream types are equal or when one stream type is above the other in the hierarchy. The resulting stream type is always the more specialized one.

The base stream type is defined by declaring the stream dependent parameters. A new stream type is derived from an existing one by restricting possible values for at least one parameter. All non restricted parameters have the same range than the parent stream type. For every parameter

there must be at least one possible value. If a parameter is restricted to only one possible value, this parameter cannot be negotiated with this stream type.

With this stream type definition we allow the declaration of QoS parameters within the boundaries given by the stream type. This allows us to use one concept for stream type definition and QoS definition and therefore it is easy to use when programming components. The type of QoS definitions is always based on the used stream type. When two components use different but compatible stream types for their ports they can be connected. When one stream type allows multiple values for a parameter and the other specializes it to exact one value, this parameter is automatically set to the right value. If we would distinguish between QoS and stream type, then one component would have a QoS parameter that the other one would not have and therefore the QoS descriptions would differ.

As an example, we can assume a microphone component which can produce audio streams with different sampling rates and a speaker component which can only work with a special sampling rate. The following example defines a base stream type `Audio` which can have different rates for sampling and samples with different sizes. This stream type is specialized by the second definition to have only different sizes for samples but only one rate. This stream type is called `Audio8kHz`. Another stream type is derived from `Audio`, where the sample size is specialized to 16 Bit. This stream type is called `Audio16Bit`. Figure 3 shows the stream type hierarchy built out of these definitions.

```
 1  STREAMTYPE Audio {
 2    int SampleRate = { 8000, 11025, 22050, 44100 };
 3    int SampleSize = { 8, 16 };
 4  };
 5  STREAMTYPE Audio8kHz : Audio {
 6    SampleRate = 8000;
 7  };
 8  STREAMTYPE Audio16Bit : Audio {
 9    SampleSize = 16;
10  };
```

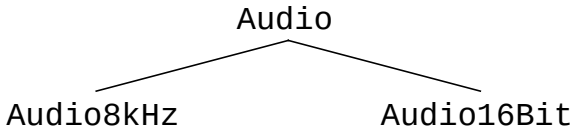

Figure 3 : Example of a stream type hierarchy

6 Discussion and Summary

A QoS description which is applicable for a user must be based on the type of media. To negotiate this QoS application specific information is needed. Within the communication subsystem the QoS is not depending on the media type but on a generic stream description handling with buffers, rates and bursts. To bring these different QoS descriptions together we need a QoS architecture containing both abstractions and the mapping between them.

We presented a QoS architecture that allows the user to specify his presentation quality in a media oriented way. An application flowspec (AFS) is introduced, that contains an application specific and a generic part of QoS parameters. The mapping from application level QoS to the generic part in the flowspec is performed by the stream handling components which can include the influence of their implementation details into the mapping. This AFS can be mapped in a transport specific link to the needed transport system flowspec, that is used for resource reservation in the network.

For defining the application level QoS, we introduced the concept of stream types which act like a class definition for a QoS definition. The stream type hierarchy gives an easy way to check for type compatibility and to provide a method to define subtypes from a stream type.

The integration of QoS handling based on this QoS architecture into the *CINEMA* system has just begun. Together with resource management and a QoS negotiation protocol for application QoS, which is in development at the moment, an easy way to build up distributed multimedia applications with guaranteed QoS will be available with the *CINEMA* system.

7 Bibliography

[AHS90] David P. Anderson, Ralf Guido Herrtwich, Carl Schaefer. SRP: A Resource Reservation Protocol for Guaranteed-Performance Communication in the Internet. Technical Report No. UCB/CSD 90/562, Computer Science Division (EECS) University of California, Berkeley, CA, 2 1990.

[ATW+90] D. P. Anderson, S.-Y. Tzou, R. Wahbe, R. Govindan, M. Andrews. Support for Continous Media in the DASH System. In *Proc of the 10th International Conference on Distributed Computing Systems*, pages 54–61, 5 1990.

[BDFR95] Ingo Barth, Gabriel Dermler, Walter Fiederer, Kurt Rothermel. A Negotiation and Resource Reservation Protocol (NRP) for Configurable Multimedia Applications. *to be published* 1995.

[CCH94] Andrew Campbell, Geoff Coulson, David Hutchison. A Quality of Service Architecture. *ACM Computer Communication Review*, 24(2):6–27, 4 1994.

[HHS+91] D. Hehmann, R. G. Herrtwich, W. Schulz, T. Schuett, R. Steinmetz. Implementing HeiTS: Architecture and Implementation Strategy of the Heidelberg High-Speed Transport System. In *2nd Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, 11 1991.

[IMA93] Hewlett-Packard Company and International Business Machines Corporation and SunSoft Inc. *Multimedia System Services, Version 1.0, available via ftp from ib minet.awdpa.ibm.com*, 7 1993.

[KrLi94] A. Krishnamurthy, T.D.C. Little. Connection-Oriented Service Renegotiation for Scalable Video Delivery. In *Proc of the International Conference on Multimedia Computing and Systems*, pages 502–507, 5 1994.

[RBH94] Kurt Rothermel, Ingo Barth, Tobias Helbig. *Architecture and Protocols for High-Speed Networks*, Chapter CINEMA - An Architecture for Distributed Multimedia Applications, pages 253–271. Kluwer Academic Publishers, 1994.

[RoHe94] Kurt Rothermel, Tobias Helbig. Clock Hierarchies: An Abstraction for Grouping and Controlling Media Streams. *Technical Report 2/94, University of Stuttgart*, 4 1994.

[SMHW95] A. Schill, C. Mittasch, T. Hutschenreuther, F. Wildenhain. A Quality of Service Abstraction Tool for Advanced Distributed Applications. In *Proc of the International Conference on Open Distributed Processing*, 2 1995.

[TaHo94] Wassim Tawbi, Eric Horlait. Expression and Management of QoS in Multimedia Communication Systems. *Annals of telecommunications*, T.49(5-6):282–296, 5-6 1994.

[Topo90] C. Topolcic. Experimental Internet Stream Protocol, Version 2 (ST-II). *RFC 1190*, 10 1990.

[ZDE+93] Lixia Zhang, Steve Deering, Debora Estrin, Scott Shanker, Daniel Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network*, pages 8-18, 9 1993.