User-to-User QoS

- Management and Monitoring -

Martina Zitterbart

IBR, TU Braunschweig, 38106 Braunschweig, Germany

zit @ ibr.cs.tu-bs.de, <http://www.ibr.cs.tu-bs.de>

Abstract

The rapid development towards networked multimedia applications leads to new and high requirements on the network as well as on the systems involved. Especially guaranteed Quality-of-Service (QoS) is required for certain applications with isochronous data streams or with highly interactive characteristics. Therefore, an integrated systemwide QoS management is needed. Among others, issues such as mapping of QoS parameters between fundamentally different levels of the system (user, application, network) and QoS monitoring are very important. This paper presents a framework for QoS management. Requirements of networked multimedia applications as well as several QoS management issues are discussed. Moreover, a QoS monitor as part of the QoS framework and its implementation is presented.

Keywords

Quality of Service (QoS), monitoring, QoS management

1 INTRODUCTION

Networked multimedia applications are getting increasingly popular. They even seem to change society rapidly towards a so-called information society. However, most available systems are rather toys. They are by no means ready for usage outside the research and development community. A popular example can be seen in the widely used MBone tools for video-conferencing in the Internet (Kumar, 1995). Dependent on the time of the day, the Internet may provide a good or a lousy service quality to the user. Especially the audio stream is very sensitive with respect to packet losses. This instability in delivering a specified service makes those tools over the Internet inapplicable to professional environments. In addition, most common tools and standards provide only very basic control and cooperation mechanisms in order to effectively support conferencing and collaborative work. Even the ITU-T standards family T.120 (T120, 1994) presents a conferencing framework only. Generally, there exists an extreme mismatch between current capabilities of networked multimedia applications and ad-

vertisements made in research and non-research communities. Emerging networked multimedia applications need much better support with respect to the QoS delivered to the user. Especially the need for continuous data delivery in case of audio and video streams contributes to these requirements. Additionally, interactivity is highly demanding. In the following, applications are categorized according to their specific QoS requirements (cf., Table 1):

- *Broadcasting applications*: typical examples are Digital Video Broadcast (DVB) and Digital Audio Broadcast (DAB).
- *Interactive playback*: typical examples include VoD (VCR-like) and teaching/tutoring applications. Mostly, playback is intended for a single individual user (or for a single end system, e.g., TV).
- *Conferencing*: conferencing applications include audio and video streams. Their main focus is on these two media. Data applications, e.g., whiteboard applications, may be incorporated to support conferencing.
- *Collaboration*: we classify such applications as collaborative, that focus on the collaborative data part. Audio and video may just be used as supportive tools. Typical examples of collaborative applications are joint editing and interactive games.
- *Traditional real-time applications*: a typical example can be seen in tele-robotics.

Table 1 QoS-driven Application Categories

Data Flow	Type	Timing Requirements	Typical User Requirements
asymmetric	broadcasting	isochronous data	low set up latency (few seconds) high downstream bandwidth (several Mbps) TV or HIFI quality (color, resolution, ...)
	playback	isochronous data (short or long streams) interactivity	high downstream bandwidth (video clips) low upstream bandwidth (interactions) low upstream and downstream latency TV or HIFI quality (color, resolution, ...)
	real-time	interactivity	typically low bandwidth and latency guaranteed maximum reaction time
symmetric	conferencing	isochronous data interactivity	high bandwidth requirements low latency high video/audio quality (video size, resolution) synchronisation (mostly audio and video)
	collaboration	interactivity isochronous data	moderate bandwidth requirements very low latency (e.g., joint editing) synchronisation (very fine granularity)

In Table 1, two basic types of timing requirements are distinguished:

- requirements from isochronous data streams, and
- requirements due to interactivity.

Isochronous requirements are introduced by applications that include audio and/or video streams which need continuous data delivery to the user. This requires guaranteed upper bounds on delay jitter. The requested QoS needs to be supported without any disruption during the entire communication phase. Additionally, reliability requirements of isochronous applications differ from those of traditional data applications:

- reliability is tight to timelines (i.e., data received after a deadline is considered as lost),
- no absolute reliability is required (tolerance depends on coding and compression).

Interactivity basically requires very low response times to user actions and, thus, very low latency. High bandwidth is not necessarily needed.

Traditional applications, such as file transfer do not require any specific guaranteed QoS. In the Internet Services framework (Braden, 1994) such applications are named elastic applications. Elasticity applies to their very loose timing requirements. Timing requirements may, however, be more strict if they are part of an application which consists of multiple streams (audio, video, data) that need to be synchronized. This implicitly may lead to high service requirements on such traditional applications as well. Generally, completely independent streams may have different requirements than streams being part of a more complex application.

Moreover, many forthcoming applications are inherently based on *multicasting*. Generally, group communication should be seen as a major communication paradigm for the future and, thus, should be directly reflected in any attempt to support QoS. It leads to heterogeneous QoS support within an interacting group due to different end system and network attachment characteristics (e.g., high-end workstations versus PDAs).

All components involved in the communication need to address these QoS requirements, including operating systems and system architectures (e.g., bus access latency and jitter). However, up to now, many investigations on QoS focus on isolated issues, such as networking or individual protocols. Generally, a lack of approaches that integrate protocol and application layer processing as well as operating systems and system architecture issues can be observed. In most cases, user-to-user services - as they are finally required - are not addressed. For that goal, an integrated systemwide QoS management approach is critical. In this context, network issues as well as protocol, application, and operating system issues need to be addressed in an integrated way. *QoS management* comprises different tasks, including, among others, local and network wide resource management as well as QoS monitoring. This paper discusses various issues considering integrated QoS management. Especially an approach towards QoS monitoring is presented in some detail.

The paper is structured as follows. Section 2 presents a clearly defined QoS framework. Section 3 concentrates on QoS management itself and on its proper structuring. The QoS monitor and its implementation is presented in section 4. Section 5 concludes the paper and gives an outlook of future work.

2 QOS FRAMEWORK

Since Quality-of-Service (QoS) forms an important part of today's research work in the communications area, one would assume that a crisp definition of QoS can be given. This is not the case, since the term QoS is defined with different degrees of abstraction. In the following, some typical examples are summarized.

2.1 QoS Definitions

Described very roughly, Quality-of-Service (QoS) refers to the support of certain service parameters in order to eventually accommodate user requirements. There exists no single definition

of QoS. Commonly various descriptions reflecting different levels of granularity are used (cf., Table 1). The spectrum includes highly generic definitions as well as detailed ones addressing specific parameters and their requirements. This variance also reflects different levels of a communication system at which QoS is applied (cf., section 2.2).

Table 2 presents some collected QoS definitions. It is by no means complete, however, it already reflects many flavours of QoS definitions. Basically, every group defines its own QoS specification dependent on the resources and requirements of the component under consideration. For example, the definition used within the ODP framework appears to be oriented towards the higher levels. It defines QoS as a set of qualitative requirements on the collective behaviour of one or more objects (Vogel, 1995). This raises the important issue of collaboration among different objects. Objects are, however, not defined any further. The ETSI definition focusses solely on the QoS specified at the user interface. In contrast, the ITU concentrates mostly on lower levels, e.g., on the network level. However, the ITU recommendation E.430 clearly states that user oriented QoS parameters most probably differ from network QoS parameters. The latter are called performance parameters. Generally, higher levels tend more to deal with qualitative issues and parameters; lower levels mainly address quantitative tasks or performance parameters. Thus, different kinds of parameters are applied dependent on different levels in the communication system. This clearly raises the mapping issue between parameters used at different levels. Since QoS definitions that deal with higher levels, such as the user and the application level, are somewhat generic, mapping them to system and network parameters is not always straight forward. Another factor that needs to be addressed here are the interactions among system components that may affect mapping rules.

Table 2 Various QoS Definitions

Origin	QoS Definition
ISO	A concept for specifying how good network services are
OSI ODP	A set of quality requirements on the collective behaviour of one or more objects
ETSI	QoS is described in terms of a set of user-perceived characteristics of the performance of a service. It is expressed in user-understandable language and manifests itself as a number of parameters, all of which have either subjective or objective values
ITU-T, I371	QoS requirements are specified in terms of objective values of some of the network performance parameters specified in Recommendation I.356
ITU-T, E.430	Distinction between user QoS and Network QoS

Independent of the individual specification, QoS eventually deals with certain requirements on the system that are expressed using various so-called *QoS parameters*. These parameters differ based on the level being addressed. QoS parameters are somewhat comparable to *regular system parameters*, such as main memory size and CPU power. In contrast to system parameters, users can not just easily be asked to increase QoS parameters. For example, to buy more network bandwidth. However, in case of traditional system parameters, such as main memory size, they usually have two choices: buy more main memory or survive with slow applications, i.e., with a low QoS. In the worst case, the application will not run at all. Such a behaviour is somehow tolerated by the users for single machines, but not for networked environments.

System parameters are local to a system, whereas QoS parameters depend on various local and non-local resources. Especially a good overall utilization of non-local resources appears to

be important. Handling of QoS parameters is somewhat more complex than handling regular system parameters, because:

- requirements address both: individual and shared resources;
- requirements may not be known in advance by the user;
- requirements may *vary highly over time*;
- parameters are *negotiated*;
- parameters may be *mapped* or translated between different QoS levels;
- multiple systems/components are involved leading to “interrelations among parameters.

Therefore, specific QoS management is required. This paper addresses various aspects of QoS parameters, including mapping and negotiation. These tasks are incorporated in a framework that addresses user-to-user QoS issues and that is currently being implemented. Due to the limited space, only selected aspects can be presented.

2.2 QoS Levels

Since Quality of Service is related to the dedication of shared resources to networked applications, it needs to be considered at different levels (i.e., for various types of resources) within a communication system. Therefore, we distinguish three QoS levels (cf., Figure 1):

- user level,
- application level, and
- network level.

Figure 1 QoS levels within a communication system

The *user level* is directly related to human perception issues. Typically, users state their QoS requirements in an imprecise and highly subjective manner. Examples are type and size of a video window (e.g., colour and large) in case of a videoconferencing application. The user should not need to have deep technical knowledge in order to operate networked multimedia applications. Thus, the requirements will usually not be specified very precisely. For example, the value large in case of the size of a video window is not “precise and, moreover, is relative to the size of the screen used. Additionally, human perception leads to implicit specifications of QoS requirements, especially in case of synchronisation. For example, in case of a videoconference including a shared whiteboard, synchronization requirements are implicitly included.

The user level comprises resources that are sources or sinks of data (camera, loudspeaker, screen, and the like). User level requirements need to be translated into application level requirements. Therefore, the communication system holds mapping tables which are used to transform the fuzzy and highly subjective and qualitative requirements to more precise and quantitative requirements needed at the application level interface.

The user level passes mostly quantitative parameters to the *application level*. Typical examples are the size of the application data unit, the maximum data rate expected, and the maximum accepted delay. The application level deals with local parameters only. Resources at the application level are application-related protocols and protocol mechanisms, the system architecture as well as the operating system.

The operating system interacts with the *network level* due to its scheduling and memory management tasks. Interrupt handling is also an important issue that influences achievable QoS guarantees. The network level deals with parameters that are concerned with the quality of service of external communication. The network level interfaces to all functionalities that are mainly dependent on the network behaviour and that regulate the end-to-end flow. Therefore, this level would most commonly be mapped to the traditional transport service interface. Thus, it can, for example, be compared with the socket interface. However, the model itself does not include any implementation details. For example, it does not consider whether the transport layer protocol is implemented inside the kernel of the operating system (as in most traditional UNIX systems) or in the user space closely attached to the application itself (what can be seen in some newer implementations).

Between the QoS levels, the type of parameters changes. Within a level, parameters may need to be translated, but the type of parameters remains the same. At the user level they are more or less solely qualitative parameters. At the application level, mostly quantitative parameters appear, mixed with few qualitative parameters. The network level deals with qualitative parameters only. These parameters are related to external communication. The network level QoS can be compared to end-to-end QoS often referred to in research papers.

Using three QoS levels seems to be a cleaner design than distinguishing many levels that only have marginal differences. For example, in (Nahrstedt, 1995) five levels are distinguished including system and device level. However, the distinction between user and device level is not needed if the model is viewed by a proper abstraction. Both levels represent sources and sinks of data. Similar arguments hold for the distinction of application and system levels. Additionally, the same type of resource applies to different levels in the model, which does not present a clean design. Another approach of decomposing the communication systems into different QoS levels is presented in (Campbell, 1996). The authors follow more or less directly the OSI reference model; they even add additional layers. A clear separation of tasks and resources is missing.

3. QOS MANAGEMENT

In order to provide the QoS requested by a networked application, resources are required in the network, the intermediate systems and the end systems. These resources are typically shared among various users (or system components) and, thus, conflicts may happen. However, some applications need to deliver data continuously (e.g., audio and video streams) to the user. They require service guarantees. Therefore, network and system resources need to be reserved. Traffic characteristics of networked applications may vary over time and are mostly

not known in advance. Thus, resources have to be allocated without complete knowledge of traffic characteristics. Since maximum reservations lead to low resource utilization, reservation strategies should be chosen, e.g., on some average and maximum values this may result in temporary overload situations and, thus, in drop of service quality experienced by some users. Resource utilization and application behaviour need to be monitored continuously in order to avoid any service problems. More generalized it can be said that communication systems need some QoS management in order to properly serve multimedia applications. A systemwide QoS management supporting guaranteed service quality is required. QoS monitoring forms an important part of QoS management.

3.1 Dual Stack Architecture

Within a communication system it should be clearly distinguished between regular communication operations and QoS-related tasks. Figure 2 presents a structure that is based on two different stacks:

- application stack and
- QoS management stack.

Figure 2 Dual-stack QoS architecture

The *QoS management stack* is responsible for handling tasks related to QoS mapping, setup and monitoring. It is subdivided into several components, among them:

- a local resource manager and
- a network resource manager.

The *local resource manager* has to form an application profile that contains application-specific resource requirements, such as audio/video encoding scheme or encryption requirements. An example of an application profile is given in table 3 considering different levels of a communication system (cf., Figure 1). Internally, the values are derived using lookup tables that contain profiles. At the user level, the local resource manager

ervations or availability checks of presentation related local resources. This can be performed without involving the operating system. At the user level, parameters that are closely related to the presentation quality itself are considered. Examples of such parameters are colour (yes/no), video (yes/no), MPEG-2 (yes/no). The reply to these parameters can be seen as a selective yes-no-type of answer. It can be returned to the user immediately after consulting the configuration of the local system. In addition, further parameters may be used in order to describe the presentation somewhat more precise. For example considering the size of a video window (small, medium, large). These parameters do have a subjective and qualitative nature and they usually do not give any precise information. The profile combined with collected experience considering the user are applied to resolve such fuzzy QoS requirement specifications.

Furthermore, the local resource manager takes care of local resources, such as processor, memory space, encoding schemes, application types, and local bandwidth. It informs the application immediately, in case that sufficient local resources with respect to the application profile cannot be made available.

Table 3 Example of application profile (see (Nahrstedt, 1995))

	Audio	Video
Application level	sample size: 8 bit (telephone quality) sample rate: 8 kHz playback point: 100-150 ms	frame rate: 30 fps frame width: ≤ 720 pixels frame height ≤ 576 pixels color resolution: 16 bit/pixel aspect ratio: 4:3 compression ratio: 2:1
Network level	end-to-end-delay: ≤ 150 ms (most) ≤ 400 ms (some) round-trip delay: ≤ 800 ms packet loss: $\leq 10^{-2}$ bandwidth: 32 kbps (audioconf.)	end-to-end delay: 250 ms packet loss (unc.): $\leq 10^{-2}$ packet loss (co.): $\leq 10^{-11}$ bit error rate: $\leq 10^{-6}$ bandwidth: ≤ 1.86 Mbps (MPEG)

The application specific profile is forwarded from the initiator of the communication association to a so-called *group mediator*. The group mediator is especially helpful in case of heterogeneous QoS requirements within a group communication. In case of a simple point-to-point communication, it may not be required. However, point-to-point communication should generally be treated as a special case of multipoint-to-multipoint communication because many emerging applications are inherently based on multipoint communication scenarios. The group mediator may adapt the application profile for group members with different QoS capabilities (e.g., considering lower capabilities of PDA systems compared to high-end workstations). The resulting application profile is then forwarded to the communication partner(s) during the set-up and negotiation phase. It is accompanied by network resource requirements. Forwarding network requirements is not sufficient since they do not provide enough information about the original user requirements. The profile presents this information to the remote user(s) in order to allow a complete user-to-user QoS coverage. Completely bidirectional translation/mapping of QoS parameters could avoid sending the profile. However, it seems not to be a practical approach because users usually make use of mostly qualitative and often fuzzy requirement specifications. This leads to difficulties with bidirectional mapping especially if some knowledge about user behaviour is included in the decision. Thus, the profile including the mapping at the original side is passed to participating communication partners. As a result, a set-up

phase is required for every communication association that needs guaranteed QoS support. This can be combined with the connection setup phase. Thus, only a marginal additional delay should be expected.

In contrast to the local resource manager, the *network resource manager* is responsible for network-related resources, i.e., for end-to-end QoS. This includes resource reservations on the communication path, if needed.

The network resource manager may initiate specific protocols for signaling and resource reservation. These protocols are typically located at the network level of the application stack. Currently, several protocols supporting the signaling task associated with the set-up and negotiation phase are under discussion. For example, Q.239B as signaling protocol within ITU-T standards (Stiller, 1995) or RSVP as reservation protocol within the Internet protocol suite (Thomas, 1996).

The QoS monitor then needs to monitor resource utilization with respect to QoS. It is part of the QoS management stack (cf., Figure 2). The design and implementation of a prototypical QoS monitor is presented in section 4.

3.2 Related Work

An overview of QoS architectures focussing on architectural perspectives can be found in (Campbell, 1996). It presents a high-level overview and compares several architectures. However, no discussion about implementations and real measurements is included. Additionally, an integrated system wide approach of QoS management seems still to be missing.

In (Nahrstedt, Smith, 1995) an architecture basically reflecting the QoS components shown in figure 2 is presented. The main component is called QoS-Broker. The QoS-Broker can be directly compared with the QoS-Manager presented in (Schmidt, 1995). Typically, it would be part of the QoS management stack introduced above. The dual-stack architecture is derived from (Schmidt, 1995). Another QoS Manager is presented in (Quang, 1995). They address memory access time and its influence on the observable QoS in a very general way.

The overview paper (Nahrstedt, 1995) focusses on resource management. It gives a summary of some QoS parameters as they are needed in several layers of a communication system. Interactions among different components are discussed at a very high level. No further details are given. In order to evaluate their QoS-Broker approach, they use tele-robotics as a test applications. They do not consider networked multimedia applications as presented in section 2. However, such applications are rapidly emerging.

More detailed experiments of QoS management are presented in (Vogt, 1995). DVI-encoded video streams are discussed in order to validate an analytical model. The goal is in reducing the amount of resources that need to be reserved and in investigating the trade-off between reservations and maximum delay experienced.

In (Ramanathan, 1995) a QoS management approach for video services is presented. The approach needs to be seen in the context of ALF (Application Level Framing (Clark, 1990)). The basic idea is that if an application frame (here a video frame) is already not valid due to missing or erroneous packets of lower layers, all remaining packets for that application frame should be discarded right away. A bit indicating the duration of a frame is used in order to provide switches with the capability to discard not needed packets.

3.3 QoS Parameters & Mapping

QoS is associated with so-called QoS parameters that are used to specify the expected quality. Different parameters are applied for the resources at the three levels introduced above. Generally, quantitative and qualitative parameters can be distinguished, as presented in (Zitterbart, 1993). Qualitative parameters basically select whether a certain service property is available or not. For example, if encryption is required or not. Moreover, qualitative parameters include subjective requirements stated at the user interface. Quantitative parameters are those that can be evaluated in terms of certain measures, such as bits per seconds, number of errors.

Table 4 Typical QoS parameters

Level		Quantitative QoS Parameter	Qualitative QoS Parameters
User		setup latency, skew (both are defined implicitly)	color, presentation quality (colour, size, synchrony, fidelity), number of streams, guarantee commitment, cost, security, group semantics
Application	Protocol, etc.	size (frame, sample), delay (end-to-end, round trip), delay jitter, loss rate (frame, sample), resolution, compression, aspect ratio	error tolerance (% corrupted, replicated, lost), inter-/intra-stream synchronization, security algorithm
	Operating system	scheduling accuracy, granularity, period time, deadline	--
Network	Protocol, etc.	loss/error rate (cell, frame, packet), bandwidth, delay, delay jitter, bit error rate, reliability (losses, duplications)	--

Table 4 lists typical QoS parameters of the network, application and user levels. For each of the quantitative parameters, mean and peak values may be given. Furthermore, an interval of acceptable values may be used for certain parameters (Zitterbart, 1993). An interesting aspect can be seen at the user level. Many qualitative parameters are listed. In addition, two quantitative parameters are given: setup latency and skew. These quantitative parameters are not explicitly specified by the user. However, due to the application type and the number of data streams involved (e.g., audio, video, whiteboard, ...) they are defined implicitly. Inter-relations among different streams are responsible for that. Typical examples are inter-relations, such as lip synchronization and pointer synchronization.

Mapping of QoS parameters between different QoS levels in the system is required in order to eventually provide user-to-user service guarantees. In the following, some examples for the mapping of QoS parameters are given. They mainly concentrate on rate and delay requirements associated with the parameter mapping. The following parameters are defined:

- $S(u)$ Size of user data unit (e.g., video frame)
- $S(a)$ Size of application data unit
- $S(n)$ Size of network data unit
- $S_d(n)$ Size of data portion of network data unit

The parameter N reflects the number of data units sent at the user level. The rate requirements R can be transformed between neighbouring levels. They depend on the data unit sizes. Thus, all operations that manipulate the size of a data unit change the rate requirements. Examples

are segmentation and reassembly or encoding. The parameter α describes the increase/decrease of $S(a)$ compared to $S(u)$. This parameter α is calculated based on information given by the protocol mechanisms selected at the application level. At the application level typically functions such as encoding are located. Segmentation and reassembly is usually not part of the application level.

- $R(u) = (N * S(u)) / \text{sec}$
- $R(a) = (N * S(a)) / \text{sec}$, with $S(a) = \alpha * S(u)$

At the network level, the parameter β reflects the changes with respect to the size of the data unit. At this level especially segmentation and reassembly need to be considered for the calculation of β . In addition, the number of data units may change due to segmentation and reassembly.

- $R(n) = v * N * S(n) / \text{sec}$, with $S(n) = \beta * S(a)$ and $v = [S(a) / S_d(n)]$

usually, rate requirements increase towards the lower levels: $R(n) \geq R(a) \geq R(u)$

In addition, other parameters need to be considered, for example delay. This involves system internal behaviour of the operating system and its scheduling mechanism at both, the application and the network level.

4 QOS MONITORING

In order to provide guaranteed services to the user, some control and monitoring is required in addition to resource reservations. Therefore, we designed a QoS monitor. In our prototypical implementation, the QoS monitor was integrated at the network level (Kanschik, 1996). In particular, it was tested with two transport level protocols, namely Sandia-XTP (XTP, 1996) and Patroclus (Braun, 1995). These protocols were chosen, since they provide some advanced QoS at their service interface.

Figure 3 QoS monitor implementation structure

The QoS monitor is implemented on SUN systems running SUN OS 4.1.3. The QoS monitor is implemented as part of the QoS management stack shown in Figure 2. It is located between

the application stack and the resource managers of the QoS management stack. Moreover, an interface to network management, namely SNMP, is provided. Thus, some longer-term QoS information can be made accessible to other systems participating in SNMP.

4.1 Interfaces

QoS monitoring is needed at different levels within end systems and intermediate systems. Thus, the design of the QoS monitor should be modular enough to allow its integration at various levels and its co-operation with multiple protocols of different levels within the application stack. In order to achieve a very open design in that sense, an SNMP interface or a socket interface could have been used. However, this would lead to a high system overhead. Ideally, the QoS monitor should add only a negligible processing overhead to the communicating systems. With that goal in mind, a direct integration of the QoS monitor into the protocol implementation of a dedicated level would be favoured. Since this does not result in a modular and flexible design, we did not choose this alternative. In order to be flexible and modular, we implemented a FIFO queue as shared memory space between the protocol and the QoS monitor (cf., figure 3). A simple data format is defined for entries in the FIFO queue. Each entry consists of four fields: a connection key, an action code, identification of the information type and an information field. The format used is defined as follows:

```
typedef struct { address64 connectionid;
                char action;
                char info_type;
                INFO information;
                } IF_TYPE; /* interface type */
```

The protocol adds an entry every time that an associated event occurred (`PUT_ELEMENT (IF_TYPE *value)`). In addition to the FIFO queue, a request line is implemented. It is used by the protocol to inform the QoS monitor that a new entry was added to the FIFO queue. It is only used in case the FIFO queue was empty before adding a new entry. Thus, the overhead introduced by the request line is kept to a minimum. The QoS monitor removes the entries from the FIFO (`GET_ELEMENT (IF_TYPE *value)`).

Figure 4 QoS monitor attached to XTP

In order to interface with SNMP network management, the QoS monitor implements an SNMP agent. It also owns a so-called QoS-MIB. The QoS-MIB holds actual information on QoS parameters, such as throughput, delay and delay jitter with respect to the established

communication associations. Generally it is preferred to have uni-directional data streams reflected in the QoS-MIB. Thus, inbound and outbound traffic characteristics are separated.

In addition to its interface to the protocol, the QoS monitor also provides a graphical user interface (cf., figures 5 and 6). This allows the visualization of monitored parameters, such as throughput and delay. However, the graphical interface adds substantial overhead. With that graphical interface, the performance of Sandia-XTP implementation was limited to a few kbps, as reflected in figure 5. The graphical interface is written in Tcl/Tk which is an interpreted language and, thus, contributes to a high system load. Typically, it would be sufficient to run the QoS monitor without this interface or only with a few parameters depicted on the screen. For example, a visualization of selected parameters in analogy to a traffic light (green - yellow - red) are supported in the current implementation of the QoS monitor. The green range shows variations of less than 5% of the negotiated value. The yellow zone signals a variation between 5% and 15%, and the red range signals a variation of more than 15%. Moreover, visualization of most information may only be needed in case of problems (e.g., system overload and red signal range), but not during regular operation. It is more important that the QoS monitor can initialize a signal to the application that currently problems are experienced.

3.2 Monitored QoS Parameters

The QoS monitor is dedicated to short term measurements of QoS parameters. The reason therefore can be seen in the need to reflect temporary overload situations. The resource managers are informed accordingly and need to decide about actions to be taken.

For measurements of the actual throughput T the following formula is applied. The parameter α reflects the sampling duration measured in milliseconds.

$$T = \frac{\alpha}{\sum_{i=1}^n \lambda_i} \sum_{i=1}^n P_i$$

P_i : size of packet i in bits

λ_i : time of packet i

This formula basically allows monitoring of the throughput behaviour in case of high load, i.e., low packet inter-arrival time. It was chosen, because we mainly aim at detecting overload situations. The graphical interface related to throughput monitoring is depicted in Figure 5.

Figure 5 Monitoring the actual throughput

Several other parameters are monitored. The delay is also of major interest. However, it can only be estimated. With the XTP implementation we make use of the *sreq-Bit*. If this bit is set, then the receiver has to respond with a control packet. A timer is started, when the packet is

sent and stopped after the reception of the associated control packet. The measured time corresponds to the round-trip-time. We estimate the end-to-end delay being half of the round-trip time. Delay jitter can also be estimated if it is part of the "traffic contract.

Figure 6 Monitoring Interface

In addition, certain measures related to reliability are supported by the QoS monitor (cf., Figure 6). For example, bit error rates can be calculated. With the QoS monitor attached to the XTP implementation, error control packets are observed. Calculation of parameters relevant to traditional reliability is initiated by the reception of an "error control packet. However, this does not allow for a distinction between bit errors and packet losses. Therefore, an extension to the XTP protocol would be needed. In order to calculate the bit error rate, the following formula is applied to calculate the number of errors:

$$errors = 8 * \left(seq - rseq - \sum_{i=1}^{n_{spans}} length(span_i) \right)$$

The discussion of monitored QoS parameters in this section shows clearly, that there is a tight dependency with the monitored protocol. Some parameters may simply not be supported by some protocols. Moreover, the calculation of some QoS parameters depends on the protocol parameters used in the corresponding protocol mechanisms. In some future work, we aim at defining some sets of meta-parameters that describe certain protocol mechanisms. They should cover the diversity of parameter usage within the protocols. In "order to adapt the QoS monitor

to a specific protocol, the corresponding set of meta parameters is selected. This will increase the flexibility of the monitor.

3.3 Evaluation

In summary, the QoS monitor has several features for monitoring and displaying various QoS parameters. The overhead associated with each of them has to be evaluated carefully. Especially the graphical interface contributes to a high overhead. However, as argued above, it is not absolutely required in all cases. The computation of monitored QoS parameters also contributes to some overhead, especially, since this is currently implemented in software. Our QoS monitor consumes about 10-15% of the resulting performance. The transfer of a 1 MByte file (transferred in 20s) stimulated slightly more than 5000 events that were forwarded to the FIFO queue. The rate of events was still slow enough to cause a context switch for every event. This appears to be acceptable. In order to further decrease this overhead, the implementation is currently being optimized with specific focus on the communication part between protocol and QoS monitor. As an alternative, the QoS monitor could be implemented using specific hardware. This would rapidly decrease the overhead on the protocol performance.

4 SUMMARY AND FUTURE WORK

Networked multimedia applications impose high requirements on communication systems. They need some guaranteed QoS provided at the user interface. Therefore, in addition to the network itself, end systems and their internal behaviour need to be addressed as well. Generally, an integrated QoS management is needed. Within this paper a framework for QoS management is presented. The paper discusses some general issues related to QoS management. Furthermore, it presents experiences collected with an implementation of a QoS monitor.

Future work will concentrate on the integration of appropriate operating system support and on high performance implementation issues. Moreover, dedicated hardware support for selected protocol functions and for the QoS monitor is planned. A complete implementation of the QoS management stack is currently under way.

ACKNOWLEDGEMENT

First, I would like to thank Ahmed Tantawy for deep and very fruitful discussions considering quality of service and its support within communication systems. He highly contributed to the content of this paper. Thomas Kanschik implemented a first version of the QoS monitor. Urs Thürmann is currently pursuing this work towards a better performing implementation.

REFERENCES

- Braden, R., Clark D., Shenker S. (1994) Integrated Services in the Internet Architecture: an Overview; RFC 1633, Juni 1994
- Braun, T., Zitterbart, M. (1992), Parallel Transport System Design, 4th IFIP Conference on High Performance Networking, Lüttich, Belgien, Dezember 1992

- Campbell, A., Aurrecochea, A., Hauw, L. (1996); Architectural Perspectives on QoS Management in Distributed Multimedia Systems; Workshop on Quality of Service, Paris, France, March 1996
- Clark, D., Tennenhouse, D. (1990); Architectural Considerations for a New Generation of Protocols, ACM SIGCOMM, 1990
- ITU-T I.356, 1993; B-ISDN ATM Layer Cell Transfer Performance; November 1993
- Kanschik, T. (1996); QoS monitor; Diploma thesis (in German), TU Braunschweig, 1996
- Kumar, V. (1995) Mbone - Interactive Multimedia on the Internet; New Riders, 1995
- Kurose, J.F. (1992); Open Issues and Challenges in Providing Quality of Service Guarantees in High-Speed Networks; ACM Computer Communication Review, 1992
- Nahrstedt, K., Steinmetz, R. (1995); Resource Management in Networked Multimedia Systems; IEEE Computer, May 1995
- Nahrstedt, K., Smith, J. (1995); The QoS Broker; IEEE Multimedia, 1995
- Quang, N.H., Bernard, G., Belaid, D. (1995); System Support for Distributed Multimedia Applications with Guaranteed Quality of Service; High Performance Networking VI, Palma de Mallorca, Spain, Chapman & Hall, 1995
- Ramanathan, S., Rangan, P.V., Vin, H., Kumar S.S. (1995); Enforcing application-level QoS by frame-induced packet discarding in video communications; Computer Communications, Vol. 18, No. 10, October 1995
- Schmidt, C., Zitterbart, M. (1995); Towards Integrated QoS Management; LAN/MAN Workshop, Florida, 1995
- Stiller, B. (1995), A Survey of UNI Signaling Systems and Protocols for ATM Networks; ACM Computer Communications Review, April, 1995
- T120 (1994) ITU-T; Proposed Draft of T.120: Data Protocols for Multimedia Conferencing; März 1994
- Vogel, A., Kerheve, B., von Bochmann, G., Gecsei, J. (1995); Distributed Multimedia and QoS: A Survey; IEEE Multimedia, Summer 1995
- Vogt, C. (1995); Quality-of-service management for multimedia streams with fixed arrival periods and variable frame sizes; IEEE Multimedia, 1995
- XTP (1996), <http://www.ca.sandia.gov/xtp/SandiaXTP/>
- Zitterbart, M., Stiller, B., Tantawy, A.N.; A Model for Flexible High-Performance Communication Subsystems; IEEE JSAC, 1993

BIOGRAPHY

Martina Zitterbart received her doctoral degree in computer science in 1990 from the University of Karlsruhe, Germany. From 1987 to 1995 she was research assistant at the University of Karlsruhe. 1991 and 1992 she was on leave of absence at the T.J. Watson Research Laboratory, New York, USA. Since 1995 she is full professor at the Technical University of Braunschweig. Her research interests are in the area of multimedia systems, protocols, mobile communication and CSCW applications.