End-to-end QoS Management for Delay-sensitive Scalable Multimedia Streams over DiffServ

Markus Albrecht, Michael Köster, Peter Martini, Matthias Frank
Institute of Computer Science IV
University of Bonn, Germany
{sukram, koester, martini, matthew}@cs.uni-bonn.de

Abstract

The migration of the Internet from classic computer communication to a platform for multimedia applications with real-time communication requires end-to-end improvements of the network-level service. The proposal known as Differentiated Services is a very promising approach for implementing quality of service in the Internet and is being discussed and developed. The challenge is to deliver end-to-end QoS on top of Differentiated Services considering multiple concurrent application-level data streams. In this paper, we propose a QoS management system for multimedia servers that benefits from the scaling properties of layered media streams. This enables the system to map application QoS demands to available network resources and to adapt the quality of individual streams according to inter-stream QoS dependencies.

1. Introduction

The best-effort service model supported by the Internet does not guarantee timely delivery of packets. Nevertheless, this approach works well with conventional Internet applications like file transfer, electronic mail, and Internet Browsing. These applications are called "elastic" because of their ability to cope with packet losses and varying packet delays. In contrast to this, the efforts to make the Internet *the* multimedia communication system of the future reveals some basic problems with the best-effort service model when applied for media streaming applications: Multimedia applications with real-time properties require tight guarantees in terms of packet delay and packet loss. Obviously, interactive multimedia communication can only be realized with upper bounded end-to-end delays. In order to meet these requirements the Internet service model has to be improved by adding additional services.

Two controversially discussed approaches for achieving Quality of Service (QoS) improvements in the Internet are *Integrated Services* (IntServ) and *Differentiated Services* (DiffServ) both being developed and refined by the Internet Engineering Task Force (IETF). The IntServ architecture intends to provide end-to-end bandwidth reservations by maintaining per-flow state information along the path from the sender to the receiver. The main disadvantage of IntServ is that it works on flow basis and thus is not scalable to large networks with thousands of flows (RFC 2208). This scalability problem resulted in the DiffServ approach where QoS is not achieved by resource reservations for individual flows, but by assigning packets to certain service classes. At network boundaries, these tagged packets are treated according to well defined per-hop behaviors specified for each service class. With DiffServ, quality differentiation is provided without giving strict guarantees.

As the DiffServ architecture only defines a basic set of building blocks for QoS enhancements, some challenging questions are left to service providers and end users: How to apply DiffServ in real life and how to manage aggregated bandwidth? Consider a multimedia server with session-based real-time streaming across the Internet where a user-initiated session may consist of several media streams (e.g. video and audio) with each stream having its specific requirements concerning bandwidth, end-to-end delay, and packet loss. In this paper we study a new approach of how to apply a differentiated network service model like DiffServ to a given multimedia server scenario with several concurrent sessions and streams.

The bandwidth available in each service class has to be shared among all multimedia streams starting from the server. It seems reasonable to make the server itself perform the inter-stream QoS and bandwidth distribution because it can use the knowledge about all outgoing streams. Additionally, the server may take the clients' payment offers into account when assigning QoS to each session. This is an important issue for scenarios with billing and charging.

To achieve a best possible QoS distribution among all kinds of application streams (i.e. audio, video, 3D media objects, etc.) the server must use meta information about application stream properties. The meta information describes the stream properties of each media type, in par-

0-7695-0912-6/00 $10.00 © 2000 IEEE

ticular media scaling properties and resource requirements in each quality level. To allow for media scaling, we assume hierarchical data formats like the ones applied in many common audio and video formats (e.g. MPEG-1 and -2, wavelet compressed audio/video [1], etc.). Hierarchical data formats also allow the subdivision of multimedia streams into *sub-streams* (i.e. base layer and extension layers) which may be assigned to different transport channels and service classes. Of course, the quality perceived by the user may be enhanced in case of heavily-loaded networks by transmitting the more important base data in a higher service class than the less important extension data.

In this paper, we propose a transport architecture for a multimedia server system that uses the knowledge of application-level streams in combination with network specific service model information to realize the best media scaling possible and to map the QoS requirements of the applications to the available network resources.

The rest of the paper is organized as follows. Section 2 presents related work and compares our proposal to related approaches. In Section 3, an application scenario is described and the problems to solve in this scenario are discussed. Section 4 outlines the overall server system and discusses the central component of the system, the QoS management. First measurement results in a DiffServ environment are given in Section 5. Section 6 concludes the paper and addresses further work.

2. Related work

There are several approaches for application-level QoS management that are related to our work. Reumann and Shin propose an adaptive resource reservation scheme for multimedia servers from an economic perspective [2]. They concentrate on resource sharing in operating systems (CPU usage) and do not consider actual network-level reservation schemes. Closely related to our work is the coordinated congestion management and bandwidth sharing approach suggested by Padmanabhan [3]. In his paper, he outlines an architecture that shares the bandwidth of a congested bottleneck link among concurrent real-time and non real-time streams with respect to the user's varying QoS preferences. However, it is not obvious from [3] how to implement this concept.

The MPEG-4 standard defines a framework for the description, visualization, and transportation of multiple media objects composed in audiovisual scenes [4]. MPEG-4 provides a generic QoS descriptions for media types. The translation of these QoS description to the network QoS is not specified. This issue is addressed by the IETF in several Internet drafts (IETF working group avt). However, QoS management on network level is not sufficiently addressed in this working group.

Balakrishnan et al. present the idea of an end-to-end network congestion management covering all data streams between two hosts (macroflow) [5]. This concept considers all types of transport protocols extracting the congestion control mechanisms from these protocols. This approach requires changes at least to the implementation of protocols such as TCP. Recently, a new IETF working group called ecm (Endpoint Congestion Management) has been formed to elaborate on this concept. At present, ECM does not address resource reservation issues.

Frank et al. proposed an end-to-end mechanism for the Regulation Of Bandwidth in Intra-Networks (ROBIN, [6]) which determines bandwidth bottlenecks with a global knowledge of the network state based on the end-to-end operation of the mechanism. ROBIN provides the concept of relative Quality of Service and tries to prevent network congestion by reducing the outgoing data rate in sending hosts. However, the ROBIN concept is limited to local or campus networks. The ideas of [6] to map ROBIN in the local network to DiffServ in the wide area network do not solve all requirements for the transmission of scalable multimedia streams. Furthermore, the transparent integration of ROBIN's mechanisms between transport and network layer protocols does not allow for a tight connection to the multimedia applications.

At the network level, the DiffServ architecture allows for service differentiation. It is not intended to provide end-to-end QoS. Therefore, Bernet et al. describe a framework combining the IntServ and DiffServ ideas in order to provide end-to-end QoS delivery to applications [7]. Here, IntServ reservations are bridge-spanned over DiffServ regions. A resource management in the DiffServ regions realizes the resource allocation for the IntServ flows. These resources may be statically or dynamically allocated.

Liao and Campbell propose a bandwidth distribution scheme in DiffServ edge routers based on aggregated bandwidth utility functions [8]. These functions describe the user's relative preference of the allocated bandwidth. This approach aims to reach a proportional utility-fair distribution per DiffServ class.

In contrast to the related work presented above our end-to-end QoS management approach distributes the layers of a media flow *among* DiffServ classes *including* the best-effort class.

3. Application scenario

This section describes the to be solved problems by the transport architecture proposed in this paper. A real-life application scenario is presented and the key problems are discussed in the framework of this scenario.

Consider the application scenario given in Figure 1: A *media server* provides several kinds of multimedia applications to possibly numerous clients throughout the Inter-

net. These applications contain for example video streams, audio streams, 3D animations, virtual reality environments, etc. A typical client/server communication scheme may be as follows: The client initiates a multimedia session and requests a video presentation accompanied by the corresponding audio stream. At a certain point in time, a second video is requested which is intended to be displayed in addition to the first video.

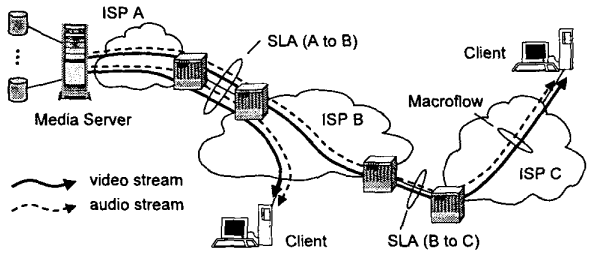

Figure 1. Application scenario

The MPEG-4 framework [4] is a good basis for the combination of any kind of media objects in scenes. MPEG-4 enables the user to arrange the media objects on the screen, to switch on/off objects and to interact with these objects. This enables a highly dynamic behavior on application level, but also results in a complicated situation for the underlying communication architecture: Network resources (i.e. bandwidth) have to be allocated immediately in order to provide a reasonable quality. Furthermore, the system's reaction time (i.e. end-to-end transmission delay) must stay within strict bounds if the user interacts with media objects. For example, if the user switches on a new video object to be displayed synchronized with other media objects, a certain maximum startup latency has to be guaranteed.

The required end-to-end QoS properties are hard to guarantee in the real Internet structured as a network of Internet Service Provider (ISP) networks. Here, the Diff-Serv approach provides some basic mechanisms to choose between services by marking packets. As depicted in Figure 1, Service Level Agreements (SLAs) will be set up between two adjacent ISPs. These SLAs include specifications of allowed traffic in both directions. For example, ISP B may accept and forward high prioritized packets coming from ISP A up to a maximum bandwidth. If this bandwidth is exceeded the out of band data are discarded. By this kind of traffic conditioning rules, different traffic classes are defined to allow the user to allocate the traffic by marking packets, i.e. to assign a real-time stream to a *"better"* class than a non real-time stream. The IETF has defined two forwarding behaviors, namely the *Expedited Forwarding* (EF, RFC 2598) and the *Assured Forwarding* (AF, RFC 2597) to be applied in addition to the best-effort service class.

Assuming a local administration of the network of ISP A in Figure 1 by the media service provider and assuming that the media server may use the total amount of bandwidth specified for each class in the SLA to ISP B for its own traffic, the key problem is how to map the QoS requirements of all streams leaving the server appropriately to the available bandwidth. The problem is even harder to solve when the aggregated bandwidth of several concurrent streams exceeds the SLA's bandwidth limitations. Here, inter-stream QoS management and adaptive media scaling have to be applied in order to reach the best overall service quality.

Our concept assumes the availability of scalable data formats, possibly multi-layered. A good example for this is the MPEG-2 video coding format where a base layer contains basic and an extension layer additional information [9]. The video quality and bandwidth consumption may be scaled down by only transmitting the base layer information. Thus, an MPEG-2 video stream may easily be subdivided into two transport sub-streams.

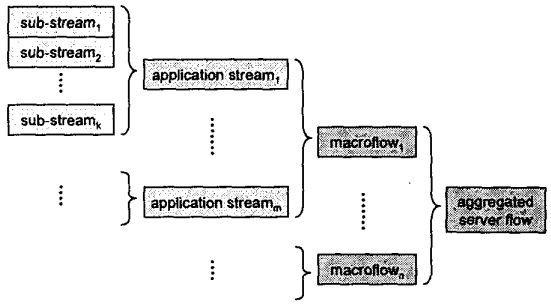

Figure 2. Stream hierarchy

Figure 2 depicts the stream hierarchy: An application stream (e.g. MPEG-2 video) consists of a number of sub-streams (layers). All application streams from the server to a specific client (e.g. video plus audio) form a "macroflow". As each macroflow contains a number of application streams with comparable end-to-end characteristics, the QoS management only considers traffic at macroflow granularity. For example, if all flows in a macroflow are running at low QoS levels, an additionally created flow should not necessarily be started with maximum QoS.

This kind of management concept assumes the server to have detailed information about the scaling properties of each stream type. More precisely, bandwidth requirements, end-to-end delay bounds, and acceptable packet loss rates for each QoS level have to be passed to the server. This information has to be provided by the media content provider.

4. End-to-end QoS management concept

This section describes the QoS management which distributes the available network resources to all active transport streams between a server and all clients. In case of network overload it has to trigger media scaling of

some streams to recover from this situation. This has to occur according to the preferences given by the application and the scaling behavior suitable for the specific media stream type.

4.1. System overview

This section presents an overview of the server's QoS and congestion management system. It also describes a possible method to attach the server to a DiffServ network. We consider a multimedia server that provides and controls real-time streaming of stored multimedia data. Client/server communication takes place on a per-session basis. Multimedia streams may be subdivided into *sub-streams*, i.e. base layer stream and (possibly several) extension layer streams that can be assigned to different transport channels. The regulation by the QoS and congestion management system presented here operates at the granularity level of these sub-streams.

gives feedback to the application in case of media scaling and controls the correct scaling. Furthermore, each RTP transport stream is tagged according to its service class. Tagging may also depend on Forward Error Correction (FEC) mechanisms.

The *QoS Management* module represents the core component of the entire server system. It dynamically maps the QoS requirements of all application streams to the available network resources. Hence, it is responsible for and triggers media scaling if the requirements exceed the resources available. In addition to the (static) QoS information of each application stream, the QoS Management takes (dynamic) inter-stream QoS information into consideration that is the relative weight of all streams belonging to one session. For example, in an MPEG-4 scene the relative QoS precedence of two streamed videos changes if the user's attention changes from one video to the other. The QoS Management maps the media streams to *abstract* service classes without detailed knowledge

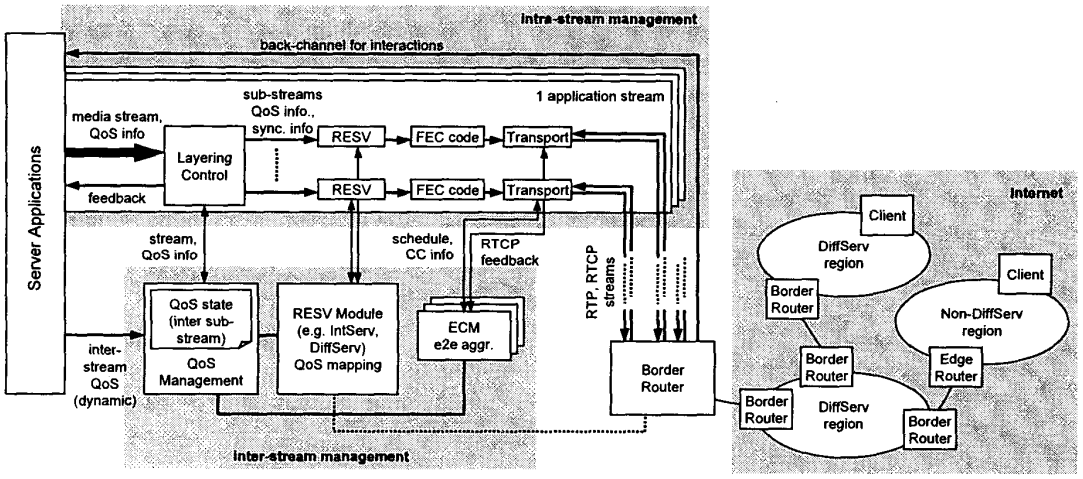

Figure 3. QoS and congestion management system

Figure 3 depicts the functional components of the server's QoS and Congestion Management system. In this scenario, the server is attached to a DiffServ-capable Border Router – either directly or via a local network. The application stream coding is processed by *Server Applications*. End-to-end transport is based on the Real-time Transport Protocol RTP (RFC 1889).

The server's management system mainly consists of an *intra-stream* and an *inter-stream management* block. The *Layering Control* module as the main component of the intra-stream management block functions as an interface between semantic-aware stream information and semantic-unaware RTP transport streams. The Layering Control uses the QoS description for an application stream provided by the server application (see Section 4.2) to manage the transport and synchronization of sub-streams. It

about the *actual* network service model. The actual assignment is done by the *RESV Module*. The separation of the two modules allows for an easy exchange of the basic network service model. In the scenario presented in Figure 3, the Border Router provides its DiffServ configuration parameters (i.e. SLA) to the RESV module.

Another possible network service model may be ATM. However, this is not considered here as ATM support is unlikely on end-to-end basis in the Internet.

Another essential component is the *inter-stream Endpoint Congestion Management* [5] that aggregates the congestion control for all streams of a session (macro-flow) on end-to-end basis. All RTCP receiver reports belonging to a session are intercepted and processed by the corresponding ECM module. The filtered QoS feedback information (i.e. packet loss rate) is not forwarded to

the application process but to the QoS Management. Finally, the ECM modules adjust packet rates and sizes by distributing congestion control information (CC info) to the *Transport Modules*. Currently, the ECM modules only exist as conceptual components. A detailed specification of its functionality and its co-operation with the QoS management will be done in future work.

At the client the corresponding RTP/RTCP handling must be realized. In case of a DiffServ client/server environment the client is not required to do reservations. In contrast to this, the client has to do the signaling if RSVP is used. The client may also provide information about its link capacity to assist the QoS manager. All other tasks are application-specific. The client application may also negotiate about billing and charging which may result in session-specific constraints for the service selection.

4.2. QoS specification

The traffic specification and scaling properties of a media stream are described in its static QoS description. It consists of layer and level specifications. A *layer* describes a part of a media stream corresponding to an RTP transport stream with specific resource requirements. Thus, a base layer may be transferred over a DiffServ EF channel while an extension layer is transferred over a best-effort channel. A *level* defines a quality level of a media stream and includes at least one layer. With this level/layer specification the QoS manager is able to scale the media streams up and down without being aware of the stream semantics.

Two scaling schemes are defined (Figure 4):

1. **The fixed level-based scaling** scheme provides at least one quality level, each consisting of at least one layer. A layer includes information about the frame size and the frames per second. Upper bounds for loss rate and end-to-end delay given in the specification may require the assignment to a specific reservation class. With this scheme, the QoS management is able to switch rapidly between the quality levels.

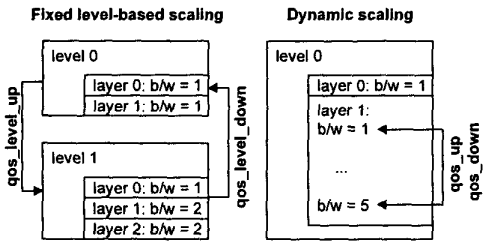

Figure 4. QoS scaling schemes

Jitter is not considered explicitly. However, a specified end-to-end delay gives an upper bound for the jitter. Here, a playback buffer at the client may be used for smoothing.

An example for this scheme is a layered MPEG-1 stream with up to 3 layers (I, P, and B frames) and 3 quality levels (I, I+P, and I+P+B frames). The stream of I frames may be mapped to a reservation class.

2. **The dynamic scaling** scheme consists of exactly one level which includes at least one layer. Rather than describing fixed bandwidth requirements, each layer specifies a valid range for the frame size. The actual frame size is derived from the bandwidth allocated to the stream.

An example suitable for this scheme is an audio wavelet coding or any progressive coding with a rate scalability where a fixed base layer gives a base quality and a variable extension layer increases the quality.

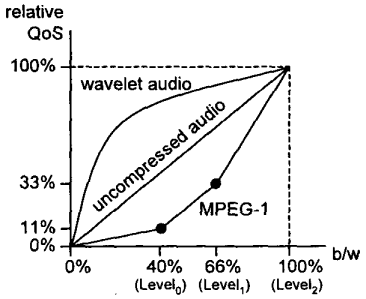

Figure 5. Relative QoS

Relative QoS: Each quality level is associated with a relative QoS value which is an important metric for calculating the fair bandwidth share for a stream. The relative QoS reflects the relation between bandwidth changes and quality provided to the user. This is similar to the utility functions used in [8]. As shown in Figure 5, dropping all B frames of an MPEG-1 coded stream with the Group of Pictures (GoP) format IBBPBBPBB reduces the frame rate to 33% but the average bandwidth needed is only reduced to 66% ($Level_1$) assuming average frame sizes of I=18 KB, P=6 KB and B=2.5 KB. Here, the frame rate is an approximation for the relative QoS. A wavelet coded audio stream has different characteristics. With 50% bandwidth the user achieves much more than 50% relative QoS. The relative QoS / bandwidth function is independent from the absolute bandwidth requirements. As an example, a video stream needs more bandwidth than an audio stream but the relative QoS function may be the same. The relative QoS function obviously depends on the coding scheme. It should be provided by the applications.

4.3. QoS distribution algorithm

As mentioned in Section 4.1, the QoS management obtains abstract static and dynamic QoS information about the media streams from the application. It also gets an abstract description of the QoS classes available at

network level from the RESV module. With this information, the QoS distribution algorithm has to assign the transport streams, i.e. the QoS layers of all media streams, to the abstract QoS classes provided by the network. The decisions are indicated to the ECM and the Transport Modules.

This section outlines a first algorithm. The algorithm's main goals are fair bandwidth distribution across all streams according to their relative QoS and a high overall average relative QoS. It has to guarantee maximum delays and loss rates to support interactive applications. The algorithm includes the following main steps:

Determine the end-to-end delays and loss rates: The end-to-end delays of flows between the server and the clients in a DiffServ environment depend on the reservation class in which the flow is transferred and the network load. Therefore, these delays have to be determined dynamically for each macroflow and for each reservation class to meet the upper bounds requested by the QoS specification of a stream type.

A single round trip time (RTT) measurement is not sufficient for this task in the asymmetric reservation scenario in a DiffServ environment (RFC 2679, RFC 2681). As an example, a downstream message from the server to the client may be transmitted within a reservation class, e.g. EF. The upstream response message may be transmitted within the best-effort class with a much higher delay in case of a congested network. The RTT measurement does not provide any useful information if the response message is sent in the best-effort class. This results in bad utilization of the reservation classes in a scenario with multiple object streams. In case of network congestion, the best-effort traffic will be delayed at the DiffServ routers. To refine the estimation we consider an additional RTT measurement with best-effort transmission in both directions. The delay of a flow in a reservation class (EF in the example) may now be estimated as follows:

$$d_{EF} = RTT_{EF} - RTT_{BE}/2$$

This estimation assumes that the delay of best-effort traffic is similar in up- and downstream direction. Therefore, an underestimation of d_{EF} may be possible. The QoS algorithm also includes moving average functionality similar to TCP's RTT estimation described by

$$d_{new} = (1-\alpha)\cdot d_{old} + \alpha \cdot d_{measured},\ \alpha = \frac{1}{8}$$

The loss rates of the active RTP flows are sent by the clients in RTCP receiver reports. The QoS management collects the loss rates for all macroflows. Here again, a smoothing function is applied:

$$loss_{new} = (1-\alpha)\cdot loss_{old} + \alpha \cdot loss_{measured},\ \alpha = \frac{1}{2}$$

The whole stream is scaled down immediately by one level when the loss rate exceeds a certain bound. If there are no active RTP flows in a specific service class, the loss rate for this class will not be updated. Therefore, the smoothed loss rate value is reset to zero after each reassignment step (see below) in order to prevent invalid loss rate values from being considered in future reassignment steps. The value α is set to 1/2 instead of 1/8 to allow for a more sensible reaction to packet loss in the short interval between two reassignment steps.

(Re-)Assign all streams to the QoS classes: Every N seconds the assignments of the streams to the QoS classes are recalculated. First, the minimal quality $level_0$ of all streams has to be provided. Thus, all layers of this level are inserted in the lowest possible QoS classes with respect to the absolute QoS demands of the stream, i.e. bounds for end-to-end delay and loss rate. If this is not possible, a higher class may be used. It may also become necessary to terminate a stream. All streams are scaled up until an upper bandwidth bound is reached. This bound is the maximum of the free bandwidth in the layer's QoS classes, of the client's bandwidth limit, and of the stream's highest bandwidth requirements. In each round, the stream with the minimal relative QoS will be scaled up first. These steps will be repeated until no more resources are available.

Insert a new stream: A new stream's initial relative QoS is the average relative QoS of all active streams. In a loop, the algorithm scales all active streams down until the QoS requirements of the new stream are met using the appropriate QoS class. In each loop, the stream with the highest relative QoS is scaled down. This calculation includes the new stream. In this step, no layer is moved from one QoS class to a different one. This may only be done in the reassignment step. In this way, frequent changes of quality levels are avoided.

Terminate an active stream: If a stream is terminated by the application, its network resources are released immediately. They are distributed to other streams in the next reassignment step.

The algorithm presented here is intended to be practically applicable rather than providing an optimum QoS distribution. Different variants of the algorithm may be applied that depend on the objectives of the application service provider. For example, the stream-based algorithm described above may be changed to a session-based algorithm to consider upper bandwidth bounds for sessions due to charging mechanisms and/or client restrictions. This issue will be considered in future work.

5. Measurement results

One major objective of the QoS management concept was its applicability in the real world. In this section, the proposed concept is demonstrated by laboratory measurements. For this purpose, a first prototype has been implemented and tested in a DiffServ environment. This

prototyping work and the performance measurements are described below.

5.1. Prototype implementation

The prototype system has been implemented on a Linux platform using a C++ server and Java clients. The server prototype includes the management system shown in Figure 3. In this first version, the ECM modules which implement TCP-friendly behavior and the FEC code modules have not yet been implemented. Both will be realized in future versions.

The media server provides RTP streaming of audio files applying an advanced encoding with up to three separate transport layers. The encoder combines sampling rates from 4410 Hz to 44100 Hz with different sampling unit sizes (from 4 to 16 bit) and mono/stereo quality resulting in 23 QoS levels. This allows for fine-grained up- and down scaling. Layer #0 in each level always is a basic quality while the extension layers #1 and #2 carry additional information (LSBs of audio samples, stereo). The QoS description of this audio stream type uses the fixed level-based scaling scheme (see Section 4.2). Figure 6 shows the bandwidth requirements of all levels including RTP overhead. Levels #0 to #4 consist of one layer, levels #5 to #10 consist of two and levels #11 to #22 consist of three layers.

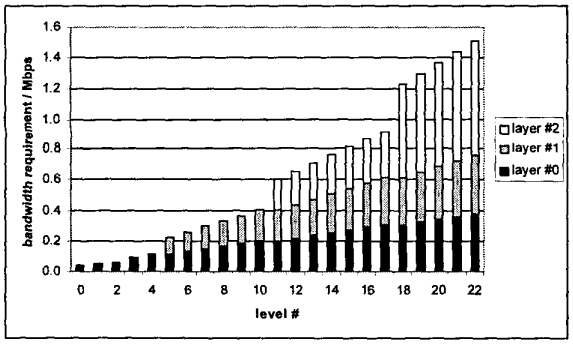

Figure 6. QoS levels of advanced audio encoding

The server considers two dynamically measured client/server network parameters as decision criteria in its QoS algorithm: First, the current packet loss rates to each client are updated every 1-2 seconds by the RTP receiver reports. The second parameter is the end-to-end delay which is estimated by round trip times (see Section 4.3). The round trip time measurements are done once per second using ICMP echo requests. The QoS algorithm's reassignment interval is chosen as $N = 5$ seconds.

5.2. Measurement environment

The measurements intend to show the validity of the system considering packet loss and delay in DiffServ routers. To observe the exact behavior of the distribution algorithm as presented in Section 4.3 we use a simple DiffServ testbed (see Figure 7). A media server transmits audio streams to two clients with one and two DiffServ hops, respectively. Furthermore, a background traffic sender transmits constant bit rate background traffic to the corresponding receiver resulting in concurrent audio and background streams in the core router. All components (server, clients, routers) shown in Figure 7 are run on Linux PCs. This simple testbed is fairly good to introduce the required delay and loss by the background traffic. In case of other background traffic patterns such as variable bit rate flows it might be difficult to distinguish the background traffic from the media flows.

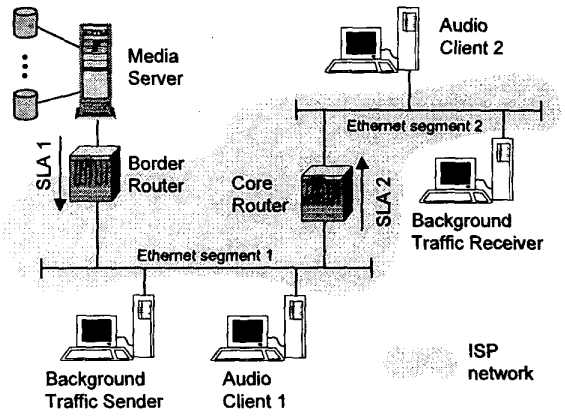

Figure 7. DiffServ network scenario

Both routers are software routers using a DiffServ Linux kernel implementation by Almesberger, Salim, and Kutznetsov [10]. Each router is configured to provide three service classes in addition to the best-effort (BE) service: Expedited Forwarding (EF) and two AF classes with one drop precedence (AF21 and AF11). AF21 is higher prioritized than AF11. The unidirectional bandwidth limitations of SLA 1 and SLA 2 are defined as follows:

	SLA 1	SLA 2
EF	2 Mbps	3 Mbps
AF21	2 Mbps	3 Mbps
AF11	2 Mbps	4 Mbps
BE	4 Mbps	5 Mbps
total	10 Mbps	15 Mbps

As described in Section 3, the total bandwidth of SLA 1 may be completely used by the server. The values of SLA 1 are known by the server and are considered by the QoS distribution algorithm. Furthermore, the server assumes EF to be a "better" service class than AF21, and AF21 better than AF11, and AF11 better than BE.

The similarity of this testbed scenario to a real-world scenario comes clear when considering a scenario where all hosts are connected to the same Internet Service Provider. The server's SLA with the ISP is given by SLA 1. Audio client 1 and 2 are both customers of the ISP where client 2 is located at the opposite side of the ISP network. Thus, the media streaming from the server to client 2 is heavily influenced by other (background) traffic in the core of the network. The core router should limit this core traffic to 15 Mbps (SLA 2).

For measurement purposes, a traffic monitor is installed in each Ethernet segment capturing all packet headers that are sent in the corresponding segment.

5.3. First measurement – packet losses

The first measurement demonstrates the fair bandwidth distribution among all media streams leaving the server and the effect of packet losses introduced by concurrent background streams. The maximum end-to-end delays for the audio streams are set to infinity so that the RTT measurements have no influence on the QoS distribution in this measurement.

Five streams are started during the measurement: The server sends two audio streams, one to each client. The other streams are background streams generated between the background traffic sender/receiver pair, one in each reservation class:

1 EF stream with 2 Mbps
1 AF21 stream with 2 Mbps
1 AF11 stream with 4 Mbps.

Concentrating on packet losses in this measurement, we expect that these background streams only affect the transmission of the audio stream to client 2 by packet losses in the core router when the SLA limits are exceeded. Note that the background streams also affect the stream to client 1 in terms of jitter.

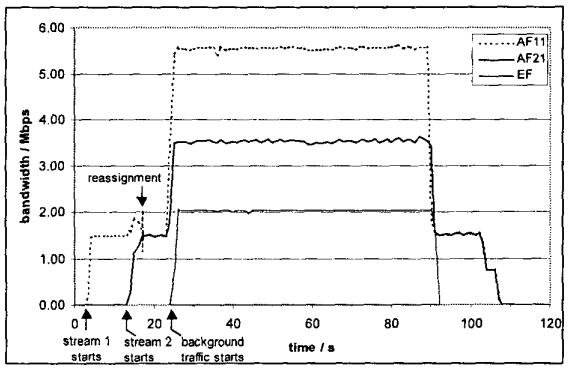

Figure 8. Throughput at the border router

Figure 8 depicts the aggregated traffic monitored in Ethernet segment 1. At time t=3s, the audio stream to client 1 is started with the maximum quality level #22 (see Figure 6). All three layers with a total amount of approximately 1.51 Mbps are marked as AF11. At time t=13s the second audio stream to client 2 is started. As the traffic specification of SLA 1 allows 2 Mbps for the AF11 class and 1.51 Mbps are already used, just one layer of stream 2 fits to this class. The other two layers are assigned to the AF21 class. In the next reassignment phase at t=17s, all six layers of both streams are completely reordered: The layers #0 and #1 of both streams are assigned to the higher prioritized AF21 class and layer #2 of both streams is assigned to AF11. The bandwidth requirement of layer #2 equals the total requirements of layers #0 and #1, so the total bandwidth consumed is now evenly shared between both AF classes. At t=23s, the three background streams are started. The monitored traffic in Ethernet segment 1 is now increased by the bandwidth amounts mentioned above.

The background traffic sender puts load on the core router and exceeds the AF11 bandwidth of SLA 2 (4 Mbps) with its AF11 stream. At the same time, layer #2 of the audio stream to client 2 uses the AF11 class with 0.75 Mbps. This results in packet loss at the core router.

Figure 9 shows the throughput of the audio stream client 2 received in Ethernet segment 2. It may be seen that the AF11 portion of stream 2 suffers packet losses from

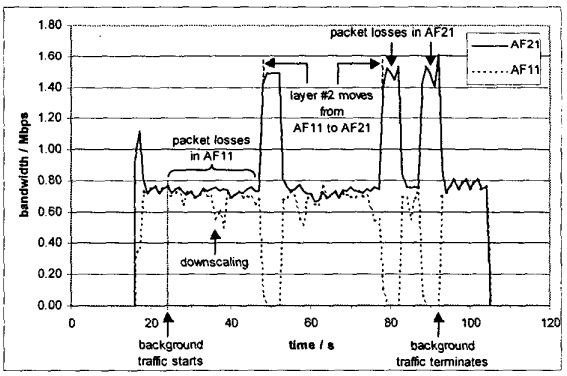

Figure 9. Throughput at client 2

the moment the background traffic is started. The server becomes aware of these losses from the RTP receiver reports and tries to solve the problem with level-wise downscaling. However, if this does not work and the smoothed packet loss rate exceeds the specified upper bound, the corrupted layer #2 of stream 2 is moved from the AF11 to the AF21 class. Now, the AF21 class is overloaded at the core router resulting in AF21 packet drops. This results in layer #2 toggling between the AF21 and the AF11 class which happens three times during the measurement.

The bandwidth shifting in stream 2 between the two AF classes blocks the other audio stream from being sent continuously with 0.75 Mbps in each AF class. Figure 10

shows the throughput of the audio stream to client 1 monitored in Ethernet segment 1. The server avoids bandwidth exceeding of SLA 1 by sending the complete

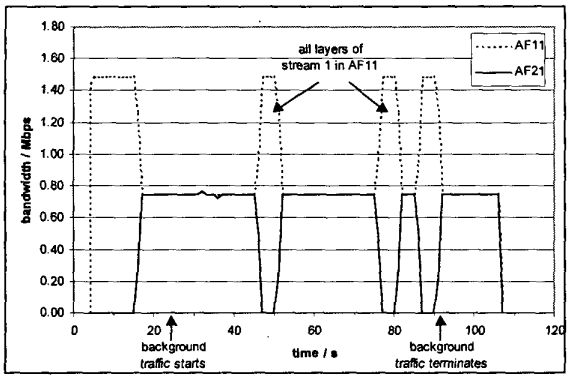

Figure 10. Throughput at client 1

audio stream to client 1 in the AF11 class when the bandwidth is needed for stream 2. However, this should not affect the perceived audio quality at client 1 because there is no background traffic disturbing this audio stream.

5.4. Second measurement – end-to-end delay

While the previous measurement showed the effect of packet losses, this section studies the effect of end-to-end delay bounds.

In this measurement, audio client 1 does not participate because there are no noticeable RTT variations between the server and this client. Instead, two audio streams are directed to audio client 2. Furthermore, two background streams are generated:

1 AF21 stream with 2 Mbps
1 AF11 stream with 4 Mbps

In conjunction with this background traffic the RTT is expected to vary over the time because of the overload in the core router.

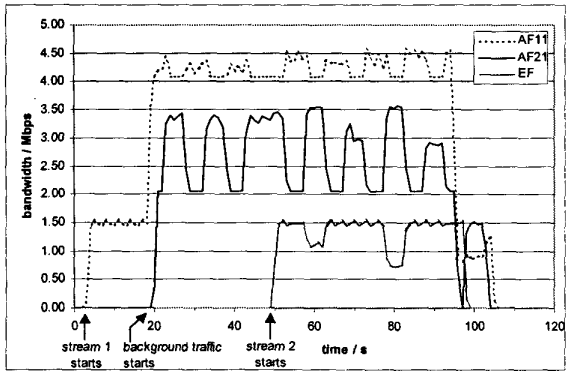

Figure 11. Throughput at the core router

Figure 11 presents the aggregated traffic across the core router as monitored in Ethernet segment 2. At time t=3s, the first audio stream is started. It has no end-to-end delay restrictions. The two background streams start approximately at time t=18s. At time t=48s, the second audio stream starts. In contrast to stream 1 it has a maximum end-to-end delay bound of 5ms which has to be considered by the QoS distribution algorithm.

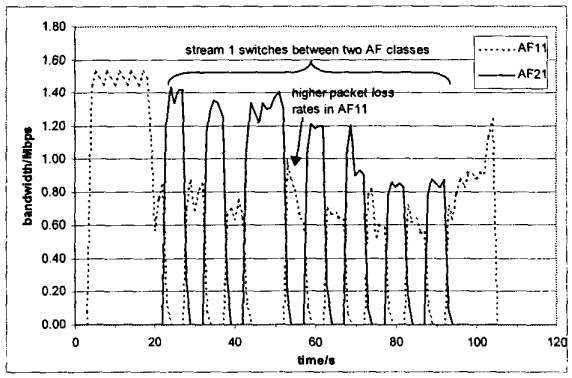

Figure 12. Throughput at client 2 – stream 1

In Figure 12 the behavior of stream 1 can be seen. When the background traffic has started, the complete stream is switched again and again between the two AF classes as described in the previous section. The packet losses in the AF11 class are higher than those in the AF21 class resulting in a higher quality level while transmitted in AF21.

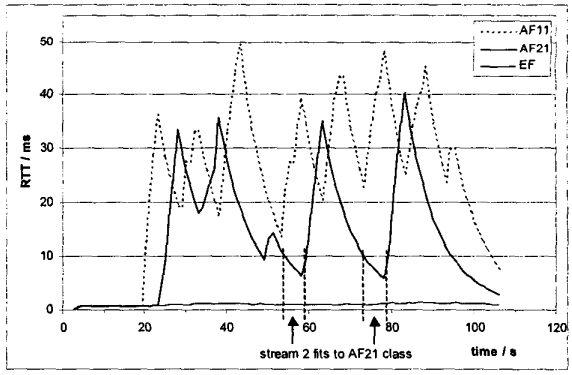

Figure 13. RTTs to client 2

The more interesting part is the transmission of stream 2 because of its end-to-end delay restriction. Figure 13 shows the smoothed round trip times for the three classes measured between the server and audio client 2. While the EF value stays nearly at a constant level, the round trip times for the two AF classes are highly variable. The end-to-end delay is estimated by RTT/2. That means that service classes with RTTs of at most 10ms are suitable for stream 2 (5ms end-to-end delay). Here, EF and, partially, AF21 would be appropriate selections.

Figure 14 depicts the per-class throughput of audio stream 2 monitored in Ethernet segment 2. Most of the stream is sent in the EF class in order to meet the end-to-end delay requirements. When comparing Figures 13 and 14, it comes clear that some parts (layers) of stream 2 are scheduled to AF21 during the same time intervals when the AF21 RTT is small enough. This additional AF21 traffic overloads the core router and, for this reason, the AF21 round trip time increases drastically. Now, all packets of stream 2 are reassigned to EF.

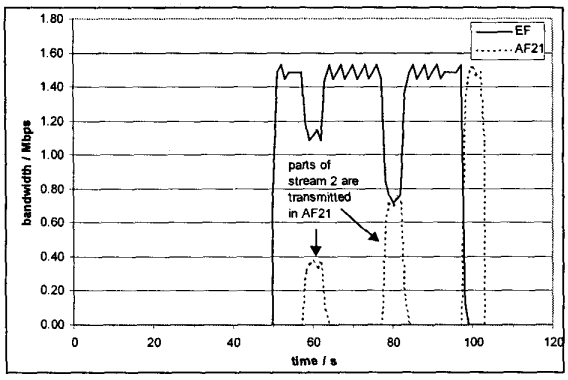

Figure 14. Throughput at client 2 – stream 2

A conclusion that can be drawn from this result is that the QoS management tries to allocate resources in the lowest possible service class and does not unnecessary waste higher service class resources.

6. Conclusions and further work

In this paper, we have proposed a QoS management system supporting end-to-end QoS for multimedia applications over DiffServ. This approach integrates application-level end-to-end quality requirements and a differentiated services network architecture. Media streams which are subdivided into sub-streams according to their scaling properties are mapped to QoS classes provided by the network. The goals of the QoS distribution algorithm are to achieve a maximum average relative QoS among all streams and to realize a fair bandwidth distribution across all streams considering their relative QoS. This algorithm has been implemented in a prototype system and evaluated in a laboratory DiffServ environment. The measurements with layered audio streams show that the QoS system is able to react properly on end-to-end delay and loss rate variations. We show that packet loss rates exceeding certain bounds result in the reassignment of the affected layers to higher service classes according to the QoS requirements of the stream. The end-to-end delay measurements show that the QoS management tries to allocate resources in the lowest possible service class and does not unnecessary waste higher service class resources. Therefore, our approach can be applied for an end-to-end QoS management system in DiffServ networks.

Our further work will include the improvement and the comparative evaluation of different variants of the QoS algorithm. The prototype will be enhanced by additional media types (e.g. scalable video). Furthermore, we will integrate a TCP-friendly congestion management for the best-effort service class. A performance analysis of the system's behavior in large Internet scenarios will be done by simulation. Additionally, a bandwidth brokering concept as presented in [11][12] will be considered to allow the server for dynamic SLA negotiation.

References

[1] J. Karlelar, U.B. Desai. Content Based Video Compression for High Perceptual Quality Videoconferencing using Wavelet Transform. Proc. of ICCIMA'99, IEEE Comput. Soc., September 1999.

[2] J. Reumann, K.G. Shin. Adaptive Quality-of-Service Session Management for Multimedia Servers. NOSSDAV'98, 1998.

[3] V.N. Padmanabhan. Coordinated Congestion Management and Bandwidth Sharing for Heterogeneous Data Streams. NOSSDAV'99, 1999.

[4] R. Koenen. Overview of the MPEG-4 Standard - (Noordwijkerhout Version). ISO/IEC JTC1/SC29/WG11 N3342, March 2000.

[5] H. Balakrishnan, H.S. Rahul, S. Seshan. An Integrated Congestion Management Architecture for Internet Hosts. ACM SIGCOMM'99, 1999.

[6] S. Baatz, M. Frank, P. Martini, A. Wenzel. Flow Aggregation and Scalability with ROBIN: End-to-End Regulation of Bandwidth in Intra-Networks. Proc. of LCN'99, Boston, MA, October 1999.

[7] Y. Bernet et al. A Framework For Integrated Services Operation Over Diffserv Networks. IETF, Internet Draft <draft-ietf-issll-diffserv-rsvp-05.txt>, May 2000.

[8] R.R.-F. Liao, A.T. Campbell. Dynamic Edge Provisioning for Core IP Networks. Proc. of IEEE IWQoS 2000, June 2000.

[9] ISO/IEC DIS 13818-2, Information technology -- Generic coding of moving pictures and associated information: Video. ISO, 1996.

[10] W. Almesberger, J.H. Salim, A. Kuznetsov. Differentiated Services on Linux. Proc. of Globecom'99, December 1999.

[11] M. Günter, T. Braun. Evaluation of Bandwidth Broker Signaling. Proc. of ICNP'99, IEEE Comput. Soc., 1999.

[12] B. Teitelbaum et al. Internet2 QBone: Building a Testbed for Differentiated Services. IEEE Network, September/October 1999.