Quality of Service Mapping in Distributed Multimedia Systems

Stefan Fischer and Ralf Keller

University of Mannheim
Praktische Informatik IV
D-68131 Mannheim, Germany
{stefis, keller}@pi4.informatik.uni-mannheim.de

Abstract

Quality of Service (QoS) is a key concept in today's high speed communication networks. New applications, such as distributed multimedia systems, need powerful constructs to express their performance and quality needs. The existing set of quality of service parameters is not suitable for this purpose.

On the other hand, application programmers want easy-to-use QoS parameters to simplify their work and human users expect meaningful QoS parameters to express their needs.

We describe the development path for a new view on quality of service followed since the beginning of the nineties and present an overall QoS architecture. Then, we concentrate on the application layer and reflect on translation of QoS parameters from the application's point of view into transport and operating system QoS parameters. Finally, we report on the implementation of these new concepts in XMovie, an existing multimedia system, and specify some of the mappings of application specific QoS parameters to transport QoS parameters in detail.

Keywords: quality of service, multimedia systems

1 Introduction

With the advent of high speed networks, *Quality of Service* was recognized as one of the key concepts in distributed systems. The high-performance characteristics of the FDDI, DQDB or ATM networks enable a new generation of applications for which the standardized protocols and services no longer suffice. Especially in the area of service quality, the new applications have no mechanism to express and communicate their needs to the lower layers. Much work still needs to be done here.

In this paper, we show how to integrate a QoS architecture into XMovie, an existing *multimedia system*. Multimedia applications are often considered to be the most important representative of the above-mentioned new generation of applications. We have gained considerable experience with the implementation of such a system and are convinced that it is quite important to provide mechanisms for QoS handling, and in particular for mapping application-level QoS parameters to communication subsystem and operating system QoS parameters.

This paper is organized as follows: In Section 2, we give a brief overview of QoS history in recent years. Our systemwide QoS architecture integrates many of the main results achieved so far. Since much work has already been done in the lower layers, we concentrate in Section 3 on the application layer. We discuss how different quality of service requirements of applications may be expressed and how they could be mapped to the lower layer interfaces. In Section 4, we show how we integrate the QoS architecture into our XMovie system and report on experiences. In addition, we present some detailed formulas for QoS parameter mapping. Section 5 concludes the paper.

2 Quality of Service: Architectural Considerations

The term *Quality of Service* was already coined before the advent of high speed networks. The OSI Basic Reference Model defines quality of service parameters for the transport subsystem [10]. The values for the parameters are negotiated between the service user and the service provider, but no statement exists on what would happen should the provider fail to fulfill the agreed values. In addition, the set of parameters is incomplete for today's application needs.

The following provides a brief overview of the work done in different areas to improve this situation. It is organized along the layers in a communication system. Efforts have been made mainly in transport, network, MAC and application layers as well as in operating systems and network nodes.

Transport Layer. Hehmann et al. [9] argued that the ISO set of quality of service parameters for transport systems was not suitable to fulfill all the needs of transport service users. The set initially proposed by ISO [10] contained *throughput, delay, residual error rate, resilience* and *transfer failure probability*. They suggested adding both a *delay jitter* (the variance in delay) and a *burstiness* (the ratio between maximum and average data rate) parameter. Many applications need an isochronous service characterized by a small jitter value. Network resources can be better used if the burstiness of the workload produced by a data stream is known in advance. Hehmann et al. also split up the existing residual error rate into bit error rate and packet error rate.

Another important observation was that transport service users are not satisfied with the semantics of QoS parameter negotiation. A transport service provider as defined in ISO/OSI has no obligation to guarantee the negotiated values [10]. The service is therefore called *best-effort* (e.g. in [3]). Several new semantics have been developed, see [1, 2, 3, 7, 17]:

- The *guaranteed* or *deterministic* service defines a strict commitment by the service provider to the negotiated service. Once agreed, values will be kept by the provider. A typical guarantee would be "The minimum throughput will be 5 MBit/s throughout the lifetime of the connection."
- The *statistical* service offers stochastic guarantees on groups of service data units, i.e. "85% of all SDUs will have a delay less than 20ms". Thus, this service offers a weaker commitment.

Both service semantics may be further subdivided into *compulsory* and *threshold* services. In the former, the service provider interrupts the connection as soon as it is no longer able to keep its commitment. In the latter, service degradations do not lead to an interruption but to an indication to the user who may then decide how to handle them.

Another important feature is the possibility of renegotiating an established connection, e.g. in the event of service degradation [16].

Network Layer. The task of the network layer (with respect to QoS) is to provide the means to implement the service semantics defined for the transport service. The most important approach is the reservation of resources in the network (e.g. [6, 8]). Algorithms have been developed to limit delay and jitter, and to guarantee a certain throughput on internetwork connections. The parameters of this service are nearly the same as those for the transport service. Whereas the reservation of resources is natural in connection-oriented networks where all intermediate nodes store state information anyway, it is much harder to implement resource reservation in connectionless networks such as IP [29]. Work is in progress to extend IP to enable isochronous data flows [5, 29].

MAC Layer. The Medium Access Control Layer enabled new developments in the QoS area. FDDI, DQDB and ATM networks are able to provide high data rates. FDDI networks are capable of supporting both synchronous and asynchronous traffic. An important feature of ATM is the possibility to transfer data streams isochronously. The possibilities offered by ATM are tailored to application requirements by ATM adaptation layers, where user or transport QoS parameters are mapped to hardware–oriented notions such as *cell loss priority, cell insertion ratio* or *cell transfer delay* [4].

Operating System. The realization of a certain QoS, especially in the layers implemented in software, requires not only the support of the respective lower layers but also operating system support. The main points here are CPU scheduling, memory management for buffering, and efficient file storage on mass media. Real–time CPU scheduling is discussed intensively in the literature [22, 23, 27]. The aim of buffer management is to avoid jitter and to minimize copy operations [27]. This may be achieved by a large number of buffers, which, however, may in turn lead to a waste of resources. Efficient file storage is achieved by storage hierarchies [21] and disk striping. Important QoS parameters for the operating system are *size* and *number of buffers*, *scheduling class* (e.g. real-time, timesharing etc.), *priority*, and *number of CPU cycles*.

Application Layer. So far, little work has been performed on QoS from the application's point of view.[1] There is some agreement that applications need a set of parameters [28] and a value domain [30] different from those for services in the lower layers. Applications should not be forced to specify a value for *jitter*, for example, as they have no understanding of such technical terms.

To allow the provision of a certain QoS, the parameters of one layer have to be mapped to those of other layers and of system resources. The resulting QoS architecture is shown in Fig. 1. Research so far

[1] To be more precise, both the application users' and the application programmers' point of view need to be analysed.

has focused mainly on lower layers, and the mapping of transport to network or of ATM adaptation layer to ATM layer [25] is quite well understood. The next section addresses the translation of application-specific QoS parameters to the transport and operating system interface.

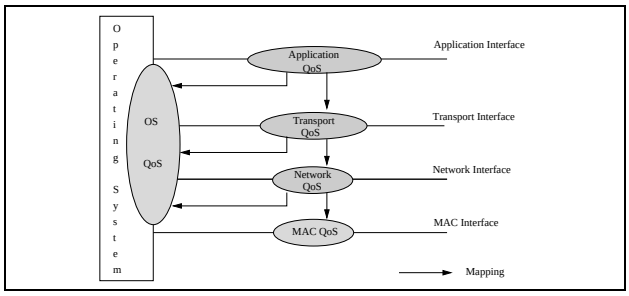

Figure 1: A Quality of Service Architecture

3 Application QoS

Unlike the set of transport QoS parameters, which are now well defined, the set of QoS parameters used in the application layer is an open set. On the one hand, there is a set of standardized application units[2] like FTAM in the OSI application layer or ftp in the Internet application layer. Each of these provides a defined set of QoS parameters which are normally negotiated during connection establishment. These parameters may be set by other application units or by the application itself. On the other hand, there may be, for a new application, a different set of parameters. Thus, we cannot give the exact number and types of all possible application QoS parameters.

On the way down to the network, these parameters have to be mapped to the transport and operating system parameters. So, for each newly defined parameter, a mapping has to be given either to existing parameters of the application units used or to the transport and operating system parameters.

From the user's point of view, a large set of parameters is unacceptable and normally useless. Users do not want to specify numerous parameters which are often meaningless to them, such as delay jitter or cell loss ratio. In addition, they have no need to give exact values for certain parameters. A frame rate of 16 frames per second will look quite similar to a frame rate of 14 fps. Therefore only a small set of meaningful parameters should be offered to the user.

[2] Application units can be used directly to build applications, or to build other application units.

One of the most important parameters from a user's point of view is the *quality* parameter. A user will always want to get the best quality available. If good and bad quality can both be provided by the underlying system, he will chose the better one.[3] If the system can only deliver bad quality, the user may decide whether to accept it or not.

Actually, we think that *cost* is an equally important parameter. However, since today's high-speed networks currently have no cost notion, we omit this parameter in our discussion.

Thus, we have a relatively small set of QoS parameters at the application and transport layer interface, while inside the application layer, complex sets of parameters may exist. The user parameters have to be mapped onto the transport parameters using the parameters of the application units.

Let us look at an example of this mapping process (see Fig. 2) and consider a multimedia news application. Various continuous media streams like video and audio streams (e.g. telephone reports) and discrete media like text or images are combined in this application.

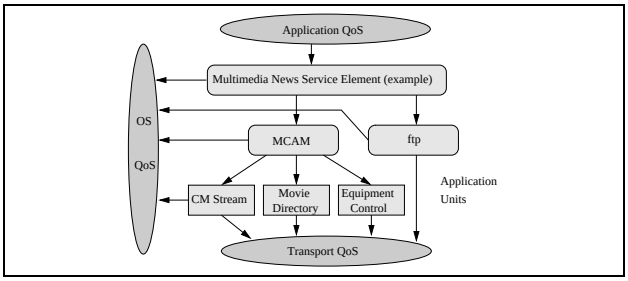

Figure 2: Quality of Service in the Application Layer

A user of this application wants to be informed about a special event which he/she specifies by some keywords like date, name of the event, place etc. He may also determine whether he wants a cheap or expensive service, when available, and whether he is ready to accept a poor quality if the best is currently not supported by the system.

These specifications are to be mapped to the MCAM and ftp service elements. MCAM (Movie Control, Access, and Management) is the application layer protocol of our XMovie project which provides support services to the user [12, 15]. The most important of these support services is the continuous media (CM) stream service allowing isochronous transmission of

[3] Provided that the fees for the higher quality are acceptable for him.

CM streams over high-speed networks. In addition, MCAM integrates movie directory and CM equipment control as support services. The movie directory is used as a repository for movie information, such as digital format and storage location. The equipment control service enables the user to control CM equipment attached to local or remote computer systems, e.g. speakers, cameras, and microphones.

MCAM determines the source of the requested stream (typically audio or video) by the use of the movie directory application unit. Then, MCAM knows format, encoding and some other parameters and can tailor the CM stream application unit to its needs by calculating the period representing the validity interval of a single stream unit, e.g. a frame. The CM stream application unit determines by itself the transport QoS, especially throughput, delay and delay jitter, taking into account the processing overhead introduced by the application layer.

The next section shows how we implemented this general application QoS architecture into the existing XMovie system. We concentrate on the mapping process *inside* the application layer.

4 Example: QoS in XMovie

XMovie is an experimental multimedia system developed at the University of Mannheim [14]. Within XMovie we have defined MCAM with its three support services. For the sake of brevity we concentrate on the CM stream part of MCAM. All other parts of MCAM have their own QoS requirements, e.g. response time of the CM equipment control service, that are not discussed here.

4.1 QoS Parameters

MCAM allows detailed control of movie transmission. To tailor the general service of MCAM to the specific needs of an application, a set of QoS parameters is provided to the application programmer (see Table 1). Parameter values are restricted to the stated domains. Since these "raw numbers" are often not handy, we provide predefined qualifiers for the parameters. If the requirements of an application cannot be met by the predefined qualifiers, other values are allowed. However, not all values might be valid; for example the MPEG audio standard defines self-contained frames of samples which should not be subdivided [11]. Since the relations between the QoS parameters are quite complex, the process of specifying application-specific QoS requirements is rather complicated, even in the limited scope of the CM stream service.

The *period* parameter determines the "speed" of the stream. To playback a movie at 25 frames per second, a period of 40 ms has to be specified. *Very fast*, *fast*, *normal*, *slow*, and *very slow* are typical examples of predefined qualifiers. If a movie is recorded and stored to disk at n frames per second, then n is the normal speed of this movie.

Parameter	Domain	Qualifiers
period	milliseconds	very fast, fast, normal, slow, very slo
quality	integer	very high, high, medium, low, very lo
reliability	percent	very high, high, medium, low, very lo
delay	milliseconds	minimal, default
start offset	milliseconds	minimal, default

Table 1: CM stream QoS parameters

The *quality* parameter is used to control the visual and/or audible quality of the CM stream. For example, in JPEG the Q factor controls the quality of a picture [24]. In addition, the quality parameter influences the usage of operating system resources. For example, to obtain a high-quality presentation, jitter in the CM stream has to be eliminated using memory buffers.

The *reliability* parameter enables the user to specify how reliable the transmission and presentation should be. For example, if high reliability is requested, no data should be lost. For both the quality and the reliability parameters we provide *very low*, *low*, *medium*, *high*, and *very high* as qualifiers. Both *very low* and *low* qualifiers are normally not requested by the user, but the system might provide only such a kind of service.

Both *delay* and *start offset*[4] parameters accept values in milliseconds. Typical qualifiers for these parameters are *minimal* and *default*. Therefore either a minimal delay (or start offset, respectively) or MCAM's default behavior can be requested.

The QoS parameters of the multimedia news application (see Sect.3) have to be mapped to the MCAM QoS parameters or, to be more precise, to the CM stream QoS parameters, which in turn have to be mapped to transport QoS parameters.

If, for example, a user of the Multimedia News Service requests a reliable service with medium quality, the CM stream QoS parameter set could be as follows: {period = normal, quality = medium, reliability =

[4] By means of the start offset parameter sender and receiver synchronize themselves at the start of the CM stream. For example, a requested stream is stored on a magnetic tape, and the locating and mounting takes a few seconds. Therefore the receiver has to know the stream starting time and can then decide if this time is acceptable.

high, delay = minimal}. For the parameters not specified explicitly, the system assumes the default values.

4.2 Mapping to Transport QoS

Table 2 shows the transport parameters available in our system. The CM stream service is currently built on top of the Internet protocols TCP and UDP. Since only the MTU (Maximum Transfer Unit) parameter of those in Table 2 can be directly controlled in the Internet protocols, we implemented an "adaptation layer" that offers a richer service interface.

Parameter	Domain
throughput	bytes per second
MTU	bytes
reliability	percent
burstiness	integer
delay	milliseconds
jitter	milliseconds

Table 2: Transport QoS parameters

The mapping of the above-mentioned QoS parameters to transport QoS parameters is depicted in Fig. 3. The properties of the requested CM stream influence all QoS mappings: The amount of data to be transmitted and processed in each period and the number of CPU cycles needed depend on the format of the CM stream. As an obvious example, MPEG streams [19] have QoS mappings totally different from those of plain audio streams because of the different amount of data to be transmitted over time and the different amount of computing power needed to decode and display the streams. The mapping of some of the MCAM parameters is rather complex. The throughput needed is determined by the requested period, quality and reliability. More stringent requirements by one or all of these parameters result in a more demanding throughput requirement and vice versa.

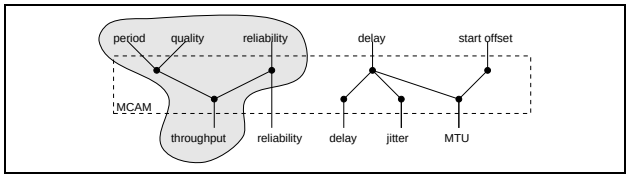

Figure 3: Mapping of MCAM QoS parameters to transport QoS parameters

The delay parameter at the MCAM service interface is influenced by the delay of the transport connection. On the one hand, longer buffer queues each result in a higher delay. On the other hand, if the user requests a minimum delay, no buffering should take place. Therefore, the quality of the CM stream might deteriorate because jitter introduced by the network or other functions cannot be completely eliminated.

Neither jitter nor MTU parameters of the transport QoS can be directly influenced by the MCAM user. MCAM keeps jitter as low as it can to provide an isochronous service. The key mechanism to eliminate jitter is to use queues in shared memory [13]. The MTU size parameter offered by the transport service is used by MCAM to realize an optimal mapping of CM stream parts to transport data units. The algorithm is simple: If only video has to be transmitted, fill each network packet completely (user data size = MTU), otherwise compute the user data size according to the delay requirements.

MCAM uses the start offset parameter to synchronize both sender and receiver at the start time of a stream. The requested start offset and delay also determine whether the MTU of the underlying transport service can be used as user data size or whether a smaller size has to be chosen.

4.2.1 Specification of the mapping to Transport QoS

In this section we present the mapping of the MCAM QoS parameters *period*, *quality* and *reliability* on the transport QoS parameter *throughput* (see the shaded region in Fig. 3).

Let CM be the CM stream to be transmitted. It can be subdivided into $nbUnits$ units U_i, with $0 \leq i \leq nbUnits$.[5] Typical examples for such units are single frames, groups of frames and groups of samples. Since each U_i is often too large to be transmitted as a whole, it has to be partitioned into $nbParts_i$ parts $pt_{i_j}, 1 \leq j \leq nbParts_i$. For simplicity, we assume that each part pt_{i_j} has the same size in bytes, i.e. $b(pt_{i_j}) = b(pt_{k_l}) \forall i,j,k,l$ and denote b_{pt} as the partsize. Let pd be the period, i.e. the number of seconds per unit U_i. Throughput is stated per time intervall t, e.g. $t = 1\ sec.$ Let $u1_t \ldots u2_t$ be the indices of the units U to be transmitted in each time period t. Then we can define the throughput T in t as

$$T_t = \frac{1}{t} b_{pt} * \sum_{i=u1_t}^{u2_t} nbParts_i \qquad (1)$$

As can be seen, the actual value of T_t depends

[5] Such a unit U_i can also be considered as a single message to be transmitted.

on the number of parts to be transmitted in t. Additionally, the number of parts depends on the period. Therefore if one or both of these parameters are changed, the throughput will change too.

Reliability and Throughput One MCAM QoS parameter that influences the number of parts to be transmitted is *reliability*. The *reliability* parameter controls our forward error correction scheme AdFEC (Adaptable Forward Error Correction [18]). AdFEC adds redundancy to the stream such that lost parts of the original stream can be reconstructed. In our current implementation, we can generate for n given parts, $n \in \{1,2,3\}$, m redundant parts, $m \in \{1,2\}$ [14]. If n of these $n+m$ parts arrive at the receiver, the original n parts can be reconstructed. Therefore we have five AdFEC types, denoted as FEC_n_m, plus the trivial case of not using AdFEC ($NoFEC$). For example, FEC_2_2 adds 100% redundancy, FEC_3_2 adds two extra parts for each set of three, i.e. creates $66\frac{2}{3}\%$ redundancy. Table 3 contains the mapping of the qualifiers for reliability to the different AdFEC types. In addition, it states the amount of redundancy added to the stream for each FEC type.

Reliability	very low	low	medium	high	very high
AdFEC type	NoFEC	FEC_3_1	FEC_2_1	FEC_3_2	FEC_2_2
Redundancy	0%	$33\frac{1}{3}\%$	50%	$66\frac{2}{3}\%$	100%

Table 3: Reliability and FEC type

Within XMovie, AdFEC is applied to units of a CM stream. The AdFEC scheme is capable of assigning different priorities to different units. These priorities are directly related to reliability, i.e. high priority implies high reliability and vice versa. As an example for assigning different priorities to different units, we consider an MPEG stream [19]: I frames may not be lost under any circumstances and therefore have a higher priority than P frames, which in turn are more important than B frames.

Let $nbParts_{i,AdFECtype}$ be the number of parts per unit U_i if AdFEC is applied and let, for simplicity, $n = nbParts_i$. For each AdFEC type we can calculate $nbParts_{i,AdFECtype}$ as follows:

$$
\begin{aligned}
nbParts_{i,NoFEC}(n) &= n \\
nbParts_{i,FEC\_1\_1}(n) &= 2 * nbParts_{i,NoFEC}(n) \\
nbParts_{i,FEC\_2\_1}(n) &= \begin{cases} 1,5 * nbParts_{i,NoFEC}(n) \\ 1,5 * nbParts_{i,NoFEC}(n-1) + nb \end{cases} \\
nbParts_{i,FEC\_2\_2}(n) &= \begin{cases} 2 * nbParts_{i,NoFEC}(n) \\ 2 * nbParts_{i,NoFEC}(n-1) + nbP \end{cases} \\
nbParts_{i,FEC\_3\_1}(n) &= \begin{cases} 1\frac{1}{3} * nbParts_{i,NoFEC}(n) \\ 1\frac{1}{3} * nbParts_{i,NoFEC}(n-1) + nb \\ 1\frac{1}{3} * nbParts_{i,NoFEC}(n-2) + nb \end{cases} \\
nbParts_{i,FEC\_3\_2}(n) &= \begin{cases} 1\frac{2}{3} * nbParts_{i,NoFEC}(n) \\ 1\frac{2}{3} * nbParts_{i,NoFEC}(n-1) + nb \\ 1\frac{2}{3} * nbParts_{i,NoFEC}(n-2) + nb \end{cases}
\end{aligned}
$$

Example :

$$
\begin{aligned}
nbParts_{i,FEC\_3\_2}(10) &= 1\frac{2}{3} * nbParts_{i,NoFEC}(9) + nbParts_{i,} \\
&= 1\frac{2}{3} * 9 + 2 * nbParts_{i,NoFEC}(1) \\
&= 15 + 2 \\
&= 17
\end{aligned}
$$

Now equation (1) can be extended to:

$$T_t = \frac{1}{t} b_{pt} * \sum_{i=u1_t}^{u2_t} nbParts_{i,AdFECtype} \qquad (2)$$

Period and Throughput As mentioned above, the MCAM period parameter also has an impact to the transport throughput parameter. If, for example, the period is increased, then the number of units in each time interval decreases, and the number of parts to be transmitted decreases.

The period pd denotes the amount of time available for each unit U_i. Therefore pd^{-1} denotes the number of units per time interval, and the following equation holds:

$$pd^{-1} = u2_t - u1_t + 1 = nbUnits_t$$

Let pd_{old} and pd_{new} be the old and new period, respectively. Then $u1_{t,new}$ and $u2_{t,new}$ in time interval t can be calculated as:

$$
\begin{aligned}
u1_{t,new} &= u2_{t-1,new} \\
u2_{t,new} &= nbUnits_t * \frac{pd_{old}}{pd_{new}} + u1_{t,new} - 1 \\
&= \frac{1}{pd_{new}} + u1_{t,new} - 1
\end{aligned}
$$

The throughput T_t after changing the period can now be determined as follows (see also equation (1):

$$T_t = \frac{1}{t} b_{pt} * \sum_{i=u1_{t,new}}^{u2_{t,new}} nbParts_i \qquad (3)$$

Quality and Throughput The MCAM QoS parameter *quality* influences the total size of each unit of the CM stream, and thereby the number of parts to be transmitted. In order to determine exact values for the relation between *quality* and *throughput*, we performed exaustive measurements.

We used a set of three different short movies, each consisting of 30 frames with an image size of 352×288 (CIF): a cartoon, an excerpt from a music video, and a sports movie. For each of the three formats Motion JPEG [24], H.261 [20] and MPEG-1 [19], we generated large sets of color movies, each movie with different settings of compression parameters such as quantization factor, motion compensation search range and group intervals of intra coded, predictive coded, and bidirectionally-predictive coded frames.[6]

The visual quality tests were performed on a true-color (24-bit) display. The following categorization was used to classify the result of these tests:

Q1: no differences to the original can be perceived

Q2: differences to the original can be perceived only in direct comparision

Q3: visual quality is acceptable, but differences to the original can be seen

Q4: differences to the original are not acceptable

Table 4 contains the relations between the file sizes of the movies in different quality categories. For example, it can be expected that the file size of a Q1 quality Motion JPEG movie is 1.6 times the file size of a Q2 quality movie.

Category	JPEG	H.261	MPEG
Q1 $\leftrightarrow$ Q2	1.6	2.3	6.2
Q2 $\leftrightarrow$ Q3	1.6	2.0	5.5
Q3 $\leftrightarrow$ Q4	1.2	1.5	n.a.

Table 4: Throughput and Quality

The MPEG coder could not be tuned to generate Q4 quality movies, therefore this entry is "n.a."

If Q1 quality movies are used as a starting point, the throughput for a lower quality level can be calculated by dividing the value resulting from equation (1) with the appropiate factor in Table 4. In the reverse order, i.e. from a lower quality level to a higher quality level, the throughput can be calculated by multiplication.

[6]Only MPEG coded movies can have all three types of frames; JPEG frames are always intra coded, and H.261 frames can be intra coded and predictive coded.

4.3 Mapping to Operating System QoS

In order to provide the requested isochronous stream service to the user, MCAM also needs operating system support in terms of real-time guarantees and shared memory buffers. Unfortunately, our current OS versions (Unix) do not have real-time support, e.g. a real-time scheduler. Therefore only the mapping of *quality* to *shared memory buffers* could be realized at this time. A requested *high* quality implies both high throughput and minimal delay jitter. If this cannot be mapped to operating system and transport guarantees, MCAM uses queues in shared memory to eliminate jitter [13].

Optimal mapping tables for combinations of QoS parameter and available system resources are still experimental.

4.4 Monitoring QoS

There are at least two options in QoS handling: either you trust your system components to fulfill the QoS requirements, or you monitor them. Since each system component can fail completely or violate the QoS contract, we opted to monitor the QoS. For this purpose every system component that has its own QoS parameters also has a QoS monitor. Within each process, one master QoS monitor is responsible to observe at regular intervals resource usage and QoS conformance by querying all QoS monitors. The master monitor reacts appropriately to resource over-utilization or QoS degradation according to one of the following failure modes, chosen at initialization time: best-effort, guaranteed-compulsory, guaranteed-threshold, statistical-compulsory or statistical-threshold (see Section 2). If, for example, the master monitor detects an underflow of the shared memory queues, it can either drop some frames for short-term synchronization or it can decrease the period of both queue users, thereby degrading the service.

By concentrating the decision logic into the master monitor we are able to experiment with different failure modes and mechanisms without affecting other system components. For example, the display of frames can be performed using either conventional X calls or our X extension. If our extension is used, the QoS monitor determines the buffer usage. This information indicates whether jitter can be reduced by shared memory buffers and the requested quality thus delivered. Otherwise any jitter increase in other system components would degrade quality, and the master QoS monitor would have to react according to

the requested failure mode.

5 Conclusion and Outlook

A small set of meaningful QoS parameters at the user interface has to be mapped to the QoS parameters of application units, and these have to mapped in turn to the transport QoS parameters. We have shown how complex the relation between the QoS parameters of application units and transport services really is, and we have used MCAM to exemplify this relationship.

The most important QoS parameter in multimedia systems is quality. Therefore the human user has a vital interest to control the system's behavior in this respect. Since each system component can fail completely or violate the QoS contract, we monitor the QoS in all components. We have shown how QoS monitoring is integrated into MCAM. The user can choose from among several failure modes, and the system reacts accordingly.

The mapping of MCAM QoS parameters to transport QoS parameters was shown in detail for the transport *throughput* parameter. We are working on a mathematical description of a complete mapping function, and of the QoS mapping in the application layer in general.

As soon as we have access to a real-time OS we want to port XMovie. In addition, more measurements need to be performed to extend the QoS mapping formulas, especially those about the *quality* mapping.

References

[1] A. Campbell, G. Coulson, F. Garcia, and D. Hutchinson. A Continuous Media Transport and Orchestration Service. *Computer Communication Review*, 22(4):99–110, Oct. 1992.

[2] D. D. Clark, S. Shenker, and L. Zhang. Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism. *Computer Communication Review*, 22(4):14–26, Oct. 1992.

[3] A. Danthine, Y. Baguette, G. Leduc, and L. Léonard. The OSI 95 Connection-Mode Transport Service – The Enhanced QoS. In A. Danthine and O. Spaniol, editors, *Proceedings of 4th IFIP Conference on High Performance Networking 92*, pages E2–1 – E2–18, Dec. 1992.

[4] M. de Prycker. *Asynchronous transfer mode: Solution for Broadband ISDN*. Ellis Horwood Limited, 1993.

[5] S. Deering. SIP: Simple Internet Protocol. *IEEE Network*, 7(5):16–28, May 1993.

[6] L. Delgrossi, R. G. Herrtwich, C. Vogt, and L. C. Wolf. Reservation Protocols for Internetworks: A Comparision of ST-II and RSVP. In Shepherd et al. [26], pages 195–203.

[7] D. Ferrari. Client Requirements for Real-Time Communication. *IEEE Communications Magazine*, 28(11):65–72, Nov. 1990.

[8] D. Ferrari. Real–Time Communication in an Internetwork. *Journal of High Speed Networks*, 1(1):79–103, 1992.

[9] D. Hehmann, M. Salmony, and H. J. Stüttgen. Transport services for multi–media applications in broadband networks. *Computer Communications*, 13(4):197–203, 1990.

[10] Information processing systems — Open Systems Interconnection — Transport Service Definition and Protocol Specification. International Standard ISO 8072/73, 1986.

[11] Information technology – Coding of Moving Pictures and Associated Audio for Digital Storage Media up to about 1.5 MBit/s (MPEG) – Part 3: Audio . International Standard ISO/IEC IS 11172-3, 1993.

[12] R. Keller and W. Effelsberg. MCAM: An Application Layer Protocol for Movie Control, Access, and Management. In *ACM Multimedia'93*, pages 21–29. ACM, New York, 1993.

[13] R. Keller, W. Effelsberg, and B. Lamparter. Performance Bottlenecks in Digital Movie Systems. In D. Shepherd, G. Blair, G. Coulson, N. Davies, and F. Garcia, editors, *Network and Operating System Support for Digital Audio and Video(NOSSDAV'93)*, Lecture Notes in Computer Science 846, pages 161–172. Springer-Verlag, Berlin Heidelberg, 1994.

[14] R. Keller, W. Effelsberg, and B. Lamparter. XMovie: Architecture and Implementation of a Distributed Movie System. *ACM TOIS*, 13(4), Oct. 1995. 471–499.

[15] R. Keller, S. Fischer, and W. Effelsberg. Implementing Movie Control, Access and Management – from a Formal Description to a Working Multimedia System. In L. Svobodova, editor, *Proceedings of the 14th IEEE International Conference on Distributed Computing Systems*, pages 276–283. IEEE Computer Society Press, Los Alamitos, CA, June 1994.

[16] A. Krishnamurthy and T. D. C. Little. Connection-Oriented Service Renegotiation for Scalable Video Delivery. In *Proceedings of the 1994 IEEE International Conference on Multimedia Computing and Systems*, pages 502–507, Boston, MA, May 1994.

[17] J. Kurose. Open Issues and Challenges in Providing Quality of Service Guarantees. *Computer Communication Review*, 23(1):6–15, Jan. 1993.

[18] B. Lamparter, O. Böhrer, W. Effelsberg, and V. Turau. Adaptable Forward Error Correction for Multimedia Data Streams. Technical Report TR-93-009, Praktische Informatik IV, Universität Mannheim, 1993. URL= ftp: //pi4.informatik.uni-mannheim.de /pub/techreports/tr-93-009.ps.gz.

[19] D. Le Gall. MPEG: A Video Compression Standard for Multimedia Applications. *Communications of the ACM*, 34(4):46–58, Apr. 1991.

[20] M. Liou. Overview of the p×64 kbit/s Video Coding Standard. *Communications of the ACM*, 34(4):59–63, Apr. 1991.

[21] T. D. Little and D. Venkatesh. Probabilistic Assignment of Movies to Storage Devices in a Video-on-Demand System. In Shepherd et al. [26], pages 204–215.

[22] R. Needham and A. Nakamura. Approach to Real-time Scheduling but is it Really a Problem for Multimedia? In P. V. Rangan, editor, *Network and Operating System Support for Digital Audio and Video(NOSSDAV'92)*, Lecture Notes in Computer Science 712, pages 32–39. Springer–Verlag Berlin Heidelberg New York, 1993.

[23] J. Nieh, J. G. Hanko, D. Northcutt, and G. A. Wall. SVR4 UNIX Scheduler Unacceptable for Multimedia Applications. In Shepherd et al. [26], pages 41–53.

[24] W. B. Pennebaker and J. L. Mitchell. *JPEG still image data compression standard*. Van Nostrand Reinhold, New York, 1993.

[25] D. Seret and J. Jung. Translation of Users' QoS Parameters into ATM performance requirements. In *Broadband Communications*, Paris, Mar. 1994. IFIP TC6.

[26] D. Shepherd, G. Blair, G. Coulson, N. Davies, and F. Garcia, editors. *Network and Operating System Support for Digital Audio and Video(NOSSDAV'93)*. Lecture Notes in Computer Science 846. Springer–Verlag Berlin Heidelberg New York, 1994.

[27] R. Steinmetz. Analyzing the Multimedia Operating System. *IEEE MultiMedia*, 2(1):68–84, Spring 1995.

[28] A. Vogel. Towards a formal computational model for distributed multimedia applications. pages 363–365, Bern, Schweiz, 1994.

[29] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network*, 7(5):8–18, Sept. 1993.

[30] M. Zitterbart, B. Stiller, and A. Tantawy. A Model for Flexible High–Performance Communication Subsystems. *IEEE Journal on Selected Areas in Communications*, 11(4):507–518, May 1993.